DISTRIBUTED AND CLOUD COMPUTING

LAB 13: K8S POD SCHEDULING

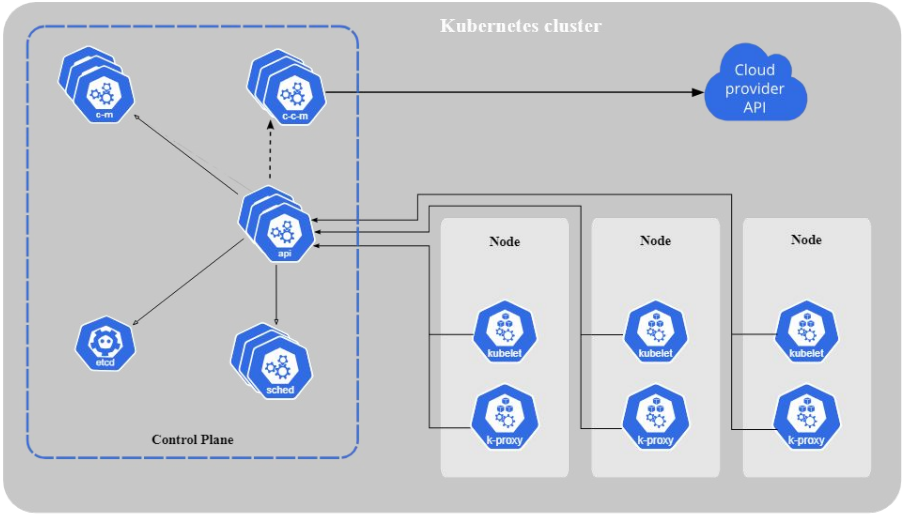
(Module: K8S & CLOUD BASICS)

https://kubernetes.io/docs/concepts/overview/components/

Kubernetes Components

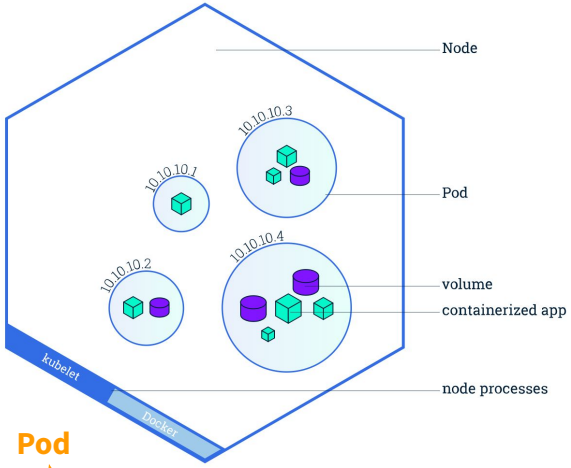
Today's focus

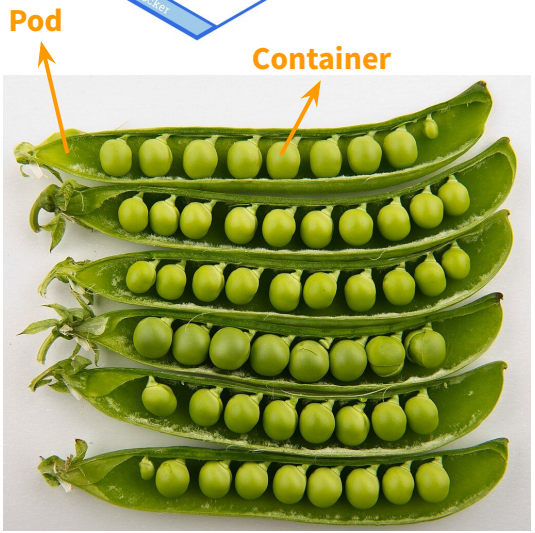
- Control Plane
  - etcd (distributed KV store following the Raft consensus protocol)
  - API Server: exposes K8s API to clients like `kubectl
  - Controller Manager: control loops that monitor system states & match desired states
  - Cloud Controller Manager (*): cloud-specific control logic
  - **Scheduler: performs pod scheduling to cluster nodes**
- Worker Node
  - kubelet
    - node agent
    - "local official"
  - kube-proxy (*)
    - network manager
  - Container Runtime
    - Containerd
    - CRI-O
    - …



