Deployment to Kubernetes

This document describes how to use the Carvel packages to deploy Spring Cloud Data Flow, and it also has a section on creating local development cluster using minikube or kind.

Deployment using Carvel

Deployment of a carvel package requires the installation of tools and specific Kubernetes controllers. Then you will add the package repository to the cluster and install the application.

Required Tools

- `kubectl` - Kubernetes CLI (Install with `brew install kubectl`)
- `carvel` - Packaging and Deployment tools

Carvel CLI can be installed using:

```
wget -O- https://carvel.dev/install.sh | bash
# or with curl...
curl -L https://carvel.dev/install.sh | bash
```

Alternative following the instructions at the bottom of the home page at carvel.dev

The following tools are use by the scripts.

- `jq` - lightweight JSON parser
- `yq` - lightweight YAML parser
- `wget` - Invoke http requests.
- `dirname` provides the directory part of a filename.
- `readlink` provides absolute path of a relative link.

NOTE	Some of these utilities are not installed in macOS or *nix by default but will be available from MacPorts or HomeBrew.

Scripts

These scripts assume you are connected to a Kubernetes cluster and `kubectl` is available.

Name	Arguments	Descriptions
`start-deploy.sh`	<broker> [scdf-type] [namespace] [release|snapshot]	Configures environmental variables needs for the rest of the scripts. `BROKER`, `NS` and `SCDF_TYPE` are set. The default `NS` is `scdf`. The namespace will be created if it doesn't exist by `setup-scdf-repo.sh`. The default `SCDF_TYPE` is `oss`. *release*|*snapshot* and *scdf-type* will determine the value of `PACKAGE_VERSION` set.
`prepare-cluster.sh`	N/A	Installs cert-manager, secretgen-controller and kapp-controller
`carvel-use-template.sh`	[scdf-type] (oss, pro)	Creates `scdf-values.yml` in current directory based on `scdf-pro-values.yml` or `scdf-oss-values.yml`
`setup-scdf-repo.sh`	[scdf-type] (oss, pro)	Creates the namespace and installs the relevant Carvel package and credentials. If the optional *scdf-type* is not provided the environmental variable `SCDF_TYPE` will be used.
`configure-prometheus-proxy.sh`	<host> <port> [step]	Configures Spring Boot Actuator properties for Data Flow, Skipper, Streams and Tasks. Default `step` is 10s
`configure-database.sh`	<app> <database> <url> <username/secret-name> [password/secret-username-key] [secret-password-key]	If only *secret-name* is provided then *secret-username-key* defaults to `username` and *secret-password-key* defaults to `password`. The following 3 combinations are allowed after the *url*: • <secret-name> • <secret-name> <username-key> <password-key> • <username> <password>
`deploy-scdf.sh`	[app-name]	Deploys the application using the package and `scdf-values.yml` in the current directory. The default *app-name* is `scdf-${SCDF_TYPE}`.
`update-scdf.sh`	[app-name]	Updated the deployed application using a modified values file. The default *app-name* is `scdf-${SCDF_TYPE}`.
`export-dataflow-ip.sh`	N/A	Will print the URL to access dataflow. If you use `source ./export-dataflow-ip.sh` it will export `DATAFLOW_URL` to be used by `register-apps.sh`
`register-apps.sh`	<broker> [stream-application-version]	*broker* must be one of rabbit or kafka. *stream-application-version* is optional and will install the latest version. The latest version is 2021.1.2

NOTE	Take note that the registration of application in the *pro* version can take a few minutes since it retrieves all version information and metadata upfront.

Prepare Configuration parameters

Executing the following script will configure the environmental variables needed.

```
source ./start-deploy.sh <broker> <namespace> [scdf-type] [release\|snapshot]
```

Where:

- `broker` is one of rabbitmq or kafka
- `namespace` A valid Kubernetes namespace other than `default`
- `scdf-type` One of oss or pro. oss is the default.
- `release\|snapshot` and `scdf-type` will determine the value of `PACKAGE_VERSION`.

The environmental variables can also be configured manually to override the values.

Name	Description	Default
`PACKAGE_VERSION`	Version of Carvel package.	Release version
`DATAFLOW_VERSION`	Version of Spring Cloud Data Flow	2.10.3
`SKIPPER_VERSION`	Version of Spring Cloud Skipper	2.9.3
`REGISTRY`	Url and repository of package registry. Format `<private-registry-host/repo-name>`. This will be used to prefix the carvel repo and package.	`docker.io/springcloud`
`BROKER`	One of `kafka` or `rabbitmq`	`rabbitmq`
`DATABASE`	One of `mariadb` or `postgresql`. The default is `postgresql`. This will only apply when you `deploy-local-database.sh`	`postgresql`
`NS`	A Kubernetes namespace other than `default`.	`scdf`
`SCDF_TYPE`	One of `oss` or `pro`.	`oss`

Prepare Configuration file

Create a file name `scdf-values.yml` by executing:

```
./carvel-use-template.sh
```

Edit the file as needed to configure the deployment. The `deploy-local-` scripts will

Uses scdf-type previously selected.

