Tareas

HOGAREMPEZANDOCONCEPTOSTAREASTUTORIALESREFERENCIACONTRIBUIR

Buscar

Traducir un archivo Docker Compose a recursos de Kubernetes

¿Qué es Kompose? Es una herramienta de conversión para todas las cosas que componen (a saber, Docker Compose) en orquestadores de contenedores (Kubernetes u OpenShift).

Se puede encontrar más información en el sitio web de Kompose en http://kompose.io .

Antes de que empieces

Debe tener un clúster de Kubernetes y la herramienta de línea de comandos kubectl debe estar configurada para comunicarse con su clúster. Si aún no tiene un clúster, puede crear uno usando Minikube , o puede usar uno de estos parques infantiles de Kubernetes:

- Katacoda

Instalar Kompose

Tenemos múltiples formas de instalar Kompose. Nuestro método preferido es descargar el binario de la última versión de GitHub.

Descargar GitHub | Construir desde la fuente | Paquete CentOS | Paquete Fedora | Homebrew (macOS)

Kompose se lanza a través de GitHub en un ciclo de tres semanas, puede ver todos los lanzamientos actuales en la página de lanzamiento de GitHub .

```
# Linux
curl -L https://github.com/kubernetes/kompose/releases/download/v1.21.0/k

# macOS
curl -L https://github.com/kubernetes/kompose/releases/download/v1.21.0/k

# Windows
curl -L https://github.com/kubernetes/kompose/releases/download/v1.21.0/k

chmod +x kompose
sudo mv ./kompose /usr/local/bin/kompose
```

Alternativamente, puede descargar el tarball .

Use Kompose

En solo unos pocos pasos, lo llevaremos de Docker Compose a Kubernetes. Todo lo que necesitas es un `docker-compose.yml` archivo existente .

1. Vaya al directorio que contiene su `docker-compose.yml` archivo. Si no tiene uno, pruebe con este.

```
services:

  redis-master:
    image: k8s.gcr.io/redis:e2e
    ports:
      - "6379"

  redis-slave:
    image: gcr.io/google_samples/gb-redisslave:v3
    ports:
      - "6379"
    environment:
      - GET_HOSTS_FROM=dns

  frontend:
    image: gcr.io/google-samples/gb-frontend:v4
    ports:
      - "80:80"
    environment:
      - GET_HOSTS_FROM=dns
    labels:
      kompose.service.type: LoadBalancer
```

2. Ejecute el `kompose up` comando para implementar directamente en Kubernetes, o salte al siguiente paso para generar un archivo para usar `kubectl` .

```
If you need different kind of resources, use the 'kompose convert' and 'k

INFO Successfully created Service: redis
INFO Successfully created Service: web
INFO Successfully created Deployment: redis
INFO Successfully created Deployment: web

Your application has been deployed to Kubernetes. You can run 'kubectl ge
```

3. Para convertir el `docker-compose.yml` archivo a archivos que puede usar `kubectl`, ejecute `kompose convert` y luego `kubectl apply -f <output file>`.

```
$ kompose convert
INFO Kubernetes file "frontend-service.yaml" created
INFO Kubernetes file "redis-master-service.yaml" created
INFO Kubernetes file "redis-slave-service.yaml" created
INFO Kubernetes file "frontend-deployment.yaml" created
INFO Kubernetes file "redis-master-deployment.yaml" created
INFO Kubernetes file "redis-slave-deployment.yaml" created
```

```
$ kubectl apply -f frontend-service.yaml,redis-master-service.yaml,redis-
service/frontend created
service/redis-master created
service/redis-slave created
deployment.apps/frontend created
deployment.apps/redis-master created
deployment.apps/redis-slave created
```

Sus implementaciones se están ejecutando en Kubernetes.

4. Acceda a su aplicación.

```
$ minikube service frontend
```

De lo contrario, ¡veamos qué IP está usando su servicio!

```
$ kubectl describe svc frontend
Name:                   frontend
Namespace:              default
Labels:                 service=frontend
Selector:               service=frontend
Type:                   LoadBalancer
IP:                     10.0.0.183
LoadBalancer Ingress:   192.0.2.89
Port:                   80      80/TCP
NodePort:               80      31144/TCP
Endpoints:              172.17.0.4:80
Session Affinity:       None
No events.
```

Si está utilizando un proveedor de la nube, su IP aparecerá junto a `LoadBalancer Ingress` .

```
$ curl http://192.0.2.89
```

Guía del usuario

- CLI
  - [`kompose convert`]
  - [`kompose up`]
  - [`kompose down`]
- Documentación

Kompose tiene soporte para dos proveedores: OpenShift y Kubernetes. Puede elegir un proveedor objetivo utilizando la opción global `--provider` . Si no se especifica ningún proveedor, Kubernetes se configura de manera predeterminada.

