



End to End Demo on Kubernetes deployments, Ingress & Monitoring/alerting tools

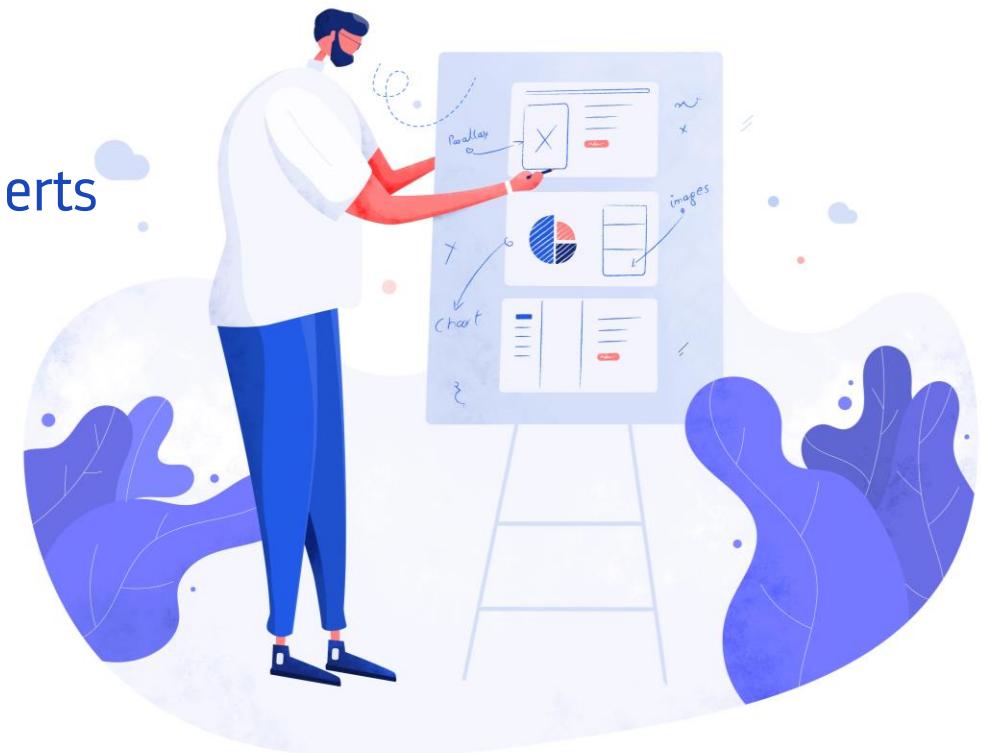




Agenda



- Multi Node K8s Cluster setup
- Application Deployments
- Kubernetes Services
- Deploying Traefik Ingress Controller
- Using host based routing ingress rules
- Deploying Prometheus & Grafana
- Deploying various metric servers
- Configuring Alert Manager for Gmail Alerts
- Installing and Configuring HA Proxy as reverse proxy
- Complete demo





Pre-Requisites

- Kubernetes Multinode cluster. Refer https://github.com/kunchalavikram1427/Kubernetes_public/blob/master/Bootstrap_K8s_Cluster_Kubeadm.pdf to setup Kubernetes cluster in Windows using VirtualBox.
- Knowledge on Docker swarm and Kubernetes. Refer the below documents if you are new to these.
Docker:
https://github.com/kunchalavikram1427/Docker_public/blob/master/Docker_Made_Easy.pdf
Kubernetes:
https://github.com/kunchalavikram1427/Kubernetes_public/blob/master/Kubernetes_Made_Easy.pdf





Note

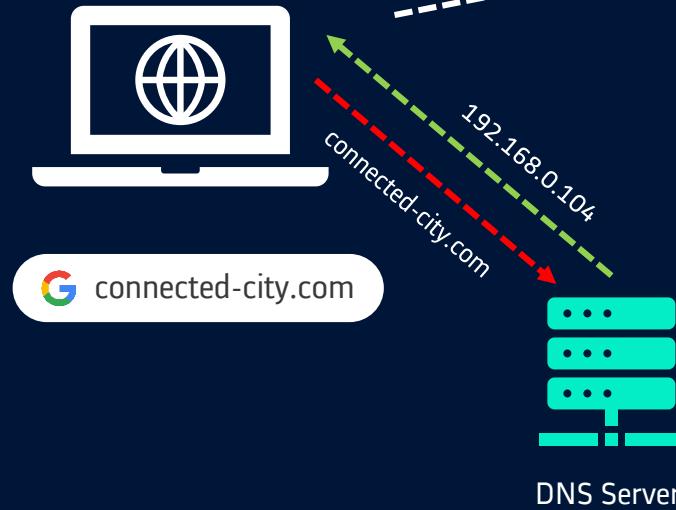
- All IP addresses mentioned in the document are for reference and to be replaced with the IPs of Nodes/Pods according to your setup.
- In all the deployments, only `emptyDir` is used as volume. In Production these should be replaced by `hostPath` or `PVC` accordingly.
- Some content is taken from free courses/blogs available online and is a sole property of the respective content creators.
- The content has been testing in a 3 node Kubernetes cluster setup using Kubeadm in Windows with VirtualBox installed. It should work flawlessly for other multinode cluster setups as well.



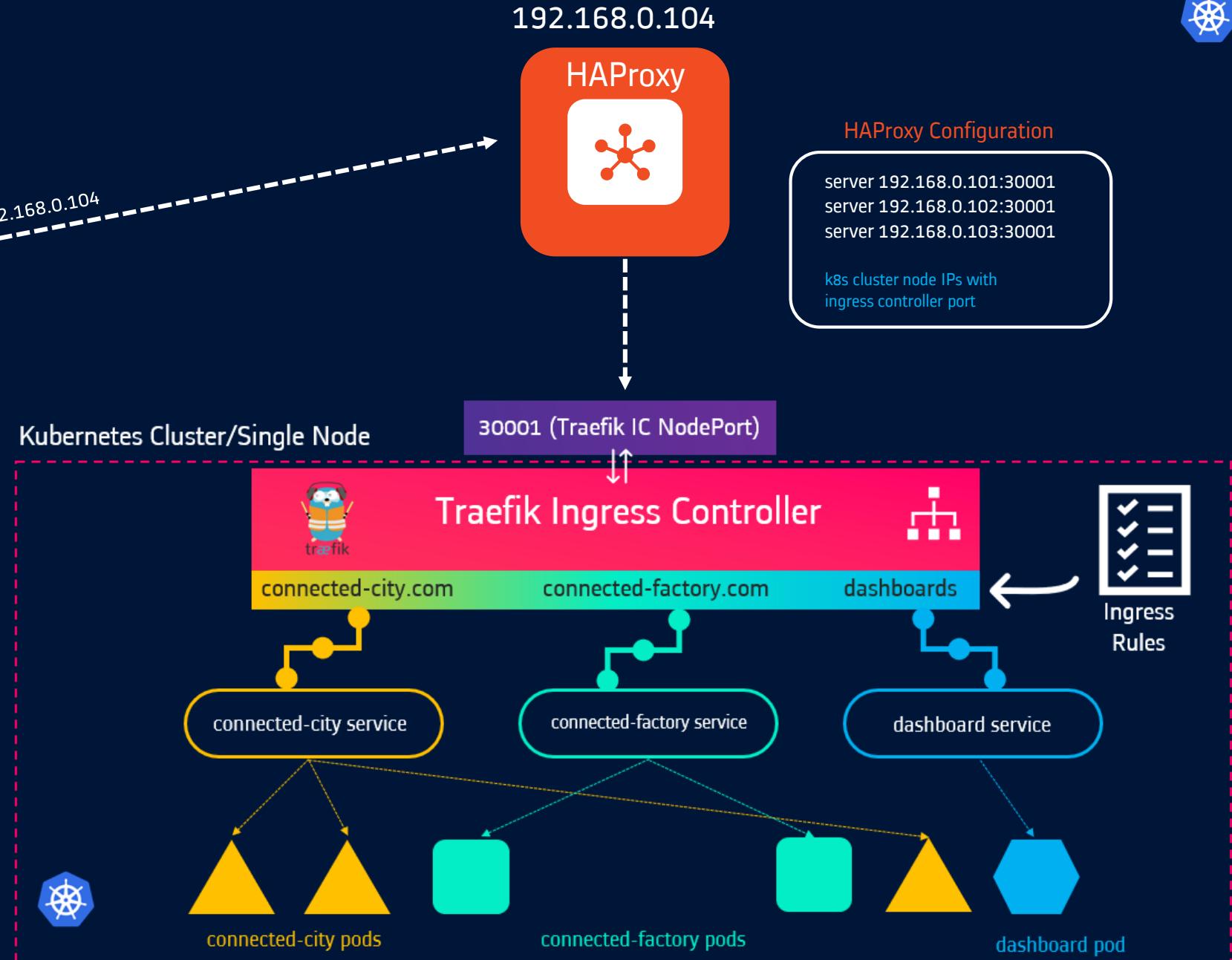


Kubernetes

Demo Architecture



- 3-node cluster + 1VM for Reverse Proxy
- 2 front-end applications
- K8s dashboard
- Traefik Ingress Controller
- Traefik dashboard
- Grafana, Prometheus, Alert Manager dashboards
- Ingress URL based routing





Kubernetes

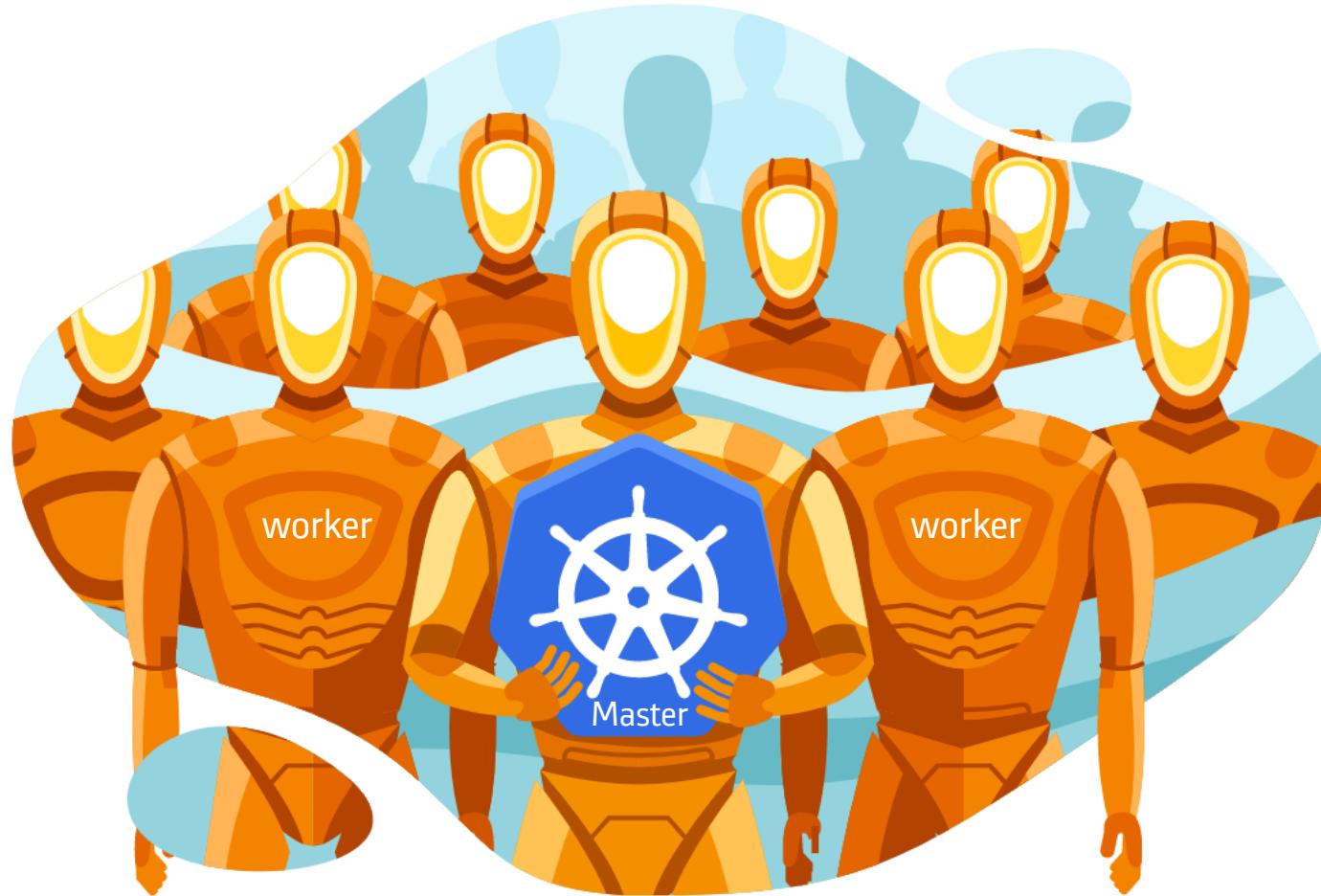
git clone https://github.com/kunchalavikram1427/Kubernetes_public.git

