How to add a custom pod scheduler in PMK?

The basic steps of adding multiple schedulers can be found in kubernetes documentation, however few more modifications as documented below are needed to do it successfully in PMK.

Package the Scheduler:

- Need to package the scheduler binary into a container image. Thus first clone the Kubernetes source code from GitHub and build the source.

```
❯ git clone --branch=v1.2x.y
https://github.com/kubernetes/kubernetes.git
❯ cd kubernetes
❯ make
```

Here one has to specify the exact branch to clone as per the PMK version in which the scheduler needs to be added, like --branch=v1.24.3

- Create a container image containing the kube-scheduler binary. Below is the *Dockerfile* to build that image:

```
FROM busybox
ADD ./_output/local/bin/linux/amd64/kube-scheduler
/usr/local/bin/kube-scheduler
```

- Build the image and push it to your registry.

```
❯ docker build -t <repo-url>/<path or name>:<version> .
❯ docker push <repo-url>/<path or name>:<version>
```

Define the kubernetes resources for deploying the scheduler:

Now that we have the scheduler in a container image, create kubernetes resources such that the scheduler can run as a pod in the Kubernetes cluster. Create a file *my-scheduler.yaml* with below contents;

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-scheduler
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: my-scheduler-as-kube-scheduler
subjects:
- kind: ServiceAccount
  name: my-scheduler
  namespace: kube-system
roleRef:
  kind: ClusterRole
  name: system:kube-scheduler
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: my-scheduler-as-volume-scheduler
subjects:
- kind: ServiceAccount
  name: my-scheduler
  namespace: kube-system
roleRef:
  kind: ClusterRole
  name: system:volume-scheduler
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: my-scheduler-config
  namespace: kube-system
data:
  my-scheduler.yaml: |
      apiVersion: kubescheduler.config.k8s.io/v1beta1
      kind: KubeSchedulerConfiguration
      clientConnection:
      kubeconfig:
/srv/kubernetes/kubeconfigs/kubeconfig-for-my-scheduler.yaml
      profiles:
      - schedulerName: my-scheduler
      plugins:
            score:
            disabled:
            - name: NodeResourcesLeastAllocated
            enabled:
            - name: NodeResourcesLeastAllocated
            weight: 3
      leaderElection:
      leaderElect: false
      healthzBindAddress: 0.0.0.0:10252
      metricsBindAddress: 0.0.0.0:10252
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: kubeconfig-for-my-scheduler
  namespace: kube-system
data:
  kubeconfig-for-my-scheduler.yaml: |
      apiVersion: v1
      kind: Config
      users:
      - name: system:kube-scheduler
      user:
      client-certificate-data: LS0tLS1CRUdJTXXXXXXFURS0tLS0tCg==
      client-key-data: LS0tLS1CRUdJTXXXXXFIEtFWS0tLS0tCg==
```

```
      clusters:
      - name: local
      cluster:
      server: https://localhost:443
      certificate-authority-data: LS0tLS1CRUdJTXXXXXFLS0tLS0K
      contexts:
      - context:
      cluster: local
      user: system:kube-scheduler
      name: service-account-context
      current-context: service-account-context
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
      component: scheduler
      tier: control-plane
  name: my-scheduler
  namespace: kube-system
spec:
  selector:
      matchLabels:
      component: scheduler
      tier: control-plane
  replicas: 1
  template:
      metadata:
      labels:
      component: scheduler
      tier: control-plane
      version: second
      spec:
      serviceAccountName: my-scheduler
      containers:
      - command:
      - /usr/local/bin/kube-scheduler
      - --secure-port=10260
      - --leader-elect=false
```

