SIG-Scheduling Deep-Dive

- Day 1/2/3 of operating kube-scheduler

Wei Huang, IBM, @Huang-Wei
Abdullah Gharaibeh, Google, @ahg-g

Lead-in

- A thousands ways of view Scheduler
- Day 1, Day 2 and Day 3 of operating Kubernetes scheduler

Day 1 – App developer

- Audience: users who write & deploy containerized application onto k8s
- Goal: Understand kube-scheduler basics, identify whether it's a scheduler issue, and how to use scheduler features
- Non-Goal: Understand scheduler internals

Day 1 – Kube-scheduler

- Kube-scheduler: Assign Pods to Nodes

```
wei.huang1@wei-mbp:~|⇒  k get pod -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
k8s-for-beginners-5b5b757bdb-97g4t   0/1     Pending   0          71s   <none>   <none>   <none>           <none>
wei.huang1@wei-mbp:~|⇒  k get po k8s-for-beginners-5b5b757bdb-97g4t -o jsonpath="{.spec.nodeName}"
wei.huang1@wei-mbp:~|⇒
```

```
wei.huang1@wei-mbp:~|⇒  k get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP           NODE          NOMINATED NODE   READINESS GATES
k8s-for-beginners-5b5b757bdb-97g4t   1/1     Running   0          11m   10.244.2.4   kind-worker2  <none>           <none>
wei.huang1@wei-mbp:~|⇒  k get po k8s-for-beginners-5b5b757bdb-97g4t -o jsonpath="{.spec.nodeName}"
kind-worker2%
```

Day 1 – Scheduler Do's and Don'ts

- kube-scheduler do's and don'ts
  - 🚫 Quota enforcement
  - 🚫 Spinning up / scaling down replicas of a Deployment/StatefulSet/etc.
  - 🚫 Evict Pods upon OutOf {Memory|Disk|CPU} error
  - 🚫 Taint nodes (kubectl taint node, or upon {Memory|Disk|CPU} pressure)
  - 🚫 Reschedule / Rebalance running Pods[1]

    Find the best Node for pending Pods
    Preempt low-priority Pods to make room for high-priority Pods

[1] A sub-project of sig-scheduling covers it: https://github.com/kubernetes-sigs/descheduler

Day 1 – Filtering

- Filtering – **Hard** Constraints, i.e., I **need** my Pod to:
  - To have 2Gi memory and 1 core CPU
  - Co-exist with some kinds of Pods
  - To tolerate taints with effect "NoSchedule"
  - …
- All hard constraints are **ANDed**
- (Almost) All **hard** constraints are from pod's spec – i.e., specified by the user

```yaml
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: foo
            operator: Exists
        topologyKey: region
```

```yaml
spec:
  containers:
  - name: test-pod
    image: k8s.gcr.io/pause:3.2
    resources:
      requests:
        cpu: 1
        memory: 2Gi
```

```yaml
spec:
  topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: kubernetes.io/hostname
    whenUnsatisfiable: DoNotSchedule
    labelSelector:
      matchLabels:
        foo: ""
```

Day 1 – Scoring

- Scoring – Soft Constraints, i.e., I prefer my Pod to:
  - To be scheduled to a node which has SSDs
  - Not to co-exist with some kinds of Pods
  - …
- Based on the soft constraints, each filtered Node gets a Score
- sum(score) for each filtered Node, and pick the highest score
- Soft constraints have 2 sources
  - Pod's spec
  - Implicit scheduler config, e.g., NodeResourcesLeastAllocated

```yaml
spec:
  affinity:
    nodeAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        preference:
          matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
```

```yaml
spec:
  topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: kubernetes.io/hostname
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        foo: ""
```

Day 1 – Preemption

- What if no node can satisfy all the **Hard** Constraints?
- Preemption
  - High-priority Pods are eligible to preempt low-priority Pods