Kubernetes Resources/Objects

- Pod: smallest deployable unit - group of containers
  - Containers from the same pod are tightly coupled.
- ReplicaSet & Deployment
  - Deployment owns ReplicaSets to manage sets of pod replicas, in order to execute **stateless** workloads.
- StatefulSet: maintains a group of pods to support stateful applications with persistent storage & network identities.
- Job: one-off tasks that run to completion and then stop.
- Service: expose applications behind a single endpoint
- AutoScalers
  - Pod AutoScaler:
    - Horizontal (scale in/out)
    - Vertical (scale up/down)
  - Cluster AutoScaler: autoscales cluster nodes
- …

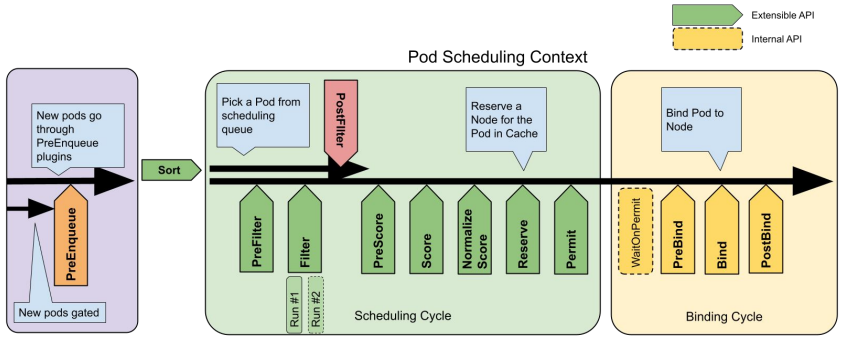




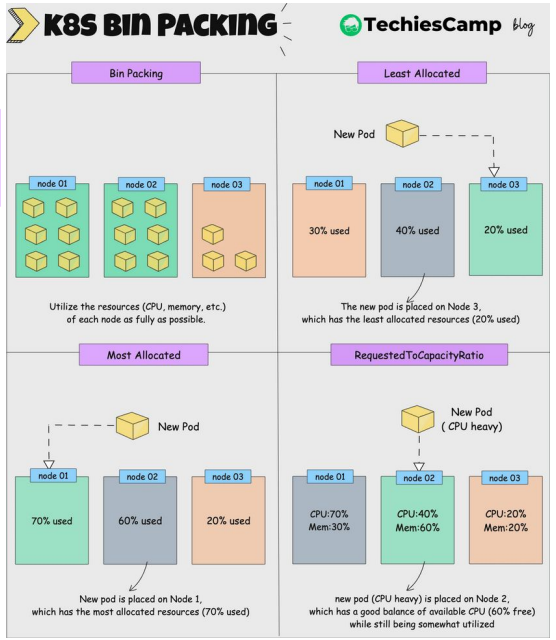
https://en.wikipedia.org/wiki/Pea
https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

Kubernetes Scheduler

- Scheduling: ensure pods are matched to nodes for kubelet to run.
- kube-scheduler is the default scheduler implementation.
- General Scheduling Framework Design:
  a. Pod Scheduling / Node Selection (serial execution)
     i. Filtering: finds a set of "feasible" nodes.
     ii. Scoring: ranks the remaining nodes to choose the most suitable pod placement.
  b. Pod Binding (concurrent execution): notifies the API Server about the decision.
- We focus on Pod Scheduling - Filtering today.

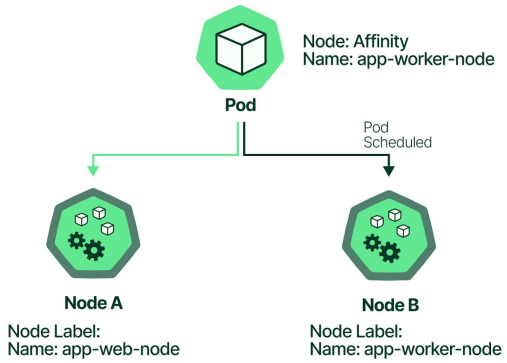


nodeSelector
Pod affinity & Anti-affinity
Taints & Tolerations



https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/#interfaces
https://blog.techiescamp.com/docs/kubernetes-bin-packing/

