



AWS
re:Invent

CON418-R

Deep dive : Kubernetes auto scaling

Chance Lee

Solutions Architect
Amazon Web services

Kubernetes auto scaling intro

	Nodes	Pods
Horizontal	# of nodes in node group (CA)	# of pods (HPA)
Vertical	Right resources for a node (N/A)	Right resources for a pod (VPA)

Kubernetes auto scaling intro

Cluster Autoscaler (CA)

Get the right number of nodes required for your workloads

Horizontal Pod Autoscaler (HPA)

Get the right number of pods required for your workloads

Vertical Pod Autoscaler (VPA)

Get the right resource (CPU/memory) requests and limits for containers in a pod

Cluster Autoscaler

Cluster Autoscaler

- Automatically adjusts the size of a node group (Amazon ASG)
- Scale up when pods failed to be scheduled due to insufficient resources
 - Checks every 10 sec (--scan-interval flag)
- Scale down when node is considered unneeded in 10 minutes
 - The sum of CPU & memory of all pods running is smaller than 50% of the node's allocatable
 - All pods running on the node can be moved to other nodes (except daemonsets)
- Some types of pods can prevent CA from removing a node
 - Annotation: "Cluster-autoscaler.kubernetes.io/scale-down-disabled" : "true"
 - Pods with restrictive [PodDisruptionBudget](#)
 - Pods that can't be moved due to lack of resources, non-matching node selectors, affinity

Cluster Autoscaler

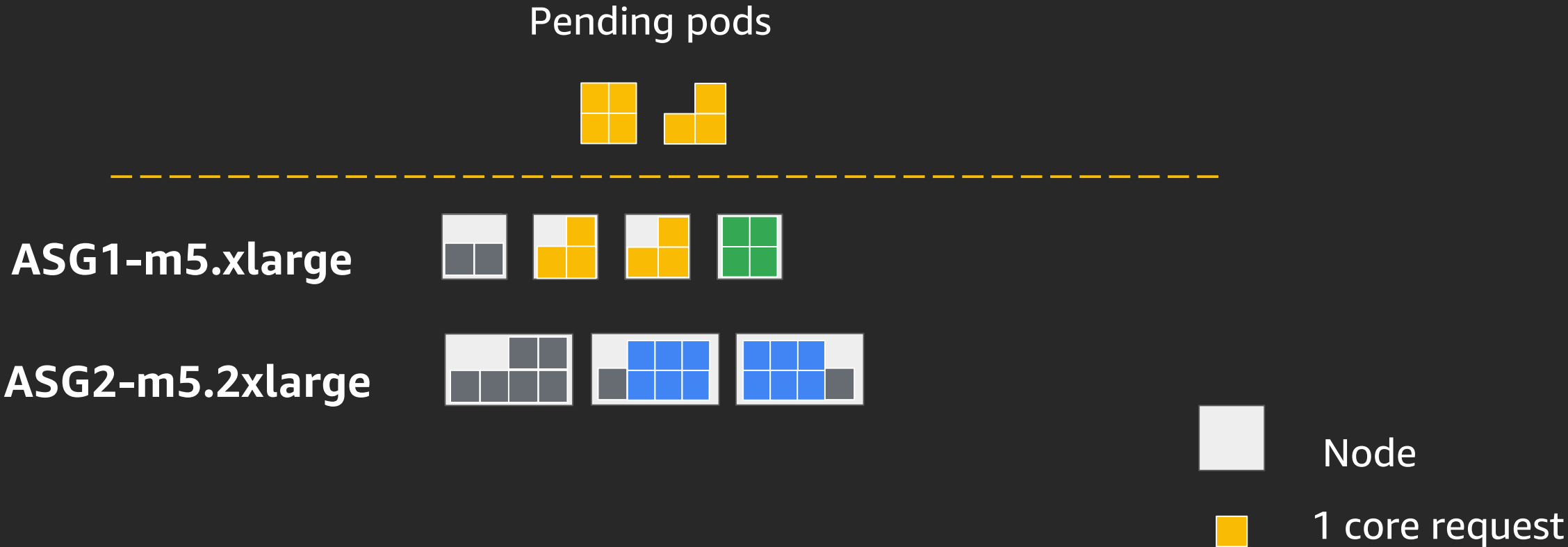
- CA runs as a deployment
- Check if your cloud provider's quota is big enough
- Metric-based cluster autoscalers are not compatible
 - HPA and VPA are for optimizing size and number of your pod
- CA directly affects your bill!

