**OpenWhisk Deployment on Kubernetes**

Apache OpenWhisk is an open source, distributed Serverless platform that executes functions (fx) in response to events at any scale. The OpenWhisk platform supports a programming model in which developers write functional logic (called Actions), in any supported programming language, that can be dynamically scheduled and run in response to associated events (via Triggers) from external sources (Feeds) or from HTTP requests.

This repository supports deploying OpenWhisk to Kubernetes. It contains a Helm chart that can be used to deploy the core OpenWhisk platform and optionally some of its Event Providers to both single-node and multi-node Kubernetes clusters.

**Table of Contents**

* [Prerequisites: Kubernetes and Helm](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#prerequisites-kubernetes-and-helm)
* [Deploying OpenWhisk](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#deploying-openwhisk)
* [Administering OpenWhisk](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#administering-openwhisk)
* [Development and Testing](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#development-and-testing)
* [Cleanup](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#cleanup)
* [Issues](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#issues)

**Prerequisites: Kubernetes and Helm**

[Kubernetes](https://kubernetes.io/) is a container orchestration platform that automates the deployment, scaling, and management of containerized applications. [Helm](https://helm.sh/) is a package manager for Kubernetes that simplifies the management of Kubernetes applications. You do not need to have detailed knowledge of either Kubernetes or Helm to use this project, but you may find it useful to review their basic documentation to become familiar with their key concepts and terminology.

**Kubernetes**

Your first step is to create a Kubernetes cluster that is capable of supporting an OpenWhisk deployment. Although there are some [technical requirements](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-technical-requirements.md) that the Kubernetes cluster must satisfy, any of the options described below is acceptable.

**Simple Docker-based options**

The simplest way to get a small Kubernetes cluster suitable for development and testing is to use one of the Docker-in-Docker approaches for running Kubernetes directly on top of Docker on your development machine. Configuring Docker with 4GB of memory and 2 virtual CPUs is sufficient for the default settings of OpenWhisk. Depending on your host operating system, we recommend the following:

1. MacOS: Use the built-in Kubernetes support in Docker for Mac version 18.06 or later. Please follow our [setup instructions](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-docker-for-mac.md) to initially create your cluster.
2. Linux: Use kubeadm-dind-cluster, but carefully follow our [setup instructions](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-dind-cluster.md) because the default setup of kubeadm-dind-cluster does *not* meet the requirements for running OpenWhisk.
3. Windows: Use the built-in Kubernetes support in Docker for Windows version 18.06 or later. Please follow our [setup instructions](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-docker-for-windows.md) to initially create your cluster.

**Using Minikube**

Minikube provides a Kubernetes cluster running inside a virtual machine (for example VirtualBox). It can be used on MacOS, Linux, or Windows to run OpenWhisk, but is somewhat less flexible than the docker-in-docker options described above. Configuring the Minikube VM with 4GB of memory and 2 virtual CPUs is sufficient for the default settings of OpenWhisk. For details on setting up Minikube, see these [setup instructions](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-minikube.md).

**Using a Kubernetes cluster from a cloud provider**

You can also provision a Kubernetes cluster from a cloud provider, subject to the cluster meeting the [technical requirements](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-technical-requirements.md). You will need at least 1 worker node with 4GB of memory and 2 virtual CPUs to deploy the default configuration of OpenWhisk. You can deploy to significantly larger clusters by scaling up the replica count of the various components and labeling multiple nodes as invoker nodes. We have detailed documentation on using Kubernetes clusters from the following major cloud providers:

* [IBM (IKS)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-ibm-public.md) and [IBM (ICP)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-ibm-private.md)
* [Google (GKE)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-google.md)
* [Amazon (EKS)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-aws.md)

We would welcome contributions of documentation for Azure (AKS) and any other public cloud providers.

**Helm**

[Helm](https://github.com/kubernetes/helm) is a tool to simplify the deployment and management of applications on Kubernetes clusters. Helm consists of the helm command line tool that you install on your development machine and the tiller runtime that is deployed on your Kubernetes cluster.

For details on installing Helm, see these [instructions](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/helm.md).

In short if you already have the helm cli installed on your development machine, you will need to execute these two commands and wait a few seconds for the tiller-deploy pod in the kube-system namespace to be in the Running state.

helm init

kubectl create clusterrolebinding tiller-cluster-admin --clusterrole=cluster-admin --serviceaccount=kube-system:default

**Deploying OpenWhisk**

Now that you have your Kubernetes cluster and have installed and initialized Helm, you are ready to deploy OpenWhisk.

**Overview**

You will use Helm to deploy OpenWhisk to your Kubernetes cluster. There are four deployment steps that are described in more detail below in the rest of this section.