Dockerfiles	Added dockerfiles for flask apps
end_to_end_demo	end to end demo
manifests	Modified
Bootstrap_K8s_Cluster_Kubeadm.pdf	Updated document
Kubernetes_Made_Easy.pdf	Uploaded Kubernetes document
Kubernetes_offline_installation_RHEL7...	Kubernetes offline installation
README.md	Initial commit



Clone the Repo

<https://github.com/kunchalavikram1427>



Cluster Setup

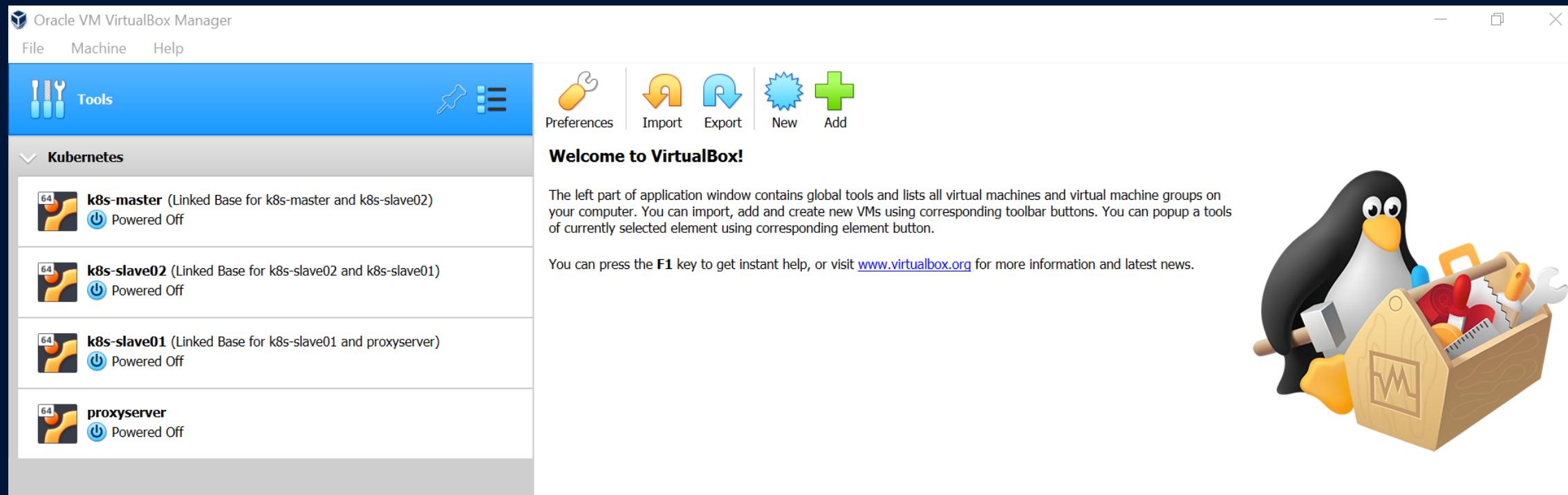
<https://github.com/kunchalavikram1427>



Kubernetes

Cluster Requirements

- 3 Kubernetes nodes: 1 Master and 2 Slaves. Refer pre-requisites section for setup.
- 1 VM to act as Reverse Proxy (Needed for on-prem. For cloud instances, a service type of LoadBalancer for Ingress controller should do)



This tutorial assumes you are running all the manifest files on on-prem Kubernetes cluster setup using any of the tools like Kubeadm. For running on cloud managed clusters like EKS/AKS/GKE, service types are to be changed accordingly.



Kubernetes

Check the cluster once it is setup...

kubectl version --short

```
root@k8s-master:/home/osboxes/Monitoring_tools# kubectl version --short
Client Version: v1.18.3
Server Version: v1.18.3
root@k8s-master:/home/osboxes/Monitoring_tools#
```

kubectl get nodes -o wide

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
node/k8s-master	Ready	master	52d	v1.18.3	192.168.0.102	<none>	Ubuntu 18.04.3 LTS	5.0.0-23-generic	docker://19.3.11
node/k8s-slave01	Ready	<none>	52d	v1.18.3	192.168.0.103	<none>	Ubuntu 18.04.3 LTS	5.0.0-23-generic	docker://19.3.10
node/k8s-slave02	Ready	<none>	52d	v1.18.3	192.168.0.113	<none>	Ubuntu 18.04.3 LTS	5.0.0-23-generic	docker://19.3.10

 We need the IPs of all the nodes in the cluster to configure HA Proxy VM later

kubectl cluster-info

```
root@k8s-master:/home/osboxes# kubectl cluster-info
Kubernetes master is running at https://192.168.50.2:6443
KubeDNS is running at https://192.168.50.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://192.168.50.2:6443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy
```

Kubernetes Deployments & Services

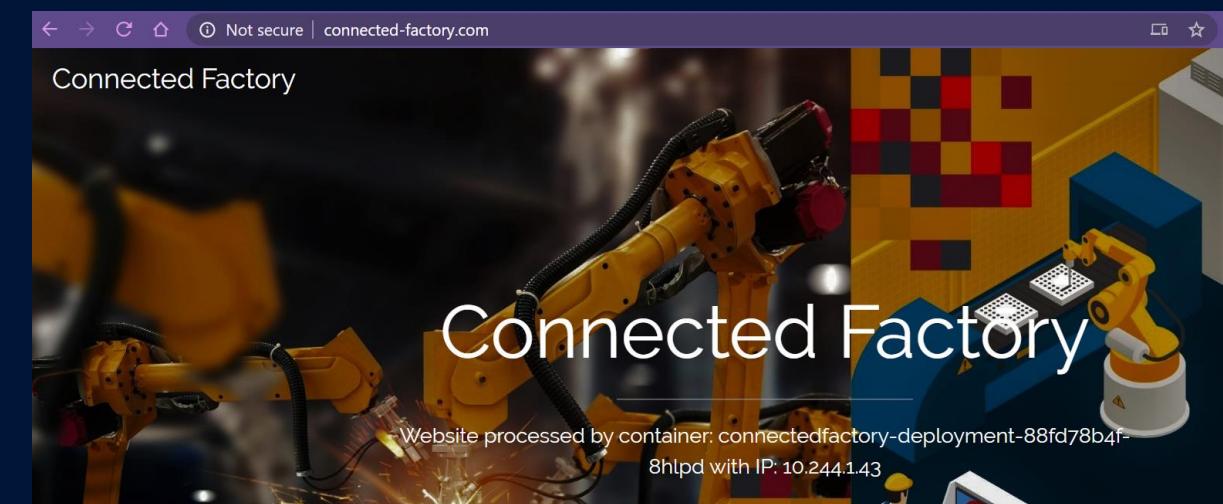
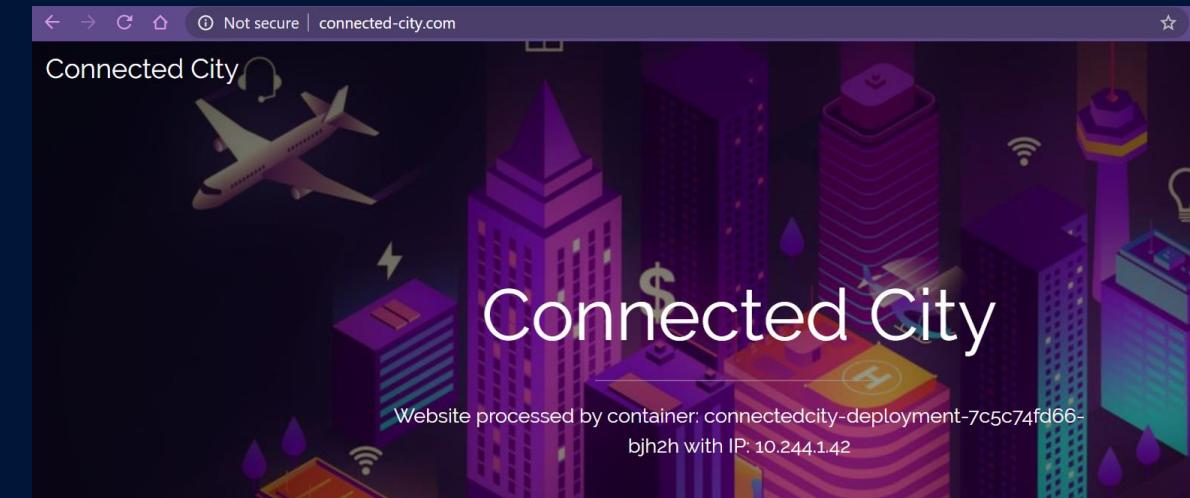




Kubernetes

Application Deployment

- We will deploy 2 flask based applications (connected city & connected factory) which will be accessed later by using the DNS names
- **Dockerfiles** for these applications can be found at
https://github.com/kunchalavikram1427/Kubernetes_public/tree/master/Dockerfiles





Kubernetes

Application Deployment and service manifest files

```
apiVersion: v1
kind: Service
metadata:
  name: connectedcity-service
spec:
  ports:
    - port: 80
      targetPort: 5000
  selector:
    app: connectedcity
```

Application-1
Deployment + ClusterIP
service

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: connectedcity-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: connectedcity
  template:
    metadata:
      labels:
        app: connectedcity
    spec:
      containers:
        - name: connectedcity
          image: kunchalavikram/connectedcity:v1
          ports:
            - containerPort: 5000
```

```
apiVersion: v1
kind: Service
metadata:
  name: connectedfactory-service
spec:
  ports:
    - port: 80
      targetPort: 5000
  selector:
    app: connectedfactory
```

Application-2
Deployment + ClusterIP
service

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: connectedfactory-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: connectedfactory
  template:
    metadata:
      labels:
        app: connectedfactory
    spec:
      containers:
        - name: connectedfactory
          image: kunchalavikram/connectedfactory:v1
          ports:
            - containerPort: 5000
```



Kubernetes

Demo files

- Change to `end_to_end_demo` directory in the GitHub project cloned before.
- All demo related files are arranged into various sub directories.

```
root@k8s-master:/home/osboxes/Kubernetes_public# cd end_to_end_demo/
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# ls -l
total 52
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 alertmanager
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 application_deployment
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 dashboard_ingress_rules
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 grafana
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 grafana_dashboards
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 HA_proxy_configuration
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 k8s_dashboard
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 kube-state-metrics
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 metrics-server
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 node-exporter
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 prometheus
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 pushgateway
drwxrwxrwx 2 osboxes osboxes 4096 Aug  3 02:56 traefik_ingress_controller
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```



Kubernetes

Deploying the Applications

- `cd application_deployment`
- `kubectl apply -f applications_deployments_with_clusterip.yml`

This will deploy all the applications with ClusterIP as the service in the default namespace

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# ls -l
total 8
-rwxrwxrwx 1 osboxes osboxes 1156 Aug  2 23:47 applications_deployments_with_clusterip.yml
-rwxrwxrwx 1 osboxes osboxes  471 Aug  2 23:46 applications_ingress_rules.yml
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# kubectl apply -f applications_deployments_with_clusterip.yml
deployment.apps/connectedcity-deployment created
service/connectedcity-service created
deployment.apps/connectedfactory-deployment created
service/connectedfactory-service created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# █
```