`kompose convert`

Kompose admite la conversión de archivos Docker Compose V1, V2 y V3 en objetos Kubernetes y OpenShift.

Kubernetes

```
$ kompose --file docker-voting.yml convert
WARN Unsupported key networks - ignoring
WARN Unsupported key build - ignoring
INFO Kubernetes file "worker-svc.yaml" created
INFO Kubernetes file "db-svc.yaml" created
INFO Kubernetes file "redis-svc.yaml" created
INFO Kubernetes file "result-svc.yaml" created
INFO Kubernetes file "vote-svc.yaml" created
INFO Kubernetes file "redis-deployment.yaml" created
INFO Kubernetes file "result-deployment.yaml" created
INFO Kubernetes file "vote-deployment.yaml" created
INFO Kubernetes file "worker-deployment.yaml" created
INFO Kubernetes file "db-deployment.yaml" created

$ ls
db-deployment.yaml  docker-compose.yml         docker-gitlab.yml  redis-deploym
db-svc.yaml         docker-voting.yml          redis-svc.yaml     result-svc.ya
```

You can also provide multiple docker-compose files at the same time:

```
INFO Kubernetes file "mongodb-service.yaml" created
INFO Kubernetes file "redis-master-service.yaml" created
INFO Kubernetes file "redis-slave-service.yaml" created
INFO Kubernetes file "frontend-deployment.yaml" created
INFO Kubernetes file "mlbparks-deployment.yaml" created
INFO Kubernetes file "mongodb-deployment.yaml" created
INFO Kubernetes file "mongodb-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "redis-master-deployment.yaml" created
INFO Kubernetes file "redis-slave-deployment.yaml" created

$ ls
mlbparks-deployment.yaml  mongodb-service.yaml                       redis-slav
frontend-deployment.yaml  mongodb-claim0-persistentvolumeclaim.yaml  redis-mast
frontend-service.yaml     mongodb-deployment.yaml                    redis-slav
redis-master-deployment.yaml
```

When multiple docker-compose files are provided the configuration is merged. Any configuration that is common will be over ridden by subsequent file.

OpenShift

```
$ kompose --provider openshift --file docker-voting.yml convert
WARN [worker] Service cannot be created because of missing port.
INFO OpenShift file "vote-service.yaml" created
INFO OpenShift file "db-service.yaml" created
INFO OpenShift file "redis-service.yaml" created
INFO OpenShift file "result-service.yaml" created
INFO OpenShift file "vote-deploymentconfig.yaml" created
INFO OpenShift file "vote-imagestream.yaml" created
INFO OpenShift file "worker-deploymentconfig.yaml" created
INFO OpenShift file "worker-imagestream.yaml" created
INFO OpenShift file "db-deploymentconfig.yaml" created
INFO OpenShift file "db-imagestream.yaml" created
INFO OpenShift file "redis-deploymentconfig.yaml" created
INFO OpenShift file "redis-imagestream.yaml" created
INFO OpenShift file "result-deploymentconfig.yaml" created
INFO OpenShift file "result-imagestream.yaml" created
```

It also supports creating buildconfig for build directive in a service. By default, it uses the remote repo for the current git branch as the source repo, and the current branch as the source branch for the build. You can specify a different source repo and branch using `--build-repo` and `--build-branch` options respectively.

```
INFO OpenShift file "foo-deploymentconfig.yaml" created
INFO OpenShift file "foo-imagestream.yaml" created
INFO OpenShift file "foo-buildconfig.yaml" created
```

Note: If you are manually pushing the OpenShift artifacts using `oc create -f`, you need to ensure that you push the imagestream artifact before the buildconfig artifact, to workaround this OpenShift issue: https://github.com/openshift/origin/issues/4518 .

`kompose up`

Kompose supports a straightforward way to deploy your "composed" application to Kubernetes or OpenShift via `kompose up`.