1. [Initial cluster setup](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#initial-setup). You will label your Kubernetes worker nodes to indicate their intended usage by OpenWhisk.
2. [Customize the deployment](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#customize-the-deployment). You will create a mycluster.yaml that specifies key facts about your Kubernetes cluster and the OpenWhisk configuration you wish to deploy.
3. [Deploy OpenWhisk with Helm](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#deploy-with-helm). You will use Helm and mycluster.yaml to deploy OpenWhisk to your Kubernetes cluster.
4. [Configure the wsk CLI](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/README.md#configure-the-wsk-cli). You need to tell the wsk CLI how to connect to your OpenWhisk deployment.

**Initial setup**

Indicate the Kubernetes worker nodes that should be used to execute user containers by OpenWhisk's invokers. Do this by labeling each node with openwhisk-role=invoker. In its default configuration, OpenWhisk assumes it has exclusive use of these invoker nodes and will schedule work on them directly, completely bypassing the Kubernetes scheduler. For a single node cluster, simply do

kubectl label nodes --all openwhisk-role=invoker

If you have a multi-node cluster, then for each node <INVOKER\_NODE\_NAME> you want to be an invoker, execute

$ kubectl label nodes <INVOKER\_NODE\_NAME> openwhisk-role=invoker

For more precise control of the placement of the rest of OpenWhisk's pods on a multi-node cluster, you can optionally label additional non-invoker worker nodes. Use the label openwhisk-role=core to indicate nodes which should run the OpenWhisk control plane (the controller, kafka, zookeeeper, and couchdb pods). If you have dedicated Ingress nodes, label them with openwhisk-role=edge. Finally, if you want to run the OpenWhisk Event Providers on specific nodes, label those nodes with openwhisk-role=provider.

**Customize the Deployment**

You must create a mycluster.yaml file to record key aspects of your Kubernetes cluster that are needed to configure the deployment of OpenWhisk to your cluster. For details, see the documentation appropriate to your Kubernetes cluster:

* [Docker for Mac](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-docker-for-mac.md#configuring-openwhisk)
* [kubeadm-dind-cluster](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-dind-cluster.md#configuring-openwhisk)
* [Minikube](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-minikube.md#configuring-openwhisk)
* [IBM Kubernetes Service (IKS)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-ibm-public.md#configuring-openwhisk)
* [IBM Cloud Private (ICP)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-ibm-private.md#configuring-openwhisk)
* [Google (GKE)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-google.md#configuring-openwhisk)
* [Amazon (EKS)](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/k8s-aws.md#configuring-openwhisk)

Beyond the Kubernetes cluster specific configuration information, the mycluster.yaml file is also used to customize your OpenWhisk deployment by enabling optional features and controlling the replication factor of the various microservices that make up the OpenWhisk implementation. See the [configuration choices documentation](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/configurationChoices.md) for a discussion of the primary options.

**Deploy With Helm**

Deployment can be done by using the following single command:

helm install ./helm/openwhisk --namespace=openwhisk --name=owdev -f mycluster.yaml

For simplicity, in this README, we have used owdev as the release name and openwhisk as the namespace into which the Chart's resources will be deployed. You can use different names, or not specify a release name at all and let Helm auto-generate one for you.

You can use the command helm status owdev to get a summary of the various Kubernetes artifacts that make up your OpenWhisk deployment. Once the install-packages Pod is in the Completed state, your OpenWhisk deployment is ready to be used.

**Configure the wsk CLI**

Configure the OpenWhisk CLI, wsk, by setting the auth and apihost properties (if you don't already have the wsk cli, follow the instructions [here](https://github.com/apache/incubator-openwhisk-cli) to get it). Replace whisk.ingress.apiHostName and whisk.ingress.apiHostPort with the actual values from your mycluster.yaml.

wsk property set --apihost <whisk.ingress.apiHostName>:<whisk.ingress.apiHostPort>

wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

**Configuring the CLI for Kubernetes on Docker for Mac and Windows**

The docker0 network interface does not exist in the Docker for Mac/Windows host environment. Instead, exposed NodePorts are forwarded from localhost to the appropriate containers. This means that you will use localhost instead of whisk.ingress.apiHostName when configuring the wsk cli and replace whisk.ingress.apiHostPort with the actual values from your mycluster.yaml.

wsk property set --apihost localhost:<whisk.ingress.apiHostPort>

wsk property set --auth 23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

**Verify your OpenWhisk Deployment**

Your OpenWhisk installation should now be usable. You can test it by following [these instructions](https://github.com/apache/incubator-openwhisk/blob/master/docs/actions.md) to define and invoke a sample OpenWhisk action in your favorite programming language.

You can also issue the command helm test owdev to run the basic verification test suite included in the OpenWhisk Helm chart.

Note: if you installed self-signed certificates, which is the default for the OpenWhisk Helm chart, you will need to use wsk -i to suppress certificate checking. This works around cannot validate certificate errors from the wsk CLI.

If your deployment is not working, check our [troubleshooting guide](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/troubleshooting.md) for ideas.