`nodeSelector` & Node Affinity



- `nodeSelector` is the simplest recommended form of node selection constraint.
- Node Affinity is a more flexible version of `nodeSelector`.
- Specifies the pods to match the nodes with expected node labels.
  - In the figure, 2 nodes with different node label values are being filtered for 1 pod.
    - Node `A` has `Name=app-web-node`.
    - Node `B` has `Name=app-worker-node`.
    - The pod wants to be scheduled to nodes having `Name=app-worker-node`.
    - As a result, the pod will be scheduled to Node `B`.
- Node **Affinity**: specifies that pods **have**/**prefer** to be on nodes with any of the conditions:
  - have a label key specified / not specified (i.e., `Exists`, `DoesNotExist`);
  - have a label value within / not within a set of options (i.e., `In`, `NotIn`);
  - have a numeric label value greater/lower than specified (i.e., `Gt`, `Lt`).
- TL;DR. Preconfigure node labels, then specify pods to match certain node labels for filtering.

https://blog.kubecost.com/blog/kubernetes-node-affinity

TASK: Pod Scheduling

Clean the resources after playing via `kubectl delete -f xxx.yaml`

```
# worker node 1
- role: worker
  labels:
    tier: frontend        # for frontend deployment
    attendance: present
# worker node 2
```

Use Kind to explore different K8s pod scheduling methods.

> Reference codebase: `k8s_scheduling`

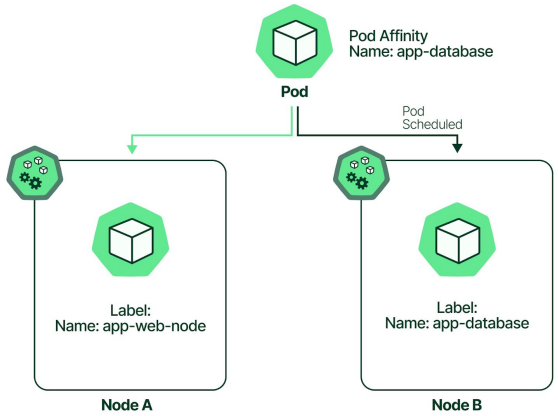
1. Set up Kubernetes CLI - `kubectl` and Kind.
2. Create/Update the `kind-config.yaml` to specify a cluster template with 1 control plane + 5 worker nodes. Then, create the multi-node cluster with name “dncc” using Kind:
   - `kind create cluster --name dncc --config kind-config.yaml`
3. Now try the node selector sub-demo.
4. Kind configuration file also defines node labels:
   - `kubectl get nodes -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.metadata.labels}{"\n"}{end}'`
5. The `m0-label-selector.yaml` file specifies a deployment with 3 pod replicas with a node selector specification. Deploy via: `kubectl apply -f m0-node-selector.yaml`
6. Then check where the nodes are scheduled via: `kubectl get pods -o wide`

```
# kubectl get nodes -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.metadata.labels}{"\n"}{end}'
dncc-control-plane      {"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-control-plane","kubernetes.io/os":"linux","node-role.kubernetes.io/control-plane":"",
"node.kubernetes.io/exclude-from-external-load-balancers":""}
dncc-worker     {"attendance":"present","beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-worker","kubernetes.io/os":"linux","tier":"frontend"}
dncc-worker2    {"attendance":"absent","beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-worker2","kubernetes.io/os":"linux","tier":"frontend"}
dncc-worker3    {"attendance":"present","beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-worker3","kubernetes.io/os":"linux","tier":"frontend"}
dncc-worker4    {"attendance":"present","beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-worker4","kubernetes.io/os":"linux","tier":"backend"}
dncc-worker5    {"attendance":"present","beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"dncc-worker5","kubernetes.io/os":"linux","purpose":"reserved"}

# kubectl get pods -o wide
NAME                            READY   STATUS    RESTARTS   AGE   IP           NODE          NOMINATED NODE   READINESS GATES
app-frontend-d968bbf96-rv57h    1/1     Running   0          8s    10.244.4.2   dncc-worker3  <none>           <none>
app-frontend-d968bbf96-wzznn    1/1     Running   0          8s    10.244.1.2   dncc-worker   <none>           <none>
app-frontend-d968bbf96-zvpz4    1/1     Running   0          8s    10.244.1.3   dncc-worker   <none>           <none>
```

```
nodeSelector:
  # all labels should match on the node
  tier: frontend
  attendance: present
```