Kubernetes

Deploying the Applications

- `kubectl get all`

To show all the pods, replicaset, deployments and services

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/connectedcity-deployment-7c5c74fd66-9dpd5   1/1     Running   0          54s
pod/connectedcity-deployment-7c5c74fd66-t245x   1/1     Running   0          54s
pod/connectedcity-deployment-7c5c74fd66-t6kz5   1/1     Running   0          54s
pod/connectedfactory-deployment-88fd78b4f-56fgl  1/1     Running   0          54s
pod/connectedfactory-deployment-88fd78b4f-g56ml  1/1     Running   0          54s
pod/connectedfactory-deployment-88fd78b4f-gxdxg  1/1     Running   0          54s

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/connectedcity-service  ClusterIP   10.99.29.180 <none>       80/TCP     54s
service/connectedfactory-service ClusterIP  10.111.82.232 <none>       80/TCP     54s
service/kubernetes            ClusterIP  10.96.0.1     <none>       443/TCP   68m

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/connectedcity-deployment  3/3     3           3           54s
deployment.apps/connectedfactory-deployment 3/3     3           3           54s

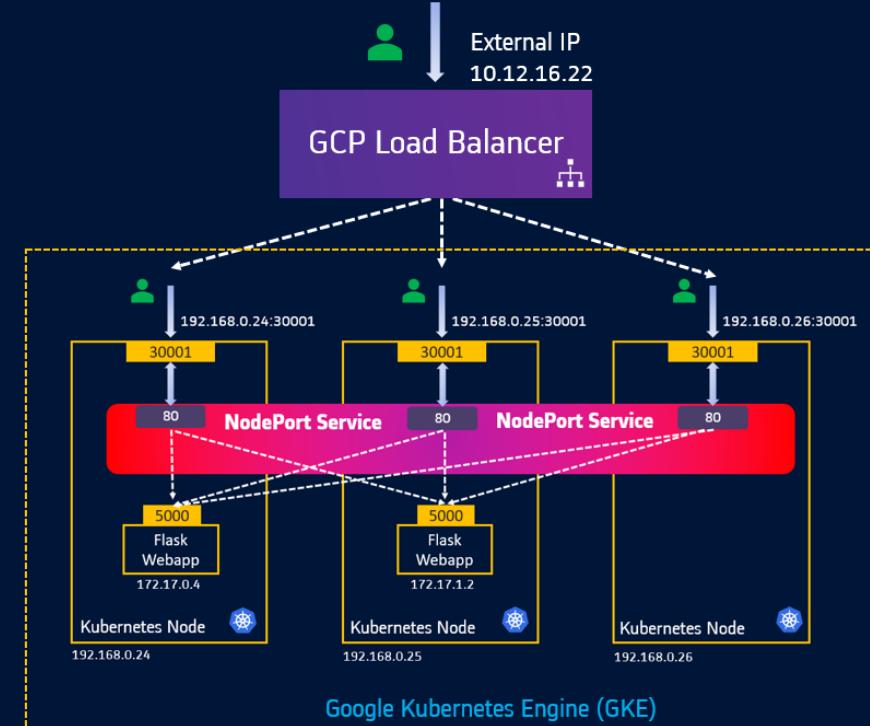
NAME           DESIRED   CURRENT   READY   AGE
replicaset.apps/connectedcity-deployment-7c5c74fd66  3         3         3         54s
replicaset.apps/connectedfactory-deployment-88fd78b4f  3         3         3         54s
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment#
```



Kubernetes

Ingress Resource(rules)

- With cloud **LoadBalancers**, we need to pay for each of the service that is exposed using LoadBalancer as the service type. As services grow in number, complexity to manage SSLs, Scaling, Auth etc., also increase
- Ingress** allows us to manage all of the above within the Kubernetes cluster with a definition file, that lives along with the rest of your application deployment files
- Ingress controller can perform load balancing, Auth, SSL and URL/Path based routing configurations by being inside the cluster living as a **Deployment** or a **DaemonSet**
- Ingress helps users access the application using a **single externally accessible URL**, that you can configure to route to different services within your cluster based on the URL path, at the same time terminate SSL/TLS





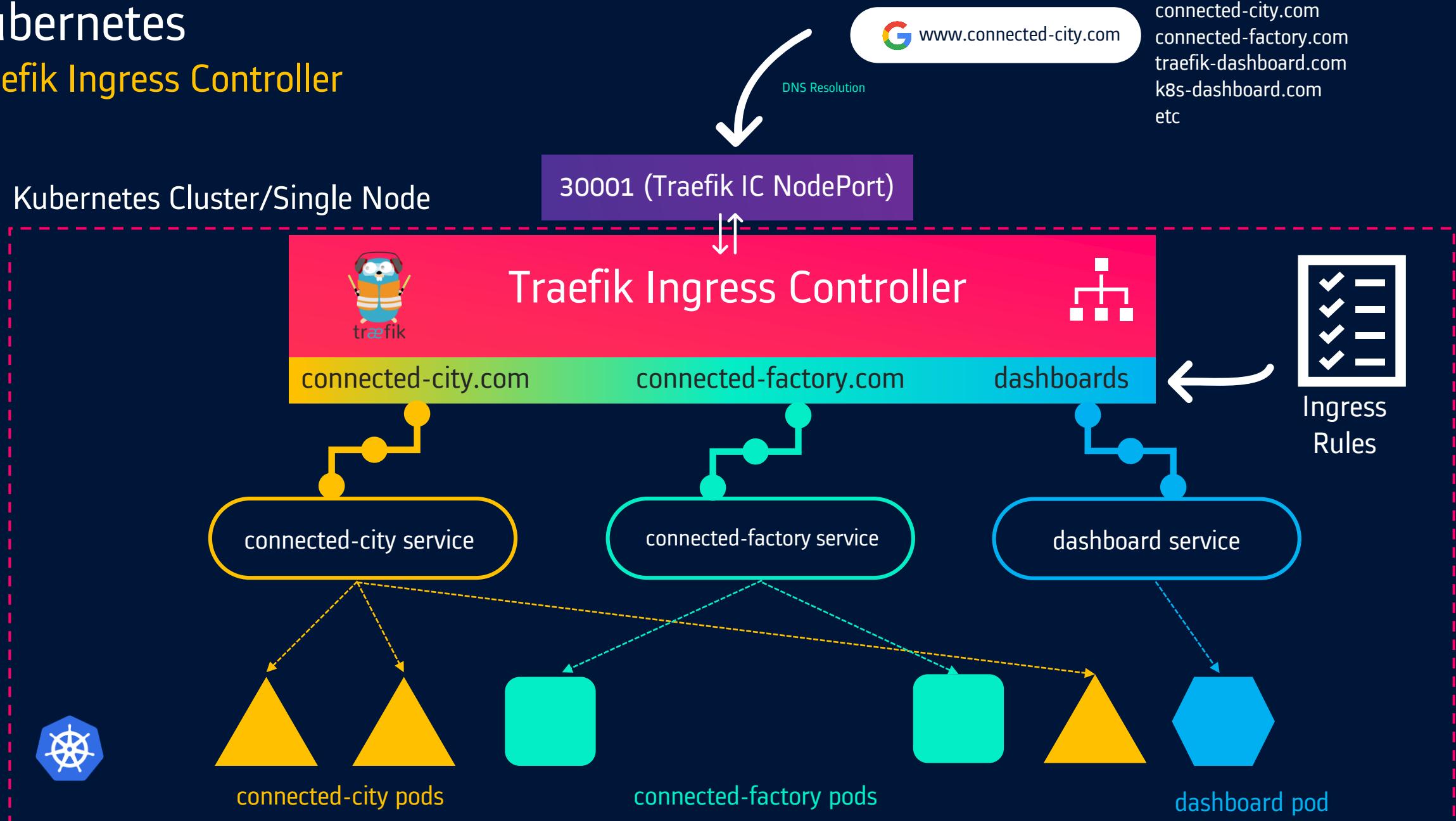
Kubernetes

Ingress Controller

- Ingress resources cannot do anything on their own. We need to have an Ingress controller in order for the Ingress resources to work
- Ingress controller implements rules defined by ingress resources
- Ingress controllers doesn't come with standard Kubernetes binary, they have to be deployed separately
- Kubernetes currently supports and maintains GCE and nginx ingress controllers
- Other popular controllers include Traefik, HAProxy ingress, istio, Ambassador etc.,
- Ingress controllers are to be exposed outside the cluster using NodePort or with a Cloud Native LoadBalancer.
- Ingress is the most useful if you want to expose multiple services under the same IP address
- Ingress controller can perform load balancing, Auth, SSL and URL/Path based routing configurations by being inside the cluster living as a Deployment or a DaemonSet

Kubernetes

Traefik Ingress Controller





Kubernetes

Deploying Ingress rules for the Applications

- `kubectl apply -f applications_ingress_rules.yml`
- `kubectl get ing application-ingress-rules`
- `kubectl describe ing application-ingress-rules`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# kubectl apply -f applications_ingress_rules.yml
ingress.networking.k8s.io/application-ingress-rules created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# kubectl get ing application-ingress-rules
NAME          CLASS      HOSTS                           ADDRESS      PORTS   AGE
application-ingress-rules <none>    connected-city.com,connected-factory.com        80          12s
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment# kubectl describe ing application-ingress-rules
Name:           application-ingress-rules
Namespace:      default
Address:
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
Host          Path  Backends
-----
connected-city.com
connected-factory.com
Annotations:   kubernetes.io/ingress.class: traefik
Events:        <none>
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment#
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: application-ingress-rules
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
    - host: connected-city.com
      http:
        paths:
          - backend:
              serviceName: connectedcity-service
              servicePort: 80
    - host: connected-factory.com
      http:
        paths:
          - backend:
              serviceName: connectedfactory-service
              servicePort: 80
```



Ingress controller will be deployed in the later part of this tutorial

Metrics Server





Kubernetes

Metrics Server

- Metrics Server is a cluster-wide aggregator of resource usage data.
- It collects metrics like CPU or memory consumption for containers or nodes, from the Summary API, exposed by Kubelet on each node.
- Metrics API can also be accessed by `kubectl top`, making it easier to debug autoscaling pipelines.
- Metric server comes installed by default in some of the K8s distributions. Do check for metrics-server pods before installing one.
- We also need `Kube-state-metrics` server to expose container and pod metrics other than those exposed by cAdvisor on the nodes.
- `kube-state-metrics` is focused on the health of the various objects inside, such as deployments, nodes and pods.



Kubernetes

Deploy Metrics Server

- `cd metrics-server`
- `kubectl apply -f metrics_server.yml`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# cd metrics-server/
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# ls
metrics_server.yml
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# kubectl apply -f metrics_server.yml
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader unchanged
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator unchanged
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader unchanged
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io unchanged
serviceaccount/metrics-server unchanged
deployment.apps/metrics-server unchanged
service/metrics-server unchanged
clusterrole.rbac.authorization.k8s.io/system:metrics-server unchanged
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server unchanged
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# █
```

View Metrics

- `kubectl top nodes`
- `kubectl top pods`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# kubectl top nodes
NAME          CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
k8s-master     88m         4%      1222Mi        64%
k8s-slave01    60m         3%      907Mi         48%
k8s-slave02    1039m       51%     938Mi        49%
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# kubectl top pods
NAME                           CPU(cores)   MEMORY(bytes)
connectedcity-deployment-7c5c74fd66-9dpd5   6m          32Mi
connectedcity-deployment-7c5c74fd66-t245x   6m          32Mi
connectedcity-deployment-7c5c74fd66-t6kz5   7m          32Mi
connectedfactory-deployment-88fd78b4f-56fgl  6m          32Mi
connectedfactory-deployment-88fd78b4f-g56ml  7m          32Mi
connectedfactory-deployment-88fd78b4f-gxdxg  6m          32Mi
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/metrics-server# █
```



Kubernetes may not generate metrics as soon as metric server is installed. Check back after sometime.



