**Configure minikube on centos 7 VM for kompose, docker stack and docker-compose tools**

**The purpose of this guide is to install the necessary tools to deploy a composed web application (such as: MessageBoardApp at** [**https://github.com/monicamarshall/MessageBoardApp**](https://github.com/monicamarshall/MessageBoardApp)**) to a kubernetes single node cluster, in this case minikube, running in a Centos 7 VM.**

**Deployments to a kubernetes cluster are accomplished in 3 different ways:**

1. **Manual deployment with ad-hoc, manually created kubernetes manifest yaml files using docker container images previously created with the docker-compose tool.**
2. **Automatic deployment with the kompose tool using the docker-compose file.**
3. **Automatic deployment with the docker stack command using the docker-compose file.**

**Description of tools:**

1. **kompose: used to automatically deploy composed applications in docker containers to minikube. The kompose tool can directly deploy to the kubernetes cluster or it can generate a set of deployment files that can be applied for deploying applications to the kubernetes cluster. Some modifications to the generated yaml files must be applied.**
2. **docker stack: used to automatically deploy composed applications in docker containers to minikube. With docker stack no kubernetes deployment files are generated.**
3. **docker-compose: used in the minikube environment to create docker images for composed applications. To be able to deploy ad-hoc, or automatically with kompose or docker stack, you must create docker images first with docker-compose. If the application does not need to compose containers, simply use the docker build command to create the docker image.**

**Docker-compose is necessary to create the container images for a composed application (ex: MessageBoardApp). These container images can also be deployed using ad-hoc kubernetes manifest yaml files instead of relying on an automatic deployment using kompose or docker stack. Automatic deployments with kompose or docker stack come in very handy in development mode.**

**None of these tools, docker-compose, docker stack, kompose, are available in an initial installation of minikube.**

**Prerequisites:**

1. **You have a working minikube environment and you can access the kubernetes dashboard.**
2. **You have downloaded the MessageBoardApp composed application source code at** [**https://github.com/monicamarshall/MessageBoardApp**](https://github.com/monicamarshall/MessageBoardApp)**. Composed applications require multiple containers networked together. An example is the MessageBoardApp requiring a container for postgres, a container for the Django/Python webapp, and one for the nginx webserver serving the MessageBoardApp webpages.**

**Deployment of the MessageBoardApp in minikube: requirements and knowledge sharing**

**One of the requirements for installing minikube on the centos 7 VM is the installation of the docker engine. For docker to be available on the host VM, (centos7 VM), the docker daemon must be started (run sudo systemctl start docker command) and also for us to start minikube. This docker daemon we will call the host docker daemon. Minikube runs in its own virtual machine and by using the docker driver we are using the host docker daemon as the minikube’s virtual machine manager.**

**Minikube has its own docker daemon and its own registry for registering images built locally inside minikube. Minikube’s docker daemon is different than the host docker daemon.**

**To be able to run docker images in minikube’s kubernetes environment we have to build these images locally (local to minikube) using the minikube’s docker daemon. Images that are built with the host docker daemon are not visible by the docker daemon running inside minikube. Therefore, they cannot be deployed to minikube’s kubernetes cluster environment. The opposite is also true: images built with the minikube’s docker daemon are not visible by the host docker daemon.**

**Before creating images locally to minikube and visible by minikube’s kube controller (kubectl) we have to switch to minikube’s docker environment with the command:**

**eval $(minikube docker-env)**

**To switch back to the host’s docker daemon issue the command:**

**eval $(minikube docker-env -u)**

**where –u stands for “undo”.**

**After switching to minikube’s docker daemon, we will have to install compose to build the images necessary to run the MessageBoardApp in minikube’s kubernetes environment. The compose tool allows us to run the command at the top messageboardapp directory where docker-compose.yaml is located and run the command:**

**docker-compose build**

**Which creates 2 docker images:**

messageboardapp\_nginx and

messageboardapp\_web.

**Install the compose tool for minikube to build docker images with docker-compose and run them with docker stack or the kompose tool or with manually created kubernetes manifest yaml files (.k8s-final)**

Reference page: <https://github.com/docker/compose-on-kubernetes/blob/master/docs/install-on-minikube.md>

NOTE: some instructions at this link are modified to reflect latest changes.

1. **INSTALL HELM**

Find the latest stable release at:

<https://github.com/helm/helm/releases>

Download the latest release [Helm v3.4.0](https://github.com/helm/helm/releases/tag/v3.4.0)

**tar -zxvf helm-v3.4.0-linux-amd64.tar.gz**

**mv linux-amd64/helm /usr/local/bin/helm**

**Verify helm version**

**helm version**

Once you have Helm ready, you can add a chart repository. One popular starting location is the official Helm stable charts:

**helm repo add stable** [**https://charts.helm.sh/stable**](https://charts.helm.sh/stable)

1. **INSTALL etcd**

**Reference page:** [**https://github.com/docker/compose-on-kubernetes/blob/master/docs/deploy-etcd.md**](https://github.com/docker/compose-on-kubernetes/blob/master/docs/deploy-etcd.md)

**minikube start**

**kubectl create namespace compose**

compose on Kubernetes requires an etcd instance (in addition to the kube-system etcd instance).

add the repository where the etcd-operator is stored

**helm repo add stable** [**https://charts.helm.sh/stable**](https://charts.helm.sh/stable)

install the etcd-operator chart.