Pod Affinity & Anti-affinity



- Pod Affinity & Anti-affinity handle inter-pod constraints: pods must/can (not) be with each other.
- Instead of node labels, pod labels are examined.
- Check the example in the figure:
  - 2 pods with different pod labels are already deployed one-to-one to 2 nodes.
  - The pod in Node `A` has `Name=app-web-node`.
  - The pod in Node `B` has `Name=app-database`.
  - The current pod wants to be scheduled with pods having `Name=app-database`.
  - As a result, the current pod will be scheduled to Node `B`.
- Soft/Hard Constraints:
  - Hard - **`required`**`DuringSchedulingIgnoredDuringExecution`
  - Soft - **`preferred`**`DuringSchedulingIgnoredDuringExecution`
    - Pods can have multiple soft pod affinity / anti-affinity constraints, specified with different weights.
    - Soft constraints with higher weights might be prioritized in scoring.

https://blog.kubecost.com/blog/kubernetes-node-affinity

TASK: Pod Scheduling

Use Kind to explore different K8s pod scheduling methods.

> Reference codebase: `k8s_scheduling`

7. Now try the pod affinity & anti-affinity demo.
8. The `m1-pod-affinity.yaml` file specifies 3 deployments representing different applications. Deploy via:
   - `kubectl apply -f m1-pod-affinity.yaml`
9. Then check where the nodes are scheduled via:
   - `kubectl get pods -o wide`
10. Check the failed pod via `kubectl describe pod xxx`

```
(base) root@RAINBOW:~/rainbow/kcl/lab/cncf/dncc-lab/k8s_cloud/0_k8s/1_scheduling# kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE    IP            NODE             NOMINATED NODE   R
app-1-8658ff7c6d-dsgrm   1/1     Running   0          2m5s   10.244.4.2    dncc-worker3     <none>           <
app-1-8658ff7c6d-xzp4d   1/1     Running   0          2m5s   10.244.1.3    dncc-worker2     <none>           <
app-2-988579d8b-mbbr8    1/1     Running   0          2m5s   10.244.1.5    dncc-worker2     <none>           <
app-2-988579d8b-nkpnd    1/1     Running   0          2m5s   10.244.1.2    dncc-worker2     <none>           <
app-2-988579d8b-t6x8w    1/1     Running   0          2m5s   10.244.1.4    dncc-worker2     <none>           <
app-3-77c8f4877-m2m7q    1/1     Running   0          2m5s   10.244.5.2    dncc-worker5     <none>           <
app-3-77c8f4877-s6zsm    0/1     Pending   0          2m5s   <none>        <none>           <none>           <
app-3-77c8f4877-t9hwq    1/1     Running   0          2m5s   10.244.2.2    dncc-worker      <none>           <
app-3-77c8f4877-wz6lb    1/1     Running   0          2m5s   10.244.3.2    dncc-worker4     <none>           <
```

```
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  33s   default-scheduler  0/6 nodes are available: 1 node(s) had
untolerated taint {node-role.kubernetes.io/control-plane: }, 5 node(s) didn't match pod anti-
affinity rules. preemption: 0/6 nodes are available: 1 Preemption is not helpful for scheduli
ng, 5 No preemption victims found for incoming pod.
```

```yaml
metadata:
  labels:
    app: app-1
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: app
              operator: In
              values:
              - app-1
        topologyKey: kubernetes.io/hostname   # Ensur
```

```yaml
metadata:
  labels:
    app: app-2
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchLabels:
            app: app-1
        topologyKey: kubernetes.io/hostname   # Ensur
```

```yaml
metadata:
  labels:
    app: app-3
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - app-1
            - app-2
        topologyKey: kubernetes.io/hostname  # Ensure
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - app-3
        topologyKey: kubernetes.io/hostname   # Ensur
```

TASK: Pod Scheduling

Try this official demo as well if you are interested!

Use Kind to explore different K8s pod scheduling methods.

> Reference codebase: `k8s_scheduling`

7. Now try the pod affinity & anti-affinity demo.
8. The `m1-pod-affinity.yaml` file specifies 3 deployments representing different applications. Deploy via:
   - `kubectl apply -f m1-pod-affinity.yaml`
9. Then check where the nodes are scheduled via:
   - `kubectl get pods -o wide`
10. Check the failed pod via `kubectl describe pod xxx`
11. Move pod self-anti-affinity rule to preferred. Then recreate the resources (`kubectl delete` & `kubectl apply`).
12. Now all 4 pod replicas from app-3 will be successfully deployed.