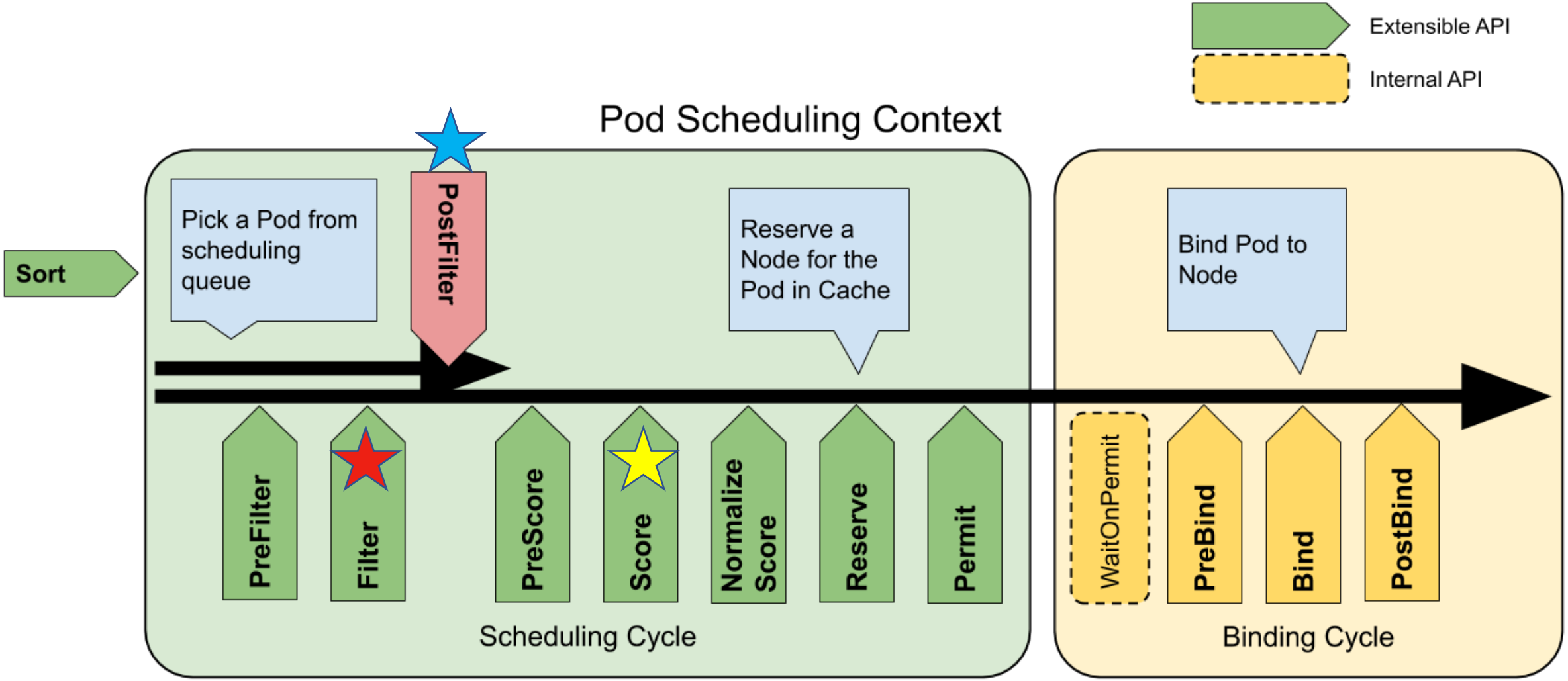
File: **pod1.yaml**

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  priorityClassName: p1
  containers:
  - name: pod1
    image: k8s.gcr.io/pause:3.2
```

File: **priority-classes.yaml**

```yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: p1
value: 1
description: "priority with value 1"
---
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: p2
value: 2
description: "priority with value 2"
```

Day 1 – Scheduling Flow

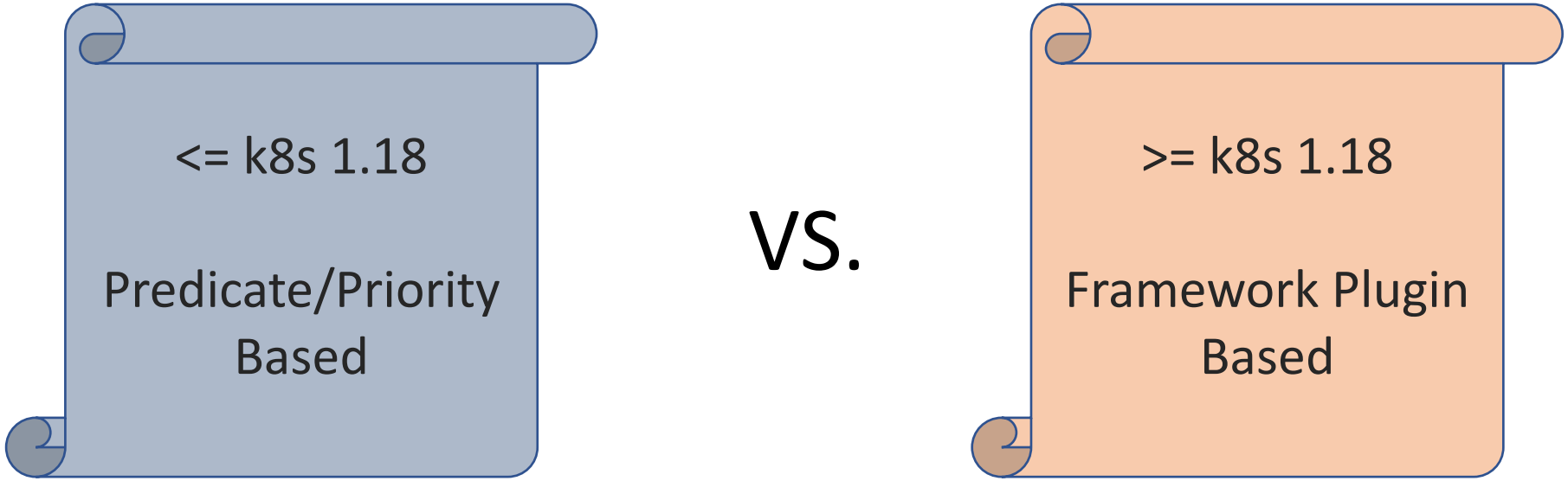


Day 2 – Cluster admin / Devops

- Audience: cluster admin / devops
- Goal: master scheduler configuration best practices, understand scheduler framework basics to make the most of kube-scheduler
- Non-goal: writing additional scheduler extender / plugin code

Day 2 – Configurations

- KubeSchedulerConfiguration (--config <config file>) over ~~CLI args~~
  - v1alph1 (<= k8s 1.17)
  - v1alph2 (k8s 1.18)
  - v1beta1 (k8s 1.19)
  - Legacy policy file based config vs. plugins based config



Day 2 – Policy-based config

KubeCon CloudNativeCon North America 2020 Virtual

File: **v1alpha1/scheduler-config.yaml**

