

Agenda

- Commands
- NameSpaces
- Resources Limits
- Taint & Tolerations
- Node Selectors
- Node Affinity

COMMANDS & ARGUMENTS

```
▶ docker run ubuntu [COMMAND]
```

```
▶ docker run ubuntu sleep 5
```



FROM Ubuntu

CMD sleep 5

CMD command param1

CMD ["command", "param1"]

CMD sleep 5

CMD ["sleep", "5"]



CMD ["sleep 5"]



```
▶ docker build -t ubuntu-sleeper .
```

```
▶ docker run ubuntu-sleeper
```



FROM Ubuntu

CMD sleep 5

```
▶ docker run ubuntu-sleeper sleep 10
```

Command at Startup: sleep 10

FROM Ubuntu

ENTRYPOINT ["sleep"]

```
▶ docker run ubuntu-sleeper 10
```

Command at Startup: sleep 10

```
▶ docker run ubuntu-sleeper
sleep: missing operand
Try 'sleep --help' for more information.
```

Command at Startup: sleep

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

```
▶ docker run ubuntu-sleeper
```

```
sleep: missing operand  
Try 'sleep --help' for more information.
```

Command at Startup: sleep 5

```
▶ docker run ubuntu-sleeper 10
```

Command at Startup: sleep 10

```
▶ docker run --entrypoint sleep2.0 ubuntu-sleeper 10
```

Command at Startup: sleep2.0 10

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name: ubuntu-sleeper
      image: ubuntu-sleeper
      args: ["10"]
```

▶ `kubectl create -f pod-definition.yml`

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

```
▶ docker run --name ubuntu-sleeper \  
  --entrypoint sleep2.0 \  
  ubuntu-sleeper 10
```

pod-definition.yml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: ubuntu-sleeper-pod  
spec:  
  containers:  
    - name: ubuntu-sleeper  
      image: ubuntu-sleeper  
      command: ["sleep2.0"]  
      args: ["10"]
```

```
▶ kubectl create -f pod-definition.yml
```


FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]



pod-definition.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: ubuntu-sleeper-pod
spec:
  containers:
    - name: ubuntu-sleeper
      image: ubuntu-sleeper
      command: ["sleep2.0"]
      args: ["10"]
```

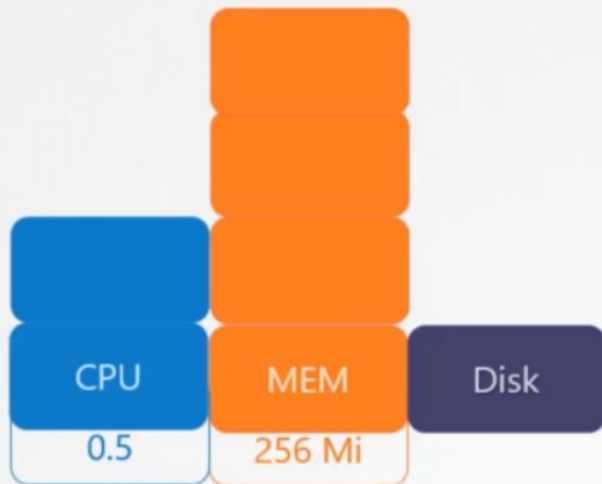
RESOURCE LIMITS

An abstract geometric pattern consisting of thin, light-orange lines connecting small circular dots. The pattern is located in the bottom-left corner of the slide, extending from the left edge towards the center. It features several interconnected triangles and polygons, with dots acting as vertices. The lines and dots are a lighter shade of orange than the background.

| Resource Requests



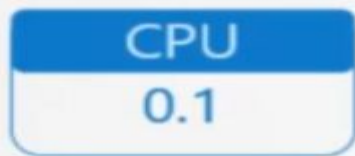
Resource Requests



pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      ports:
        - containerPort: 8080
      resources:
        requests:
          memory: "1Gi"
          cpu: 1
```