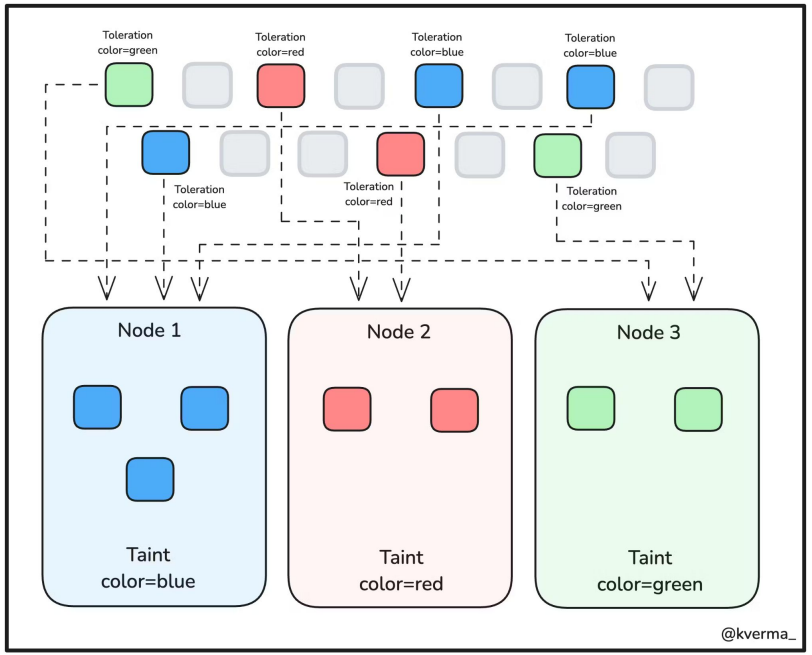
```
kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE           
app-1-8658ff7c6d-bwv9n   1/1     Running   0          56s   10.244.2.4   dncc-worker    
app-1-8658ff7c6d-psmf4   1/1     Running   0          56s   10.244.4.6   dncc-worker3   
app-2-988579d8b-77l7q    1/1     Running   0          56s   10.244.2.5   dncc-worker    
app-2-988579d8b-h2vqh    1/1     Running   0          56s   10.244.2.6   dncc-worker    
app-2-988579d8b-knlzj    1/1     Running   0          56s   10.244.4.5   dncc-worker3   
app-3-7b9b4bc87-ds2m4    1/1     Running   0          56s   10.244.3.5   dncc-worker4   
app-3-7b9b4bc87-hwwzd    1/1     Running   0          56s   10.244.3.4   dncc-worker4   
app-3-7b9b4bc87-m76cq    1/1     Running   0          56s   10.244.5.6   dncc-worker5   
app-3-7b9b4bc87-vxtll    1/1     Running   0          56s   10.244.1.7   dncc-worker2   
```

```yaml
metadata:
  labels:
    app: app-3
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - app-1
            - app-2
        topologyKey: kubernetes.io/hostname  # Ensure
      - labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - app-3
        topologyKey: kubernetes.io/hostname  # Ensur
```

```yaml
    topologyKey: kubernetes.io/hostname  # Ensure
  # - labelSelector:
  #     matchExpressions:
  #     - key: app
  #       operator: In
  #       values:
  #       - app-3
  #   topologyKey: kubernetes.io/hostname  # Ensu
  preferredDuringSchedulingIgnoredDuringExecution:
  # There will be one pod replica pending, since i
  # as a solution, switch the second label selecto
  # -- UNCOMMENT CONFIG BELOW --
  - weight: 6
    podAffinityTerm:
      labelSelector:
        matchExpressions:
        - key: app
          operator: In
          values:
          - app-3
      topologyKey: kubernetes.io/hostname  # Ensu
```

Taints & Tolerations

- Node Affinity: nodes with certain labels **attract** these pods.
- **Taints**: nodes with certain taints **repel** pods
  - Format: `key=value:effect`
  - E.g., `user=admin:NoSchedule`
- **Tolerations**: Pods are configured so that they can **tolerate** certain node taints and become schedulable to relevant nodes.
- Common use cases:
  - Dedicated Node Specification
    i. Hardware Requirements
    ii. Business-level Logic
  - Temporary Node Isolation
    i. for responsibility managing (e.g., control-plane)
    ii. for maintenance/troubleshooting



https://blog.kubesimplify.com/kubernetes-scheduling-the-complete-guide

TASK: Pod Scheduling

Use Kind to explore different K8s pod scheduling methods.