**Administering OpenWhisk**

[Wskadmin](https://github.com/apache/incubator-openwhisk/tree/master/tools/admin) is the tool to perform various administrative operations against an OpenWhisk deployment.

Since wskadmin requires credentials for direct access to the database (that is not normally accessible to the outside), it is deployed in a pod inside Kubernetes that is configured with the proper parameters. You can run wskadmin with kubectl. You need to use the <namespace> and the deployment <name> that you configured with --namespace and --name when deploying.

You can then invoke wskadmin with:

kubectl -n <namespace> -ti exec <name>-wskadmin -- wskadmin <parameters>

For example, is your deployment name is owdev and the namespace is openwhisk you can list users in the guest namespace with:

$ kubectl -n openwhisk -ti exec owdev-wskadmin -- wskadmin user list guest

23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9lyGVCGuMDGIwP

Check [here](https://github.com/apache/incubator-openwhisk/tree/master/tools/admin) for details about the available commands.

**Development and Testing OpenWhisk on Kubernetes**

This section outlines how common OpenWhisk development tasks are supported when OpenWhisk is deployed on Kubernetes using Helm.

**Running OpenWhisk test cases**

Some key differences in a Kubernetes-based deployment of OpenWhisk are that deploying the system does not generate a whisk.properties file and that the various internal microservices (invoker, controller, etc.) are not directly accessible from the outside of the Kubernetes cluster. Therefore, although you can run full system tests against a Kubernetes-based deployment by giving some extra command line arguments, any unit tests that assume direct access to one of the internal microservices will fail. The system tests can be executed in a batch-style as shown below, where WHISK\_SERVER and WHISK\_AUTH are replaced by the values returned by wsk property get --apihost and wsk property get --auth respectively.

cd $OPENWHISK\_HOME

./gradlew :tests:testSystemBasic -Dwhisk.auth=$WHISK\_AUTH -Dwhisk.server=https://$WHISK\_SERVER -Dopenwhisk.home=`pwd`

You can also launch the system tests as JUnit test from an IDE by adding the same system properties to the JVM command line used to launch the tests:

-Dwhisk.auth=$WHISK\_AUTH -Dwhisk.server=https://$WHISK\_SERVER -Dopenwhisk.home=`pwd`

**Deploying a locally built docker image.**

If you are using Kubernetes in Docker, it is straightforward to deploy local images by adding a stanza to your mycluster.yaml. For example, to use a locally built controller image, just add the stanza below to your mycluster.yaml to override the default behavior of pulling a stable openwhisk/controller image from Docker Hub.

controller:

imageName: "whisk/controller"

imageTag: "latest"

**Selectively redeploying using a locally built docker image**

You can use the helm upgrade command to selectively redeploy one or more OpenWhisk componenets. Continuing the example above, if you make additional changes to the controller source code and want to just redeploy it without redeploying the entire OpenWhisk system you can do the following:

# Execute these commands in your openwhisk directory

./gradlew distDocker

docker tag whisk/controller whisk/controller:v2

Then, edit your mycluster.yaml to contain

:

controller:

imageName: "whisk/controller"

imageTag: "v2"

Redeploy with Helm by executing this commaned in your openwhisk-deploy-kube directory:

helm upgrade ./helm/openwhisk --namespace=openwhisk --name=owdev -f mycluster.yaml

**Deploying Lean Openwhisk version.**

To have a lean setup (no Kafka, Zookeeper and no Invokers as separate entities):

controller:

lean: true

**Cleanup**

Use the following command to remove all the deployed OpenWhisk components:

helm delete owdev

Helm does keep a history of previous deployments. If you want to completely remove the deployment from helm, for example so you can reuse owdev to deploy OpenWhisk again, use the command:

helm delete owdev --purge

**Issues**

If your OpenWhisk deployment is not working, check our [troubleshooting guide](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/blob/master/docs/troubleshooting.md) for ideas.

Report bugs, ask questions and request features [here on GitHub](https://github.com/gprateek93/incubator-openwhisk-deploy-kube/issues).

You can also join our slack channel and chat with developers. To get access to our slack channel, request an invite [here](http://slack.openwhisk.org).

**Disclaimer**

Apache OpenWhisk Deployment on Kubernetes is an effort undergoing incubation at The Apache Software Foundation (ASF), sponsored by the Apache Incubator. Incubation is required of all newly accepted projects until a further review indicates that the infrastructure, communications, and decision making process have stabilized in a manner consistent with other successful ASF projects. While incubation status is not necessarily a reflection of the completeness or stability of the code, it does indicate that the project has yet to be fully endorsed by the ASF.

* © 2019 GitHub, Inc.
* [Terms](https://github.com/site/terms)
* [Privacy](https://github.com/site/privacy)
* [Security](https://github.com/security)
* [Status](https://githubstatus.com/)
* [Help](https://help.github.com)