CA on AWS gotchas

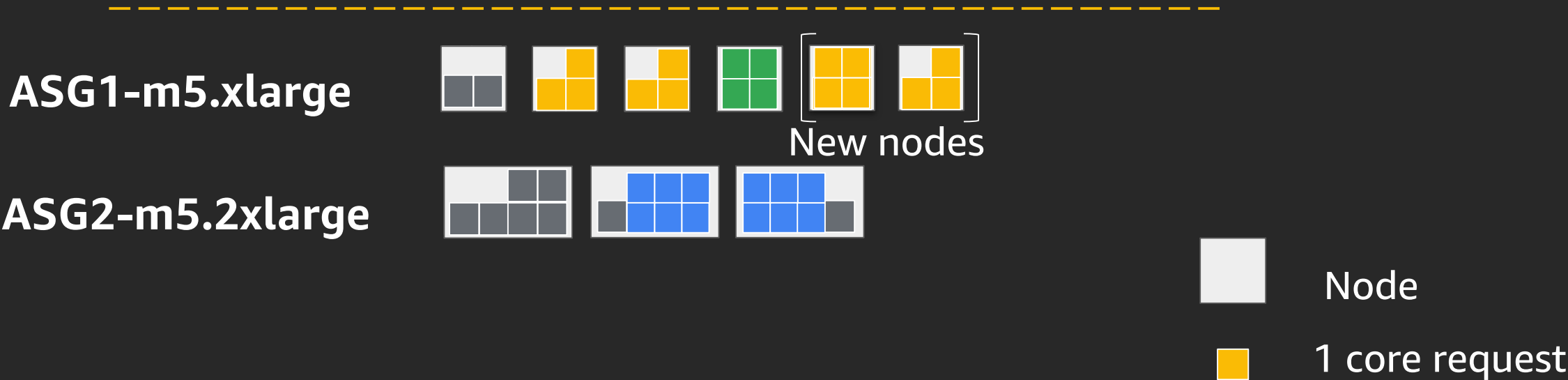
- Auto-discovery setup is preferred (`--node-group-auto-discovery` flag)
 - If you want to adjust the min/max of the group, adjust size on ASG directly, CA will fetch latest change when talking to ASG
 - The min/max configuration change in CA won't make the corresponding change to ASG
- MixedInstancesPolicy (The minimum version is v1.14.x.)
 - Diversification across on-demand and spot instances, instance types in a single ASG
 - The instance types should have the same amount of RAM and number of vCPU
 - See AWS CloudFormation example

<https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/cloudprovider/aws/MixedInstancePolicy.md>