```
    - --v=2
    - --scheduler-name=my-scheduler
    - --config=/etc/kubernetes/my-scheduler/my-scheduler.yaml
    image: <repo-url>/<path or name>:<version>
    livenessProbe:
          httpGet:
          path: /healthz
          port: 10260
          scheme: HTTPS
          initialDelaySeconds: 15
    name: kube-second-scheduler
    readinessProbe:
          httpGet:
          path: /healthz
          port: 10260
          scheme: HTTPS
    resources:
          requests:
          cpu: '0.1'
    securityContext:
          privileged: false
    volumeMounts:
          - name: config-file
          mountPath: /etc/kubernetes/my-scheduler
          - name: kubeconfig-file
          mountPath: /srv/kubernetes/kubeconfigs
          readOnly: true
    nodeSelector:
    node-role.kubernetes.io/master: ""
    hostNetwork: true
    hostPID: false
    volumes:
    - name: config-file
          configMap:
          name: my-scheduler-config
    - name: kubeconfig-file
          configMap:
          name: kubeconfig-for-my-scheduler
```

Important points to note in the above manifest file:

- In the first configmap, a resource of kind KubeSchedulerConfiguration is being created to add the required custom configuration to the kubernetes scheduler. The apiVersion of it is currently shown as kubescheduler.config.k8s.io/v1beta1 , however the same may vary as per the k8s version on which this is being deployed. Refer to this k8s doc for the exact version corresponding to your cluster version. This configmap can also be created separately using following command: `kubectl create configmap my-scheduler-config --from-file=my-scheduler.yaml --namespace=kube-system`
- The custom scheduler configuration added under *profiles* section of KubeSchedulerConfiguration may vary as per requirement. There are lots of parameters to tweak/add/change, etc and one may refer to this k8s doc for the available options corresponding to your cluster version.
- The second configmap is there to load kubeconfig for the custom scheduler to communicate. We can use the same existing kubeconfig of default scheduler found on the master nodes of the PMK cluster at location */etc/pf9/kube.d/kubeconfigs/kube-scheduler.yaml.* This configmap can also be created separately using following command: `kubectl create configmap kubeconfig-for-my-scheduler --from-file=kubeconfig-for-my-scheduler.yaml --namespace=kube-system`
- In the deployment resource, under container config there are a few kube-scheduler command options, these options may change as per the k8s version, one can refer to this k8s doc, to know the available/valid options corresponding to your cluster version.
- The default scheduler in PMK runs as a container within *k8s-master-<master_node-IP>* pod which is a static pod that runs on master node(s) and has `hostNetwork: true`.

The above custom scheduler deployment also has *nodeSelector* set to master node(s) and along with that it has `hostNetwork: true` as well, with this it creates port conflict with the default scheduler process on the master nodes and thus this custom scheduler is configured to run on different ports i:e 10260 for scheduler process and 10252 for healthz and metrics.(kube-scheduler default ports are 10259 and 10251).

- Add correct docker `image:` name that was created and pushed earlier.

Update RBAC permissions:

- With RBAC enabled on the cluster, you must update the system:kube-scheduler cluster role. Add your scheduler name to the *resourceNames* of the rule applied for endpoints and leases resources and also add appropriate permissions for the *apiGroups* “” to access resource configmaps.

```
❯ kubectl edit clusterrole system:kube-scheduler -n kube-system

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:kube-scheduler
rules:
[...]
[...]
- apiGroups:
  - coordination.k8s.io
  resourceNames:
  - kube-scheduler
  - my-scheduler  <----
  resources:
  - leases
  verbs:
  - get
  - update
  - list
  - watch

- apiGroups:
  - ""
  resourceNames:
  - kube-scheduler
  - my-scheduler   <----
  resources:
```

```
  - endpoints
  verbs:
  - get
  - update
  - list
  - watch

- apiGroups:       // add this apiGroups config.
  - ""
  resources:
  - configmaps
  verbs:
  - get
  - list
  - watch
```

- Update the system::extension-apiserver-authentication-reader *rolebinding* to add the *serviceaccount* corresponding to the custom scheduler.

```
❯ kubectl edit rolebindings
system::extension-apiserver-authentication-reader -n kube-system

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system::extension-apiserver-authentication-reader
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: extension-apiserver-authentication-reader
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
```

```
  name: system:kube-controller-manager
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: system:kube-scheduler
- kind: ServiceAccount            // add this section.
  name: my-scheduler
  namespace: kube-system
```

Create scheduler resources:

- Finally create all the resources related to the scheduler using below command;

```
❯ kubectl create -f my-scheduler.yaml
```

- Once everything goes fine, *my-scheduler* pod can be seen running under the kube-system namespace. Below we have shown the default *k8s-master-<master_node-IP>* pod and the newly created *my-scheduler* pod running on the master node.