Kubernetes

```
INFO Successfully created service: redis-master
INFO Successfully created service: redis-slave
INFO Successfully created service: frontend
INFO Successfully created deployment: redis-master
INFO Successfully created deployment: redis-slave
INFO Successfully created deployment: frontend

Your application has been deployed to Kubernetes. You can run 'kubectl get depl

$ kubectl get deployment,svc,pods
NAME                                  DESIRED       CURRENT       U
deployment.extensions/frontend        1             1             1
deployment.extensions/redis-master    1             1             1
deployment.extensions/redis-slave     1             1             1

NAME                      TYPE             CLUSTER-IP     EXTERNAL-IP   POR
service/frontend          ClusterIP        10.0.174.12    <none>        80/
service/kubernetes        ClusterIP        10.0.0.1       <none>        443
service/redis-master      ClusterIP        10.0.202.43    <none>        637
service/redis-slave       ClusterIP        10.0.1.85      <none>        637

NAME                            READY         STATUS        RESTARTS     AG
pod/frontend-2768218532-cs5t5   1/1           Running       0            4m
pod/redis-master-1432129712-63jn8  1/1        Running       0            4m
pod/redis-slave-2504961300-nve7b   1/1        Running       0            4m
```

Note:

- You must have a running Kubernetes cluster with a pre-configured kubectl context.
- Only deployments and services are generated and deployed to Kubernetes. If you need different kind of resources, use the `kompose convert` and `kubectl apply -f` commands instead.

OpenShift

```
INFO Successfully created service: redis-slave
INFO Successfully created service: frontend
INFO Successfully created service: redis-master
INFO Successfully created deployment: redis-slave
INFO Successfully created ImageStream: redis-slave
INFO Successfully created deployment: frontend
INFO Successfully created ImageStream: frontend
INFO Successfully created deployment: redis-master
INFO Successfully created ImageStream: redis-master

Your application has been deployed to OpenShift. You can run 'oc get dc,svc,is'

$ oc get dc,svc,is
NAME                REVISION                              DESIRED       CURRENT
dc/frontend         0                                     1             0
dc/redis-master     0                                     1             0
dc/redis-slave      0                                     1             0
NAME                CLUSTER-IP                            EXTERNAL-IP   PORT(S)
svc/frontend        172.30.46.64                          <none>        80/TCP
svc/redis-master    172.30.144.56                         <none>        6379/TCP
svc/redis-slave     172.30.75.245                         <none>        6379/TCP
NAME                DOCKER REPO                           TAGS          UPDATED
is/frontend         172.30.12.200:5000/fff/frontend
is/redis-master     172.30.12.200:5000/fff/redis-master
is/redis-slave      172.30.12.200:5000/fff/redis-slave    v1
```

Note:

- You must have a running OpenShift cluster with a pre-configured `oc` context (`oc login`)

`kompose down`

Once you have deployed "composed" application to Kubernetes, `$ kompose down` will help you to take the application out by deleting its deployments and services. If you need to remove other resources, use the 'kubectl' command.

```
INFO Successfully deleted service: redis-slave
INFO Successfully deleted deployment: redis-slave
INFO Successfully deleted service: frontend
INFO Successfully deleted deployment: frontend
```

Note:

- You must have a running Kubernetes cluster with a pre-configured kubectl context.

Build and Push Docker Images

Kompose supports both building and pushing Docker images. When using the `build` key within your Docker Compose file, your image will:

- Automatically be built with Docker using the `image` key specified within your file
- Be pushed to the correct Docker repository using local credentials (located at `.docker/config`)

Using an example Docker Compose file:

```
version: "2"

services:
    foo:
        build: "./build"
        image: docker.io/foo/bar
```

Using `kompose up` with a `build` key:

```
INFO Image 'docker.io/foo/bar' from directory 'build' built successfully
INFO Pushing image 'foo/bar:latest' to registry 'docker.io'
INFO Attempting authentication credentials 'https://index.docker.io/v1/
INFO Successfully pushed image 'foo/bar:latest' to registry 'docker.io'
INFO We are going to create Kubernetes Deployments, Services and PersistentVolu

INFO Deploying application in "default" namespace
INFO Successfully created Service: foo
INFO Successfully created Deployment: foo

Your application has been deployed to Kubernetes. You can run 'kubectl get depl
```

In order to disable the functionality, or choose to use BuildConfig generation (with OpenShift) `--build (local|build-config|none)` can be passed.

```
# Disable building/pushing Docker images
$ kompose up --build none

# Generate Build Config artifacts for OpenShift
$ kompose up --provider openshift --build build-config
```