```yaml
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: "/var/run/kubernetes/scheduler.kubeconfig"
algorithmSource:
  policy:
    file:
      path: "/root/config/scheduler-policy.yaml"
```

File: **v1alpha1/scheduler-policy.yaml**

```yaml
apiVersion: v1 # or kubescheduler.config.k8s.io/v1
kind: Policy
predicates:
- name: PodFitsHost
- name: PodFitsResources
...
priorities:
- name: InterPodAffinityPriority
  weight: 50
- name: LeastRequestedPriority
  weight: 10
# - name: MostRequestedPriority
#   weight: 10
...
```

- https://kubernetes.io/docs/reference/scheduling/policies/
- Will be deprecated
- Provide config and policy yaml
- Policy API are not user-friendly – have a specify a full list of predicates/priorities

Day 2 – Plugins-based config

- https://kubernetes.io/docs/reference/scheduling/config/
- Aligned with scheduler framework
- User friendly - enable/disable plugins
- Support multi-profile config

File: **v1beta1.yaml**

```yaml
apiVersion: kubescheduler.config.k8s.io/v1beta1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: /etc/kubernetes/scheduler.conf
profiles:
- schedulerName: default-scheduler
  plugins:
    score:
      enabled:
      - name: NodeResourcesMostAllocated
        weight: 50000
      disabled:
      - name: NodeResourcesLeastAllocated
```

Day 2 – Multi-profile scheduler


```yaml
apiVersion: kubescheduler.config.k8s.io/v1beta1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: /etc/kubernetes/scheduler.conf
leaderElection:
  leaderElect: false
profiles:
- schedulerName: default-scheduler
- schedulerName: image-first
  plugins:
    score:
      enabled:
      - name: ImageLocality
        weight: 50000
      disabled:
      - name: ImageLocality
- schedulerName: binpack
  plugins:
    score:
      enabled:
      - name: NodeResourcesMostAllocated
        weight: 50000
      disabled:
      - name: NodeResourcesLeastAllocated
- schedulerName: skip-score
  plugins:
    preScore:
      disabled:
      - name: "*"
    score:
      disabled:
      - name: "*"
```

File: **ubuntu.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ubuntu
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ubuntu
  template:
    metadata:
      labels:
        app: ubuntu
    spec:
      schedulerName: image-first
      containers:
      - name: ubuntu
        image: ubuntu
        # Wait forever
        command: [ "/bin/bash", "-c", "--" ]
        args: [ "while true; do sleep 3600; done;" ]
```