CA scale-up



CA scale-up



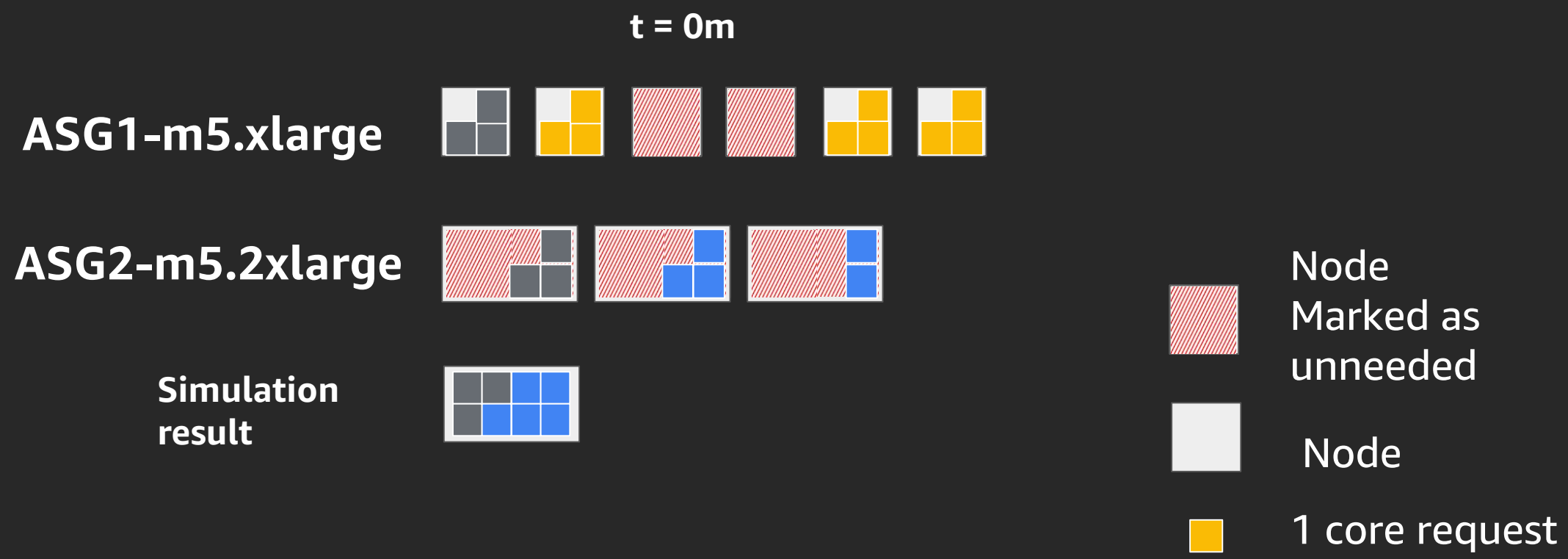
CA scale-up

- Which node group do we scale-up
- Can we customize it?

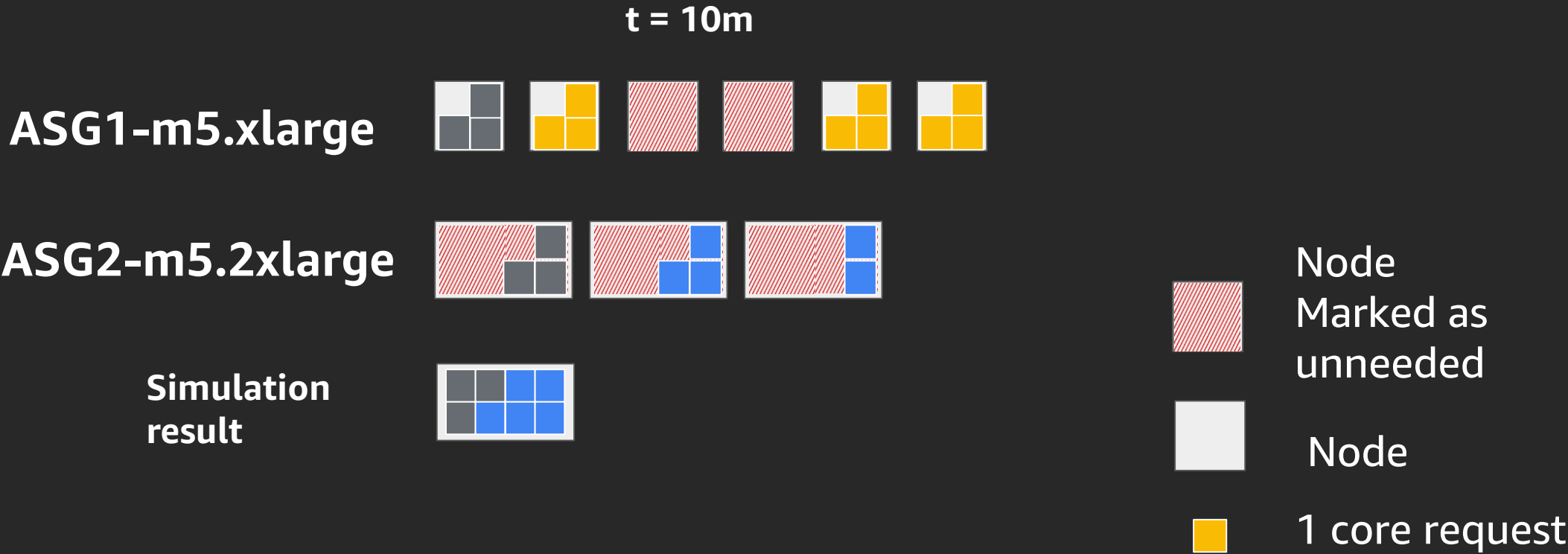
CA scale-up: Expanders

- If there is more than 1 node group, it has to decide which to expand
- Expanders provides strategies for selecting the node group
 - Random (by default)
 - Least wastage
 - The node group that will have the least idle resource after scale-up
 - Most pods
 - The node group that would be able to schedule the most pods
 - Use this when you absolutely need to schedule as many pods as possible quickly
 - Priority
 - The node group that has the highest priority assigned by the user