Prepare cluster and add repository

Login to docker and optionally registry.pivotal.io for Spring Cloud Data Flow Pro.

```
# When deploying SCDF Pro.
export TANZU_DOCKER_USERNAME="<tanzu-net-username>"
export TANZU_DOCKER_PASSWORD="<tanzu-net-password>"
docker login --username $TANZU_DOCKER_USERNAME --password $TANZU_DOCKER_PASSWORD
registry.pivotal.io

# Always required to ensure you don't experience rate limiting with Docker HUB
export DOCKER_HUB_USERNAME="<docker-hub-username>"
export DOCKER_HUB_PASSWORD="<docker-hub-password>"
docker login --username $DOCKER_HUB_USERNAME --password $DOCKER_HUB_PASSWORD
index.docker.io
```

Install carvel kapp-controller, secretgen-controller and certmanager

```
./prepare-cluster.sh
```

Load scdf repo package for the *scdf-type*

```
./setup-scdf-repo.sh
```

Install supporting services

In a production environment you should be using supported database and broker services or operators along with shared observability tools.

For local development or demonstration the following can be used to install database, broker and prometheus.

Deploy local database.

```
./deploy-local-database.sh <database>  ①
```

① `database` must be one of `postgresql` or `mariadb`. Default is postgresql or configure in `DATABASE` using `start-deploy.sh`.

NOTE This script updates `scdf-values.yml` with the correct secret name.

Deploy local message broker.

```
./deploy-local-broker.sh
```

Deploy local Prometheus and proxy.

```
./deploy-local-prometheus.sh
```

This script also configures the Grafana endpoint in `scdf-values.yml`

Configure Prometheus proxy

In the case where and existing prometheus and prometheus proxy is deployed the proxy can be configured using:

```
./configure-prometheus-proxy.sh <host> <port> [step]
```

Deploy Spring Cloud Data Flow

```
./deploy-scdf.sh
# This should display Dataflow URL: <url-to-access-dataflow>
source ./export-dataflow-ip.sh
./register-apps.sh
```

Update deployed application.

You can modify the values file used during installation and then update the deployment using `update-scdf.sh`

Configure Kubernetes for local development or testing

Prerequisites

You will need to install kubectl and then kind or minikube for a local cluster.

All the examples assume you have cloned the `spring-cloud-dataflow` repository and are executing the scripts from `src/local/k8s`.

On macOS you may need to install `realpath` from Macports or `brew install realpath`

NOTE	The scripts require a shell like `bash` or `zsh` and should work on Linux, WSL 2 or

macOS.

Steps

- Choose Kubernetes provider. Kind, Minikube or remote GKE or TMC.
- Decide the namespace to use for deployment if not `default`.
- Configure Kubernetes and loadbalancer.
- Choose Broker with `export BROKER=kafka|rabbitmq`
- Build or Pull container images for Skipper and Data Flow Server.
- Deploy and Launch Spring Cloud Data Flow.
- Export Data Flow Server address to env.

Kubernetes Provider

How do I choose between minikube and kind? kind will generally provide quicker setup and teardown time than Minikube. There is little to choose in terms of performance between the 2 apart from being able to configure limits on CPUs and memory when deploying minikube. So in the case where you have memory constraints or need to enforce memory limitations Minikube will be a better option.

Kubectl

You will need to install kubectl in order to configure the Kubernetes cluster

Kind

Kind is Kubernetes in docker and ideal for local development.

- Installation
- LoadBalancer

The LoadBalancer will be installed by the `configure-k8s.sh` script by will require an update to a yaml file to provide the address range available to the LoadBalancer.

Minikube

Minikube uses one of a selection of drivers to provide a virtualization environment.

- Installation
- LoadBalancer

NOTE Delete existing Minikube installation if you have any. `minikube delete`

Remote TMC Cluster

Tanzu Mission Control

Building and loading containers.

For local development you need control of the containers used in the local environment.