Kubernetes

Deploy Kube-state-metrics Server

- Before deploying kube-state-metrics, we will create a namespace called monitoring
- `kubectl create ns monitoring`
- `cd kube-state-metrics`
- `kubectl apply -f kube_state_metrics.yaml`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/kube-state-metrics# kubectl create ns monitoring
namespace/monitoring created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/kube-state-metrics# k apply -f kube_state_metrics.yaml
clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics unchanged
clusterrole.rbac.authorization.k8s.io/kube-state-metrics unchanged
rolebinding.rbac.authorization.k8s.io/kube-state-metrics created
role.rbac.authorization.k8s.io/kube-state-metrics-resizer created
serviceaccount/kube-state-metrics created
deployment.apps/kube-state-metrics created
service/kube-state-metrics created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/kube-state-metrics# █
```



Please note `kube_state_metrics` deployments will be unavailable state until we deploy some objects in monitoring namespace.

Kubernetes Dashboard





Kubernetes

K8 Dash

- K8 Dash is one of the many dashboards available for Kubernetes.
- It is the easiest way to visualize and manage your Kubernetes cluster.
- It provides full cluster management: Namespaces, Nodes, Pods, Replica Sets, Deployments, Storage, RBAC and more.
- Blazing fast and Always Live: no need to refresh pages to see the latest
- Quickly visualize cluster health at a glance: Real time charts help quickly track down poorly performing resources

<https://github.com/indeedeng/k8dash>

<https://github.com/kunchalavikram1427>



Kubernetes

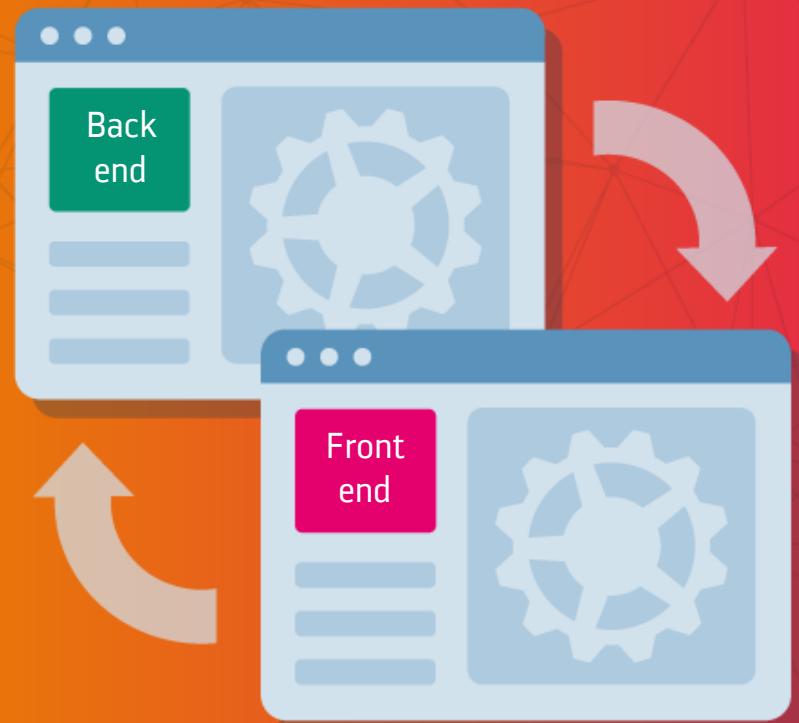
Deploy K8 Dash & its Ingress rules

- We will deploy K8dash in **kube-system** namespace
- cd k8s_dashboard/
- kubectl apply -f k8s_dashboard_with_clusterip.yml -f k8s_dashboard_ingress_rules.yml
- kubectl get all -n kube-system | grep -i k8dash

k8s_dashboard_ingress_rules.yml

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: kubernetes-web-ui
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: k8s-dashboard.com
    http:
      paths:
      - backend:
          serviceName: k8dash
          servicePort: 80
```

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/k8s_dashboard# kubectl get all -n kube-system | grep -i k8dash
pod/k8dash-84978cd779-lmmk2           1/1     Running   2          8d
service/k8dash                         ClusterIP  10.98.217.226  <none>    80/TCP   8d
deployment.apps/k8dash                 1/1     1          1          8d
replicaset.apps/k8dash-84978cd779     1         1          1          1          8d
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/k8s_dashboard#
```



Ingress Controller



Kubernetes

Traefik Ingress Controller deployment

- We will now deploy traefik ingress controller and Ingress rules to access its dashboard via a DNS name.
 - All these objects will be deployed in **kube-system** namespace.
-
- `cd traefik_nginx_controller/`
 - `kubectl apply -f traefik_nginx_controller.yml -f traefik_dashboard_ingress_rules.yml`

traefik_dashboard_ingress_rules.yml

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: traefik-web-ui
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: traefik-dashboard.com
    http:
      paths:
      - backend:
          serviceName: traefik-web-ui
          servicePort: 80
```



```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# cd traefik_nginx_controller/
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_nginx_controller# ls
traefik_dashboard_ingress_rules.yml  traefik_nginx_controller.yml
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_nginx_controller# kubectl apply -f traefik_nginx_controller.yml -f traefik_dashboard_ingress_rules.yml
clusterrole.rbac.authorization.k8s.io/traefik-ingress-controller unchanged
clusterrolebinding.rbac.authorization.k8s.io/traefik-ingress-controller unchanged
serviceaccount/traefik-ingress-controller unchanged
deployment.apps/traefik-ingress-controller unchanged
service/traefik-ingress-service unchanged
service/traefik-web-ui unchanged
ingress.networking.k8s.io/traefik-web-ui configured
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_nginx_controller#
```



Kubernetes

Traefik Ingress Controller deployment

- `kubectl get all -n kube-system | grep -i traefik`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_ingress_controller# kubectl get all -n kube-system | grep -i traefik
pod/traefik-ingress-controller-78b4959fdf-vqf57  1/1    Running   2          8d
service/traefik-ingress-service  NodePort  10.98.4.68  <none>      80:30001/TCP,8080:30933/TCP  8d
service/traefik-web-ui          ClusterIP 10.99.74.220 <none>      80/TCP                8d
deployment.apps/traefik-ingress-controller 1/1    1          1          8d
replicaset.apps/traefik-ingress-controller-78b4959fdf  1        1        1        8d
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_ingress_controller#
```

 Traefik Ingress controller is listening on NodePort 30001 which is manually configured in the NodePort service.



Prometheus & Grafana



Kubernetes

Prometheus and Grafana



- Monitoring is a crucial aspect of any Ops pipeline and for technologies like Kubernetes which is a rage right now, a robust monitoring setup can bolster your confidence to migrate production workloads from VMs to Containers.
- The combination of **Prometheus** and **Grafana** is becoming a more and more common monitoring stack for storing and visualizing time series data. Prometheus acts as the storage backend and Grafana as the interface for analysis and visualization.
- **Prometheus** exposes a wide variety of metrics that can be easily monitored. By adding Grafana as a visualization layer, we can easily set up a monitoring stack .
- **Prometheus** collects metrics from pre-defined targets via a pull model. Targets can be found by Service Discovery or manually configured to pull data from endpoints of your application like an API server, redis server or SQL server.
- **Grafana** supports lots of data sources such as CloudWatch, Stackdriver, Elasticsearch, and sure, Prometheus. After connecting Grafana with the data sources, you could create dashboard and panel manually, or import already existing dashboards.



Kubernetes

Alert Manager, Node Exporter and Pushgateway service

- Along with Prometheus and Grafana, we will also deploy **Alert Manager** to generate alerts and select notifications via emails, slack notifications etc. We will configure the alert rules and deploy them along with Prometheus
- Node Exporter is used to collect Node specific metrics like hardware and OS metrics exposed by NIX kernels.
- The **Pushgateway** is an intermediary service which allows you to push metrics from jobs which cannot be scraped by Prometheus. It allow ephemeral and batch jobs to expose their metrics to Prometheus. Since these kinds of jobs may not exist long enough to be scraped, they can instead push their metrics to a Pushgateway. The Pushgateway then exposes these metrics to Prometheus



Kubernetes

Deploying Prometheus & Grafana

- Prometheus and Grafana will be deployed in **Monitoring** namespace.
- The metrics and endpoints to scrape along with the alert rules are all injected as configmaps into Prometheus. Check the manifest files for complete configuration.
- `kubectl apply -f prometheus/`
- `kubectl apply -f grafana`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# ls
alertmanager      dashboard-ingress-rules  grafana_dashboards  k8s_dashboard    metrics-server   prometheus   traefik-ingress-controller
application_deployment  grafana          HA_proxy_configuration  kube-state-metrics  node-exporter  pushgateway
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f prometheus/
configmap/prometheus-server-conf created
serviceaccount/monitoring created
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
deployment.apps/prometheus-deployment created
service/prometheus-service created
configmap/prometheus-rules created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f grafana
configmap/grafana-datasources created
deployment.apps/grafana created
service/grafana created
secret/grafana-secret created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```



Kubernetes

Deploying Alert Manager, Node Exporter and Pushgateway service

- Configure SMTP server settings in the Alert Manager configMap before deploying the Alert Manager.
- We can also use 1 time App password from Google services, instead of providing the real password in the configuration. Refer <https://grafana.com/blog/2020/02/25/step-by-step-guide-to-setting-up-prometheus-alertmanager-with-slack-pagerduty-and-gmail/>
- kubectl apply -f alertmanager/
- kubectl apply -f pushgateway/
- kubectl apply -f node-exporter/

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/grafana# cd ..
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f alertmanager/
configmap/alertmanager created
deployment.apps/alertmanager created
service/alertmanager created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f pushgateway/
deployment.apps/pushgateway-deployment created
service/pushgateway-service created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f node-exporter/
daemonset.apps/node-exporter created
service/node-exporter-service created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: alertmanager
  namespace: monitoring
data:
  config.yml: |-  

    global:  

      resolve_timeout: 1m  

  templates:  

  - '/etc/alertmanager-templates/*.tmpl'  

route:  

  receiver: 'email-notifications'  

  routes:  

  - match:  

    team: devops  

    receiver: 'email-notifications'  

    continue: true  

  - match:  

    team: dev  

    receiver: 'email-notifications'  

    continue: true  

receivers:  

- name: 'email-notifications'  

  email_configs:  

  - to: kunchalavikram@gmail.com  

  from: kunchalavikram@gmail.com  

  smarthost: smtp.gmail.com:587  

  auth_username: kunchalavikram@gmail.com  

  auth_identity: kunchalavikram@gmail.com  

  auth_password: xxxx  

  send_resolved: true
```