File: **normal.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pause
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pause
  template:
    metadata:
      labels:
        app: pause
    spec:
      # schedulerName: default-scheduler
      containers:
      - name: pause
        image: k8s.gcr.io/pause:3.2
```

File: **binpack.yaml**

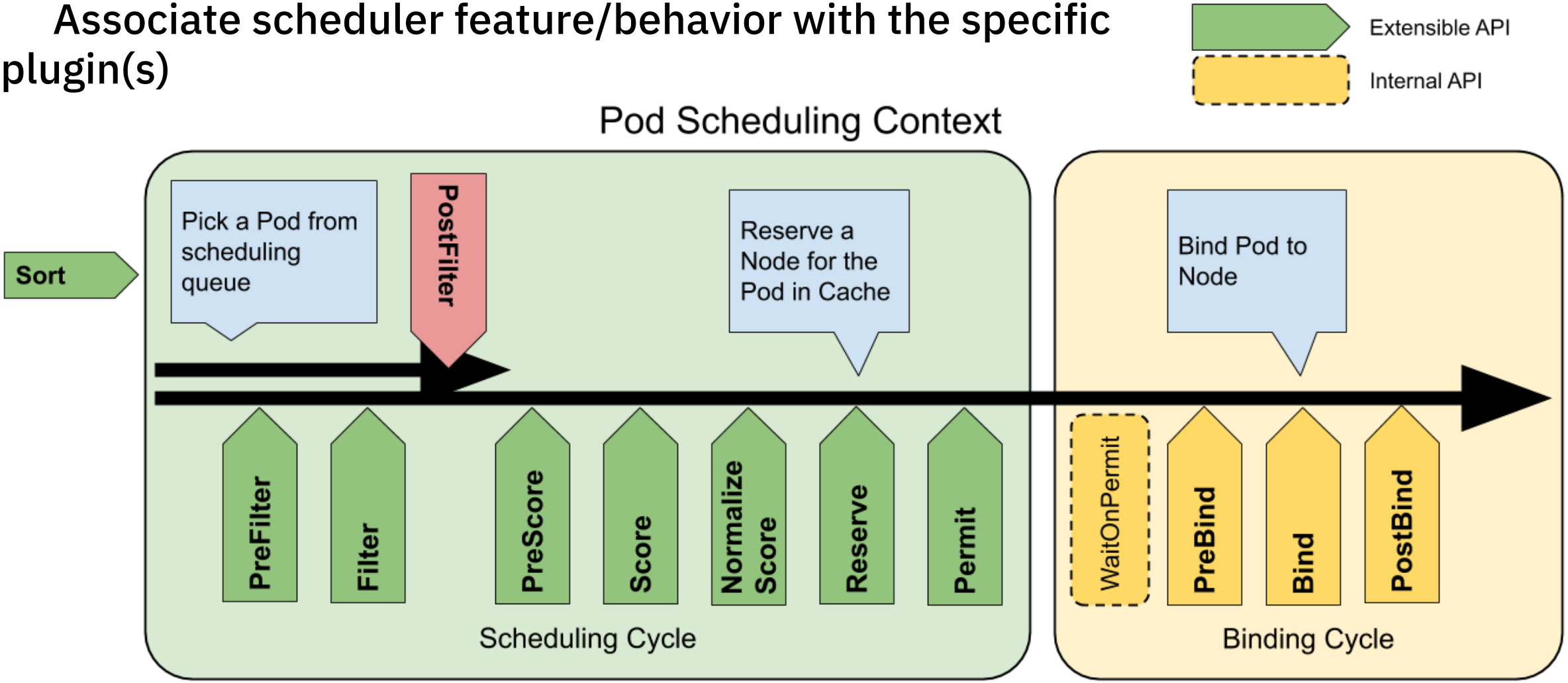
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pause
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pause
  template:
    metadata:
      labels:
        app: pause
    spec:
      schedulerName: binpack
      containers:
      - name: pause
        image: k8s.gcr.io/pause:3.2
```

