Resource Management

Introduction:

Each node has a specific CPU, memory and disk space available. Every pod consumes some resource of the node. If node does not have the sufficient resource, then pods will be placed in another node. If there is no sufficient resource available in any node, the Kubernetes holds back scheduling the pod and pod will be in pending status with reason insufficient CPU.

The kube-scheduler uses the request information to decide which node to place the Pod on. When you specify a resource *limit* for a container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set. The kubelet also reserves at least the *request* amount of that system resource specifically for that container to use

When the kubelet starts a container as part of a Pod, the kubelet passes that container's requests and limits for memory and CPU to the container runtime. the container runtime typically configures kernel cgroups that apply and enforce the limits you defined.

Objectives:

1. Check the resources used by the pods in the cluster
2. Schedule pods with resource request and limit

1. Check the resources used by the pods in the cluster

We are having 3 nodes in this cluster, one master and two worker nodes. Master nodes is having the management pods deployed on it whereas other pods will be deployed on worker nodes.

Let's check the current resources available on all the nodes using the describe command.

```
kubectl get nodes
```

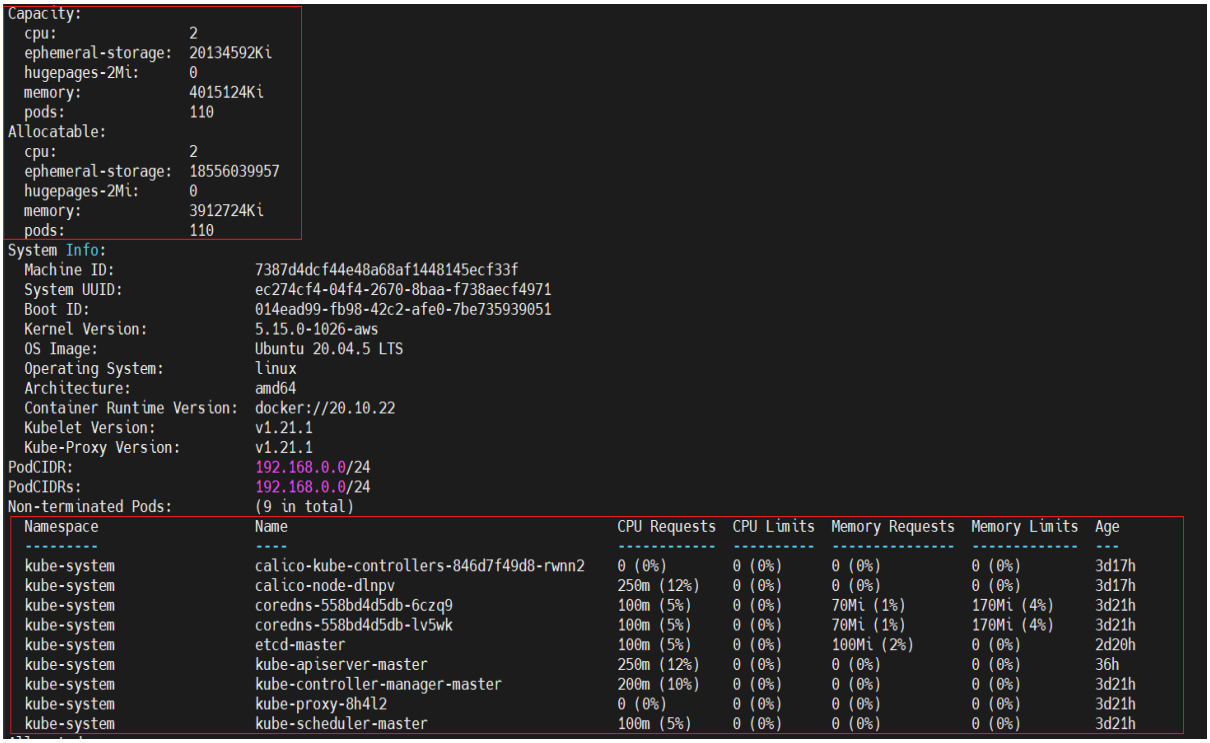
```
root@master:~# kubectl get nodes
NAME      STATUS   ROLES                  AGE     VERSION
master    Ready    control-plane,master   3d21h   v1.21.1
worker1   Ready    <none>                 3d20h   v1.21.1
worker2   Ready    <none>                 3d20h   v1.21.1
```

We will use the describe command to check the resources of available on the nodes.