Kubernetes

Deploying Ingress rules for all dashboards

- We will deploy ingress rules to access Prometheus, Grafana and Alerts Manager dashboards via a DNS name.
- `kubectl apply -f dashboard_ingress_rules/`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl apply -f dashboard_ingress_rules/
ingress.networking.k8s.io/monitoring-ingress-rules created
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```

monitoring_dashboards_ingress_rules.yml

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: monitoring-ingress-rules
  namespace: monitoring
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
    - host: grafana-dashboard.com
      http:
        paths:
          - backend:
              serviceName: grafana
              servicePort: 3000
    - host: prometheus-dashboard.com
      http:
        paths:
          - backend:
              serviceName: prometheus-service
              servicePort: 9090
    - host: alerts-dashboard.com
      http:
        paths:
          - backend:
              serviceName: alertmanager
              servicePort: 9093
```



Kubernetes

Check all objects in Monitoring namespace

- `kubectl get all -n monitoring`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl get all -n monitoring
NAME                                         READY   STATUS    RESTARTS   AGE
pod/alertmanager-7f6d85c47b-mcfpf           1/1     Running   0          15m
pod/grafana-84d9874d6-hhbvd                 1/1     Running   0          22m
pod/kube-state-metrics-67b4c8d854-ccw9k      2/2     Running   0          30m
pod/node-exporter-45hkb                      1/1     Running   0          9m51s
pod/node-exporter-4lx56                       1/1     Running   0          9m51s
pod/node-exporter-76jh8                       1/1     Running   0          9m51s
pod/prometheus-deployment-5c5dfc7d66-9q5qz   1/1     Running   0          31m
pod/pushgateway-deployment-7df68cf778-jfqgw  1/1     Running   0          15m

NAME                TYPE        CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
service/alertmanager   ClusterIP   10.104.29.207   <none>           9093/TCP      18m
service/grafana        ClusterIP   10.109.3.26    <none>           3000/TCP      26m
service/kube-state-metrics   ClusterIP   10.102.103.247  <none>           8080/TCP,8081/TCP  9h
service/node-exporter-service  ClusterIP   10.109.200.201  <none>           9100/TCP      18m
service/prometheus-service   ClusterIP   10.103.218.51   <none>           9090/TCP      31m
service/pushgateway-service   ClusterIP   10.101.37.210   <none>           9091/TCP      18m

NAME            DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
daemonset.apps/node-exporter  3         3        3      3           3           <none>        18m

NAME            READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/alertmanager  1/1      1           1           18m
deployment.apps/grafana        1/1      1           1           26m
deployment.apps/kube-state-metrics  1/1      1           1           9h
deployment.apps/prometheus-deployment  1/1      1           1           31m
deployment.apps/pushgateway-deployment  1/1      1           1           18m

NAME            DESIRED  CURRENT  READY  AGE
replicaset.apps/alertmanager-7f6d85c47b  1         1           1           15m
replicaset.apps/grafana-84d9874d6       1         1           1           22m
replicaset.apps/kube-state-metrics-67b4c8d854  1         1           1           9h
replicaset.apps/prometheus-deployment-5c5dfc7d66  1         1           1           31m
replicaset.apps/pushgateway-deployment-7df68cf778  1         1           1           15m
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```



Kubernetes

Check all Ingress rules in all namespaces

- `kubectl get ing -A`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl get ing -A
NAMESPACE      NAME          CLASS      HOSTS                               ADDRESS      PORTS      AGE
default        application-ingress-rules <none>     connected-city.com,connected-factory.com
kube-system    kubernetes-web-ui       <none>     k8s-dashboard.com
kube-system    traefik-web-ui         <none>     traefik-dashboard.com
monitoring     monitoring-ingress-rules <none>     grafana-dashboard.com,prometheus-dashboard.com,alerts-dashboard.com
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# █
```



Kubernetes

Configure HA Proxy

- Provision a fresh VM to use it as a Reverse proxy
- Install HA Proxy for your distribution. For this tutorial, I'm using Ubuntu VM with APT package manager

```
apt install haproxy -y
```

- Stop the HAProxy service
`systemctl stop haproxy`

- Edit the `haproxy.conf` file at `/etc/haproxy/haproxy.cfg` and append the contents given in the `\end_to_end_demo\HA_proxy_configuration\haproxy_configuration.txt`.

- Modify the IP address of the Kubernetes nodes and save the file
- Restart the HAProxy service and enable it to start at boot

```
systemctl start haproxy
```

```
systemctl enable haproxy
```



Kubernetes

Configure HA Proxy

```
GNU nano 2.9.3          /etc/haproxy/haproxy.cfg          Modified

errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http

# Configure HAProxy to listen on port 80
frontend http_front
    bind *:80
    default_backend http_back

# Configure HAProxy to route requests to swarm nodes on port 8080
backend http_back
    balance roundrobin
    mode http
    server srv1 192.168.0.106:30001
    server srv2 192.168.0.105:30001
    server srv3 192.168.0.104:30001
```



In my case, IP of master node is 192.168.0.106 and slave 1 & slave 2 are 192.168.0.105 and 192.168.0.104 respectively.

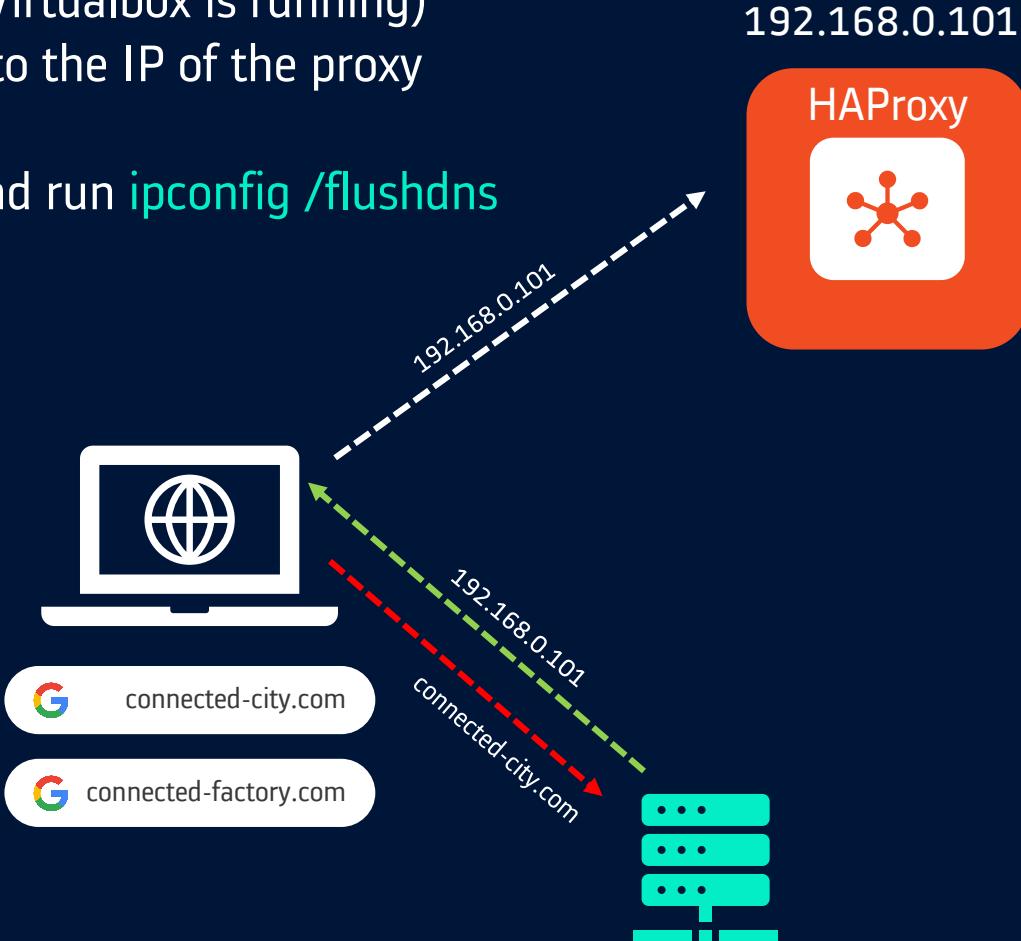


Kubernetes

Update dummy DNS entries

- Edit the hosts file in the host machine(windows machine where Virtualbox is running) with the DNS names configured in all the ingress rules mapping to the IP of the proxy server with HAProxy installed
- In windows, edit `C:\Windows\System32\drivers\etc\hosts` file and run `ipconfig /flushdns` in CMD to load the changes
- In linux, edit `/etc/hosts` file

```
192.168.0.101 connected-city.com
192.168.0.101 connected-factory.com
192.168.0.101 k8s-dashboard.com
192.168.0.101 traefik-dashboard.com
192.168.0.101 grafana-dashboard.com
192.168.0.101 prometheus-dashboard.com
192.168.0.101 alerts-dashboard.com
```

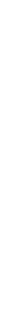


i In my case, 192.168.0.101 is the IP of the HAProxy server. Replace it with your IP and append these entries to the hosts file



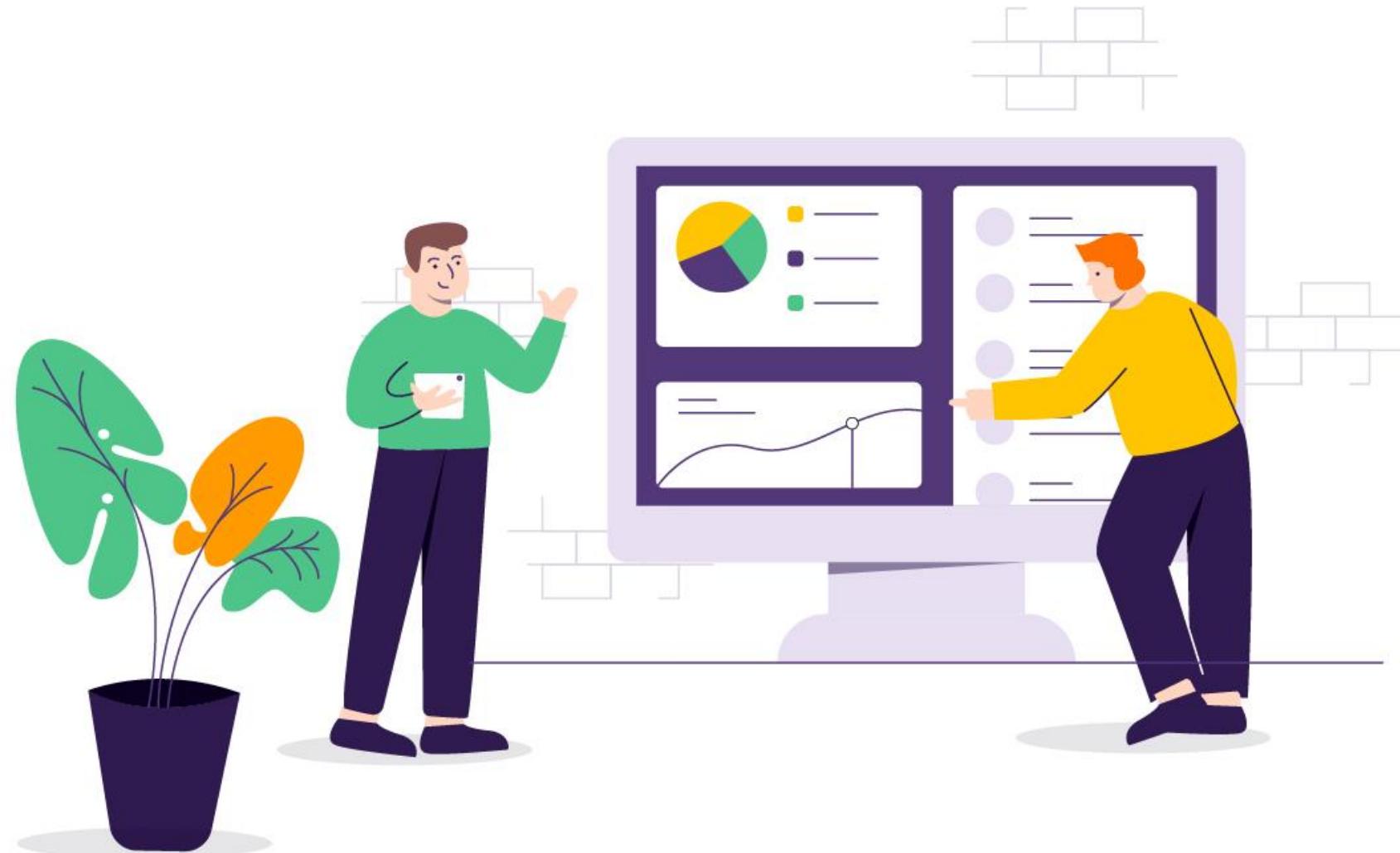
Kubernetes

Update dummy DNS entries



```
C:\Windows\System32\drivers\etc\hosts - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hosts
#1 # Added by Docker Desktop
#2 192.168.0.102 host.docker.internal
#3 192.168.0.102 gateway.docker.internal
#4 # To allow the same kube context to work on the host and the container:
#5 127.0.0.1 kubernetes.docker.internal
#6 # End of section
#7
#8
#9 192.168.0.101 connected-city.com
#10 192.168.0.101 connected-factory.com
#11 192.168.0.101 k8s-dashboard.com
#12 192.168.0.101 traefik-dashboard.com
#13 192.168.0.101 grafana-dashboard.com
#14 192.168.0.101 prometheus-dashboard.com
#15 192.168.0.101 alerts-dashboard.com
```