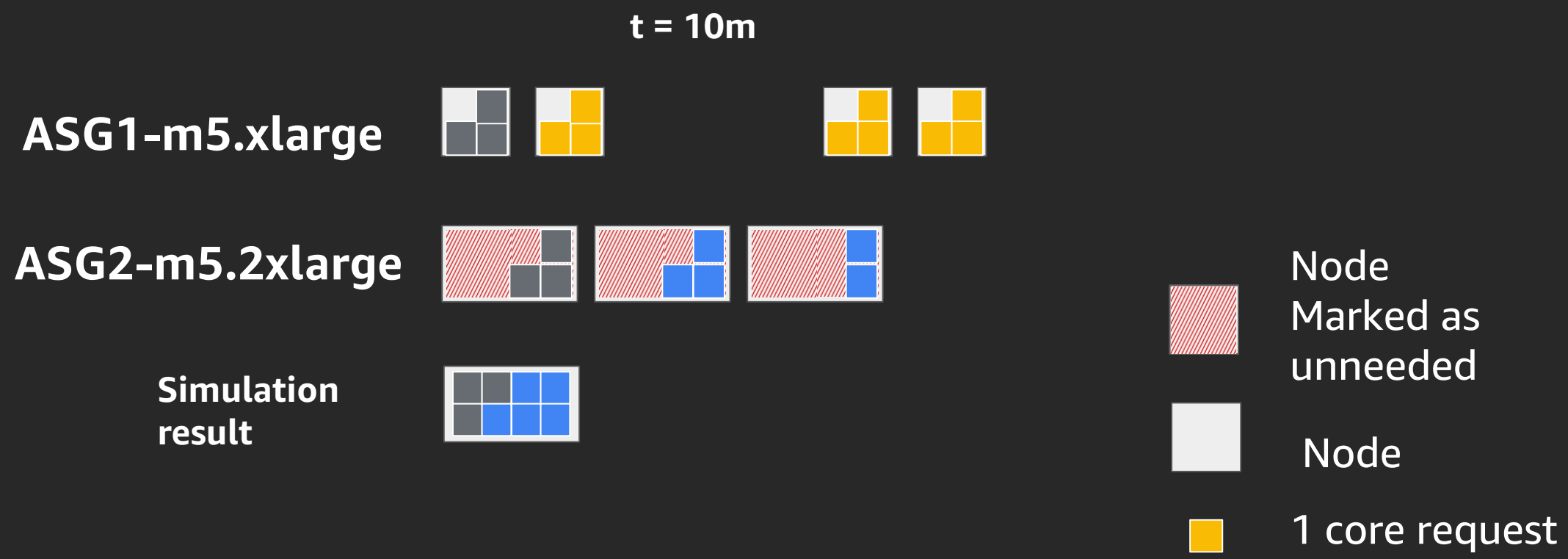
CA scale-down



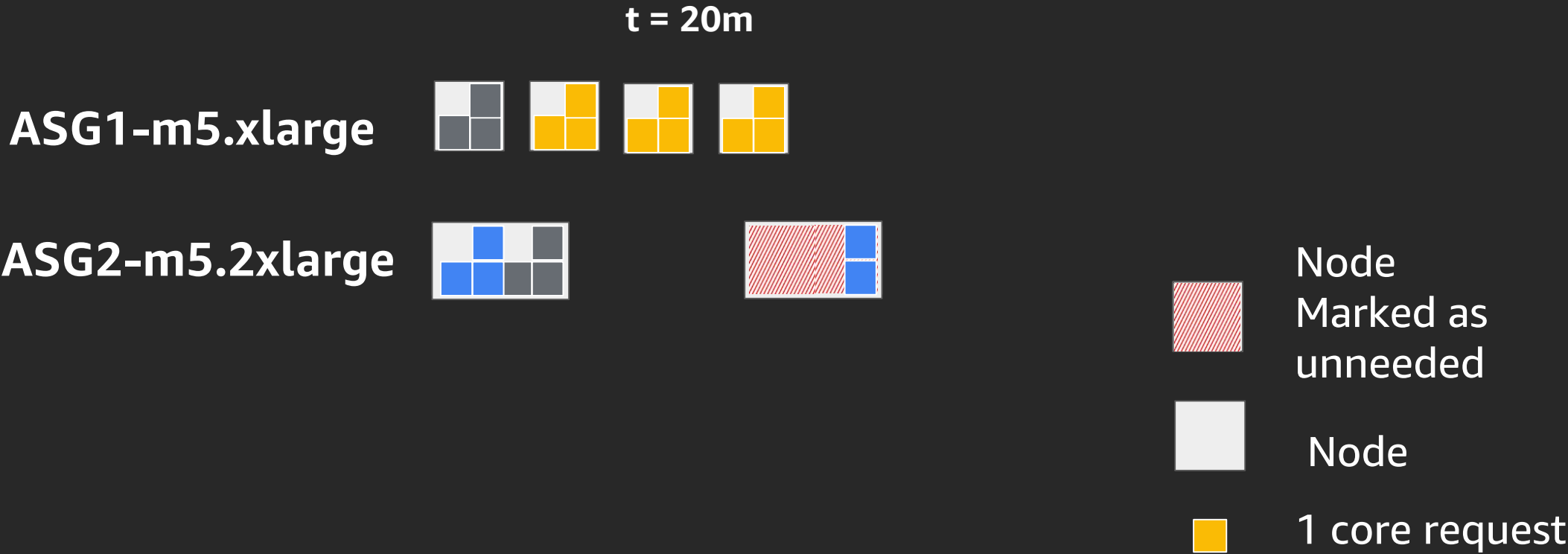
CA scale-down



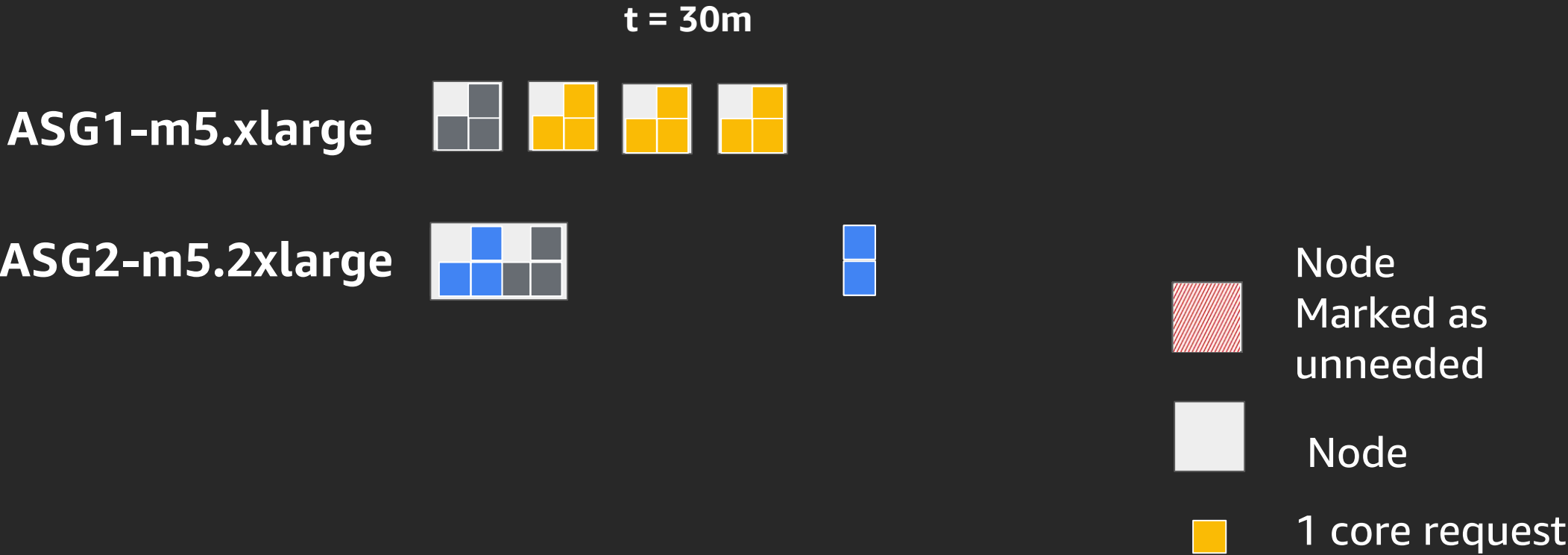
CA scale-down



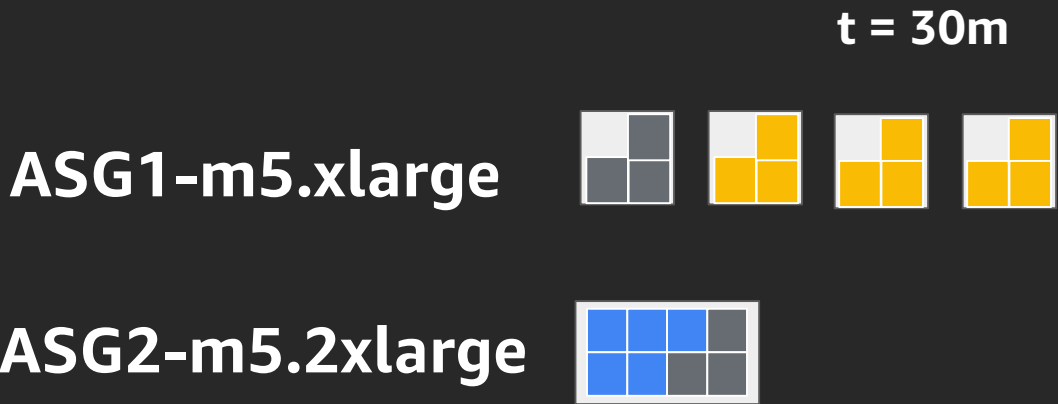
CA scale-down






CA scale-down



CA scale-down



-  Node
Marked as
unnneeded
-  Node
-  1 core request

CA scale-down tuning

- The tunables are set to reasonable defaults
- However, they can be optimized for requirements