Master Node:

```
kubectl describe node master
```

Below we can see the resources available on **Master node** and resources consumed by pods.



```
Capacity:
  cpu:                2
  ephemeral-storage:  20134592Ki
  hugepages-2Mi:      0
  memory:             4015124Ki
  pods:               110
Allocatable:
  cpu:                2
  ephemeral-storage:  18556039957
  hugepages-2Mi:      0
  memory:             3912724Ki
  pods:               110
System Info:
  Machine ID:                 7387d4dcf44e48a68af1448145ecf33f
  System UUID:                ec274cf4-04f4-2670-8baa-f738aecf4971
  Boot ID:                    014ead99-fb98-42c2-afe0-7be735939051
  Kernel Version:             5.15.0-1026-aws
  OS Image:                   Ubuntu 20.04.5 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://20.10.22
  Kubelet Version:            v1.21.1
  Kube-Proxy Version:         v1.21.1
PodCIDR:                      192.168.0.0/24
PodCIDRs:                     192.168.0.0/24
Non-terminated Pods:          (9 in total)
  Namespace                   Name                                       CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                       ------------  ----------  ---------------  -------------  ---
  kube-system                 calico-kube-controllers-846d7f49d8-rwnn2   0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d17h
  kube-system                 calico-node-dlnpv                          250m (12%)    0 (0%)      0 (0%)           0 (0%)         3d17h
  kube-system                 coredns-558bd4d5db-6czq9                   100m (5%)     0 (0%)      70Mi (1%)        170Mi (4%)     3d21h
  kube-system                 coredns-558bd4d5db-lv5wk                   100m (5%)     0 (0%)      70Mi (1%)        170Mi (4%)     3d21h
  kube-system                 etcd-master                                100m (5%)     0 (0%)      100Mi (2%)       0 (0%)         2d20h
  kube-system                 kube-apiserver-master                      250m (12%)    0 (0%)      0 (0%)           0 (0%)         36h
  kube-system                 kube-controller-manager-master             200m (10%)    0 (0%)      0 (0%)           0 (0%)         3d21h
  kube-system                 kube-proxy-8h4l2                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d21h
  kube-system                 kube-scheduler-master                      100m (5%)     0 (0%)      0 (0%)           0 (0%)         3d21h
```

Worker1 Node:

```
kubectl describe node worker1
```

Below we can see the resources available on **worker1 node** and resources consumed by pods.

```
Capacity:
  cpu:                1
  ephemeral-storage:  20134592Ki
  hugepages-2Mi:      0
  memory:             989428Ki
  pods:               110
Allocatable:
  cpu:                1
  ephemeral-storage:  18556039957
  hugepages-2Mi:      0
  memory:             887028Ki
  pods:               110
System Info:
  Machine ID:                 cb6a7c5c75c24a6e87cd7dd321cfbe7d
  System UUID:                ec29ba2b-4581-9c30-b1c7-01984e5f9651
  Boot ID:                    9c993fbe-a5e0-4fbe-b680-4b7586856f90
  Kernel Version:             5.15.0-1026-aws
  OS Image:                   Ubuntu 20.04.5 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://20.10.22
  Kubelet Version:            v1.21.1
  Kube-Proxy Version:         v1.21.1
PodCIDR:                      192.168.2.0/24
PodCIDRs:                     192.168.2.0/24
Non-terminated Pods:          (11 in total)
  Namespace                   Name                                CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                ------------  ----------  ---------------  -------------  ---
  default                     dev-deploy-57b86d5ccc-kxspz         0 (0%)        0 (0%)      0 (0%)           0 (0%)         13h
  default                     dev-deploy-57b86d5ccc-zxvq6         0 (0%)        0 (0%)      0 (0%)           0 (0%)         13h
  default                     prod-deploy-6d56cd4795-kmb7p        0 (0%)        0 (0%)      0 (0%)           0 (0%)         12h
  default                     prod-deploy-6d56cd4795-zhf5v        0 (0%)        0 (0%)      0 (0%)           0 (0%)         12h
  development                 dev1                                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  development                 dev2                                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  istio-system                web-pod                             0 (0%)        0 (0%)      0 (0%)           0 (0%)         46h
  kube-system                 calico-node-7zc5t                   250m (25%)    0 (0%)      0 (0%)           0 (0%)         3d17h
  kube-system                 kube-proxy-vjfv2                    0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d21h
  testing                     test                                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  yavin                       pod1                                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d19h
```