In order to ensure to manage the specific versions of data flow and skipper containers you can set SKIPPER_VERSION and DATAFLOW_VERSION environmental variable and then invoke `./pull-dataflow.sh` and `./pull-skipper.sh` or if you want to use a locally built application you can invoke `./build-skipper-image.sh` and `./build-dataflow.sh`

Configure k8s environment

You can invoke one of the following scripts to choose the type of installation you are targeting:

```
use-kind.sh [<namespace>] [<database>] [<broker>]
use-mk-docker.sh [<namespace>] [<database>] [<broker>]
use-mk-kvm2.sh [<namespace>] [<database>] [<broker>]
use-mk.sh <driver> [<namespace>] [<database>] [<broker>] ①
use-tmc.sh <cluster-name> [<namespace>] [<database>] [<broker>]
use-gke.sh <cluster-name> [<namespace>] [<database>] [<broker>]
```

① <driver> must be one of `kvm2`, `docker`, `vmware`, `virtualbox`, `vmwarefusion` or `hyperkit`. `docker` is the recommended option for local development.

NOTE	`<namespace>` will be `default` if not provided. The default `<database>` is `postgresql` and the default `<broker>` is `kafka`.

Since these scripts export environmental variable they need to be executes as in the following example:

```
source ./use-mk-docker.sh test-ns postgresql rabbitmq
```

TMC or GKE Cluster in Cloud

The cluster must exist before use and you should use the relevant cli to login before executing `source ./use-gke.sh`

Create Local Cluster.

The following script will create the local cluster.

```
# Optionally add to control cpu and memory allocation.
export MK_ARGS="--cpus=8 --memory=12g"
./configure-k8s.sh
```

- For **kind** follow instruction to update `src/local/k8s/yaml/metallb-configmap.yaml` and then apply using `kubectl apply -f src/local/k8s/yaml/metallb-configmap.yaml`
- For **minikube** launch a new shell and execute `minikube tunnel`

Deploy Spring Cloud Data Flow.

Configure Broker

```
export BROKER=<broker> ①
```

① <broker> one of `kafka` or `rabbitmq`

Configure Database

```
export DATABASE=<database> ①
```

① <database> one of `mariadb` or `postgresql`

NOTE	This is still optional and PostgreSQL support isn't available yet but will follow soon.

```
./install-scdf.sh
source ./export-dataflow-ip.sh
```

Delete the deployment from the cluster.

```
./delete-scdf.sh
```

Delete the cluster

This script will also delete the TMC cluster if you have configured one.

```
./destroy-k8s.sh
```

Utilities

The following list of utilities may prove useful.

Name	Description
k9s	k9s is a text based monitor to explore the Kubernetes cluster.
kail	Extra and tail the logs of various pods based on various naming criteria.

`kail`

- Using kail to log activity related to a specific stream.

```
kail --label=spring-group-id=<stream-name>
```

- Using kail to log all pods in specific namespace.

```
kail --ns=<namespace>
```

Scripts

Some of the scripts apply to local containers as well and can be found in `src/local`, the Kubernetes specific scripts are in `src/local/k8s`

Script	Description
`build-app-images.sh`	Build all images of Restaurant Sample Stream Apps
`pull-app-images.sh`	Pull all images of Restaurant Sample Stream Apps from Docker Hub
`pull-dataflow.sh`	Pull dataflow from DockerHub based on `DATAFLOW_VERSION`.
`pull-scdf-pro.sh`	Pull Dataflow Pro from Tanzu Network based on `SCDF_PRO_VERSION`.
`pull-skipper.sh`	Pull Skipper from DockerHub base on the `SKIPPER_VERSION`.
`build-dataflow-image.sh`	Build a docker image from the local repo of Dataflow
`build-scdf-pro-image.sh`	Build a docker image from the local repo of Dataflow Pro. Set `USE_PRO=true` in environment to use Dataflow Pro
`build-skipper-image.sh`	Build a docker image from the local repo of Skipper.
`configure-k8s.sh`	Configure the Kubernetes environment based on your configuration of K8S_DRIVER.
`delete-scdf.sh`	Delete all Kubernetes resources create by the deployment.
`destroy-k8s.sh`	Delete cluster, kind or minikube.
`export-dataflow-ip.sh`	Export the url of the data flow server to `DATAFLOW_IP`
`export-http-url.sh`	Export the url of an http source of a specific flow by name to `HTTP_APP_URL`
`install-scdf.sh`	Configure and deploy all the containers for Spring Cloud Dataflow
`load-images.sh`	Load all container images required by tests into kind or minikube to ensure you have control over what is used.
`load-image.sh`	Load a specific container image into local kind or minikube.
`local-k8s-acceptance-tests.sh`	Execute acceptance tests against cluster where `DATAFLOW_IP` is pointing.
`register-apps.sh`	Register the Task and Stream apps used by the unit tests.

IMPORTANT	Please report any errors with the scripts along with detail information about the relevant environment.