| Resource - CPU



| Resource - Memory



1 G (Gigabyte) = 1,000,000,000 bytes

1 M (Megabyte) = 1,000,000 bytes

1 K (Kilobyte) = 1,000 bytes

1 Gi (Gibibyte) = 1,073,741,824 bytes

1 Mi (Mebibyte) = 1,048,576 bytes

1 Ki (Kibibyte) = 1,024 bytes

Resource Limits



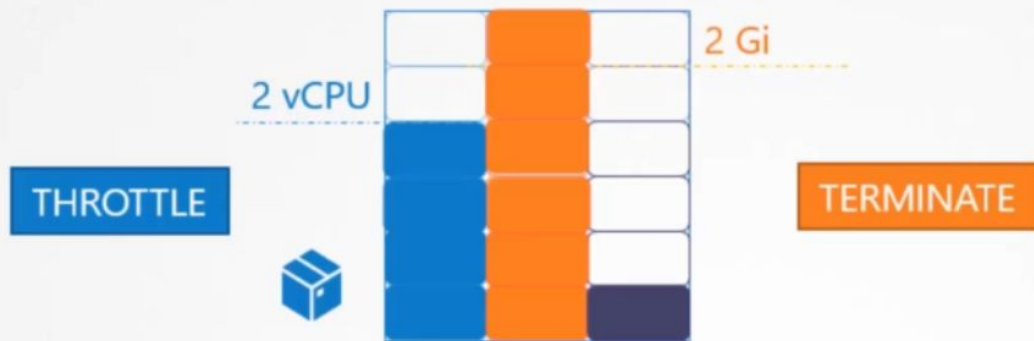
Resource Limits



pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
  labels:
    name: simple-webapp-color
spec:
  containers:
    - name: simple-webapp-color
      image: simple-webapp-color
      ports:
        - containerPort: 8080
      resources:
        requests:
          memory: "1Gi"
          cpu: 1
        limits:
          memory: "2Gi"
          cpu: 2
```


Exceed Limits



Demo: Specify a memory request and a memory limit

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: mem-example
spec:
  containers:
    - name: memory-demo-ctr
      image: polinux/stress
      resources:
        limits:
          memory: "200Mi"
        requests:
          memory: "100Mi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

Demo: Exceed a Container's memory limit

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-2
  namespace: mem-example
spec:
  containers:
    - name: memory-demo-2-ctr
      image: polinux/stress
      resources:
        requests:
          memory: "50Mi"
        limits:
          memory: "100Mi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "250M", "--vm-hang", "1"]
```

Demo: Specify a memory request that is too big for your Nodes

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-3
  namespace: mem-example
spec:
  containers:
    - name: memory-demo-3-ctr
      image: polinux/stress
      resources:
        limits:
          memory: "1000Gi"
        requests:
          memory: "1000Gi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

If you do not specify a memory limit

If you do not specify a memory limit for a Container, one of the following situations applies:

- The Container has no upper bound on the amount of memory it uses. The Container could use all of the memory available on the Node where it is running which in turn could invoke the OOM Killer. Further, in case of an OOM Kill, a container with no resource limits will have a greater chance of being killed.
- The Container is running in a namespace that has a default memory limit, and the Container is automatically assigned the default limit. Cluster administrators can use a [LimitRange](#) to specify a default value for the memory limit.

Demo: Specify a CPU request and a CPU limit

```
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo
  namespace: cpu-example
spec:
  containers:
    - name: cpu-demo-ctr
      image: vish/stress
      resources:
        limits:
          cpu: "1"
        requests:
          cpu: "0.5"
      args:
        - -cpus
        - "2"
```

If you do not specify a CPU limit

If you do not specify a CPU limit for a Container, then one of these situations applies:

- The Container has no upper bound on the CPU resources it can use. The Container could use all of the CPU resources available on the Node where it is running.
- The Container is running in a namespace that has a default CPU limit, and the Container is automatically assigned the default limit. Cluster administrators can use a [LimitRange](#) to specify a default value for the CPU limit.

If you specify a CPU limit but do not specify a CPU request

If you specify a CPU limit for a Container but do not specify a CPU request, Kubernetes automatically assigns a CPU request that matches the limit. Similarly, if a Container specifies its own memory limit, but does not specify a memory request, Kubernetes automatically assigns a memory request that matches the limit.

Limit Range (Memory):

Provides constraints to limit resource consumption per Containers or Pods in a namespace.

```
kubectl create namespace default-mem-example
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-limit-range
spec:
  limits:
  - default:
      memory: 512Mi
    defaultRequest:
      memory: 256Mi
    type: Container
```

```
kubectl apply -f memory-defaults.yaml --namespace=default-mem-example
```

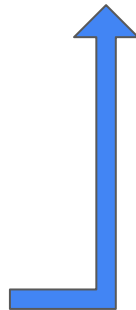

Now if you create a Pod in the default-mem-example namespace, and any container within that Pod does not specify its own values for memory request and memory limit, then the control plane applies default values: a memory request of 256MiB and a memory limit of 512MiB.

```
apiVersion: v1
kind: Pod
metadata:
  name: default-mem-demo
spec:
  containers:
  - name: default-mem-demo-ctr
    image: nginx
```

```
containers:
- image: nginx
  imagePullPolicy: Always
  name: default-mem-demo-ctr
  resources:
    limits:
      memory: 512Mi
    requests:
      memory: 256Mi
```

```
kubectl apply -f memory-defaults-pod.yaml --namespace=default-mem-example
```

```
kubectl get pod default-mem-demo --output=yaml --namespace=default-mem-example
```



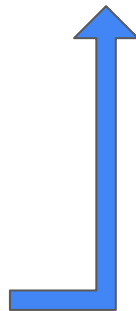
What if you specify a container's limit, but not its request?

```
apiVersion: v1
kind: Pod
metadata:
  name: default-mem-demo-2
spec:
  containers:
  - name: default-mem-demo-2-ctr
    image: nginx
    resources:
      limits:
        memory: "1Gi"
```

```
resources:
  limits:
    memory: 1Gi
  requests:
    memory: 1Gi
```

```
kubectl apply -f memory-defaults-pod-2.yaml --namespace=default-mem-example
```

```
kubectl get pod default-mem-demo-2 --output=yaml --namespace=default-mem-example
```