Worker2 Node:

kubectl describe node worker2

Below we can see the resources available on **worker2 node** and resources consumed by pods.

```
Capacity:
  cpu:                1
  ephemeral-storage:  20134592Ki
  hugepages-2Mi:      0
  memory:             989424Ki
  pods:               110
Allocatable:
  cpu:                1
  ephemeral-storage:  18556039957
  hugepages-2Mi:      0
  memory:             887024Ki
  pods:               110
System Info:
  Machine ID:                 d5ff271fa013490c94f79bc77cd503c2
  System UUID:                ec25b099-de70-191a-1a66-1b19333e33c3
  Boot ID:                    4d4773c6-0d97-4781-b2de-7aa01cadc2c6
  Kernel Version:             5.15.0-1027-aws
  OS Image:                   Ubuntu 20.04.5 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://20.10.22
  Kubelet Version:            v1.21.1
  Kube-Proxy Version:         v1.21.1
PodCIDR:                      192.168.1.0/24
PodCIDRs:                     192.168.1.0/24
Non-terminated Pods:          (10 in total)
  Namespace                   Name                                CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                ------------  ----------  ---------------  -------------  ---
  default                     dev-deploy-57b86d5ccc-cbhff         0 (0%)        0 (0%)      0 (0%)           0 (0%)         13h
  default                     prod-deploy-6d56cd4795-8jq5m        0 (0%)        0 (0%)      0 (0%)           0 (0%)         12h
  dev-team                    dev-team                            0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d19h
  dev-team                    dev-team1                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d19h
  dev-team                    user-service                        0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  dev                         dev-pod                             0 (0%)        0 (0%)      0 (0%)           0 (0%)         44h
  kube-system                 calico-node-bsbmx                   250m (25%)    0 (0%)      0 (0%)           0 (0%)         3d17h
  kube-system                 kube-proxy-bkrxq                    0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d21h
  monitoring                  test-pod                            0 (0%)        0 (0%)      0 (0%)           0 (0%)         45h
  yavin                       pod2                                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d19h
```

2. **Schedule pods with resource request and limit:**

Now we are going to set a request and limit to our pods which will be deployed either on worker1 or worker2 node. Let's use the below given yaml code to deploy our pods.

```
# vi resource-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: rsu-pod1
  labels:
    env: dev
spec:
  containers:
    - name: container1
      image: httpd:2.4
      resources:
        requests:
          cpu: "0.10"
          memory: "100M"
        limits:
          cpu: "1"
          memory: "500M"
---
apiVersion: v1
kind: Pod
metadata:
  name: rsu-pod2
  labels:
    env: dev
spec:
  containers:
    - name: container1
      image: httpd:2.4
      resources:
        requests:
          cpu: "4"
          memory: "100M"
        limits:
          cpu: "4"
          memory: "500M"
```

We are going to deploy 2 pods rsu-pod1 and rsu-pod2 with requests and limits configured.

As per our node's description, we do not have enough resources for rsu-pod2. Let's find out what will happen rsu-pod2 after applying the configuration.

kubectl apply -f resource-pod.yaml

```
root@master:~# kubectl apply -f resource-pod.yaml
pod/rsu-pod1 created
pod/rsu-pod2 created
root@master:~#
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
rsu-pod1   1/1     Running   0          6s
rsu-pod2   0/1     Pending   0          6s
root@master:~# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP               NODE      NOMINATED NODE   READINESS GATES
rsu-pod1   1/1     Running   0          17s   192.168.235.186   worker1   <none>           <none>
rsu-pod2   0/1     Pending   0          17s   <none>           <none>    <none>           <none>
root@master:~#
```

From the above output, we can see that **rsu-pod1** has been scheduled on **worker1** node whereas **rsu-pod2** is still in pending state.

