Guidelines for Labels and Annotations for OpenShift applications

Version 0.9, 07.01.2019

Table of Contents

Terminology

Software System

Highest level of abstraction that delivers value to its users, whether they are human or not. A software system is composed of one or more `Application`s.

Application

An abstraction that encapsulates the value delivered by the software system. An application is composed of one or more `Component`s.

Component

A component is a set of kubernetes resources, for hosting code or data, that needs to be running in order for the overall software system to work. Each component is a seperately deployable and runnable unit.

Labels

The following table defines common labels applications deployed on OpenShift should use Labels that are marked *REC* are recommended. Those marked *OPT* are optional. The labels should not be relied upon for operational purposes.

You can find more information on the Kubernetes labels, prefixed with app.kubernetes.io, in the Kubernetes documentation.

TIP	Labels must be applied to ALL resources, DeploymentConfigs, BuildConfigs, Services, Routes, ConfigMaps, PersistenVolumeClaims etc., including any custom resource definitions (CRDs).

Table 1. Labels

Name	Status	Description	Example
app.kubernetes.io/part-of	REC	The name of the top level software system this resource is part of	ticketmonster
app.kubernetes.io/name	REC	The name, reflecting component.	mysql
app.kubernetes.io/component	REC	This is the role/type of the component	frontend
app.kubernetes.io/managed-by	OPT	The tool being used to manage the operation of the component	odo
app.kubernetes.io/instance	REC	A name identifying the application, usually used if different from `app.kubernetes.io/name`	accounts
app.kubernetes.io/version	OPT	The current version of the component (e.g., a semantic version, revision hash, etc.)	1.0.0
app.openshift.io/runtime	OPT	The runtime to be used to bootstrap the component	spring-boot

Name	Status	Description	Example
app.openshift.io/runtime-version:	OPT	The version of the runtime	e.g. `1.5.17` for Spring Boot

Table 2. Values for `app.kubernetes.io/component` label

Value	Description
frontend	Serves the UI or part of the UI for an application
backend	Usually an application code that is running on a runtime or framework.
database	Data persistence
integration	Integration middleware such as API gateways or single-sign-on software
cache	
queue	

Annotations

Table 3. Annotations

Name	Status	Description	Example
app.openshift.io/vcs-uri	REC	URI for the source code under version control	git://git.kernel.org/pub/scm/linux.git
app.openshift.io/vcs-ref	REC	Identifier for the version of the source code. Can be branch, tag or commit SHA for the application. If omitted master head of the default branch is assumed	v1.0.0

Examples

Simple microservice with a database

Node.js based rest service with a database (mongodb) backing

```
@startuml
rectangle cool-service {
    database mongodb   [
               \t<b>mongodb for storing data
            ----
            <b>app.kubernetes.io/part-of :</b>cool-service
            <b>app.kubernetes.io/name :</b> mongodb
            <b>app.kubernetes.io/component :</b> database
            <b>app.kubernetes.io/managed-by :</b> odo
            <b>app.kubernetes.io/version :</b> 3.6
        ]
    rectangle nodejs   [
            \t\t<b>The node.js service with code</b>
            ----
            <b>app.kubernetes.io/part-of :</b> cool-service
            <b>app.kubernetes.io/name :</b> nodejs
            <b>app.kubernetes.io/component :</b> frontend
            <b>app.kubernetes.io/managed-by :</b> odo
            <b>app.kubernetes.io/version</b> : 1.0.1
            <b>app.openshift.io/runtime</b>: nodejs
            <b> app.openshift.io/runtime-version</b>: 10.14.1
            ----
            <b>app.openshift.io/vcs-uri :</b> git://git@github.com:gorkem/cool-
service.git
            <b>app.openshift.io/vcs-ref :</b> master
        ]

}

@enduml
```

A complex system with multiple services

```
@startuml
left to right direction

rectangle coolstore {
    together {
        node cart [
                    <b>The node.js service with code</b>
                    ----
                    <b>app.kubernetes.io/part-of :</b> coolstore
                    <b>app.kubernetes.io/instance :</b> cart
                    <b>app.kubernetes.io/name :</b> nodejs
                    <b>app.kubernetes.io/component :</b> frontend
                    <b>app.kubernetes.io/managed-by :</b> odo
                    <b>app.kubernetes.io/version</b> : 10
                    <b>app.openshift.io/runtime</b>: nodejs
```

```
                    <b> app.openshift.io/runtime-version</b>: 10.14.1
                    ----
                    <b>app.openshift.io/vcs-uri :</b>
git://git@github.com:gorkem/cart-service.git
                    <b>app.openshift.io/vcs-ref :</b> master
            ]
    }
   together {
    rectangle catalog {
            database catalogdb  [
                    <b>mongodb for storing data
                    ----
                    <b>app.kubernetes.io/part-of :</b> coolstore
                    <b>app.kubernetes.io/instance :</b> catalog
                    <b>app.kubernetes.io/name :</b> mongodb
                    <b>app.kubernetes.io/component :</b> database
                    <b>app.kubernetes.io/managed-by :</b> odo
                    <b>app.kubernetes.io/version :</b> 3.6
                ]
            rectangle catalog   [
                    <b>The node.js service with code</b>
                    ----
                    <b>app.kubernetes.io/part-of :</b> coolstore
                    <b>app.kubernetes.io/instance :</b> catalog
                    <b>app.kubernetes.io/name :</b> nodejs
                    <b>app.kubernetes.io/component :</b> frontend
                    <b>app.kubernetes.io/managed-by :</b> odo
                    <b>app.kubernetes.io/version</b> : 10
                    <b>app.openshift.io/runtime</b>: nodejs
                    <b> app.openshift.io/runtime-version</b>: 10.14.1
                    ----
                    <b>app.openshift.io/vcs-uri :</b>
git://git@github.com:gorkem/catalog-service.git
                    <b>app.openshift.io/vcs-ref :</b> master
                ]
        }
    }
    together{
        rectangle inventory {
            database postgresql [
                    <b>postgresql database
                    ----
                    <b>app.kubernetes.io/part-of :</b> coolstore
                    <b>app.kubernetes.io/instance :</b> in
                    <b>app.kubernetes.io/name :</b> postgresql
                    <b>app.kubernetes.io/component :</b> database
                    <b>app.kubernetes.io/managed-by :</b> odo
                    <b>app.kubernetes.io/version :</b> 11
                ]
            rectangle java8 [
                    <b>The node.js service with code</b>
```

```
                    ----
                    <b>app.kubernetes.io/part-of :</b> coolstore
                    <b>app.kubernetes.io/instance :</b> inventory
                    <b>app.kubernetes.io/name :</b> nodejs
                    <b>app.kubernetes.io/component :</b> frontend
                    <b>app.kubernetes.io/managed-by :</b> odo
                    <b>app.kubernetes.io/version</b> : 10
                    <b>app.openshift.io/runtime</b>: nodejs
                    <b>app.openshift.io/runtime-version</b>: 10.14.1
                    ----
                    <b>app.openshift.io/vcs-uri :</b>
git://git@github.com:gorkem/inventory-service.git
                    <b>app.openshift.io/vcs-ref :</b> master
                ]
        }
    }
}

@enduml
```

TIP

It is a good practice to check if combination of `app.kubernetes.io/part-of`, `app.kubernetes.io/instance`, `app.kubernetes.io/name` labels lead to a meaningful identifier without repeating parts. For instance `coolstore.catalog.mongodb`. A repeated part is usually an indicator of a label that can be avoided.