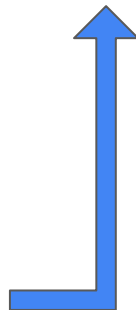
What if you specify a container's request, but not its limit?

```
apiVersion: v1
kind: Pod
metadata:
  name: default-mem-demo-3
spec:
  containers:
  - name: default-mem-demo-3-ctr
    image: nginx
    resources:
      requests:
        memory: "128Mi"
```

```
resources:
  limits:
    memory: 512Mi
  requests:
    memory: 128Mi
```

```
kubectl apply -f memory-defaults-pod-3.yaml --namespace=default-mem-example
```

```
kubectl get pod default-mem-demo-3 --output=yaml --namespace=default-mem-example
```



Limit Range (CPU):

```
kubectl create namespace default-cpu-example
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.5
    type: Container
```

```
kubectl apply -f cpu-defaults.yaml --namespace=default-cpu-example
```

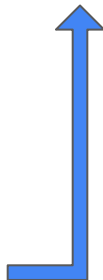
Now if you create a Pod in the default-cpu-example namespace, and any container in that Pod does not specify its own values for CPU request and CPU limit, then the control plane applies default values: a CPU request of 0.5 and a default CPU limit of 1.

```
apiVersion: v1
kind: Pod
metadata:
  name: default-cpu-demo
spec:
  containers:
  - name: default-cpu-demo-ctr
    image: nginx
```

```
containers:
- image: nginx
  imagePullPolicy: Always
  name: default-cpu-demo-ctr
  resources:
    limits:
      cpu: "1"
    requests:
      cpu: 500m
```

```
kubectl apply -f cpu-defaults-pod.yaml --namespace=default-cpu-example
```

```
kubectl get pod default-cpu-demo --output=yaml --namespace=default-cpu-example
```



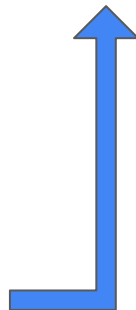
What if you specify a container's limit, but not its request?

```
apiVersion: v1
kind: Pod
metadata:
  name: default-cpu-demo-2
spec:
  containers:
  - name: default-cpu-demo-2-ctr
    image: nginx
    resources:
      limits:
        cpu: "1"
```

```
resources:
  limits:
    cpu: "1"
  requests:
    cpu: "1"
```

```
kubectl apply -f cpu-defaults-pod-2.yaml --namespace=default-cpu-example
```

```
kubectl get pod default-cpu-demo-2 --output=yaml --namespace=default-cpu-example
```



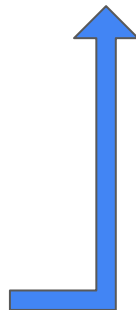
What if you specify a container's request, but not its limit?

```
apiVersion: v1
kind: Pod
metadata:
  name: default-cpu-demo-3
spec:
  containers:
  - name: default-cpu-demo-3-ctr
    image: nginx
    resources:
      requests:
        cpu: "0.75"
```

```
resources:
  limits:
    cpu: "1"
  requests:
    cpu: 750m
```

```
kubectl apply -f cpu-defaults-pod-3.yaml --namespace=default-cpu-example
```

```
kubectl get pod default-cpu-demo-3 --output=yaml --namespace=default-cpu-example
```



Configure Minimum and Maximum Memory Constraints for a Namespace

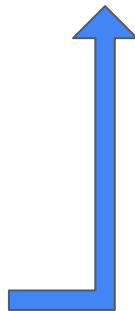
```
kubectl create namespace constraints-mem-example
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-min-max-demo-lr
spec:
  limits:
  - max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

```
limits:
- default:
    memory: 1Gi
  defaultRequest:
    memory: 1Gi
  max:
    memory: 1Gi
  min:
    memory: 500Mi
  type: Container
```

```
kubectl apply -f memory-constraints.yaml --namespace=constraints-mem-example
```

```
kubectl get limitrange mem-min-max-demo-lr --namespace=constraints-mem-example --output=yaml
```

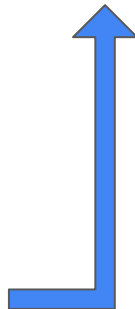



```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo
spec:
  containers:
  - name: constraints-mem-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
      requests:
        memory: "600Mi"
```

```
resources:
  limits:
    memory: 800Mi
  requests:
    memory: 600Mi
```

```
kubectl apply -f memory-constraints-pod.yaml --namespace=constraints-mem-example
```

```
kubectl get pod constraints-mem-demo --output=yaml --namespace=constraints-mem-example
```



Attempt to create a Pod that exceeds the maximum memory constraint

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo-2
spec:
  containers:
  - name:
constraints-mem-demo-2-ctr
  image: nginx
  resources:
    limits:
      memory: "1.5Gi"
    requests:
      memory: "800Mi"
```

```
kubectl apply -f memory-constraints-pod-2.yaml --namespace=constraints-mem-example
```

```
Error from server (Forbidden): error when creating "memory-constraints-pod-2.yaml":
pods "constraints-mem-demo-2" is forbidden: maximum memory usage per Container is 1Gi, but limit is
1536Mi.
```