Let's check for rsu-pod1 first.

kubectl describe node worker1

```
Capacity:
  cpu:                1
  ephemeral-storage:  20134592Ki
  hugepages-2Mi:      0
  memory:             989428Ki
  pods:               110
Allocatable:
  cpu:                1
  ephemeral-storage:  18556039957
  hugepages-2Mi:      0
  memory:             887028Ki
  pods:               110
System Info:
  Machine ID:                 cb6a7c5c75c24a6e87cd7dd321cfbe7d
  System UUID:                ec29ba2b-4581-9c30-b1c7-01984e5f9651
  Boot ID:                    9c993fbe-a5e0-4fbe-b680-4b7586856f90
  Kernel Version:             5.15.0-1026-aws
  OS Image:                   Ubuntu 20.04.5 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://20.10.22
  Kubelet Version:            v1.21.1
  Kube-Proxy Version:         v1.21.1
PodCIDR:                      192.168.2.0/24
PodCIDRs:                     192.168.2.0/24
Non-terminated Pods:          (8 in total)
  Namespace                   Name                CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                ------------  ----------  ---------------  -------------  ---
  default                     rsu-pod1            100m (10%)    1 (100%)    100M (11%)       500M (55%)     3m5s
  development                 dev1                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  development                 dev2                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  istio-system                web-pod             0 (0%)        0 (0%)      0 (0%)           0 (0%)         46h
  kube-system                 calico-node-7zc5t   250m (25%)    0 (0%)      0 (0%)           0 (0%)         3d17h
  kube-system                 kube-proxy-vjfv2    0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d21h
  testing                     test                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
  yavin                       pod1                0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d20h
```

Above we can see in above output that rsu-pod1 has received the requested CPU and memory.

Now let’s check rsu-pod2 description why it is still in pending state.

```
root@master:~# kubectl describe pod rsu-pod2
Name:         rsu-pod2
Namespace:    default
Priority:     0
Node:         <none>
Labels:       env=dev
Annotations:  <none>
Status:       Pending
IP:
IPs:          <none>
Containers:
  container1:
    Image:      httpd:2.4
    Port:       <none>
    Host Port:  <none>
    Limits:
      cpu:     4
      memory:  500M
    Requests:
      cpu:        4
      memory:     100M
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-sfccv (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-sfccv:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  46s (x11 over 10m)  default-scheduler  0/3 nodes are available: 1 node(s) had taint {node-role.kubernetes.io/master: }, that the
pod didn't tolerate, 2 Insufficient cpu.
```

In above output, we can see that the pod is in pending state and under events field we can find the reason of this state. The reason is **2 Insufficient CPU** , as we are having 2 worker nodes and both the nodes are not having the resources what the pod is requesting for.

Let’s edit the rsu-pod2 definition file and change the request / limit field.

```
kubectl edit pod rsu-pod2
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"labels":{"env":"dev"},"name":"rsu-pod2","namespace":"default"},"spec":{"containers":[{"im
ge":"httpd:2.4","name":"container1","resources":{"limits":{"cpu":"4","memory":"500M"},"requests":{"cpu":"4","memory":"100M"}}}]}}
  creationTimestamp: "2023-01-12T07:30:25Z"
  labels:
    env: dev
  name: rsu-pod2
  namespace: default
  resourceVersion: "152062"
  uid: ce0d67a6-8da2-4ef2-a9a6-43c5908e7785
spec:
  containers:
  - image: httpd:2.4
    imagePullPolicy: IfNotPresent
    name: container1
    resources:
      limits:
        cpu: "1"
        memory: 500M
      requests:
        cpu: "0.5"
        memory: 100M
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-sfccv
      readOnly: true
```

Once we make the changes, it will store in definition in a temporary file. Apply the definition file using the below command.

```
kubectl replace --force -f /tmp/kubectl-edit-5z1vt.yaml
```

```
root@master:~# kubectl edit pod rsu-pod2
error: pods "rsu-pod2" is invalid
A copy of your changes has been stored to "/tmp/kubectl-edit-5z1vt.yaml"
error: Edit cancelled, no valid changes were saved.
root@master:~#
root@master:~#
root@master:~# kubectl replace --force -f /tmp/kubectl-edit-5z1vt.yaml
pod "rsu-pod2" deleted
pod/rsu-pod2 replaced
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
rsu-pod1   1/1     Running   0          23m
rsu-pod2   1/1     Running   0          106s
root@master:~#
root@master:~# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP               NODE      NOMINATED NODE   READINESS GATES
rsu-pod1   1/1     Running   0          23m     192.168.235.186   worker1   <none>           <none>
rsu-pod2   1/1     Running   0          2m12s   192.168.235.187   worker1   <none>           <none>
root@master:~#
root@master:~#
```

After applying the changes our rsu-pod2 has been scheduled on **worker1** node as nodes are fulfilling the resources it is asking for.