Accessing Dashboards

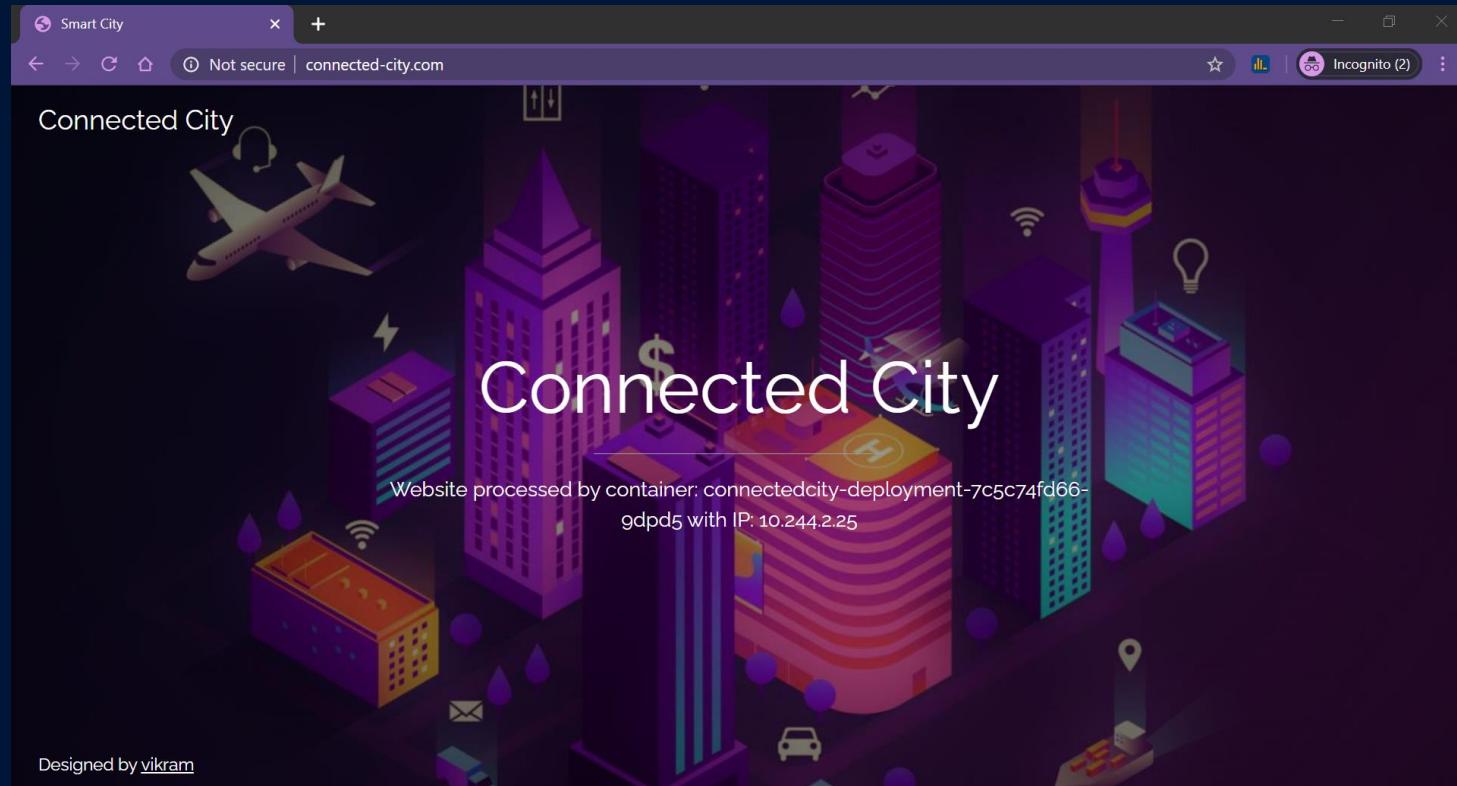
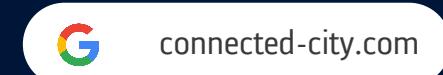




Kubernetes

Accessing Applications

App - 1



Since there are 3 instances of frontend applications running, the service load balances the requests.

 You can view this effect by refreshing the page and see the actual content changing like POD serving the request and its IP.

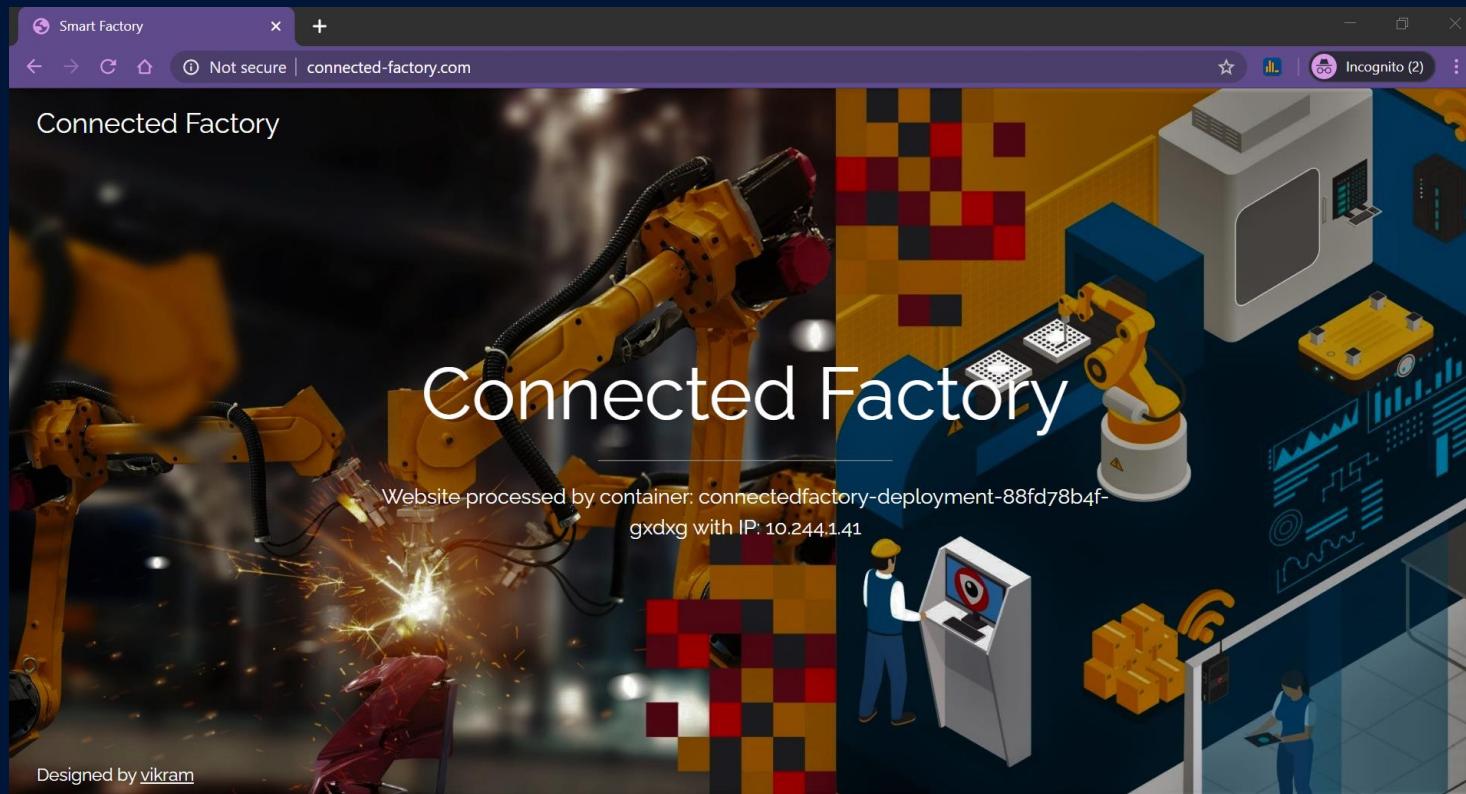
<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Applications

App -2



Since there are 3 instances of frontend applications running, the service load balances the requests.



You can view this effect by refreshing the page and see the actual content changing like POD serving the request and its IP.

<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Dashboards

Traefik dashboard

traefik-dashboard.com

The screenshot shows the Traefik dashboard interface. At the top, there are tabs for PROVIDERS and HEALTH, with V1.7.24 / MAROILLES and DOCUMENTATION links on the right. Below this, there are two main sections: FRONTENDS and BACKENDS.

FRONTENDS:

- alerts-dashboard.com**:
 - Main
 - Route Rule: Host:alerts-dashboard.com
 - Entry Points: http
 - Backend: alerts-dashboard.com
- connected-city.com**:
 - Main
 - Route Rule: Host:connected-city.com
 - Entry Points: http
 - Backend: connected-city.com

BACKENDS:

- alerts-dashboard.com**:
 - Main
 - Server: http://10.244.2.29:9093 Weight: 1
- connected-city.com**:
 - Main
 - Server: http://10.244.2.25:5000 Weight: 1
 - http://10.244.1.44:5000 Weight: 1
 - http://10.244.1.43:5000 Weight: 1
- connected-factory.com**:
 - Main
 - Server: http://10.244.2.25:5000 Weight: 1