Attempt to create a Pod that does not meet the minimum memory request

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo-3
spec:
  containers:
  - name:
constraints-mem-demo-3-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
      requests:
        memory: "100Mi"
```

```
kubectl apply -f memory-constraints-pod-3.yaml --namespace=constraints-mem-example
```

```
Error from server (Forbidden): error when creating "memory-constraints-pod-3.yaml":
pods "constraints-mem-demo-3" is forbidden: minimum memory usage per Container is 500Mi, but request
is 100Mi.
```

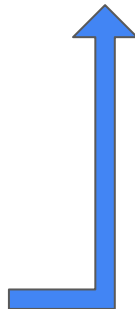
Create a Pod that does not specify any memory request or limit

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-mem-demo-4
spec:
  containers:
  - name: constraints-mem-demo-4-ctr
    image: nginx
```

```
resources:
  limits:
    memory: 1Gi
  requests:
    memory: 1Gi
```

```
kubectl apply -f memory-constraints-pod-4.yaml --namespace=constraints-mem-example
```

```
kubectl get pod constraints-mem-demo-4 --namespace=constraints-mem-example --output=yaml
```



Configure Minimum and Maximum CPU Constraints for a Namespace

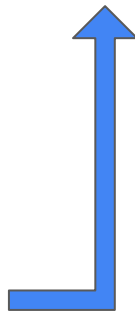
```
kubectl create namespace constraints-cpu-example
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-min-max-demo-lr
spec:
  limits:
  - max:
      cpu: "800m"
    min:
      cpu: "200m"
    type: Container
```

```
limits:
- default:
    cpu: 800m
  defaultRequest:
    cpu: 800m
  max:
    cpu: 800m
  min:
    cpu: 200m
  type: Container
```

```
kubectl apply -f cpu-constraints.yaml --namespace=constraints-cpu-example
```

```
kubectl get limitrange cpu-min-max-demo-lr --output=yaml --namespace=constraints-cpu-example
```

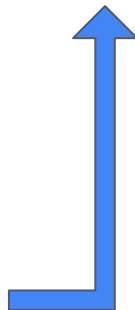


```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-cpu-demo
spec:
  containers:
  - name: constraints-cpu-demo-ctr
    image: nginx
    resources:
      limits:
        cpu: "800m"
      requests:
        cpu: "500m"
```

```
resources:
  limits:
    cpu: 800m
  requests:
    cpu: 500m
```

```
kubectl apply -f cpu-constraints-pod.yaml --namespace=constraints-cpu-example
```

```
kubectl get pod constraints-cpu-demo --output=yaml --namespace=constraints-cpu-example
```



Attempt to create a Pod that exceeds the maximum CPU constraint

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-cpu-demo-2
spec:
  containers:
  - name:
constraints-cpu-demo-2-ctr
    image: nginx
    resources:
      limits:
        cpu: "1.5"
      requests:
        cpu: "500m"
```

```
kubectl apply -f cpu-constraints-pod-2.yaml --namespace=constraints-cpu-example
```

```
Error from server (Forbidden): error when creating "cpu-constraints-pod-2.yaml":
pods "constraints-cpu-demo-2" is forbidden: maximum cpu usage per Container is 800m, but limit is
1500m.
```

Attempt to create a Pod that does not meet the minimum CPU request

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-cpu-demo-3
spec:
  containers:
  - name:
constraints-cpu-demo-3-ctr
  image: nginx
  resources:
    limits:
      cpu: "800m"
    requests:
      cpu: "100m"
```

```
kubectl apply -f cpu-constraints-pod-3.yaml --namespace=constraints-cpu-example
```

```
Error from server (Forbidden): error when creating "cpu-constraints-pod-3.yaml":
pods "constraints-cpu-demo-3" is forbidden: minimum cpu usage per Container is 200m, but request is
100m.
```

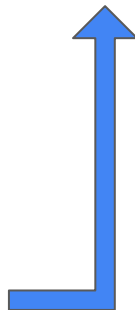

Create a Pod that does not specify any CPU request or limit

```
apiVersion: v1
kind: Pod
metadata:
  name: constraints-cpu-demo-4
spec:
  containers:
  - name: constraints-cpu-demo-4-ctr
    image: vish/stress
```

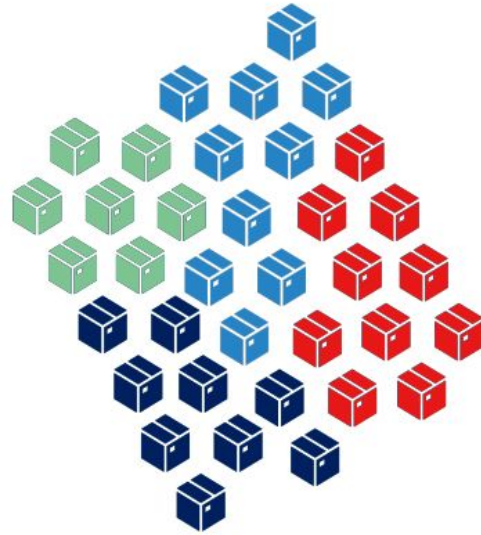
```
resources:
  limits:
    cpu: 800m
  requests:
    cpu: 800m
```