File: **skipscore.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pause
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pause
  template:
    metadata:
      labels:
        app: pause
    spec:
      schedulerName: skip-score
      containers:
      - name: pause
        image: k8s.gcr.io/pause:3.2
```

Day 2 – Dive a bit deeper

Associate scheduler feature/behavior with the specific plugin(s)



Day 2 – Enabled plugin list

KubeCon CloudNativeCon North America 2020 Virtual

Scheduling plugins

The following plugins, enabled by default, implement one or more of these extension points:

- `SelectorSpread` : Favors spreading across nodes for Pods that belong to Services, ReplicaSets and StatefulSets Extension points: `PreScore` , `Score` .
- `ImageLocality` : Favors nodes that already have the container images that the Pod runs. Extension points: `Score` .
- `TaintToleration` : Implements taints and tolerations. Implements extension points: `Filter` , `Prescore` , `Score` .
- `NodeName` : Checks if a Pod spec node name matches the current node. Extension points: `Filter` .
- `NodePorts` : Checks if a node has free ports for the requested Pod ports. Extension points: `PreFilter` , `Filter` .
- `NodePreferAvoidPods` : Scores nodes according to the node annotation `scheduler.alpha.kubernetes.io/preferAvoidPods` . Extension points: `Score` .
- `NodeAffinity` : Implements node selectors and node affinity. Extension points: `Filter` , `Score` .