Kubernetes

Accessing Dashboards

k8s dashboard

- For accessing k8s dashboard, we need a token that is generated during the deployment
 - `kubectl get secret`
 - `kubectl describe secret <k8s-dash-secret-name>`

```
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl get secret
NAME           TYPE      DATA  AGE
default-token-l6wnb  kubernetes.io/service-account-token  3    52d
k8dash-sa-token-h7rs8  kubernetes.io/service-account-token  3    8d
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo# kubectl describe secret k8dash-sa-token-h7rs8
Name:         k8dash-sa-token-h7rs8
Namespace:   default
Labels:      <none>
Annotations: kubernetes.io/service-account.name: k8dash-sa
              kubernetes.io/service-account.uid: f37ef39e-9706-4c92-9399-bae1ec1850ce
Type:        kubernetes.io/service-account-token

Data
=====
ca.crt:     1025 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6Ijd0MFMwcEdFMjJqbHM5VDlCY2VQanZNNfWT3VjM1VMVVZUQW9EUVFTVXMifQ.eyJpc3MiOiJrdWJlcmt5ldGVzL3NlcnZpY2VhY2NvdW50Iiwi3ViZXJuZXRlcyc5pbryzZXJ2aWNlyWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhWx0Iiwiia3ViZXJuZXRlcyc5pbryzZXJ2aWNlyWNjb3VudC9zZWNyZXQubmFtZSI6Ims4ZGFzaC1zYS10b2tlbi1oN3JzOCIsImt1YmVybmV0ZXMuaw8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LmbhWUiOiJr0GRhc2gtc2EiLCJrdWJlcmt5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOjmZndlZjM5ZS05NzA2LTRj0TIt0TM50S1YWXzWMx0DUwY2UiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdp0Rhc2gtc2Eif0.0dIdIcREFv49iF5W4xbcHNil6i0e_ajxRQm6TjQd2oS7rPHntu_7U0Fg53i1BLTz0ICkgxdyieuHB2Bqgxyb4600pBdfNzdqxVR8h0lBZ44XL6L-vJQWPfsHrqnvMlwGW42tvdm-ErnpkagNLcxfkZR0huhti8icGhsVNJNYtN23ew7Ult4uGPsi44BXZNWqS9gz76eJa7P6R3eTMv4JFhtghuF_X5vFKdzjfqbfQaboOIOCrfk1rjnjuWDdpS0Upx6kplIzV8QE4CAFcwAaPn7vkMU8EnVhzGMaCDI-fE3WNA_x3bUL-YoR5a4BytW1CKExi0BQWbjIxv6WpYQ
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```



Copy the token to use it at the login page

<https://github.com/kunchalavikram1427>

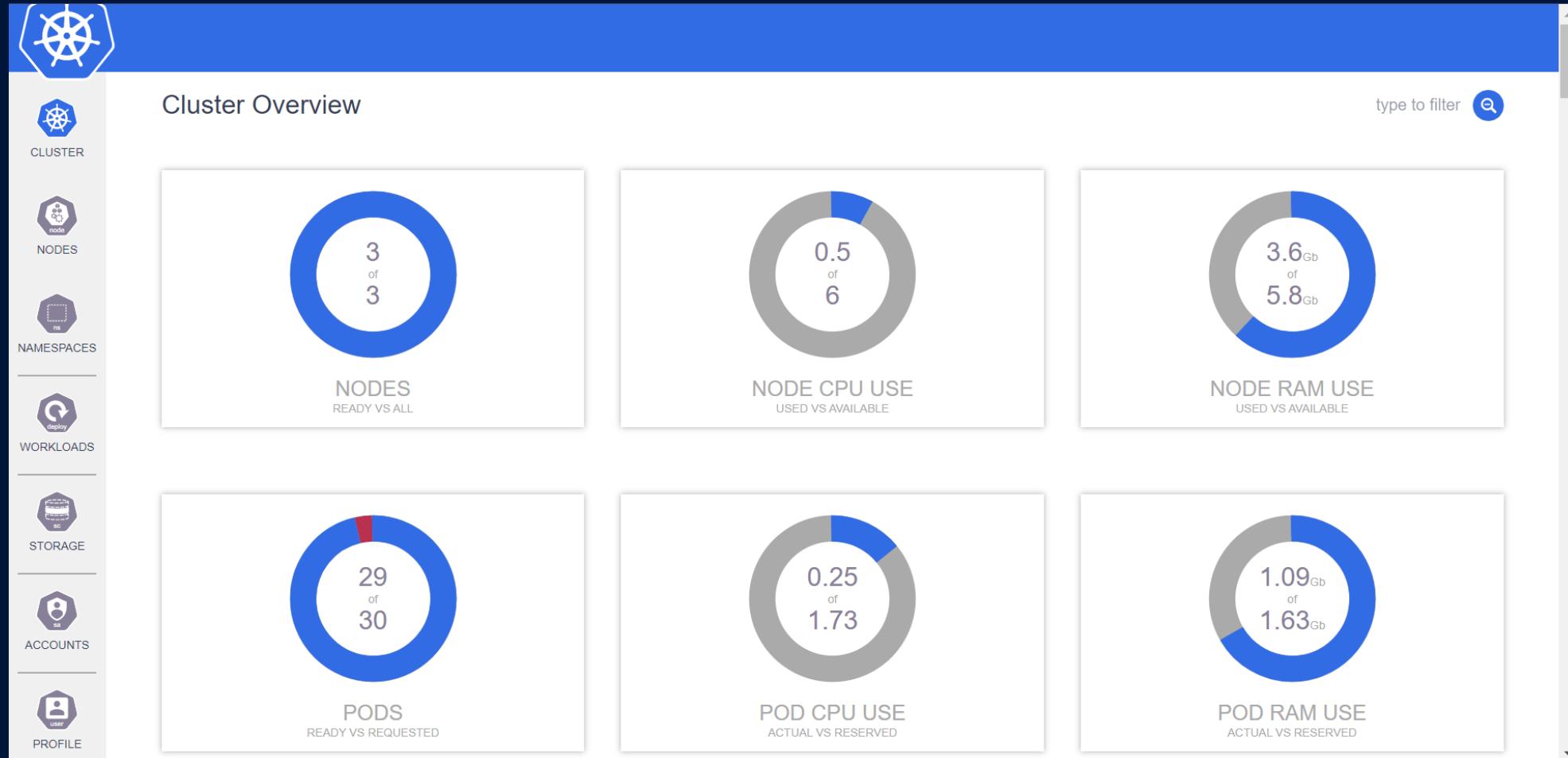


Kubernetes

Accessing Dashboards

k8s dashboard

 k8s-dashboard.com





Kubernetes

Accessing Dashboards

Prometheus dashboard



The screenshot shows the Prometheus Time Series Collector interface in a web browser. The title bar reads "Prometheus Time Series Collector". The address bar shows "Not secure | prometheus-dashboard.com/graph". The main menu includes "Prometheus", "Alerts", "Graph", "Status", and "Help". On the left, there is a checkbox for "Enable query history" and a text input field for "Expression (press Shift+Enter for newlines)". Below it are two buttons: "Execute" (highlighted in blue) and "- insert metric at cursor -". There are two tabs: "Graph" (selected) and "Console". Under the "Graph" tab, there is a "Moment" input field with arrows for navigation. A table below shows one entry: "Element" (no data) and "Value". At the bottom right of the graph area is a "Remove Graph" link. At the bottom left is a "Add Graph" button.



Kubernetes

Accessing Dashboards

Prometheus dashboard

prometheus-dashboard.com

The screenshot shows a web browser window with the title "Prometheus Time Series Collection". The address bar indicates the URL is "prometheus-dashboard.com/alerts" and the connection is "Not secure". The browser interface includes standard navigation buttons, a search bar, and an "Incognito (2)" tab.

The main content area is titled "Alerts". Below it, there are three buttons: "Inactive (10)", "Pending (0)", and "Firing (0)". A checkbox labeled "Show annotations" is present. The page displays two sections of alert rules:

- /etc/prometheus-rules/alert.rules > Deployment**
 - Deployment at 0 Replicas** (0 active)
 - HPA Scaling Limited** (0 active)
 - HPA at MaxCapacity** (0 active)
- /etc/prometheus-rules/alert.rules > Nodes**
 - High Node CPU Usage** (0 active)
 - High Node Disk Usage** (0 active)
 - High Node Memory Usage** (0 active)
 - InstanceDown** (0 active)



Kubernetes

Accessing Dashboards

Grafana dashboard

The screenshot shows a web browser window with the address bar containing "grafana-dashboard.com". The main content is the Grafana welcome screen. On the left, there's a sidebar with icons for Home, Dashboards, Data Sources, Metrics, Annotations, Events, and Help. The main area has a dark blue header with "Welcome to Grafana" and "Need help? Documentation Tutorials Community Public Slack". Below this, there are three main sections: "Basic" (with a sub-section "TUTORIAL DATA SOURCE AND DASHBOARDS Grafana fundamentals" about setting up and understanding Grafana), "COMPLETE" (with a sub-section "Add your first data source" and a "Learn how in the docs" link), and "DASHBOARDS" (with a sub-section "Create your first dashboard" and a "Learn how in the docs" link). At the bottom, there are sections for "Dashboards" (Starred dashboards) and "Latest from the blog" (an article titled "How to stream Graphite metrics to Grafana Cloud using carbon-relay" posted on Aug 03).



default credentials for Grafana are admin & admin

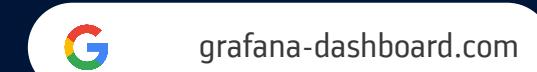
<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Dashboards

Importing Grafana dashboards



The screenshot shows the Grafana home page at grafana-dashboard.com/?orgId=1. A red box highlights the 'Create' dropdown menu in the top-left corner, which includes options for 'Dashboard', 'Folder', and 'Import'. The 'Import' option is selected. To the right of the 'Import' button, a tooltip reads: 'The steps below will guide you to quickly finish setting up your Grafana installation.' Below the 'Create' menu, there are three main sections: 'TUTORIAL' (Data Source and Dashboards, Grafana Fundamentals), 'COMPLETE' (Add your first data source), and 'DASHBOARDS' (Create your first dashboard). Each section has a 'Learn how in the docs' link.



Import Grafana dashboard templates from `end_to_end_demo\grafana_dashboards`

<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Dashboards

Importing Grafana dashboards

Import

Import dashboard from file or Grafana.com

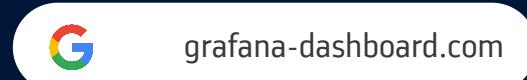
Upload JSON file

Import via grafana.com

Grafana.com dashboard url or id

Load

Import via panel json



end_to_end_demo > grafana_dashboards

Name	Status	Date modified	Type	Size
Kubernetes_Application_Metrics	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	31 KB
Kubernetes_Cluster_Monitoring	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	55 KB
Kubernetes_Deployment_Metrics	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	41 KB
Kubernetes_Node_Exporter_01	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	584 KB
Kubernetes_Node_Exporter_02	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	42 KB
Kubernetes_Pod_Metrics	✓ ⓘ	20-06-2020 09:55 AM	JSON Source File	82 KB



Kubernetes

Accessing Dashboards

Importing Grafana dashboards



The screenshot shows the Grafana interface for creating a new dashboard. On the left is a sidebar with various icons: a yellow circle with a red 'G', a magnifying glass, a plus sign, a grid, a circular arrow, a bell, a gear, a shield, a green hexagon with a white 'H', and a question mark. The main area has a dark background. At the top right is a search bar with the placeholder 'Import dashboard from file or Grafana.com'. Below it is a section titled 'Options' with fields for 'Name' (set to 'Kubernetes Nodes - 01 (Node Exporter)'), 'Folder' (set to 'General'), and 'Unique identifier (uid)' (containing 'Vq1Wj2_mz'). There is a blue button labeled 'Change uid'. Below these is a dropdown menu for 'DS_Prometheus' with the option 'Select a Prometheus data source' and 'prometheus' selected. At the bottom of the screen, there are links for 'Documentation', 'Support', 'Community', 'Open Source', and the version 'v7.1.1 (3039f9c3bd)'.



After importing a template, select Prometheus as the data source

<https://github.com/kunchalavikram1427>

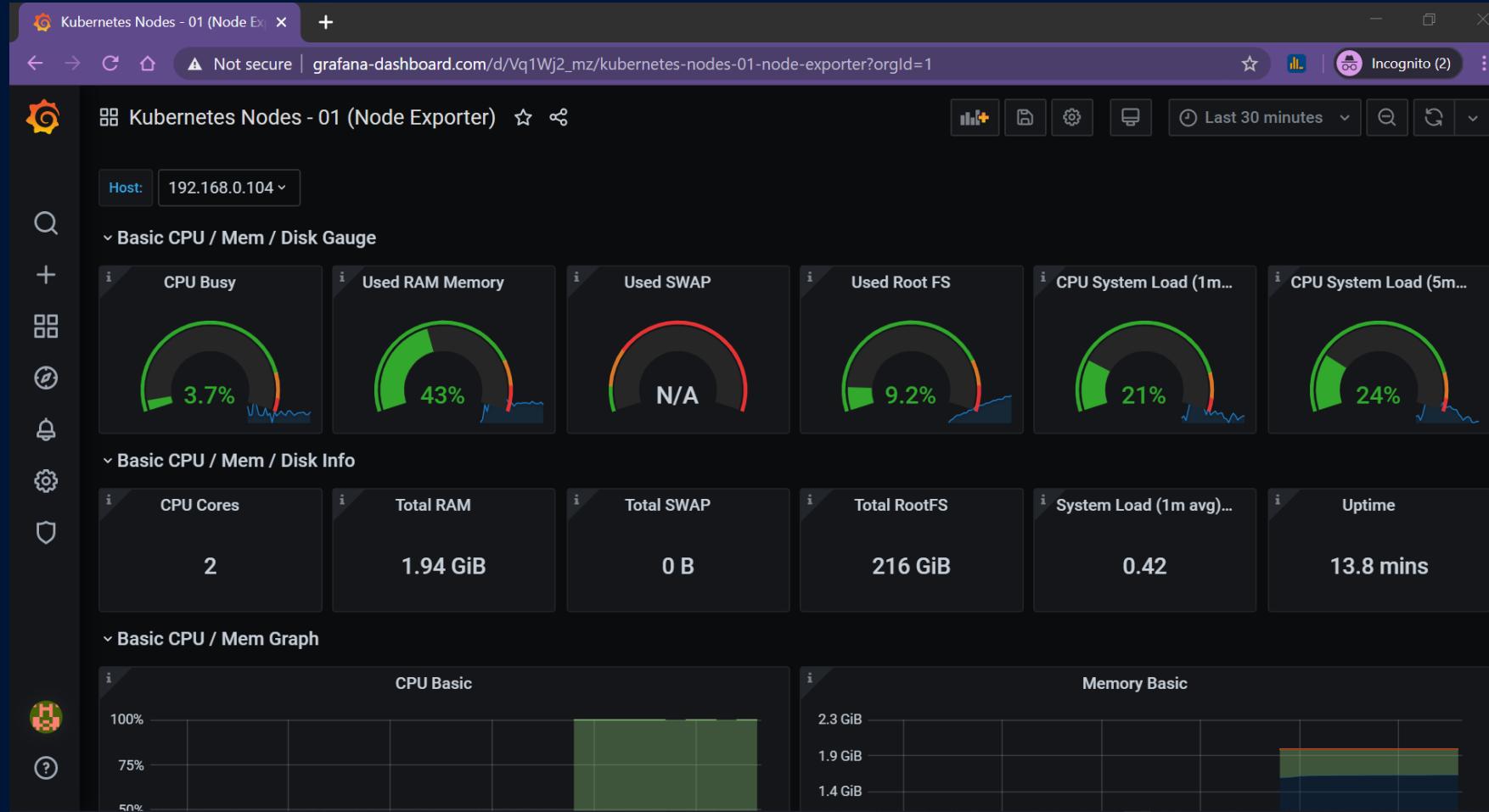


Kubernetes

Accessing Dashboards

Importing Grafana dashboards

grafana-dashboard.com





Kubernetes

Accessing Dashboards

Alert Manager dashboards



The screenshot shows the Alertmanager dashboard interface. At the top, there are tabs for Alertmanager, Alerts, Silences, Status, and Help, with 'Alerts' selected. A 'New Silence' button is located in the top right. Below the tabs is a search/filter section with 'Filter' and 'Group' tabs, and a 'Custom matcher, e.g. env="production"' input field. To the right of the filter are buttons for 'Receiver: All', 'Silenced', and 'Inhibited'. Under the search section, there's a '+ Silence' button. The main area displays a list of alerts. One alert is expanded to show its details:

05:17:21, 2020-08-04 (UTC) + Info ↗ Source ✘ Silence

alertname="InstanceDown" + instance="192.168.50.2:6443" + job="kubernetes-apiservers" + severity="critical" +

05:17:26, 2020-08-04 (UTC) + Info ↗ Source ✘ Silence

alertname="InstanceDown" + beta_kubernetes_io_arch="amd64" + beta_kubernetes_io_os="linux" + instance="k8s-master" + job="kubernetes-nodes-cadvisor" + kubernetes_io_arch="amd64" + kubernetes_io_hostname="k8s-master" + kubernetes_io_os="linux" +



For testing alerts, bring down Pushgateway deployments(`kubectl delete -f pushgateway/`) and verify if Gmail alerts are triggering

<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Dashboards

Gmail alert when a service is down

1 alert for

[View In AlertManager](#)

[1] Firing

Labels

alertrname = InstanceDown
instance = pushgateway-service:9091
job = pushgateway
severity = critical

Annotations

description = pushgateway-service:9091 of job pushgateway has been down for more than 1 minute.
summary = Prometheus target missing (instance pushgateway-service:9091)

[Source](#)



Identify the alert for Pushgateway service that we brought down for testing

<https://github.com/kunchalavikram1427>



Kubernetes

Accessing Dashboards

Gmail Alerts

4 alerts for

[View In AlertManager](#)

[3] Firing

Labels