CA tunables

- scale-down-unneeded-time
- scale-down-delay-after-delete
- scale-down-utilization-threshold
- max-empty-bulk-delete

CA tunables --scale-down-unneeded-time

- How long a node should be unneeded before it is eligible for scale in
 - 10 mins. by default
 - If you reduce it, faster scale in but could be too aggressive
 - If you increase it, more graceful but nodes could be wasted

CA tunables --scale-down-delay-after-delete

- How long after node deletion that scale-down evaluation resumes
 - Default to scan-interval (10 sec.)
 - If you reduce it, faster scale in but could be too aggressive
 - If you increase it, more graceful but nodes could be wasted

CA tunables --scale-down-utilization-threshold

- The utilization of a node has to be below this threshold to be considered for scale-down
 - 50% by default
 - If you reduce it, it could cause tighter packing of nodes
 - If you increase it, it could cause looser packing of nodes or over-provisioning

CA tunables --max-empty-bulk-delete

- Denotes how many empty nodes can be terminated in bulk
 - Up to 10 nodes by default
 - If you reduce it, it provides more graceful node scale-down
 - If you increase it, faster cleanup is available in larger clusters

CA tunables --max-empty-bulk-delete

Make base scale-down faster

- `--scale-down-unneeded-time`
- `--scale-down-utilization-threshold`
- `--scale-down-unready-time`

Make serial scale-downs faster

- `--scale-down-delay-after-add`
- `--max-empty-bulk-delete`
- `--scan-interval`
- `--scale-down-delay-after-failure`
- `--scale-down-delay-after-delete`
- `--unremovable-node-recheck-timeout`

Horizontal Pod Autoscaler

Kubernetes auto scaling intro

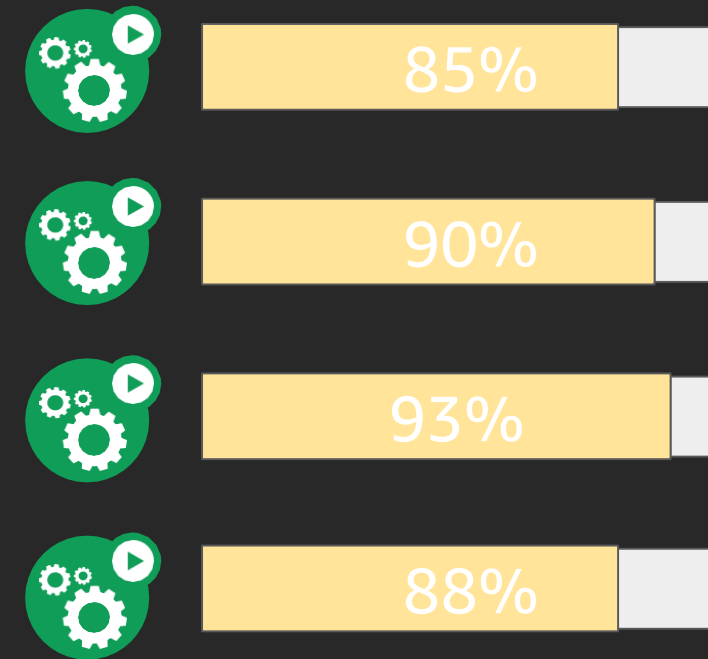
	Nodes	Pods
Horizontal	# of nodes in node group (CA)	# of pods (HPA)
Vertical	Right resources for a node (N/A)	Right resources for a pod (VPA)

HPA intro

- Automatically scales the number of replica in deployment
- Based on target metrics
 - CPU utilization by default
 - Custom & external metrics

