Implement Replication with ReplicaSet Resource

Definition

Un ReplicaSet est une primitive de l'API Kubernetes qui maintient un nombre spécifié de répliques de pods en fonctionnement. Il assure que le nombre de pods, c'est-à-dire des instances d'une application, reste constant même en cas de défaillance. Le ReplicaSet est souvent utilisé par des déploiements pour automatiser la mise à jour et le scaling des applications.

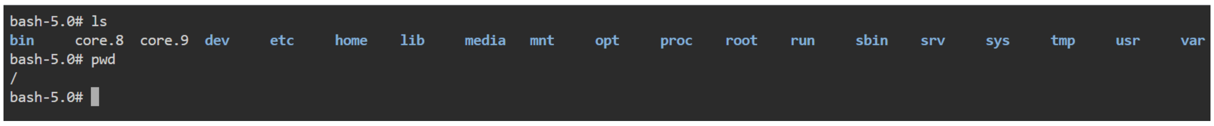
Task Details

1. Launching Lab Environment.
2. SSH into the EC2 instance through the Whiz terminal
3. Create the YML declarative file for Rs
4. Describe the resource
5. Delete the ReplicaSet

Lab Steps

Task 1: Launching Lab Environment

1. Launch the lab environment by clicking on the **Start Lab** button.
2. Please wait until the lab environment is provisioned. It will take around 2 minutes to provision the lab environment which contains the single node Kubernetes cluster. The unique Username, Public IP address, and the Password are displayed on the right corner of the screen.
3. Alongside with the EC2 instance, the Whizlabs terminal is also being provisioned.



```
bash-5.0# ls
bin     core.8  core.9  dev     etc     home    lib     media   mnt     opt     proc    root    run     sbin    srv     sys     tmp     usr     var
bash-5.0# pwd
/
bash-5.0# 
```

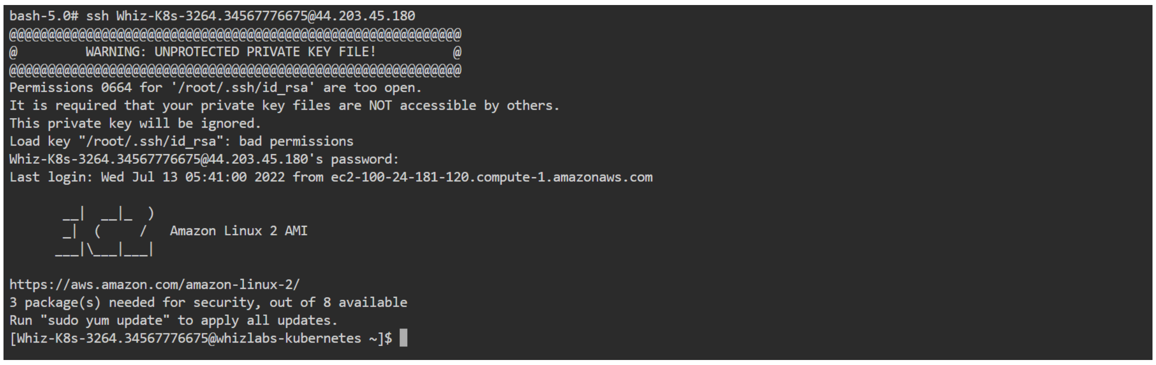
Note : If you face any issues, please go through FAQs and Troubleshooting for Labs.

Task 2: SSH into the EC2 instance through the Whiz terminal

1. Copy the IP address and the Password for SSH to the EC2 instance configured with the Kubernetes cluster.
2. Run the following command to SSH into the system:

```
ssh username@ip_address
```

1. Replace the unique username and the IP address with the details of your own lab environment.
2. Type yes if prompted for the SSH permission for the first time. Then, Insert the password in the prompt as it is.



```
bash-5.0# ssh Whiz-K8s-3264.34567776675@44.203.45.180
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0664 for '/root/.ssh/id_rsa' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/root/.ssh/id_rsa": bad permissions
Whiz-K8s-3264.34567776675@44.203.45.180's password:
Last login: Wed Jul 13 05:41:00 2022 from ec2-100-24-181-120.compute-1.amazonaws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[Whiz-K8s-3264.34567776675@whizlabs-kubernetes ~]$ 
```

3. Now switch the user to root with the command below:

```
sudo su root
```

1. This ec2 instance is configured with Kubernetes packages and all prerequisites are fulfilled. This is a single node cluster and we will use the same node as the client node to communicate and control the containers.