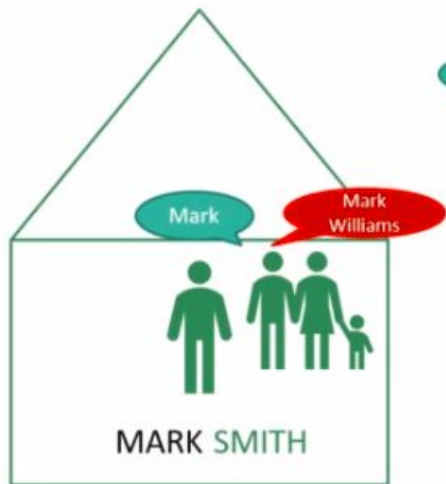
```
kubectl apply -f cpu-constraints-pod-4.yaml --namespace=constraints-cpu-example
```

```
kubectl get pod constraints-cpu-demo-4 --namespace=constraints-cpu-example --output=yaml
```



Namespace





View Namespaces

```
> kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	20m
kube-system	Active	20m

View Resources in a Namespace

```
> kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
myapp-pod	1/1	Running	0	20s

```
> kubectl get pods --namespace=kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
etcd-kubemaster	1/1	Running	2	14d
kube-apiserver-kubemaster	1/1	Running	4	14d
kube-controller-manager-kubemaster	1/1	Running	4	14d
kube-dns-86f4d74b45-zgh8p	3/3	Running	10	14d
kube-flannel-ds-gmg5r	1/1	Running	4	14d
kube-flannel-ds-kt74d	1/1	Running	4	14d
kube-flannel-ds-qtlg8	1/1	Running	4	14d
kube-proxy-4nkb6	1/1	Running	3	14d
kube-proxy-b6wnm	1/1	Running	3	14d
kube-proxy-rph7b	1/1	Running	2	14d
kube-scheduler-kubemaster	1/1	Running	3	14d
metrics-server-6fbfb84cdd-jt5q9	1/1	Running	0	3d

Create Namespace

```
namespace-definition.yml
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: team1

spec:
```

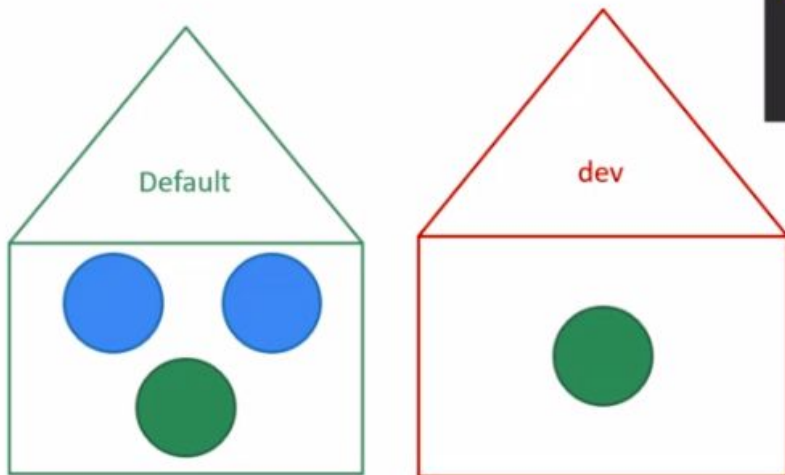
```
kubectl create -f namespace-definition.yml
namespace "team1" created
```

```
> kubectl create -f pod-definition.yml
```

```
pod/myapp-pod created
```

```
> kubectl create -f pod-definition.yml
```

```
pod/myapp-pod created
```

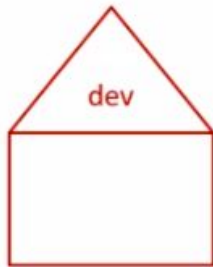


pod-definition.yml

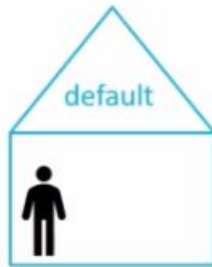
```
apiVersion: v1
kind: Pod

metadata:
  name: myapp-pod
  namespace: dev
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Switch



```
> kubectl get pods --namespace=dev
```

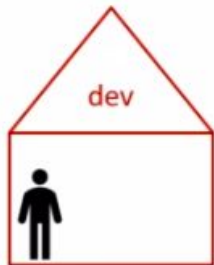


```
> kubectl get pods
```

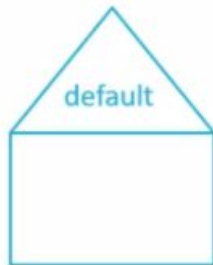


```
> kubectl get pods --namespace=prod
```

Switch



```
> kubectl get pods --namespace=dev
```



```
> kubectl get pods
```



```
> kubectl get pods --namespace=prod
```