> Reference codebase: `k8s_scheduling`

13. Now try the taints & tolerations demo.
14. Uncomment the node taints configuration for worker node 5 in `kind-config.yaml`. Recreate the cluster (`kind delete` & `kind create`). Check this taint:
    - `kubectl get nodes -o jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.spec.taints}{"\n"}{end}'`
15. The `m2-taint-toleration.yaml` file specifies a deployment with 3 pod replicas with a node selector specification. Deploy via: `kubectl apply -f m2-taint-toleration.yaml`
16. Then check where the nodes are scheduled via: `kubectl get pods -o wide`
17. Check the failed pod via `kubectl describe pod xxx`

```
# worker node 5
- role: worker
  # https://github.com/kubernetes-sigs/kind/issues/3775
  kubeadmConfigPatches:
  - |
    kind: JoinConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        # no tier label
        node-labels: "attendance=present,purpose=reserve
      # Uncomment the following lines to test the taints
      taints:
      - key: user
        value: "admin"
        effect: NoSchedule
```

```
1 node(s) had untolerated taint {user: admin}
```

```
dncc-control-plane      [{"effect":"NoSchedule","key":"node-role.kubernetes.io/control-plane"}]
dncc-worker
dncc-worker2
dncc-worker3
dncc-worker4
dncc-worker5    [{"effect":"NoSchedule","key":"user","value":"admin"}]
```

```
NAME                                READY   STATUS    RESTARTS   AGE   IP           NODE          NOMINATED NODE   READINESS GATES
app-normal-8557598db7-gxh6c         0/1     Pending   0          5s    <none>       <none>        <none>           <none>
app-special-74d4f49658-7kwnd        1/1     Running   0          5s    10.244.4.2   dncc-worker5  <none>           <none>
```

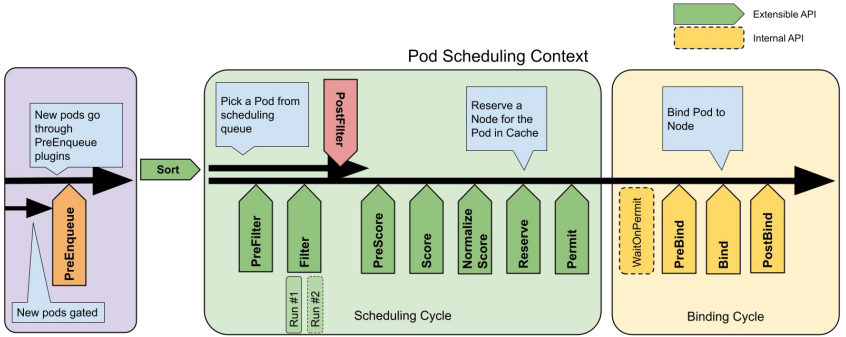
```
metadata:
  labels:
    app: special
spec:
  nodeSelector:
    purpose: reserved
  tolerations:
  - key: "user"
    operator: "Equal"
    value: "admin"
    effect: "NoSchedule"
```

```
template:
  metadata:
    labels:
      app: normal
  spec:
    nodeSelector:
      purpose: reserved
```

Summary

Kubernetes Scheduler

- kube-scheduler as the default implementation
- Framework
  - Pod Scheduling / Node Selection
    - **Filtering**
      - **`nodeSelector` & Node Affinity**
      - **Pod Affinity & Anti-affinity**
      - **Taints & Tolerations**
      - **…**
    - Scoring
      - Lest Allocated
      - Most Allocated
      - Requested-to-Capacity Ratio
      - …
  - Pod Binding



That's all for the K8s lab! There are still so much more about K8s. If you are interested, you can continue with the official documentation :)

The next lab will introduce a bit about Cloud Computing, including common cloud providers & services, and Infrastructure as Code (IaC).

https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/#interfaces