**helm install etcd-operator --set customResources.createEtcdClusterCRD=true stable/etcd-operator --namespace compose**

check that etcd-operator containers were created and are in running state

**kubectl get pods --namespace compose**

**you should have:**

etcd-operator-etcd-operator-etcd-backup-operator

etcd-operator-etcd-operator-etcd-operator

etcd-operator-etcd-operator-etcd-restore-operator

**Install** etcd-cluster.yaml with command:

**kubectl apply –f etcd-cluster.yaml**

Content of etcd-cluster.yaml:

apiVersion: "etcd.database.coreos.com/v1beta2"

kind: "EtcdCluster"

metadata:

name: "compose-etcd"

namespace: "compose"

spec:

size: 3

version: "3.3.15"

pod:

affinity:

podAntiAffinity:

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 100

podAffinityTerm:

labelSelector:

matchExpressions:

- key: etcd\_cluster

operator: In

values:

- compose-etcd

topologyKey: kubernetes.io/hostname

check that the etcd-cluster containers are in running state

**kubectl get pods --namespace compose**

**You should have 3 etcd-cluster in running state:**

NAME READY STATUS RESTARTS AGE

compose-etcd-5gk95j4ms6 1/1 Running 0 21m

compose-etcd-nqmcwk4gdf 1/1 Running 0 21m

compose-etcd-sxplrdthp6 1/1 Running 0 20m

**minikube stop**

**minikube start**

1. **INSTALL docker-compose**

**Download compose at this page: The name of the file to be downloaded is:** [installer-linux](https://github.com/docker/compose-on-kubernetes/releases/download/v0.4.25-alpha1/installer-linux)

**View the releases at:**

[**https://github.com/docker/compose-on-kubernetes/releases**](https://github.com/docker/compose-on-kubernetes/releases)

**chmod +x installer-linux**

**Run the installer-linux:**

**./installer-linux -namespace=compose -etcd-servers=http://compose-etcd-client:2379**

**Verify installation:**

**kubectl api-versions | grep compose**

**minikube stop**

**minikube start**

**CREATE DOCKER IMAGES FOR THE MESSAGEBOARDAPP**

**Change directory to the main directory of the messageboardapp. Make sure you have your docker-compose.yaml file, your Dockerfile, and your .env file. Run the command:**

**docker-compose build**

**Verify that you have built messageboardapp\_web and messageboardapp\_nginx as docker images with command**

**docker images**

**you should see both images in the minikube docker registry:**

**[monica@localhost .kompose]$ docker images**

**REPOSITORY TAG IMAGE ID CREATED SIZE**

**messageboardapp\_nginx latest 460bd400d29a 32 hours ago 21.3MB**

**messageboardapp\_web latest 6d72b79d6514 32 hours ago 986MB**

**DEPLOY MESSAGEBOARDAPP TO KUBERNETES WITH DOCKER STACK**

**Use the docker-compose file that is located in the .docker-stack folder. Note that no kubernetes yaml deployment files are generated.**

**To deploy messageboardapp to kubernetes with docker stack, run:**

**docker stack deploy --orchestrator=kubernetes -c docker-compose.yml messageboardapp-stack**

**To undeploy messageboardapp run this command:**

**docker stack remove --orchestrator=kubernetes messageboardapp-stack**

**Map the minikube ip to a dns name:**

**Edit the hosts file:**

sudo vim /etc/hosts

**Find the ip of minikube:**

**minikube ip**

**Add a line at the end of /etc/host:**

**192.168.49.2 messageboard.com (192.168.49.2 is the minikube ip on my centos7 VM)**

**To view the main page retrieve the NodePort:**

**kubectl get svc**

**NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE**

**db ClusterIP 10.102.65.224 <none> 5432/TCP 7h38m**

**kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d1h**

**nginx NodePort 10.107.61.128 <none> 1338:31883/TCP 7h38m**

**web NodePort 10.104.78.119 <none> 8000:30546/TCP 7h38m**

**Inside minikube dashboard, shell into the web pod (kubectl get pods) and run the 3 python commands:**

**Python manage.py collectstatic**

**Python manage.py –run-syncdb**

**Python manage.py createsuperuser**

**View the main page in a browser at the web-published NodePort:**

[**http://messageboard.com:30546**](http://messageboard.com:30546)

**DEPLOY MESSAGEBOARDAPP TO KUBERNETES WITH kompose**

**Download kompose**

**curl -L https://github.com/kubernetes/kompose/releases/download/v1.21.0/kompose-linux-amd64 -o kompose**

**chmod +x kompose**

**sudo mv ./kompose /usr/local/bin/kompose**

**Change directory for the messageboardapp where docker-compose.yaml is located**

**mkdir .kompose**

**Generate a set of deployable kubernetes manifest files for the messageboardapp. Run this command at the top of the messageboardapp directory. Make sure to use the appropriate docker-compose.yaml file. Copy the one located in the .docker-stack directory.**