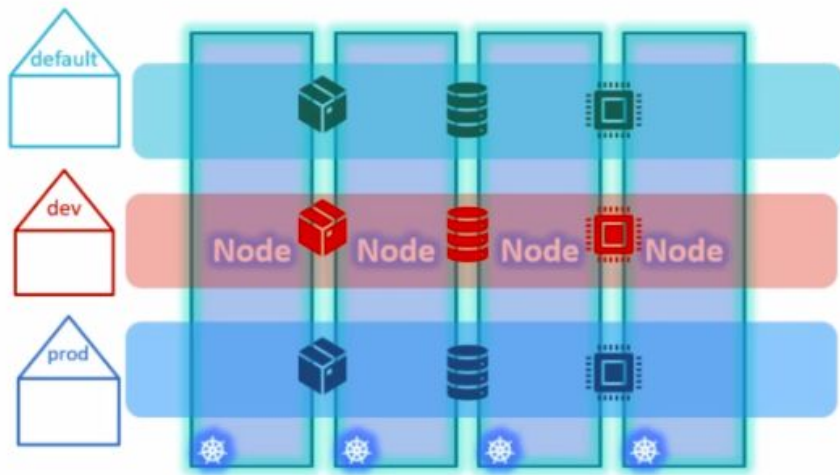
```
> kubectl config set-context $(kubectl config current-context) --namespace=dev
```

```
> kubectl get pods
```

```
> kubectl get pods --namespace=default
```

```
> kubectl get pods --namespace=prod
```


Resource Quota



Note: you could also limit the total number of other kinds of object. For example, you might decide to limit how many Deployments that can live in a single namespace.

Compute-quota.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-quota
  namespace: dev
spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: 5Gi
    limits.cpu: "10"
    limits.memory: 10Gi
```

```
> kubectl create -f compute-quota.yaml
```

Configure Memory and CPU Quotas for a Namespace

```
kubectl create namespace quota-mem-cpu-example
```

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

```
kubectl apply -f quota-mem-cpu.yaml --namespace=quota-mem-cpu-example
```

```
kubectl get resourcequota mem-cpu-demo --namespace=quota-mem-cpu-example --output=yaml
```

The ResourceQuota places these requirements on the quota-mem-cpu-example namespace:

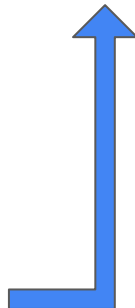
- For every Pod in the namespace, each container must have a memory request, memory limit, cpu request, and cpu limit.
- The memory request total for all Pods in that namespace must not exceed 1 GiB.
- The memory limit total for all Pods in that namespace must not exceed 2 GiB.
- The CPU request total for all Pods in that namespace must not exceed 1 cpu.
- The CPU limit total for all Pods in that namespace must not exceed 2 cpu.

```
apiVersion: v1
kind: Pod
metadata:
  name: quota-mem-cpu-demo
spec:
  containers:
  - name: quota-mem-cpu-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
        cpu: "800m"
      requests:
        memory: "600Mi"
        cpu: "400m"
```

```
status:
  hard:
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.cpu: "1"
    requests.memory: 1Gi
  used:
    limits.cpu: 800m
    limits.memory: 800Mi
    requests.cpu: 400m
    requests.memory: 600Mi
```

```
kubectl apply -f quota-mem-cpu-pod.yaml --namespace=quota-mem-cpu-example
```

```
kubectl get resourcequota mem-cpu-demo --namespace=quota-mem-cpu-example --output=yaml
```



Attempt to create a second Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: quota-mem-cpu-demo-2
spec:
  containers:
  - name: quota-mem-cpu-demo-2-ctr
    image: redis
    resources:
      limits:
        memory: "1Gi"
        cpu: "800m"
      requests:
        memory: "700Mi"
        cpu: "400m"
```

```
kubectl apply -f quota-mem-cpu-pod-2.yaml --namespace=quota-mem-cpu-example
```

```
Error from server (Forbidden): error when creating "quota-mem-cpu-pod-2.yaml":
pods "quota-mem-cpu-demo-2" is forbidden: exceeded quota: mem-cpu-demo,
requested: requests.memory=700Mi,used: requests.memory=600Mi, limited:
requests.memory=1Gi
```

Configure a Pod Quota for a Namespace

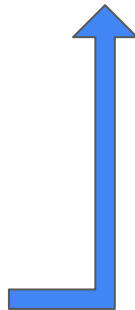
```
kubectl create namespace quota-pod-example
```

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: pod-demo
spec:
  hard:
    pods: "2"
```

```
status:
  hard:
    pods: "2"
  used:
    pods: "0"
```

```
kubectl apply -f quota-pod.yaml --namespace=quota-pod-example
```

```
kubectl get resourcequota pod-demo --namespace=quota-pod-example --output=yaml
```



Create a deployment

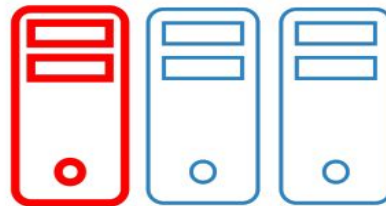
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pod-quota-demo
spec:
  selector:
    matchLabels:
      purpose: quota-demo
  replicas: 3
  template:
    metadata:
      labels:
        purpose: quota-demo
    spec:
      containers:
        - name: pod-quota-demo
          image: nginx
```

```
kubectl apply -f quota-pod-deployment.yaml
--namespace=quota-pod-example
```

```
kubectl get deployment pod-quota-demo
--namespace=quota-pod-example --output=yaml
```

```
spec:
  ...
  replicas: 3
  ...
status:
  availableReplicas: 2
  ...
lastUpdateTime: 2021-04-02T20:57:05Z
  message: 'unable to create pods: pods
"pod-quota-demo-1650323038-" is forbidden:
  exceeded quota: pod-demo, requested: pods=1,
used: pods=2, limited: pods=2'
```

