Health-Checks Report

Abstract:  
In this report we have documented how we configured heath-checks for some popular Kubernetes stateful addons. The stateful Kubernetes add-ons are as follows:

* HashiCorp Vault
* Fluentbit
* Apache Zookeeper

Environment:

The Environment we used was using a minikube cluster running on a **Ubuntu VM** inside **Oracle VM VirtualBox Manager.**

## Pre-Requisites:

Before moving on to setting up these add-ons to run in our cluster, we need to have the following

* A running Minikube cluster
* Helm must be configured and installed on our machine

## Installing Helm:

Helm is a package manager for Kubernetes. Since we are running a Ubuntu VM, we will be using apt to install helm. We used the following command

curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

sudo apt-get install apt-transport-https --yes

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

sudo apt-get update

sudo apt-get install helm

# Installing the add-ons to our Minikube Cluster

Now we installed our add-ons to our cluster. Now first we install the HashiCorp Consul OSS which will be where we register our health-checks and monitor them.

## HashiCorp Consul OSS

First make sure that there is no namespace named “consul”

Then run the following commands in the terminal

helm repo add hashicorp <https://helm.releases.hashicorp.com>

helm search repo hashicorp/consul

helm install consul hashicorp/consul --set global.name=consul --create-namespace --namespace consul

This will have the consul pods and service running on your cluster. Wait for them to be ready. You can check by

minikube kubectl get statefulsets

## Vault Installation to minikube via Helm with Integrated Storage:

To install the vault we will be first adding our hashicorp repository but since we have already added it before for the installation of consul oss. We can skip this. Then we will be creating a file named “helm-vault-raft-values.yaml”. Its content will be found below this paragraph. Then we will install the Vault using helm chart which will create 3 Instances of the vault. They will take some time to get up and running. After this we will be initializing our first instance of vault with one key share and one threshold. Now we will unseal this instance of vault and then join the rest of the two instance of this vault to the raft cluster as well as unseal these two vaults with the same key as well. The steps to do the above in terminal are as follows, but first we need to create the yaml file.

|  |
| --- |
| cat > helm-vault-raft-values.yml <<EOF  server:  affinity: ""  ha:  enabled: true  raft:  enabled: true  setNodeId: true  config: |  cluster\_name = "vault-integrated-storage"  storage "raft" {  path = "/vault/data/"  }  listener "tcp" {  address = "[::]:8200"  cluster\_address = "[::]:8201"  tls\_disable = "true"  }  service\_registration "kubernetes" {}  EOF |

helm install vault hashicorp/vault --values helm-vault-raft-values.yml

kubectl exec vault-0 -- vault operator init \

-key-shares=1 \

-key-threshold=1 \

-format=json > cluster-keys.json

VAULT\_UNSEAL\_KEY=$(jq -r ".unseal\_keys\_b64[]" cluster-keys.json)

kubectl exec vault-0 -- vault operator unseal $VAULT\_UNSEAL\_KEY

kubectl exec -ti vault-1 -- vault operator raft join <http://vault-0.vault-internal:8200>

kubectl exec -ti vault-2 -- vault operator raft join <http://vault-0.vault-internal:8200>

kubectl exec -ti vault-1 -- vault operator unseal $VAULT\_UNSEAL\_KEY

kubectl exec -ti vault-2 -- vault operator unseal $VAULT\_UNSEAL\_KEY

Optional:   
To setup secrets in the vault and then view these secrets in the browser through a web application deployment. Use the official Vault Documentation and look for **Set a secret in the Vault**

<https://developer.hashicorp.com/vault/tutorials/kubernetes/kubernetes-minikube-raft>

## Registering the Health Check for Vault on Consul

The Hashicorp vault has a built-in endpoint which can be used to check the status of the vault. It is the /sys/health endpoint for the Vault. This basically means that if we send an http get request at <http://(ip> of vault):[port at which vault is exposed] /sys/health we will receive back a status code telling us about the health of vault.  
Different status codes we may get are as follows

|  |  |
| --- | --- |
| 200 | if initialized, unsealed, and active |
| 429 | if unsealed and standby |
| 472 | if disaster recovery secondary (both active and standby nodes within the DR secondary) |
| 473 | if performance standby |
| 501 | if not initialized |
| 502 | if sealed |

