Kubernetes Workflows

This page is a placeholder for information and processes surrounding Kubernetes (K8).

A quick-start guide can be found here: https://kubernetes.io/docs/setup/pick-right-solution/

This will set up a Kubernetes cluster running locally on your VM.

Pods

A Kubernetes Pod consists of one or more Docker containers running on the same local network.

This allows related or tightly-coupled services to run together with ease, communicating via localhost.

The Kubernetes Pod Spec

Pods can be configured to run with any number of custom configuration options, such as:

- Ports utilized / exposed
- Environment variables
- Custom commands and arguments
- Custom labels for further specification of functionality or purpose

See http://kubernetes.io/docs/user-guide/pods/multi-container/#the-spec-schema for more details on the Kubernetes spec.

Replication Controllers

Pods are mortal, and can crash when things go wrong.

In such cases, it is wise to set up something to monitor a pod or pods and recreate them when necessary.

Enter Replication Controllers, which perform just the task we need!

You can specify all of the inputs to each image of the pod in the same "spec" fashion described above, but also specifying a number of replicas to keep running.

This tells the Controller the if it does not have the desired number of pods, to create or destroy them as necessary to maintain our desired state.

This is immensely powerful in keeping production system running long-term.

Services

A Kubernetes Service allows a set of pods to receive traffic from within the cluster, which is accomplished by sharing the IPs and Ports of the services is through injected environment variables:

```
"SPARK_MASTER_SERVICE_HOST=10.132.232.14",
"SPARK_MASTER_SERVICE_PORT=7077",
"SPARK_MASTER_PORT=tcp://10.132.232.14:7077",
"SPARK_MASTER_PORT_7077_TCP=tcp://10.132.232.14:7077",
"SPARK_MASTER_PORT_7077_TCP_PROTO=tcp",
"SPARK_MASTER_PORT_7077_TCP_PORT=7077",
"SPARK_MASTER_PORT_7077_TCP_ADDR=10.132.232.14",
"SERVICE_HOST=10.132.232.14",
```

Any Replication Controllers started after a service will have several environment variable injected into it regarding the connection details of that service.

You can then reference these environment variable in the RC / Pod spec to use the injected values.

NodePort

Setting up a service with a NodePort will allow the service to receive traffic through the node's public (external) IP as well.

The alternative to NodePort is using a LoadBalancer (see below).

Ingress Loadbalancer

If you're running on GCE, you will need to run a GLBC to handle routing your ingress traffic.

For all other platforms, an nginx instance running within your cluster can serve as the Ingress Loadbalancer

In addition to routing traffic to your running services, these loadbalancers can also handle things like TLS-termination, authentication via basic-auth (working) or OAuth2 (still in ongoing development), and providing unified custom error messages for all services.

Caveats:

- The pod must run on a node with a Public IP
- Your server must have a real DNS **OR** you must run your own bind DNS server within the cluster

Labels and Selectors

You can choose which pods a service affects by applying labels to the Pod(s) in question.

You then simply need to provide a matching set of labels to the Service.

See http://kubernetes.io/docs/user-guide/labels/ for more information.Namespaces

Initially, there is only one namespace name **default** that runs your Kubernetes master.

Namespaces are synonymous with "users" in NDS Labs, and allow you to encapsulate services from one another even further - Services cannot communicate between namespaces.