**Run the command:**

**kompose convert -o .kompose (-o stands for output. This means that kompose will generate the manifest files in the .kompose folder)**

**Before deployment apply a couple changes to some of the generated yaml files:**

**[monica@localhost .kompose]$ ll**

**total 36**

**-rw-r--r-- 1 monica monica 2048 Nov 10 14:10 db-deployment.yaml**

**-rw-r--r-- 1 monica monica 347 Nov 10 14:10 db-service.yaml**

**-rw-r--r-- 1 monica monica 351 Nov 10 14:10 env-configmap.yaml**

**-rw-r--r-- 1 monica monica 1003 Nov 10 14:43 nginx-deployment.yaml**

**-rw-r--r-- 1 monica monica 371 Nov 10 14:40 nginx-service.yaml**

**-rw-r--r-- 1 monica monica 247 Nov 10 14:10 postgres-data-persistentvolumeclaim.yaml**

**-rw-r--r-- 1 monica monica 247 Nov 10 14:10 static-volume-persistentvolumeclaim.yaml**

**-rw-r--r-- 1 monica monica 2162 Nov 10 14:43 web-deployment.yaml**

**-rw-r--r-- 1 monica monica 367 Nov 10 14:41 web-service.yaml**

**In nginx-deployment.yaml and web-deployment.yaml add “IfNotPresent” for the ImagePullPolicy.**

**imagePullPolicy: "IfNotPresent"**

**in web-service.yaml and nginx-service.yaml add the line:**

**type: NodePort**

**web-service.yaml**

**apiVersion: v1**

**kind: Service**

**metadata:**

**annotations:**

**kompose.cmd: kompose convert -o .kompose**

**kompose.version: 1.21.0 (992df58d8)**

**creationTimestamp: null**

**labels:**

**io.kompose.service: web**

**name: web**

**spec:**

**ports:**

**- name: "8000"**

**port: 8000**

**targetPort: 8000**

**selector:**

**io.kompose.service: web**

**type: NodePort**

**status:**

**loadBalancer: {}**

**nginx-service.yaml**

**apiVersion: v1**

**kind: Service**

**metadata:**

**annotations:**

**kompose.cmd: kompose convert -o .kompose**

**kompose.version: 1.21.0 (992df58d8)**

**creationTimestamp: null**

**labels:**

**io.kompose.service: nginx**

**name: nginx**

**spec:**

**ports:**

**- name: "1338"**

**port: 1338**

**targetPort: 80**

**selector:**

**io.kompose.service: nginx**

**type: NodePort**

**status:**

**loadBalancer: {}**

**This will allow the container port to be accessed from outside the kubernetes cluster, for example by a web browser.**

**Apply the manifest files that kompose created with command. This command will deploy messageboardapp to kubernetes.**

**kubectl apply –f .kompose**

**To undeploy messageboardapp run the command:**

**kubectl delete –f .kompose**

**Verify deployment in the minikube dashboard.**

**To view the main page retrieve the NodePort:**

**kubectl get svc**

**NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE**

**db ClusterIP 10.102.65.224 <none> 5432/TCP 7h38m**

**kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d1h**

**nginx NodePort 10.107.61.128 <none> 1338:31883/TCP 7h38m**

**web NodePort 10.104.78.119 <none> 8000:30546/TCP 7h38m**

**Inside minikube dashboard, shell into the web pod (kubectl get pods) and run the 3 python commands:**

**Python manage.py collectstatic**

**Python manage.py –run-syncdb**

**Python manage.py createsuperuser**

**View the messageboardapp main page in a browser at the web-published NodePort:**

[**http://messageboard.com:30546**](http://messageboard.com:30546)

**MANUAL DEPLOYMENT OF MESSAGEBOARDAPP VIA KUBERNETES MANIFEST FILES**

**To deploy messageboardapp to kubernetes run the command:**

**kubectl apply –f .k8s-final**

**To undeploy messageboardapp run the command:**

**kubectl delete –f .k8s-final**

**Verify deployment in the minikube dashboard.**

**Inside minikube dashboard, shell into the web pod (kubectl get pods) and run the 3 python commands:**

**Python manage.py collectstatic**

**Python manage.py –run-syncdb**

**Python manage.py createsuperuser**

**Access messageboard app in a browser using the same method listed for docker-stack and kompose.**

**To view the main page retrieve the NodePort:**

**kubectl get svc**

**NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE**

**db ClusterIP 10.102.65.224 <none> 5432/TCP 7h38m**

**kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 2d1h**

**nginx NodePort 10.107.61.128 <none> 1338:31883/TCP 7h38m**

**web NodePort 10.104.78.119 <none> 8000:30546/TCP 7h38m**

**View the messageboardapp main page in a browser at the web-published NodePort:**

[**http://messageboard.com:30546**](http://messageboard.com:30546)