- `PodTopologySpread` : Implements Pod topology spread. Extension points: `PreFilter` , `Filter` , `PreScore` , `Score` .
- `NodeUnschedulable` : Filters out nodes that have `.spec.unschedulable` set to true. Extension points: `Filter` .
- `NodeResourcesFit` : Checks if the node has all the resources that the Pod is requesting. Extension points: `PreFilter` , `Filter` .
- `NodeResourcesBalancedAllocation` : Favors nodes that would obtain a more balanced resource usage if the Pod is scheduled there. Extension points: `Score` .
- `NodeResourcesLeastAllocated` : Favors nodes that have a low allocation of resources. Extension points: `Score` .
- `VolumeBinding` : Checks if the node has or if it can bind the requested volumes. Extension points: `PreFilter` , `Filter` , `Reserve` , `PreBind` .
- `VolumeRestrictions` : Checks that volumes mounted in the node satisfy restrictions that are specif to the volume provider. Extension points: `Filter` .
- `VolumeZone` : Checks that volumes requested satisfy any zone requirements they might have. Extension points: `Filter` .
- `NodeVolumeLimits` : Checks that CSI volume limits can be satisfied for the node. Extension points: `Filter` .
- `EBSLimits` : Checks that AWS EBS volume limits can be satisfied for the node. Extension points: `Filter` .
- `GCEPDLimits` : Checks that GCP-PD volume limits can be satisfied for the node. Extension points: `Filter` .