Alternative Conversions

The default `kompose` transformation will generate Kubernetes Deployments and Services, in yaml format. You have alternative option to generate json with `-j`. Also, you can alternatively generate Replication Controllers objects, Daemon Sets, or Helm charts.

```
$ kompose convert -j
INFO Kubernetes file "redis-svc.json" created
INFO Kubernetes file "web-svc.json" created
INFO Kubernetes file "redis-deployment.json" created
INFO Kubernetes file "web-deployment.json" created
```

The `*-deployment.json` files contain the Deployment objects.

```
INFO Kubernetes file "redis-replicationcontroller.yaml" created
INFO Kubernetes file "web-replicationcontroller.yaml" created
```

The `*-replicationcontroller.yaml` files contain the Replication Controller objects. If you want to specify replicas (default is 1), use `--replicas` flag: `$ kompose convert --replication-controller --replicas 3`

```
$ kompose convert --daemon-set
INFO Kubernetes file "redis-svc.yaml" created
INFO Kubernetes file "web-svc.yaml" created
INFO Kubernetes file "redis-daemonset.yaml" created
INFO Kubernetes file "web-daemonset.yaml" created
```

The `*-daemonset.yaml` files contain the Daemon Set objects

If you want to generate a Chart to be used with Helm simply do:

```
$ kompose convert -c
INFO Kubernetes file "web-svc.yaml" created
INFO Kubernetes file "redis-svc.yaml" created
INFO Kubernetes file "web-deployment.yaml" created
INFO Kubernetes file "redis-deployment.yaml" created
chart created in "./docker-compose/"

$ tree docker-compose/
docker-compose
├── Chart.yaml
├── README.md
└── templates
    ├── redis-deployment.yaml
    ├── redis-svc.yaml
    ├── web-deployment.yaml
    └── web-svc.yaml
```

The chart structure is aimed at providing a skeleton for building your Helm charts.

Labels

`kompose` supports Kompose-specific labels within the `docker-compose.yml` file in order to explicitly define a service’s behavior upon conversion.

```
version: "2"
services:
  nginx:
    image: nginx
    dockerfile: foobar
    build: ./foobar
    cap_add:
      - ALL
    container_name: foobar
    labels:
      kompose.service.type: nodeport
```

- `kompose.service.expose` defines if the service needs to be made accessible from outside the cluster or not. If the value is set to "true", the provider sets the endpoint automatically, and for any other value, the value is set as the hostname. If multiple ports are defined in a service, the first one is chosen to be the exposed.
  - For the Kubernetes provider, an ingress resource is created and it is assumed that an ingress controller has already been configured.
  - For the OpenShift provider, a route is created.

For example:

```
version: "2"
services:
  web:
    image: tuna/docker-counter23
    ports:
     - "5000:5000"
    links:
     - redis
    labels:
      kompose.service.expose: "counter.example.com"
  redis:
    image: redis:3.0
    ports:
     - "6379"
```

The currently supported options are:

kompose.service.expose	true / hostname

Note: The `kompose.service.type` label should be defined with `ports` only, otherwise `kompose` will fail.

Restart

If you want to create normal pods without controllers you can use `restart` construct of docker-compose to define that. Follow table below to see what happens on the `restart` value.

`docker-compose` `restart`	object created	Pod `restartPolicy`
`""`	controller object	`Always`
`always`	controller object	`Always`
`on-failure`	Pod	`OnFailure`
`no`	Pod	`Never`

Note: The controller object could be `deployment` or `replicationcontroller`, etc.

For example, the `pival` service will become pod down here. This container calculated value of `pi`.

```
version: '2'

services:
  pival:
    image: perl
    command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
    restart: "on-failure"
```

Warning about Deployment Config’s

same time.

If the Docker Compose file has service name with `_` in it (eg. `web_service`), then it will be replaced by `-` and the service name will be renamed accordingly (eg. `web-service`). Kompose does this because "Kubernetes" doesn't allow `_` in object name.

Please note that changing service name might break some `docker-compose` files.

Docker Compose Versions

Kompose supports Docker Compose versions: 1, 2 and 3. We have limited support on versions 2.1 and 3.2 due to their experimental nature.

A full list on compatibility between all three versions is listed in our conversion document including a list of all incompatible Docker Compose keys.

Feedback

Was this page helpful?

Yes No

Create an Issue Edit This Page

Page last modified on March 01, 2020 at 2:12 AM PST by Update some instances of latin abbreviation e.g. to alternative phrases (#19182) (Page History)

Case Studies

Contribute