https://github.com/kunchalavikram1427

End to End Demo on

Kubernetes deployments, Ingress & Monitoring/alerting tools

Vikram
IoT Application Dev

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

https://github.com/kunchalavikram1427

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

connected-city.com

connected-factory.com

Kubernetes

Application Deployment and service manifest files

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

```yaml
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
```

Application-1
Deployment + ClusterIP service

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

```yaml
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
```

Application-2
Deployment + ClusterIP service

https://github.com/kunchalavikram1427

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#
```

https://github.com/kunchalavikram1427

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_cluste
rip.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#
```

https://github.com/kunchalavikram1427

Kubernetes

Deploying the Applications

- kubectl get all

  To show all the pods, replicasets, 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#
```

https://github.com/kunchalavikram1427

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

https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0
https://cloud.google.com/kubernetes-engine/docs/concepts/ingress

https://github.com/kunchalavikram1427

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

https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/

Kubernetes

Traefik Ingress Controller

https://github.com/kunchalavikram1427

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
                            connectedcity-service:80 (10.244.1.36:5000,10.244.1.39:5000,10.244.2.21:5000)
  connected-factory.com
                            connectedfactory-service:80 (10.244.1.37:5000,10.244.1.38:5000,10.244.2.20:5000)
Annotations:             kubernetes.io/ingress.class: traefik
Events:                  <none>
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/application_deployment#
```

```yaml
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

https://github.com/kunchalavikram1427

Metrics Server

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.

https://github.com/kunchalavikram1427

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

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

k8s_dashboard_ingress_rules.yml

```yaml
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
```

- kubectl get all -n kube-system | grep -i k8dash

```
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       8d
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/k8s_dashboard#
```

https://github.com/kunchalavikram1427

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_ingress_controller/
- kubectl apply -f traefik_ingress_controller.yml -f traefik_dashboard_ingress_rules.yml

traefik_dashboard_ingress_rules.yml

```yaml
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_ingress_controller/
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_ingress_controller# ls
traefik_dashboard_ingress_rules.yml  traefik_ingress_controller.yml
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo/traefik_ingress_controller# kubectl apply -f traefik_ingress_controller.yml -f traefik_dashboard_ingres
s_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_ingress_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.

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 will 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#
```

https://github.com/kunchalavikram1427

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#
```

```yaml
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
```

https://github.com/kunchalavikram1427

Kubernetes

Deploying Ingress rules for all dashboards

- We will deploy ingress rules to access Prometheus, Grafana and Alerts Manager dashboards vis 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

```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
```

https://github.com/kunchalavikram1427

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-hhbdv               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#
```

https://github.com/kunchalavikram1427

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                                         80      19h
kube-system   kubernetes-web-ui           <none>   k8s-dashboard.com                                                                80      8d
kube-system   traefik-web-ui              <none>   traefik-dashboard.com                                                            80      8d
monitoring    monitoring-ingress-rules    <none>   grafana-dashboard.com,prometheus-dashboard.com,alerts-dashboard.com              80      3m8s
root@k8s-master:/home/osboxes/Kubernetes_public/end_to_end_demo#
```

https://github.com/kunchalavikram1427

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 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
```

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

```
# Added by Docker Desktop
192.168.0.102 host.docker.internal
192.168.0.102 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section


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
```

Accessing Dashboards

Kubernetes

Accessing Applications

App - 1

connected-city.com

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

connected-factory.com

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

https://github.com/kunchalavikram1427

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:      eyJhbGciOiJSUzI1NiIsImtpZCI6Ijd0MFMwcEdFMjJqbHM5VDlCY2VQanZNNFlWT3VjM1VMVVZUQW9EUVFTVXMifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2Nv
dW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Ims4ZGFzaC1
zYS10b2tlbi1oN3JzOCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrOGRhc2gtc2EiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW
50L3NlcnZpY2UtYWNjb3VudC51aWQiOiJmMzdlZjM5ZS05NzA2LTRjOTItOTM5OS1iYWUxZWMxODUwY2UiLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDprOGRhc2gtc
2EifQ.OdIcREFv49iF5W4zxbcHNil6iOe_ajxRQm6TjQd2oS7rPHntu__7UOFg53i1BLTZ0ICkgxdyieuHB2Bqgxyb460OpBdfNzdqxVR8h0lBZ44XL6L-vJQWPfsHrqnvMWwGW42tvdm-E
rnpkagNlCxfkZROhuzti8icGhsVNJNYtN23ew7Ult4uGPs144BXZNWqS9gz76eJa7P6R3eTMv4JFHtghuF_X5vFKdzzjfqbLfQaboOIOCrfk1rjnjUWDdpS0Upx6kplIzV8QE4CAfcwAaPn
7vkMU8EnVHhzGMaCDI-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

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Prometheus dashboard

prometheus-dashboard.com

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Prometheus dashboard

prometheus-dashboard.com

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Grafana dashboard

grafana-dashboard.com

default credentials for Grafana are admin & admin

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Importing Grafana dashboards

grafana-dashboard.com

Import Grafana dashboard templates from end_to_end_demo\grafana_dashboards

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Importing Grafana dashboards

grafana-dashboard.com

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Importing Grafana dashboards

grafana-dashboard.com/

After importing a template, select Prometheus as the data source

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Importing Grafana dashboards

grafana-dashboard.com

https://github.com/kunchalavikram1427

Kubernetes

Accessing Dashboards

Alert Manager dashboards

alerts-dashboard.com

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

alertname = 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
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
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
alertname = 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
alertname = 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/@JockDaRock/metalloadbalancer-kubernetes-on-prem-baremetal-loadbalancing-101455c3ed48
  https://medium.com/@cashisclay/kubernetes-ingress-82aa960f658e
- Ingress
  https://www.youtube.com/watch?v=QUfn0EDMmtY&list=PLVSHGLlFuAh89j0mcWZnVhfYgvMmGI0lF&index=18&t=0s
- K8s Dashboard
  https://github.com/kubernetes/dashboard
  https://github.com/indeedeng/k8dash
- YAML
  https://kubeyaml.com/

Thank You