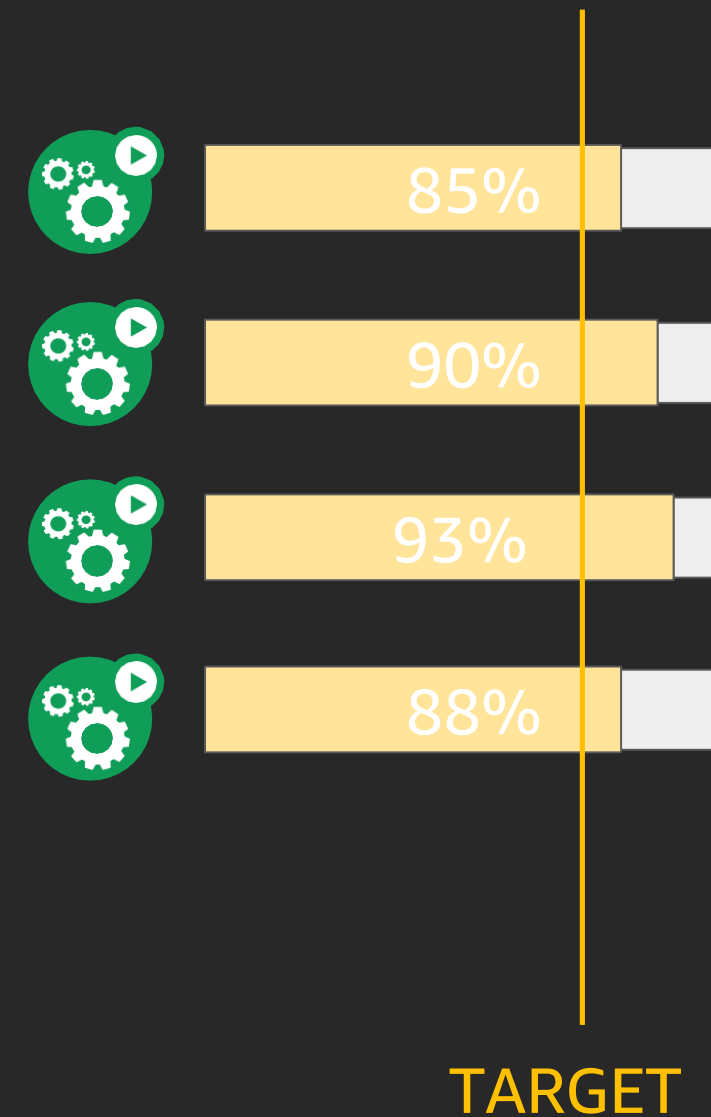
HPA intro

- If pods are heavily loaded, then starting new pods may bring average load down
- If pods are barely loaded, then stopping pods will free resources



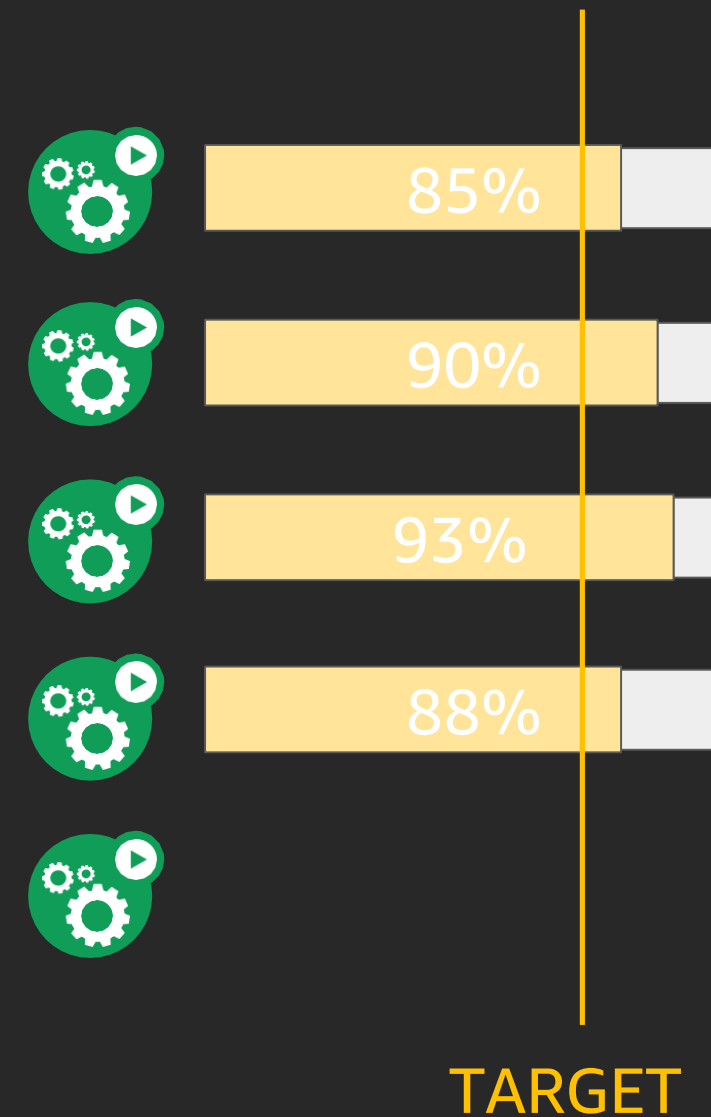
HPA intro

- If pods are heavily loaded, then starting new pods may bring average load down
- If pods are barely loaded, then stopping pods will free resources
- Specify the target for the load
 - Ex: target = CPU utilization 70%