```
❯ kubectl get pods -owide -n kube-system | egrep
"my-scheduler|k8s-master"
k8s-master-10.128.145.27                    3/3     Running     0
17d     10.128.145.27    10.128.145.27    <none>           <none>
my-scheduler-84cc776d45-wrcrt               1/1     Running     0
2d22h   10.128.145.27    10.128.145.27    <none>           <none>
```

Checking the custom scheduler logs and configuration:

The pod logs show the configuration and all the parameters/options with which the scheduler has been loaded up. Below we have shown a snippet of few options.

```
❯ kubectl logs my-scheduler-84cc776d45-wrcrt -n kube-system
```

```
I1014 15:56:42.843048       1 flags.go:59] FLAG:
--add-dir-header="false"
I1014 15:56:42.843636       1 flags.go:59] FLAG: --address="0.0.0.0"
I1014 15:56:42.843656       1 flags.go:59] FLAG:
--algorithm-provider=""
[...]
I1014 15:56:42.844511       1 flags.go:59] FLAG: --port="10251"
I1014 15:56:42.843872       1 flags.go:59] FLAG:
--config="/etc/kubernetes/my-scheduler/my-scheduler.yaml"
[...]
I1014 15:56:42.844681       1 flags.go:59] FLAG:
--scheduler-name="my-scheduler"
I1014 15:56:42.844700       1 flags.go:59] FLAG:
--secure-port="10260"
[...]
I1014 15:56:42.844994       1 flags.go:59] FLAG: --v="2"
[...]
I1014 15:56:44.186694       1 configfile.go:72] Using component
config:
apiVersion: kubescheduler.config.k8s.io/v1beta1
clientConnection:
  acceptContentTypes: ""
  burst: 100
  contentType: application/vnd.kubernetes.protobuf
  kubeconfig:
/srv/kubernetes/kubeconfigs/kubeconfig-for-my-scheduler.yaml
[...]
healthzBindAddress: 0.0.0.0:10252
kind: KubeSchedulerConfiguration
leaderElection:
  leaderElect: false
metricsBindAddress: 0.0.0.0:10252
profiles:
profiles:
- plugins:
    [...]
    score:
      enabled:
    [...]
```

```
      - name: NodeResourcesLeastAllocated   <--- Config that was
added via 'KubeSchedulerConfiguration' configmap.
        weight: 3                           <---
  schedulerName: my-scheduler
```

Deploy applications using custom scheduler:

- Below is an apache application deployment using the newly created *my-scheduler* kubernetes scheduler. We can see pods successfully scheduled using our custom scheduler.

```
❯ kubectl get deploy testapache -n apache
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
testapache   2/2     2            2           4d22h
❯
❯ kubectl get deploy testapache -n apache -o yaml | grep -i scheduler
      schedulerName: my-scheduler
❯
❯ kubectl get pods -owide -n apache -l
app.kubernetes.io/name=testapache
NAME                         READY   STATUS    RESTARTS   AGE     IP
NODE             NOMINATED NODE   READINESS GATES
testapache-649986566d-29sjw   1/1     Running   0          4h19m
10.20.120.132   10.128.145.27    <none>           <none>
testapache-649986566d-8bdjp   1/1     Running   0          4h18m
10.20.168.72    10.128.145.157   <none>           <none>
❯
```

- Below are the '*my-scheduler*' scheduler logs corresponding to above shown pods;

```
❯ kubectl logs my-scheduler-84cc776d45-wrcrt -n kube-system |less
```

```
[...]
I1017 10:59:24.725039       1 scheduler.go:604] "Successfully bound
pod to node" pod="apache/testapache-649986566d-8bdjp"
node="10.128.145.157" evaluatedNodes=4 feasibleNodes=4
I1017 10:58:44.476134       1 scheduler.go:604] "Successfully bound
pod to node" pod="apache/testapache-649986566d-29sjw"
node="10.128.145.27" evaluatedNodes=4 feasibleNodes=4
```

Thus we have seen how to create a custom scheduler and further deploy and schedule application pods using that custom scheduler.

–=< END >=–