Taints And Tolerations



A


B

C

D

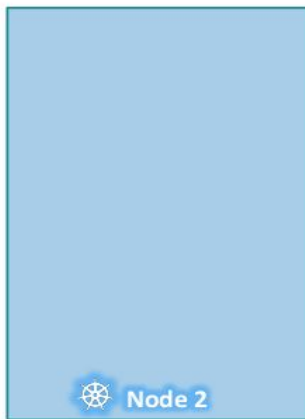
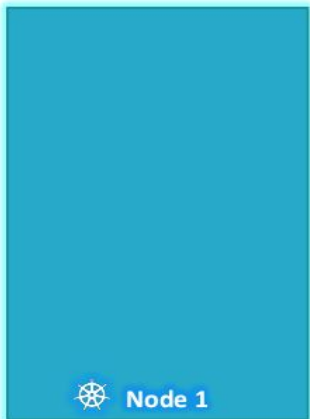
 Node 1

 Node 2

 Node 3



Taint=blue



D



Node 1

C

A



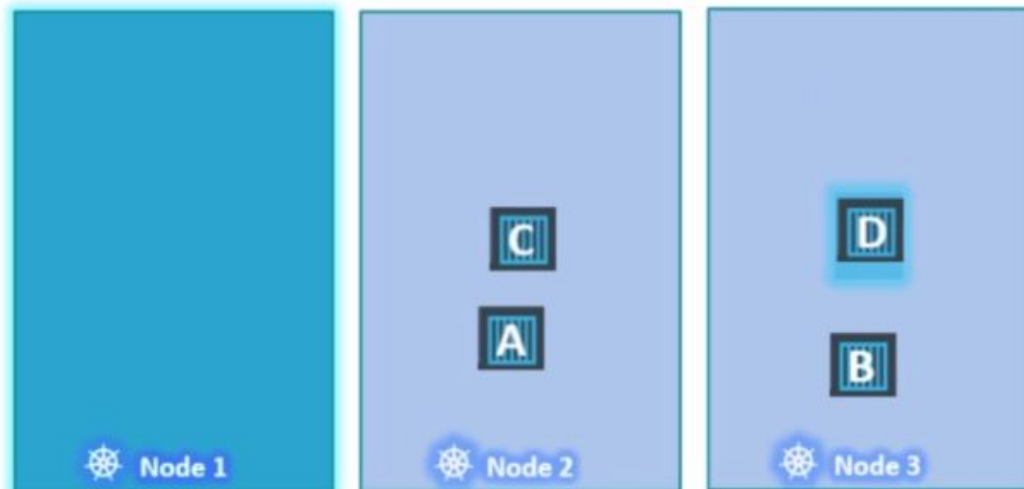
Node 2

B



Node 3

Taint=blue





Taints - Node

```
kubectl taint nodes node-name key=value:taint-effect
```

NoSchedule | PreferNoSchedule | NoExecute

What happens to PODs
that do not tolerate this taint?

```
kubectl taint nodes node1 app=myapp:NoSchedule
```

★ Tolerations - PODs

```
kubectl taint nodes node1 app= blue :NoSchedule
```

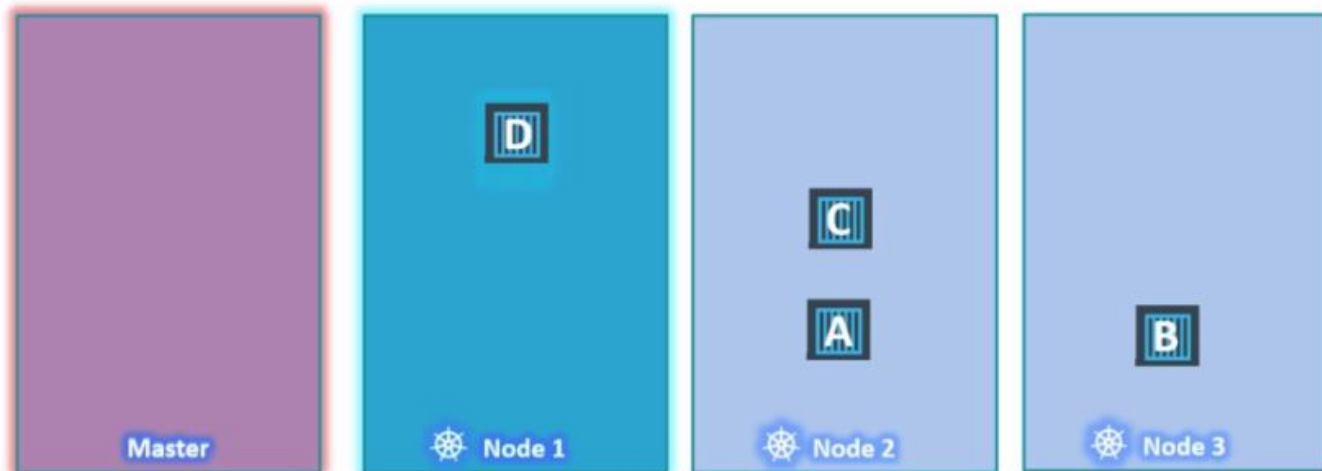
pod-definition.yml

```
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: nginx-container
    image: nginx

  tolerations:
  - key: "app"
    operator: "Equal"
    value: " blue"
    effect: " NoSchedule "
```

```
kubectl describe node kubemaster | grep Taint
```

```
Taints:                node-role.kubernetes.io/master:NoSchedule
```

