Lesson 2 Demo 2: Labels and Selectors in ReplicaSet

This section will guide you to:

* Define labels and selectors in a ReplicaSet

This lab has one sub-section, namely:

1. Defining labels and selectors in a ReplicaSet

**Note:** If you don’t have an existing Kubernetes cluster, refer to the Demo 1.1 of Lesson 1.

**Step 1:** Defining labels and selectors in a ReplicaSet

* Create a ReplicaSet configuration file

*vi replica.yaml*

* To create a simple frontend *nginx* app with 3 replicas, add the following code to the *replica.yaml* file and save it:

*apiVersion: apps/v1*

*kind: ReplicaSet*

*metadata:*

*name: frontendapp*

*labels:*

*app: guestbook*

*tier: frontendapp*

*spec:*

*# modify replicas according to your case*

*replicas: 3*

*selector:*

*matchLabels:*

*tier: frontendapp*

*template:*

*metadata:*

*labels:*

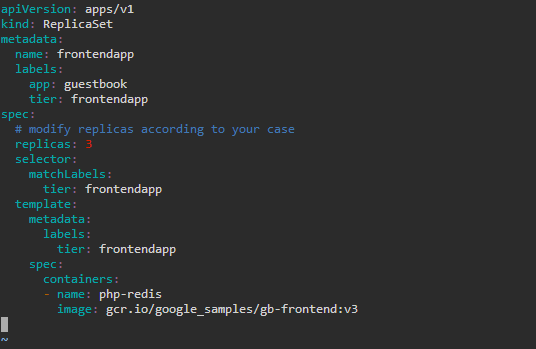
*tier: frontendapp*

*spec:*

*containers:*

*- name: php-redis*

*image: gcr.io/google\_samples/gb-frontend:v3*



**Note:** Press **Esc** button and enter **:wq** to save and exit the text editor

* Run the replica.yaml file to create the ReplicaSet

*kubectl create -f replica.yaml*



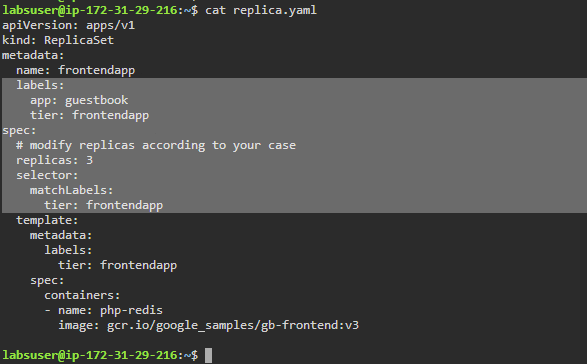
**Note:** In case you get an error about an existing **frontend** ReplicaSet, execute the following commands and run the *kubectl create* command again:

*kubectl get rs*

*kubectl delete replicaset frontend*

* Use the following command to display the replica.yaml showing the Labels and Selectors:

*cat replica.yaml*



**Note:** Labels are key/value pairs that are attached to an object, such as pods. An example of this is the yaml labels shown below:

*labels:*

*app: guestbook*

*tier: frontendapp*

**Note:** Via a label selector, the client/user can identify a set of objects. The label selector is the core grouping primitive in Kubernetes. An example of this is the yaml labels shown below:

*selector:*

*matchLabels:*

*tier: frontendapp*