Task 3: Configure ReplicaSet

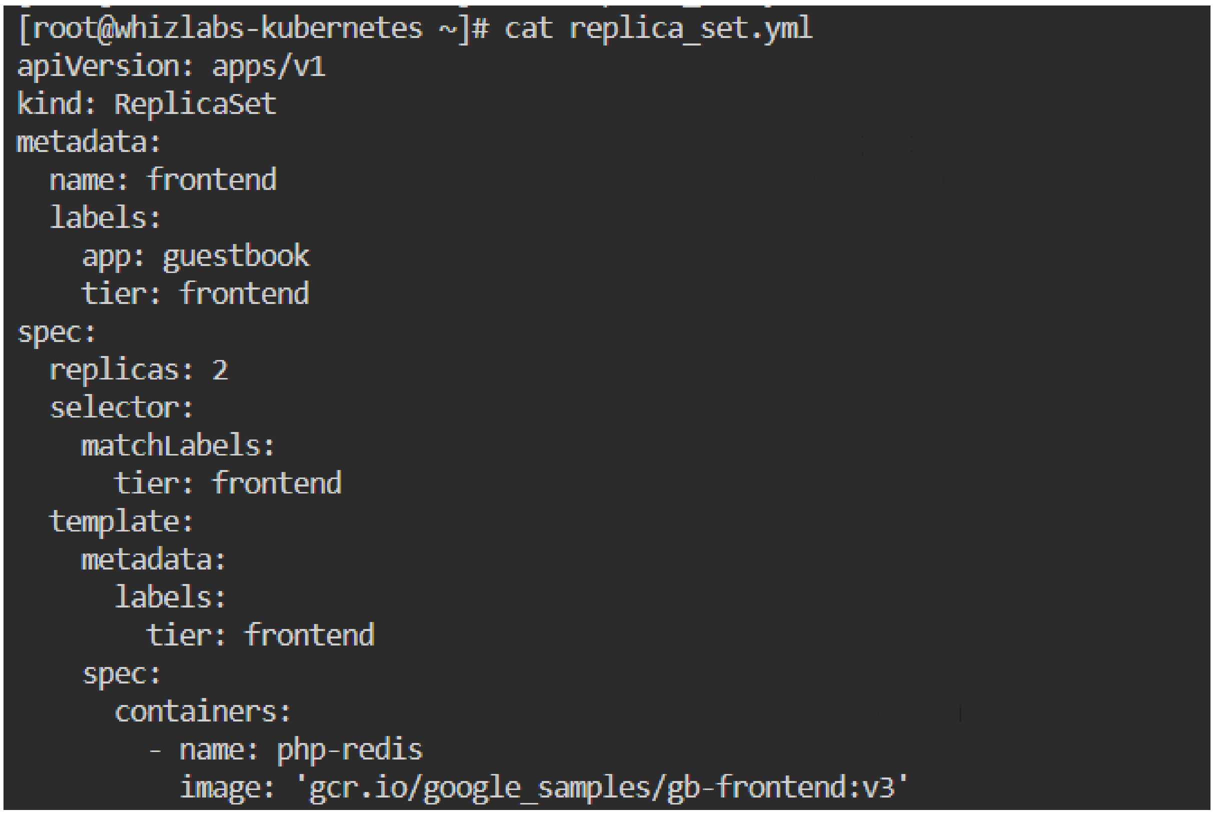
We can use the command-line and YAML approach to create ReplicaSet and both approaches give the same results. However, the YAML approach is more used and recommended as the declarative approach gives more flexibility and we can maintain the whole configuration of the resource in a single file and maintain versioning which can be helpful at the time of rollback to previous versions.

The main attributes in the replica set are labels because the whole program works to match the labels on the underlying resources. Take note of the kind and labels keywords in the declarative file below. Here, we've attached multiple tags to the pods based on the environment and provide a unique identity to the resource.

ReplicaSet using YAML approach:

1. Create a declarative YAML file using the below command.

```
vim replica_set.yml
```



```
[root@whizlabs-kubernetes ~]# cat replica_set.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: php-redis
          image: 'gcr.io/google_samples/gb-frontend:v3'
```

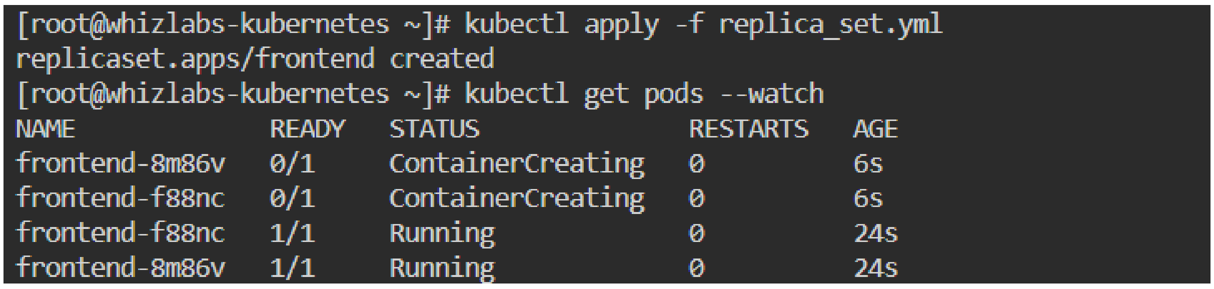
2. Now paste the YAML file below in the editor. The pod label as a specification to indicate the rs to watch the pods matching the set-based rule and maintain uptime for the same. For creating the YAML code, kind would be "ReplicaSet"

and apiVersion would be "apps/v1" that indicates ``rs`` has been released in a later part than RC

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: 'gcr.io/google_samples/gb-frontend:v3'
```

2. Having this file completed, run the below command to apply and create the ReplicaSet as mentioned:

```
kubectl apply -f replica_set.yml
```



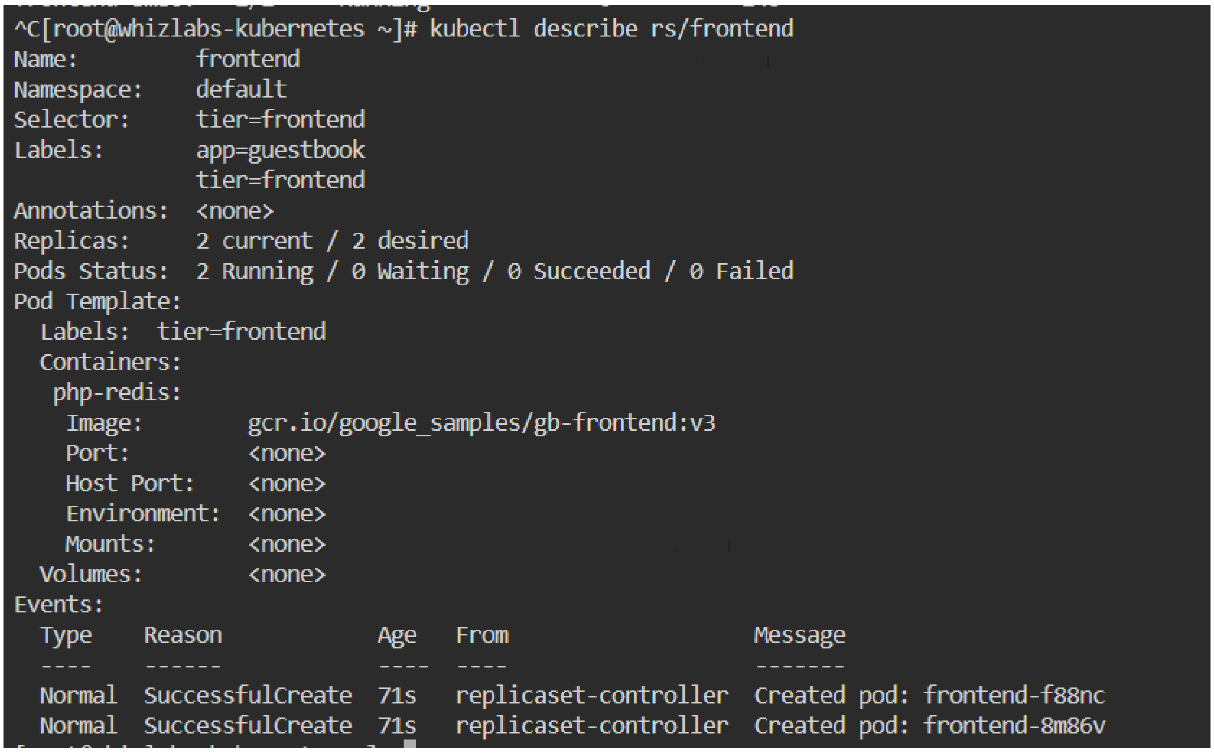
```
[root@whizlabs-kubernetes ~]# kubectl apply -f replica_set.yml
replicaset.apps/frontend created
[root@whizlabs-kubernetes ~]# kubectl get pods --watch
NAME              READY   STATUS              RESTARTS   AGE
frontend-8m86v    0/1     ContainerCreating   0          6s
frontend-f88nc    0/1     ContainerCreating   0          6s
frontend-f88nc    1/1     Running             0          24s
frontend-8m86v    1/1     Running             0          24s
```

So, this is one of the ways to create an independent ReplicaSet and ensure the uptime for the application pods. The watch command will stream the whole container deployment process and we can see the desired number of pods up and running after a few seconds. First-time execution may take a little more time as the container images are being pulled from the hub.

Task 4: Describe the ReplicaSet:

The task of creating the ReplicaSet is done. Now is the time to fetch the details about it and understand it in a better format. Use the describe command to get all the details about ``rs``.

```
kubectl describe replicaset/frontend
```



```
^C[root@whizlabs-kubernetes ~]# kubectl describe rs/frontend
Name:         frontend
Namespace:    default
Selector:     tier=frontend
Labels:       app=guestbook
              tier=frontend
Annotations:  <none>
Replicas:     2 current / 2 desired
Pods Status:  2 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  tier=frontend
  Containers:
   php-redis:
    Image:        gcr.io/google_samples/gb-frontend:v3
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  71s   replicaset-controller  Created pod: frontend-f88nc
  Normal  SuccessfulCreate  71s   replicaset-controller  Created pod: frontend-8m86v
```

In ReplicaSet, pod status and desired number of pods are the key outputs to watch out for. From these details, we can get a whole idea about the process. Also, the events section in the bottom states all the details about the pod and throws the error in case of deployment failure. With the describe command, we can see the actual running pods and desired number of pods and take needful actions if there is some fault.
Bare pods are not a recommended practice if you're thinking of an environment at scale. Pods being watched by a program like ``rs`` gives assurance of minimal downtime.

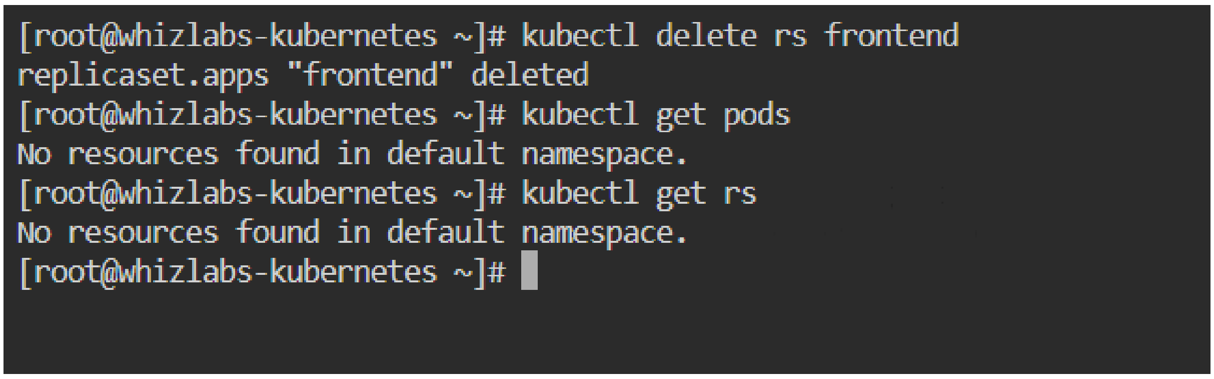
Task 5 : Delete the resources

We learned how to create and describe rs resources and now we'll see how to delete the created rs using the command line.

1. **Delete the whole ReplicaSet:** This approach is used when we want to destroy the rs environment completely. Here, the rs and underlying pods will be deleted and further uptime won't be maintained as the watching program rs is not on the head. It is recommended to take backup of the container data befor running this command as it will erase the whole temporary data.

```
kubectl delete replicaset frontend
```

A program named garbage collector will clean up the environment. Only the data stored in the persistent storage will remain present. Rest everything will be wiped out.



```
[root@whizlabs-kubernetes ~]# kubectl delete rs frontend
replicaset.apps "frontend" deleted
[root@whizlabs-kubernetes ~]# kubectl get pods
No resources found in default namespace.
[root@whizlabs-kubernetes ~]# kubectl get rs
No resources found in default namespace.
[root@whizlabs-kubernetes ~]#
```

1. **Reference: Delete only the ReplicaSet or isolate the pods from rs:** This approach is used when we only want to detach the rs from the underlying pods. After this, the unlinked pods will behave like bare pods and will no longer have the monitoring program to maintain the uptime.

```
kubectl delete replicaset frontend --cascade=orphan
```

2. Changing labels can also be useful to move the pods inside/outside the boundary of the set-based selector. Set-based selector performs pod query operations based on various sets of conditions and this resource becomes handy in larger environments as the pod tagges need to be unique and specific enough to identity as well as moderate under the service of replication.

Completion and Conclusion

1. You have successfully created a ReplicaSet resource inside a single-node Kubernetes cluster.
2. You have successfully logged the Whiz terminal via SSH into the cluster and performed the mentioned operations.
3. You have successfully described and deleted the created resources through it.
4. You got an idea about the difference between RC and RS