In order to register this health check to the console service, we would need to add it to the console server config. This config can be edited through the configmap which was created for consul by the Helm chart. We edit the config map to look like this and then restart the consul server Statefulset. The configmap for consul should look like this. The important section is extra-from-values.json. The values such as uid and creationtimestamp can be different. This will register the healthcheck on the consul-server under the name vault and will use the vault service to send a get request to the health api endpoint at an interval of 50s with a timeout of 2 seconds. These values can be adjusted according to our own need. The 50s and 2s are arbitrary values

apiVersion: v1

data:

central-config.json: |-

{

"enable\_central\_service\_config": true

}

extra-from-values.json: '{ "services": [

{

"name": "vault",

"tags": ["api"],

"port": 8200,

"check": {

"id": "vault-api",

"name": "Vault API Health Check",

"http": "http://vault.default.svc.cluster.local:8200/v1/sys/health",

"tls\_skip\_verify": true,

"method": "GET",

"interval": "50s",

"timeout": "2s"

}

}

] }'

kind: ConfigMap

metadata:

annotations:

meta.helm.sh/release-name: consul

meta.helm.sh/release-namespace: default

creationTimestamp: "2024-07-23T05:40:18Z"

labels:

app: consul

app.kubernetes.io/managed-by: Helm

chart: consul-helm

component: server

heritage: Helm

release: consul

name: consul-server-config

namespace: default

resourceVersion: "75816"

uid: e442fed7-df6e-400d-9bba-adc9c787d0b7

Now we will restart the consul server

kubectl rollout restart statefulset consul

Here we have completed the healthcheck for our first stateful-addon which was the hashicorp vault. To check the health-check at the consul ui, skip to the Monitoring the Health checks through the consul-ui

# Fluentbit

The process of installing and configuring the healthcheck for this addon is also similar to vault. First we install fluentbit using Helm charts.

helm repo add fluent <https://fluent.github.io/helm-charts>

helm upgrade --install fluent-bit fluent/fluent-bit

This will launch fluentbit as a daemonset on minikube and also launch its service which we can see by using kubectl get svc command

The api endpoint to check the fluentbit health is similar to the vault. It is /api/v1/health. It uses the equation

Health status = (HC\_Errors\_Count > 5) OR (HC\_Retry\_Failure\_Count > 5) IN 5 seconds

If (HC\_Errors\_Count > 5) OR (HC\_Retry\_Failure\_Count > 5) IN 5 seconds is TRUE, then it's unhealthy.

If (HC\_Errors\_Count > 5) OR (HC\_Retry\_Failure\_Count > 5) IN 5 seconds is FALSE, then it's healthy.

And the status code are as follows:

* HTTP status 200 and "ok" in response to healthy status
* HTTP status 500 and "error" in response for unhealthy status

## Registering the Health-check:

Just as we registered the Vault health-check, we need to add the health-check to consul server config in addition to the already existing previous vault health-check. We can add this health check by adding a comma (,) at the end of previous one and then editing this in our configmap of the consul-server

{

"name": "fluent-bit",

"tags": ["logging"],

"port": 2020,

"check": {

"id": "fluent-bit-health",

"name": "Fluent Bit Health Check",

"http": "http://fluent-bit.default.svc.cluster.local:2020/api/v1/health",

"method": "GET",

"interval": "30s",

"timeout": "5s"

}

}

]

Followed by

kubectl rollout restart statefulset consul

This will have created and registered our health-check at the consul-server-ui for fluentbit addon.

# Apache-Zookeeper

Apache-zookeeper is another addon that we will be deploying. We did not use helm charts for this but rather standard yaml files for the statefulset and its service and for the health-check, we implement a rather simpler health check than the previous two which test the TCP connect at the port 2181 every 10 seconds with a timeout set for 1 second.

As previously, we can add the health check in addition to the already existing previous health checks

{

"name": "zookeeper",

"tags": ["datastore"],

"port": 2181,

"check": {

"id": "zookeeper-health",

"name": "ZooKeeper TCP Health Check",

"tcp": "zookeeper-service.default.svc.cluster.local:2181",

"interval": "10s",

"timeout": "1s"

}

}

# Monitoring the Health checks through the consul-ui

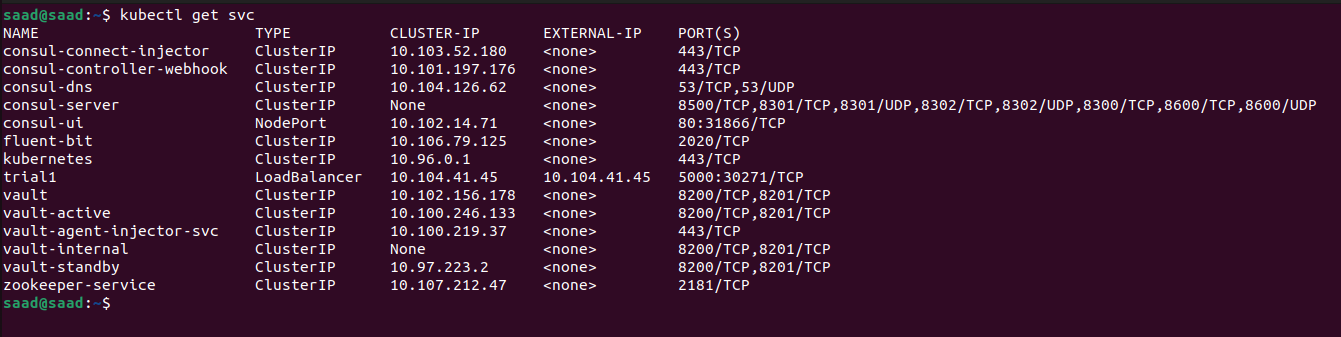
First we need to find the ip of our minikube cluster which can be found by

