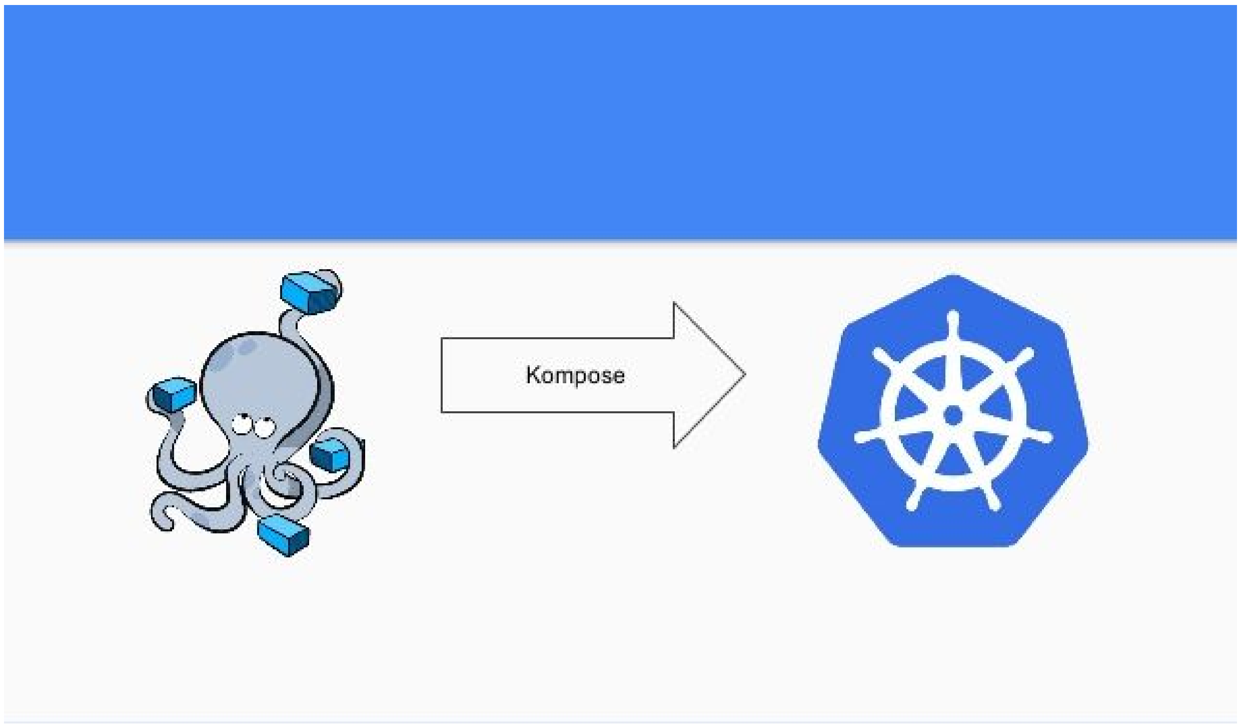
Migrating a docker-compose 3 tier application stack to kubernetes.

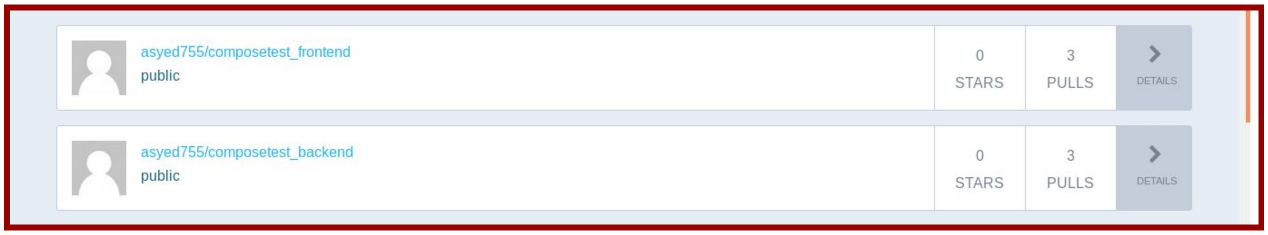
kompose is a tool to help users who are familiar with docker-compose move to Kubernetes.

kompose takes a Docker Compose file and translates it into Kubernetes resources.
kompose is a convenience tool to go from local Docker development to managing your application with Kubernetes. Transformation of the Docker Compose format to Kubernetes resources manifest may not be exact, but it helps tremendously when first deploying an application on Kubernetes.



1. ***git clone*** [*https://github.com/LovesCloud/Docker-compose-demo.git*](https://github.com/LovesCloud/Docker-compose-demo.git)
2. ***cd Docker-compose-demo/***
3. ***vim docker-compose.yaml*** *(file - https://pastebin.com/raw/0etUFJX0)*

Replace $DOCKER_ID_USER/<tag_name> in the docker-compose.yaml file with the tag and username you created when uploading the docker images to docker hub.



asyed755/composetest_frontend
public
0 STARS 3 PULLS DETAILS

asyed755/composetest_backend
public
0 STARS 3 PULLS DETAILS

4. Run the below command to convert the docker-compose yaml and deploy the stack on the kubernetes cluster.
   ```
   $ kompose convert -f docker-compose.yaml
   ```

5. **The compose will convert the docker-compose.yaml to compatible deployment objects that can be then deployed as services and deployments on the cluster.**

   ```
   $ ls

   INFO Kubernetes file "backend-service.yaml" created
   INFO Kubernetes file "frontend-service.yaml" created
   INFO Kubernetes file "redis-service.yaml" created
   INFO Kubernetes file "backend-deployment.yaml" created
   INFO Kubernetes file "frontend-deployment.yaml" created
   INFO Kubernetes file "redis-deployment.yaml" created
   ```

7. Now run run the below commands to deploy the stack on the K8s cluster.

   ```
   $ kubectl apply -f redis-deployment.yaml
   $ kubectl apply -f redis-service.yaml
   $ kubectl apply -f frontend-deployment.yaml
   $ kubectl apply -f frontend-service.yaml
   $ kubectl apply -f backend-deployment.yaml
   $ kubectl apply -f service-service.yaml
   ```

   Run the below command to check the deployment details
   ```
   $ kubectl get deployment
   ```
   Run the below command to check the services details
   ```
   $ kubectl get svc
   ```

8. **Login to the Kubernetes Dashboard** and check the deployment details under the deployment section.

   You can see that the stack has been deployed on the K8s cluster. One more step before we can access the application. You can observe under services that the frontend application has not been exposed externally. Under the deployment section click on the frontend and expose the replica externally.

9. Exposing the frontend service

$ kubectl get service

```
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP          PORT(S)          AGE
backend        ClusterIP      100.71.213.149    <none>              5000/TCP         15m
frontend       ClusterIP      100.68.146.118    <none>              80/TCP           15m
Kubernetes      ClusterIP     100.64.0.1       <none>               443/TCP          55m
redis          ClusterIP      100.68.25.60     <none>               6379/TCP         15m
```

$ kubectl expose deployment frontend --type=LoadBalancer --name=frontend1

$ kubectl get service

```
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP          PORT(S)          AGE
backend        ClusterIP      100.71.213.149    <none>              5000/TCP         15m
frontend       ClusterIP      100.68.146.118    <none>              80/TCP           15m
frontend1     Loadbalancer    100.65.46.247    xxxamazonaws.com    80:30952/TCP      37s
Kubernetes      ClusterIP     100.64.0.1       <none>               443/TCP          55m
redis          ClusterIP      100.68.25.60     <none>               6379/TCP          15m
```

Now, under Discovery and Load Balancing > Services, you can see that the frontend1 service has been deployed and exposed externally with a External endpoint attached as a AWS load balancer.

10. Cleaning Up

To delete the stack deployed on Kubernetes using kompose

$ kubectl delete deployment frontend
$ kubectl delete deployment backend
$ kubectl delete deployment redis

And to delete the service we created to expose frontend app

$ kubectl delete svc frontend
$ kubectl delete svc backend
$ kubectl delete svc redis
$ kubectl delete svc frontend1