```
alername = InstanceDown
beta_kubernetes_io_arch = amd64
beta_kubernetes_io_os = linux
instance = k8s-master
job = kubernetes-nodes-cadvisor
kubernetes_io_arch = amd64
kubernetes_io_hostname = k8s-master
kubernetes_io_os = linux
severity = critical
```

Annotations

```
description = k8s-master of job kubernetes-nodes-cadvisor has been down for more than 1 minute.
summary = Prometheus target missing (instance k8s-master)
```

[Source](#)

5 alerts for

[View In AlertManager](#)

[5] Resolved

Labels

```
alername = InstanceDown
instance = 192.168.50.2:6443
job = kubernetes-apiservers
severity = critical
```

Annotations

```
description = 192.168.50.2:6443 of job kubernetes-apiservers has been down for more than 1 minute.
summary = Prometheus target missing (instance 192.168.50.2:6443)
```

[Source](#)

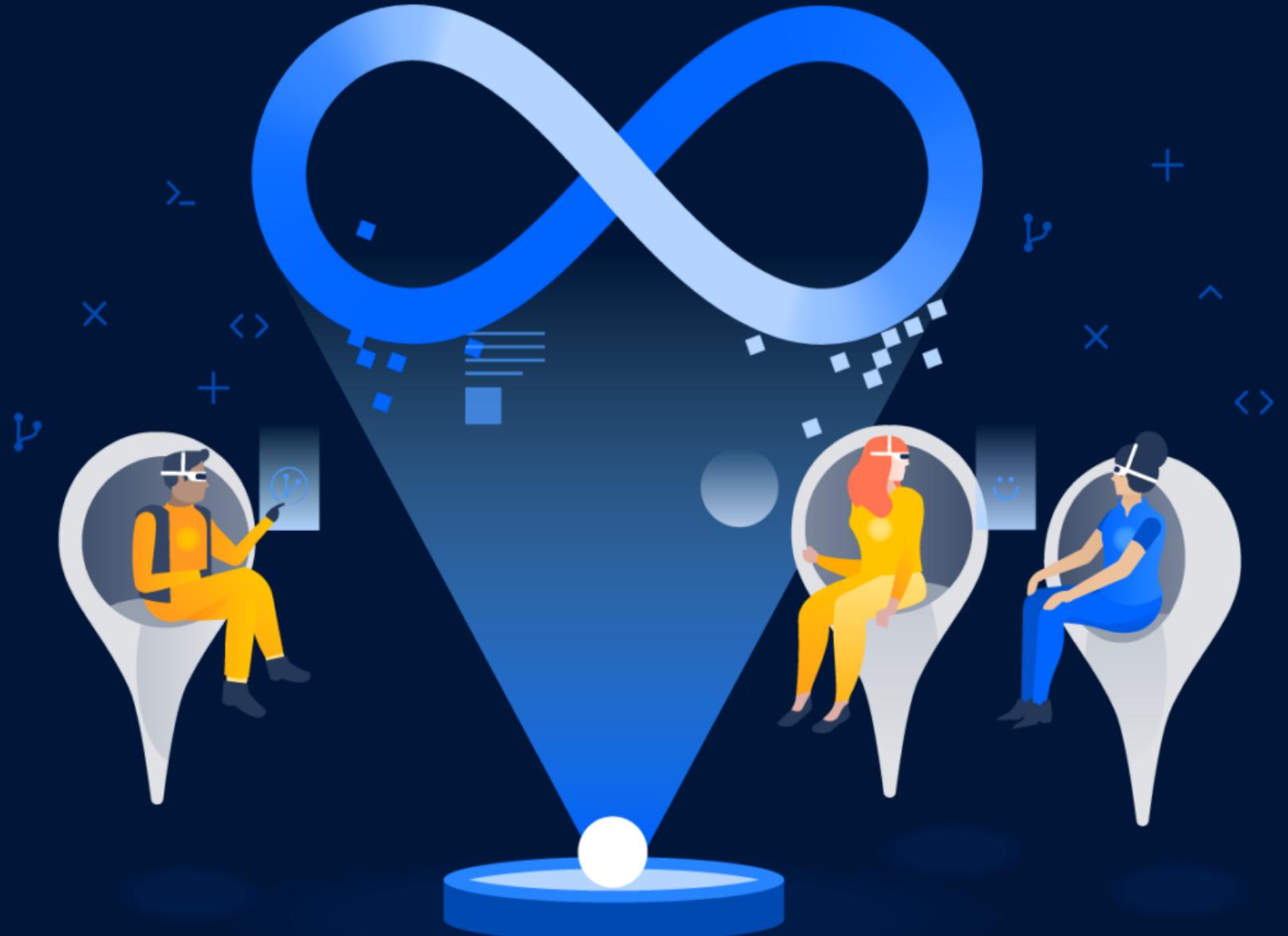
Labels

```
alername = InstanceDown
beta_kubernetes_io_arch = amd64
beta_kubernetes_io_os = linux
instance = k8s-master
```



References

- Kubernetes 101
 - <https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16>
 - https://jamesdefabia.github.io/docs/user-guide/kubectl/kubectl_run/
- Kubeadm
 - <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>
- Kubectl commands
 - <https://kubernetes.io/docs/reference/kubectl/cheatsheet/>
 - <https://kubernetes.io/docs/reference/kubectl/overview/>
- Deployments
 - <https://www.bmc.com/blogs/kubernetes-deployment/>
 - <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
- Services
 - <https://kubernetes.io/docs/concepts/services-networking/service/#headless-services>
 - <https://www.edureka.co/community/19351/clusterip-nodeport-loadbalancer-different-from-each-other>
 - <https://theithollow.com/2019/02/05/kubernetes-service-publishing/>
 - <https://www.ovh.com/blog/getting-external-traffic-into-kubernetes-clusterip-nodeport-loadbalancer-and-ingress/>
 - <https://medium.com/@lockDaRock/metalloadbalancer-kubernetes-on-prem-baremetal-loadbalancing-101455c3ed48>
 - <https://medium.com/@cashisclay/kubernetes-ingress-82aa960f658e>
- Ingress
 - <https://www.youtube.com/watch?v=QUfn0EDMmtY&list=PLVSHGLIFuAh89jomcWZnVhfYgvMmGIoIF&index=18&t=0s>
- K8s Dashboard
 - <https://github.com/kubernetes/dashboard>
 - <https://github.com/indeedeng/k8dash>
- YAML
 - <https://kubeyaml.com/>





If you like my work, Kindly consider supporting me.
This will help me come up with more tutorials on DevOps tools and help me in launching my own YouTube Channel.
Direct all your support to the below mentioned payment channels.
Your support is highly appreciated.

Check my Github for other tutorials on Docker, Kubernetes, K3s, Helm, Git, Ansible, Jenkins, Terraform, AWS etc.,

Subscribe to my [Facebook](#) page: <https://www.facebook.com/vikram.devops> and join my group: <https://www.facebook.com/groups/171043094400359> for all latest updates on DevOps



Need original presentation PPTs? Send an email to below address

kunchalavikram1427@gmail.com

+91-9885683322
 paypal.me/kunchalavikram