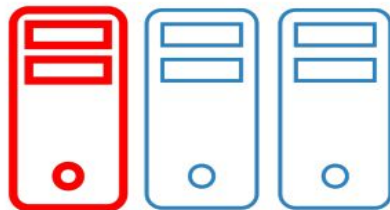


Manual Scheduling

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 8080
  nodeName: node02
```


Node Selectors

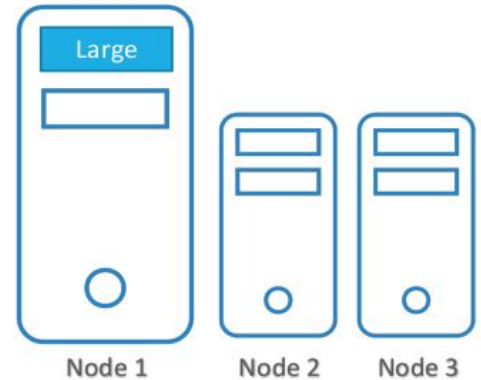


Node Selectors

pod-definition.yml

```
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: data-processor
    image: data-processor

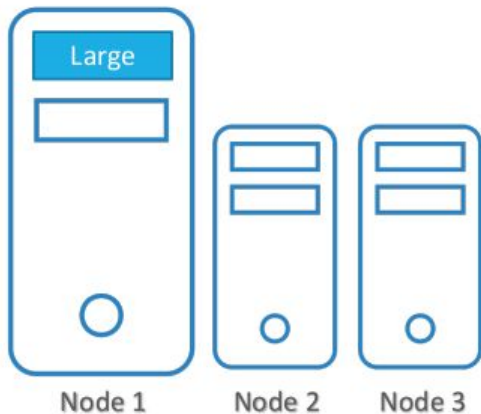
  nodeSelector:
    size: Large
```



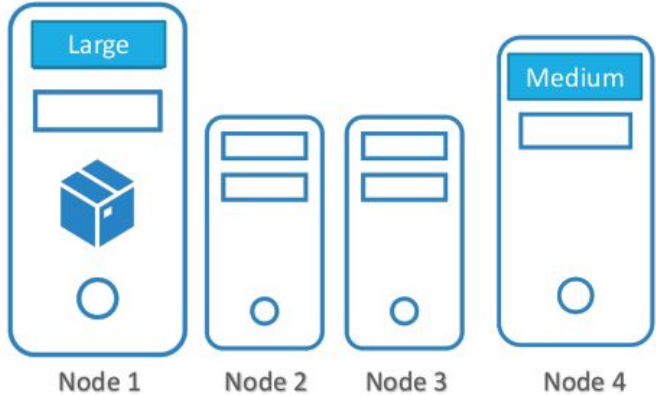
Label Nodes

```
▶ kubectl label nodes <node-name> <label-key>=<label-value>
```

```
▶ kubectl label nodes node-1 size=Large
```

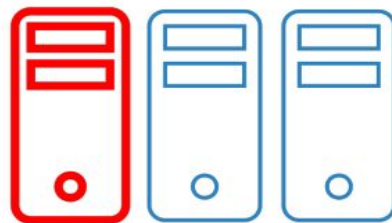


Node Selector - Limitations



- Large OR Medium?
- NOT Small

Node Affinity



Node Affinity

pod-definition.yml

```
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
    - name: data-processor
      image: data-processor

  nodeSelector:
    size: Large
```

pod-definition.yml

```
apiVersion:
kind:
metadata:
  name: myapp-pod
spec:
  containers:
    - name: data-processor
      image: data-processor

  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: size
                operator: In
                values:
                  - Large
```

Node Affinity Types

Available:

requiredDuringSchedulingIgnoredDuringExecution

preferredDuringSchedulingIgnoredDuringExecution

Planned:

requiredDuringSchedulingRequiredDuringExecution



Blue



Red



Green



Other



Other

References:

- <https://www.udemy.com/course/certified-kubernetes-administrator-with-practice-tests>
- <https://www.udemy.com/course/certified-kubernetes-application-developer>
- <https://kubernetes.io/docs>