Schedule and execute tasks with Jobs and CronJobs

Point Clés du lab

Task Details

1. Launching Lab Environment.
2. SSH into the EC2 instance through the Whiz terminal
3. Create the manifest file for the sample DaemonSet
4. Verify the DaemoSet resource using Kubectl command
5. Delete the provisioned resources.

Lab Steps

Task 1: Launching Lab Environment
Launch the lab environment by clicking the button.

Please wait until the lab environment is provisioned, which will take about 2 minutes. This environment includes a single-node Kubernetes cluster. The unique Username, Public IP address, and Password will be displayed in the right corner of the screen.

Along with the EC2 instance, the Whizlabs terminal is also being provisioned.

Note: If you face any issues, please refer to FAQs and Troubleshooting for Labs.

Task 2: SSH into the EC2 Instance via the Whiz Terminal:
Copy the IP address and Password for SSH to the EC2 instance configured with the Kubernetes cluster.

Run the following command to SSH into the system:

```
bashCopy code
ssh username@ip_address
```

Replace the unique username and IP address with the details of your own lab environment.

Type "yes" if prompted for SSH permission for the first time. Then, insert the password as it is when prompted.

Now switch the user to root with the command below:

```
bashCopy code
sudo su root
```

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: Create the Manifest File for the Sample DaemonSet

In this step, we will create a manifest file for a sample Prometheus application (monitoring tool) and launch it as a DaemonSet. In this lab, we are implementing the concept over a multi-node Kubernetes cluster. You can verify the same by running the `kubectl get nodes` command. You will see two nodes in the output of the above command. DaemonSet is included under the `apps/v1` apiVersion. In the YAML declarative file, we need to mention `daemonset` as the kind of the resource. Also, in the template block, we specify the labels and selector for internal management of the program to ensure that one copy of the pod with the mentioned label is running on each node (or specified nodes).

Here we will use the sample Prometheus node exporter container image and launch it as a single pod. Make sure that the labels and selectors are the same; otherwise, the selector won't be able to keep track of the daemon.

Using the YAML approach:

Create a YAML file for the deployment of the sample Prometheus node exporter DaemonSet using the command below:

```
bashCopy code
Vim daemonset.yml
```

Copy the following code into the YML file, which contains the declaration of the DaemonSet resource with the required blocks explained above.

```
yamlCopy code
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: prometheus-daemonset
spec:
  selector:
    matchLabels:
      tier: monitoring
      name: prometheus-exporter
  template:
    metadata:
      labels:
        tier: monitoring
        name: prometheus-exporter
    spec:
      containers:
      - name: prometheus
        image: prom/node-exporter
        ports:
        - containerPort: 80
```

The next step is to apply the YAML file to launch the deployment using the command below:

```
bashCopy code
kubectl apply -f daemonset.yml
```

Task 4: Verify the DaemonSet Resource Using the Kubectl Command

Let's verify the deployed DaemonSet with the command below:

```
bashCopy code
kubectl get daemonset
```

Here, we can see the running and desired count along with the health status of the launched pods from the DaemonSet.

To get more insights from the resource, run the `describe` command and note a few more details about the launched resource.

```
bashCopy code
kubectl describe daemonset
```

In the output, we can see that the labels and selectors match with the labels of the pods.

With the DaemonSet, we can control the scheduling of the daemon pod and restrict the provisioning of the pods to selective nodes only. In the output of the above command, we can see that the Node Selector is set to none, which means all nodes will get one copy of the Daemon. By mentioning specific nodes with this feature, we can limit the deployment of the daemon pods.

Task 5: How to Delete the DaemonSet?
This will delete the DaemonSet configuration and clean up all the provisioned daemon pods in all the nodes.

```
bashCopy code
kubectl delete daemonset prometheus-daemonset
```

One thing to note is that we can restrict the deletion of the daemon pods by setting up taints and tolerations for specific nodes.

Also, directly deleting the pods will only result in the deployment of the same pods again from the DaemonSet program, as it is tracking the pods with label selectors and, on not matching the desired pod count, it will compensate with new pods in the nodes.

Completion and Conclusion

You have successfully created a DaemonSet resource in a multi-node Kubernetes cluster.

You have successfully observed the resource configuration.

You have deleted the configured DaemonSet and removed the daemon pods from all nodes.

End Lab
Exit from the terminal SSH session.

You have successfully completed the lab.

Once you have completed the steps, click the button from your Whizlabs dashboard.