- `AzureDiskLimits` : Checks that Azure disk volume limits can be satisfied for the node. Extension points: `Filter` .
- `InterPodAffinity` : Implements inter-Pod affinity and anti-affinity. Extension points: `PreFilter` , `Filter` , `PreScore` , `Score` .
- `PrioritySort` : Provides the default priority based sorting. Extension points: `QueueSort` .
- `DefaultBinder` : Provides the default binding mechanism. Extension points: `Bind` .
- `DefaultPreemption` : Provides the default preemption mechanism. Extension points: `PostFilter` .

Day 2 – Disabled plugin list

- `NodeResourcesMostAllocated`: Favors nodes that have a high allocation of resources. Extension points: `Score`.
- `RequestedToCapacityRatio`: Favor nodes according to a configured function of the allocated resources. Extension points: `Score`.
- `NodeResourceLimits`: Favors nodes that satisfy the Pod resource limits. Extension points: `PreScore`, `Score`.
- `CinderVolume`: Checks that OpenStack Cinder volume limits can be satisfied for the node. Extension points: `Filter`.
- `NodeLabel`: Filters and / or scores a node according to configured label(s). Extension points: `Filter`, `Score`.
- `ServiceAffinity`: Checks that Pods that belong to a Service fit in a set of nodes defined by configured labels. This plugin also favors spreading the Pods belonging to a Service across nodes. Extension points: `PreFilter`, `Filter`, `Score`.

Day 2 – Wrap-up

- Understand plugins and plugin arguments
- Some global settings:
  - percentageOfNodesToScore
- What's new
  - DefaultTopologySpread: beta in 1.19
  - Prioritizing nodes based on volume capacity [KEP 1845]
  - Try out .status.nominatedNodeName as a shortcut [KEP 1923]
  - [Draft] Simplified version of topology manager in kube-scheduler [#1858]
  - [Draft] Add default node affinity constraints to NodeAffinity plugin [#95738]
- 🎉 Eventual goal: offer an out-of-box "multi-flavored" scheduler

Day 3 – Enthusiast / Innovator

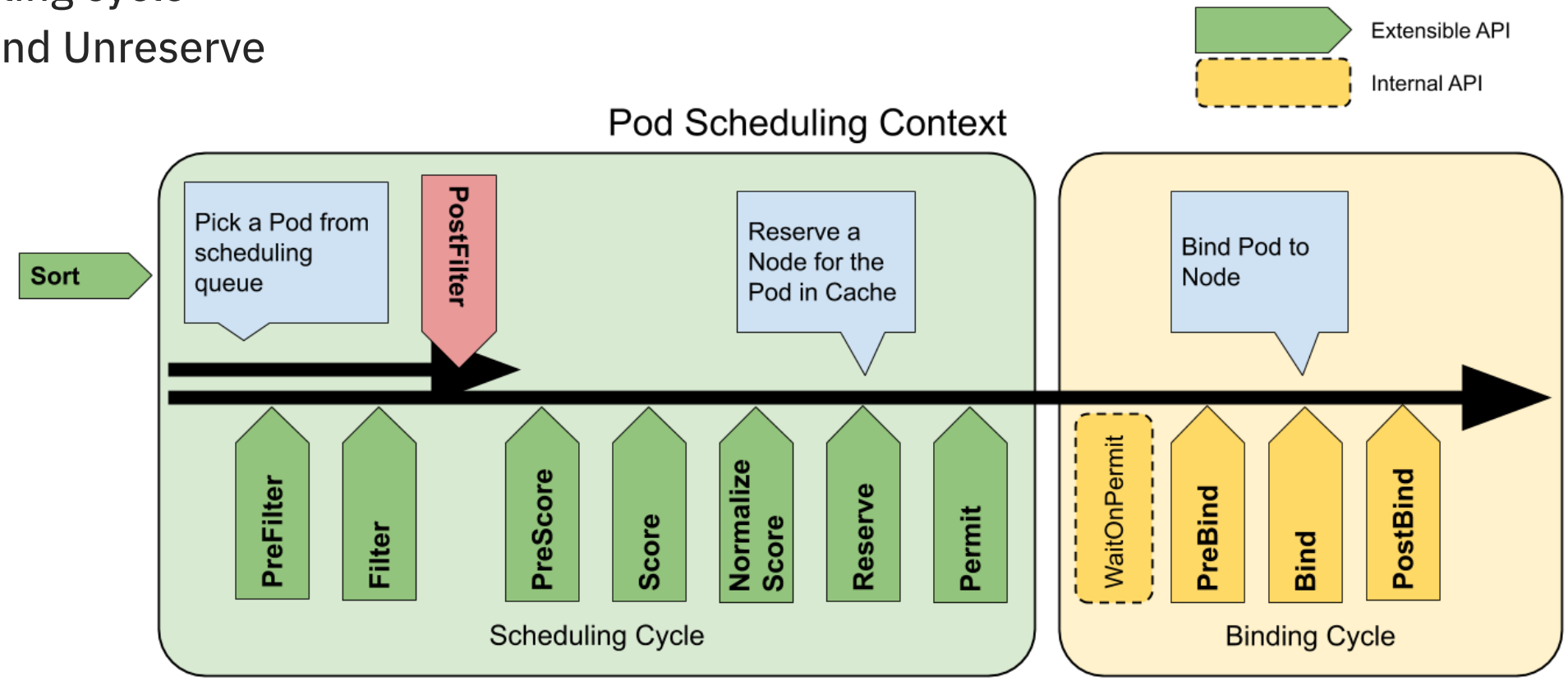
- Audience: scheduling enthusiast / innovator
- Goal: extend scheduler to fit diverse workloads, by writing as minimum code as possible
- Non-Goal: start from scratch to write a secondary scheduler

Day 3 – Deep dive into scheduler framework

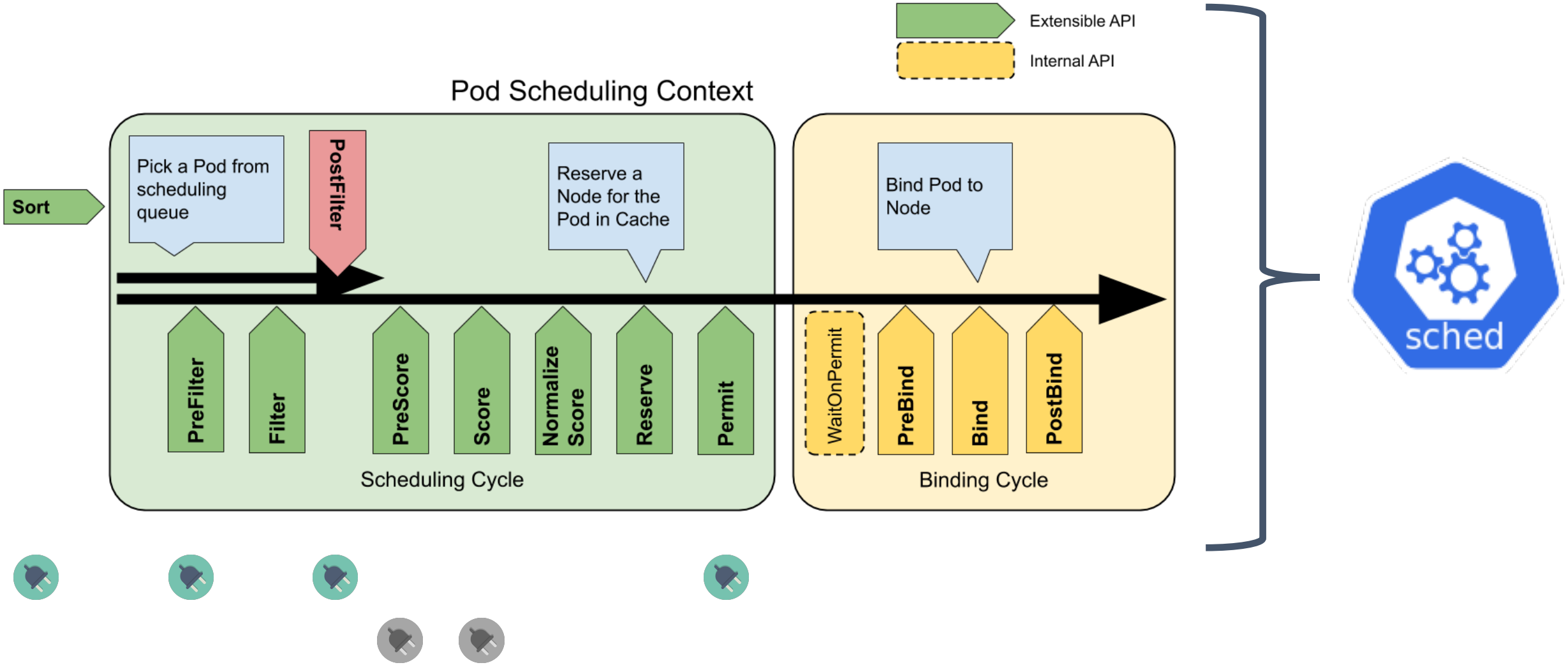
What's new

- Refined PostFilter in 1.19
- Permit in scheduling cycle
- Merge Reserve And Unreserve
- 🎉 GA in 1.20 🎉

Master details of each extension point



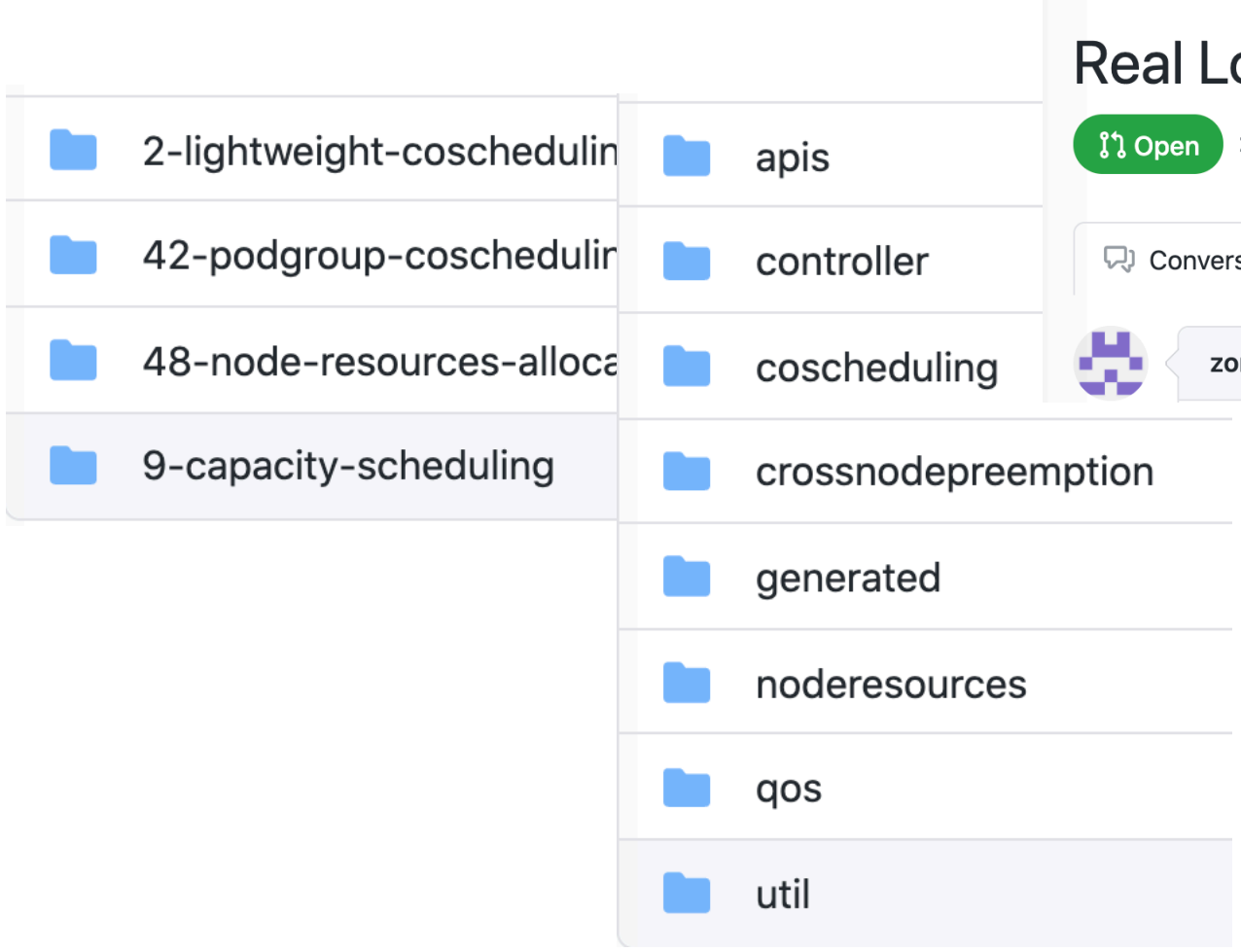
Day 3 – Build your own scheduler

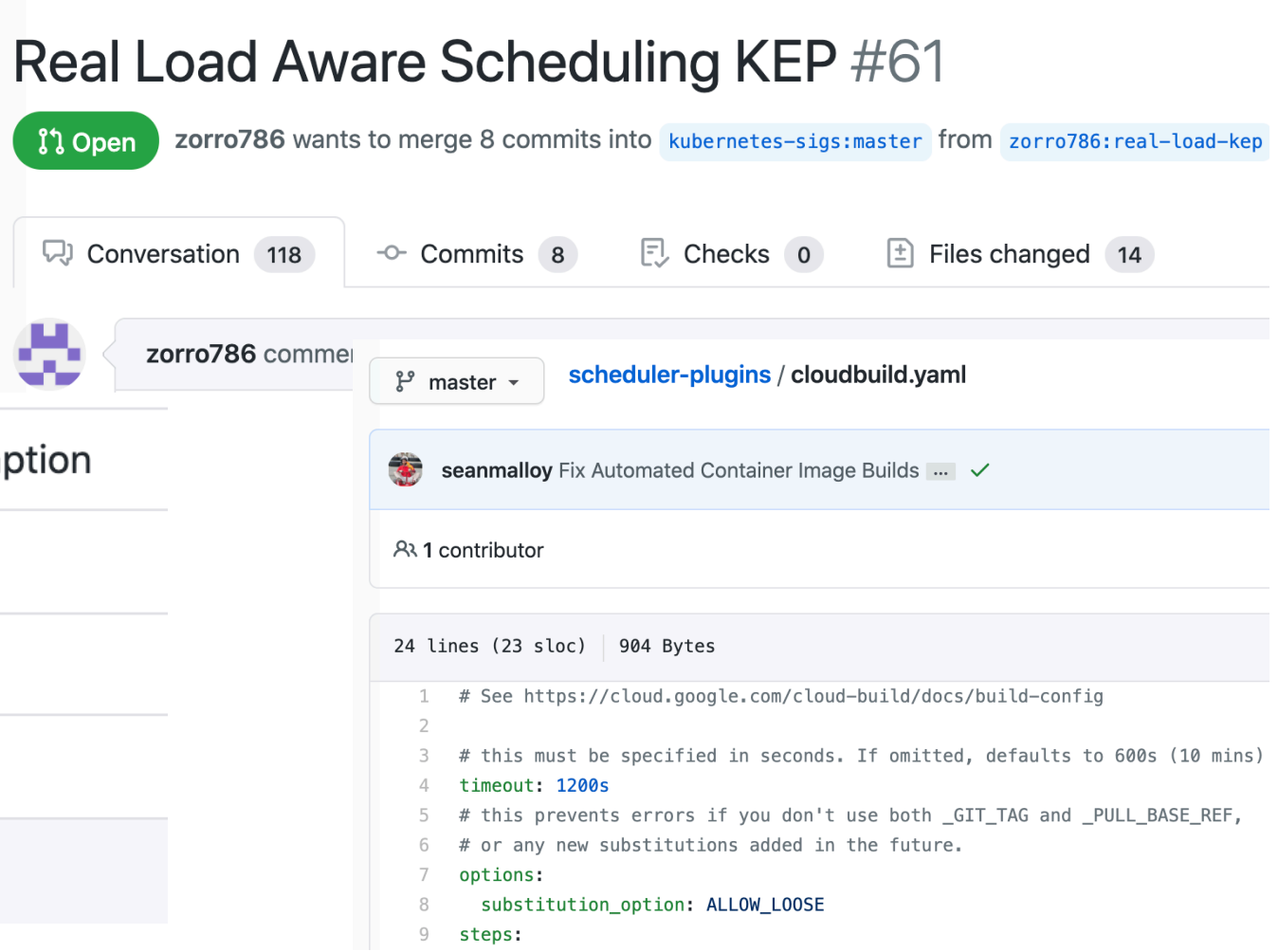


Day 3 – scheduler-plugins

KubeCon CloudNativeCon North America 2020 Virtual

- scheduler-plugins: a sig-scheduling sponsored project





Contact Us

KubeCon CloudNativeCon North America 2020 Virtual

- SIG-Chairs
  - @ahg-g, Google
  - @Huang-Wei, IBM
- Home page
- Slack channel: #sig-scheduling
- Mailing list
- Weekly meeting

Q & A


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual