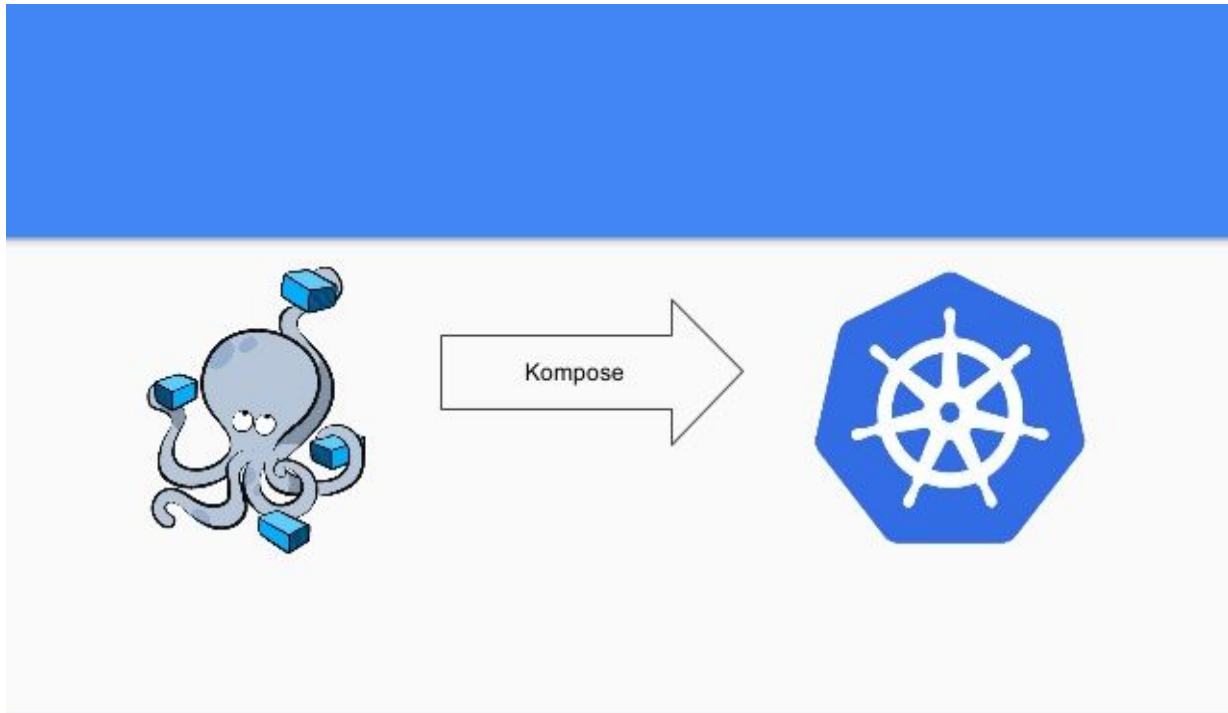


## Migrating a docker-compose 3 tier application stack to kubernetes.

**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. **`sudo git clone`** <https://github.com/LovesCloud/Docker-compose-demo.git>
2. **`cd Docker-compose-demo/`**
3. **`vim docker-compose.yaml`** (file - <https://pastebin.com/raw/MifjkX0W>)
4. **Add <your-name> after backend, frontend and redis in the `docker-compose-demo.yaml`**



The docker-compose-demo.yaml should look something like below.

```

version: '3'
services:
  backend-arshad:
    image: asyed755/composetest_backend
    ports:
      - "5000"
    links:
      - redis
  frontend-arshad:
    image: asyed755/delldemo:v1
    ports:
      - "80:80"
    links:
      - backend
  redis-arshad:
    image: "redis:alpine"
    ports:
      - "6379"

```

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	<a href="#">&gt; DETAILS</a>
 asyed755/composetest_backend public	0 STARS	3 PULLS	<a href="#">&gt; DETAILS</a>

- 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.**

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

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

```
$ kubectl apply -f redis-<your-name>-deployment.yaml
$ kubectl apply -f redis-<your-name>-service.yaml
$ kubectl apply -f frontend-<your-name>-deployment.yaml
$ kubectl apply -f frontend-<your-name>-service.yaml
$ kubectl apply -f backend-<your-name>-deployment.yaml
$ kubectl apply -f backend-<your-name>-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 expose deployment frontend
--type=NodePort--name=frontend1-<your-name>
```

```
$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
backend-arshad	ClusterIP	100.71.213.149	<none>	5000/TCP	15m
frontend-arshad	ClusterIP	100.68.146.118	<none>	80/TCP	15m
frontend1-arshad	NodePort	100.65.46.247	<none>	80:30952/TCP	37s

Kubernetes	ClusterIP	100.64.0.1	<none>	443/TCP	55m
redis-arshad	ClusterIP	100.68.25.60	<none>	6379/TCP	15m

Now, login to the **AWS Console** and Copy the **Public IP** of the NODE where the POD has been deployed to access the application.

Access the application as shown below

<http://<node-public-ip>:NodePort> ##see step 5 to check the port

Example

<http://18.208.206.161:32587/>

## 10. Cleaning Up

To delete the stack deployed on Kubernetes using kompose

```
$ kubectl delete deployment frontend-<your-name>
```

```
$ kubectl delete deployment backend-<your-name>
```

```
$ kubectl delete deployment redis-<your-name>
```

And to delete the service we created to expose frontend app

```
$ kubectl delete svc frontend-<your-name>
```

```
$ kubectl delete svc backend-<your-name>
```

```
$ kubectl delete svc redis-<your-name>
```

```
$ kubectl delete svc frontend1-<your-name>
```