minikube ip e.g. 192.168.4.2

Note this Ip

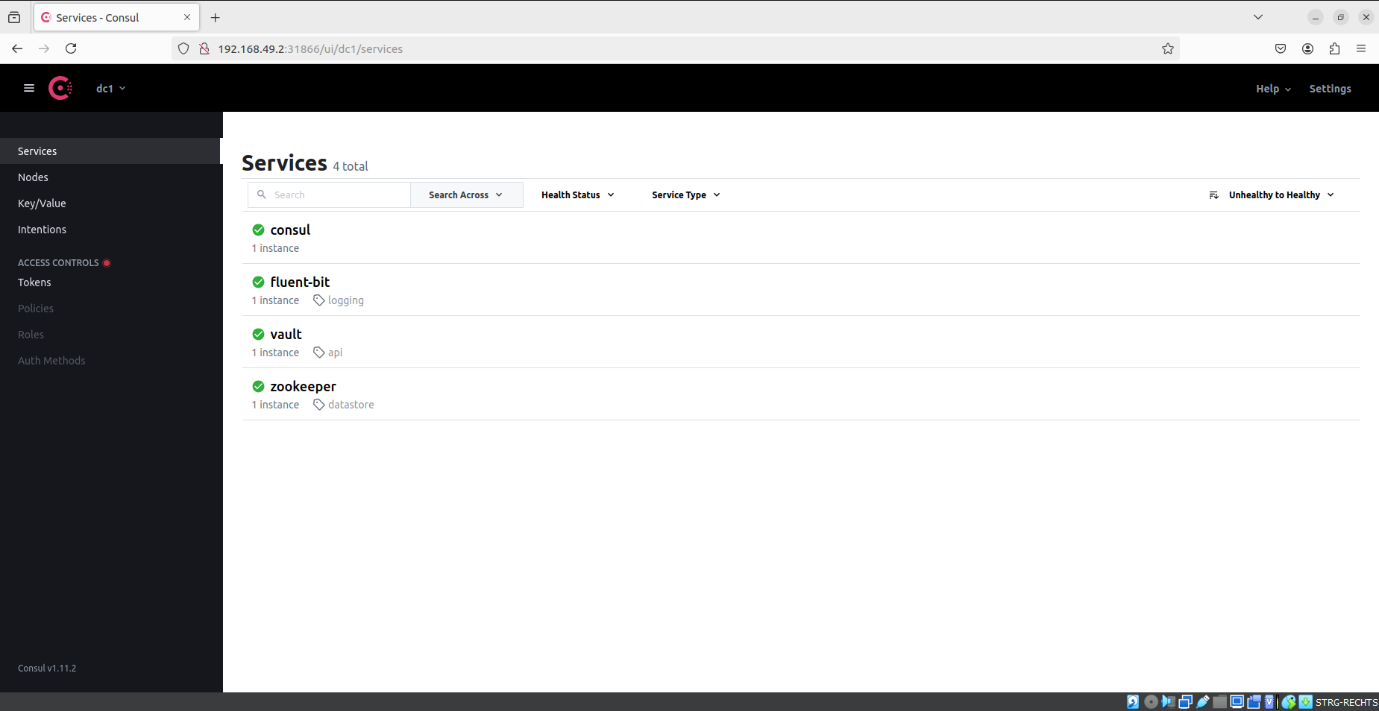
Now do

kubectl get svc



Find the consul service of type Nodeport and note down its port e.g. 31182

Now run your browser and go to   
<http://192.168.4.2:31882>  
  
The consul-ui will open and show you the state of healthchecks as follows



As it can be seen that all the health checks we created and registered to the consul server are now available at the consul ui

References:

Following references were used for implementing these

<https://developer.hashicorp.com/vault/tutorials/kubernetes/kubernetes-raft-deployment-guide>

[Kubernetes | Fluent Bit: Official Manual](https://docs.fluentbit.io/manual/installation/kubernetes)

[Consul on Kubernetes | Consul | HashiCorp Developer](https://developer.hashicorp.com/consul/docs/k8s)

Some other addons for which we aim to apply are

* SQL (with Patroni)
* MongoDB
* Redis