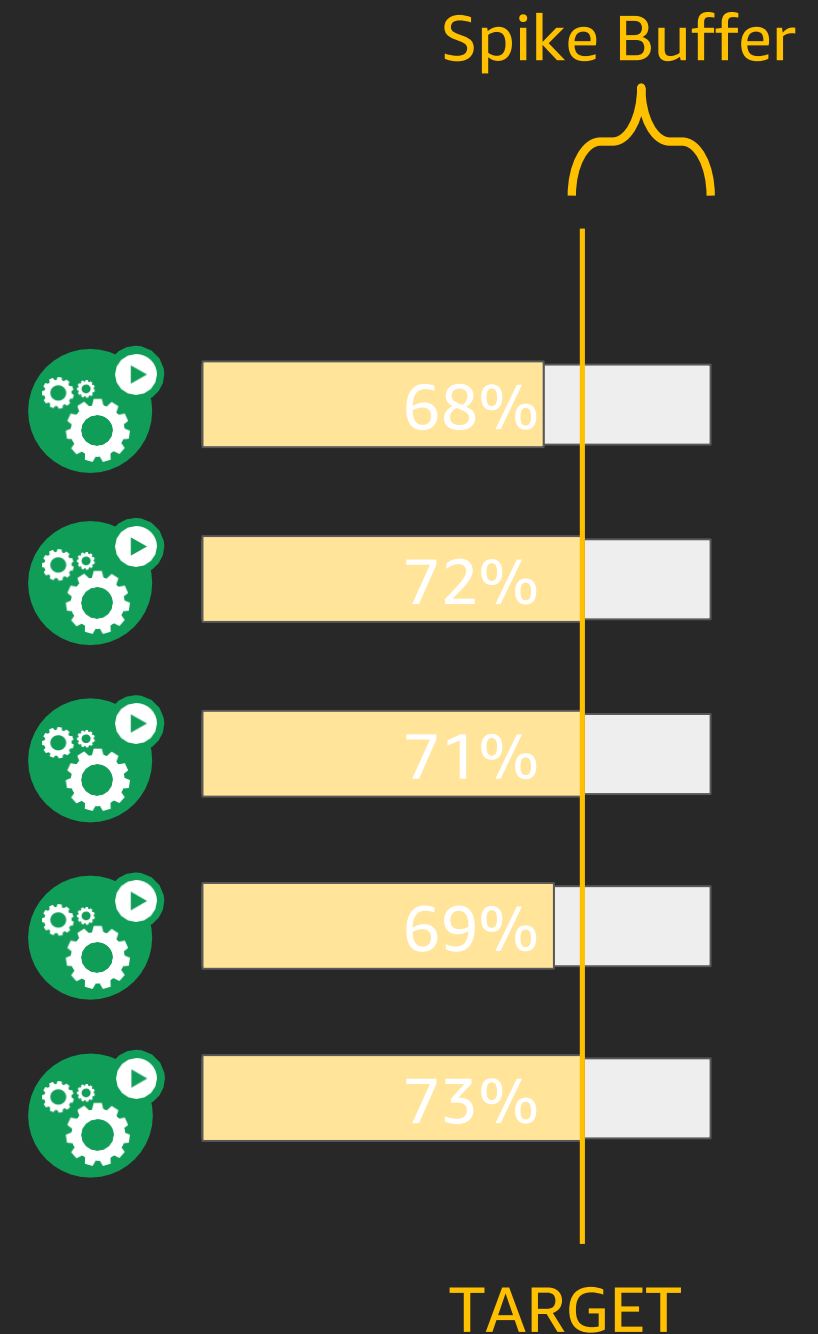
HPA intro

- If pods are heavily loaded, then starting new pods may bring average load down
- If pods are barely loaded, then stopping pods will free resources
- Specify the target for the load
 - Ex: target = CPU utilization 70%



HPA intro

- If pods are heavily loaded, then starting new pods may bring average load down
- If pods are barely loaded, then stopping pods will free resources
- Specify the target for the load
 - Ex: target = CPU utilization 70%
 - Too small spike buffer may overload your replicas
 - Too big buffer causes resource waste



Replica count

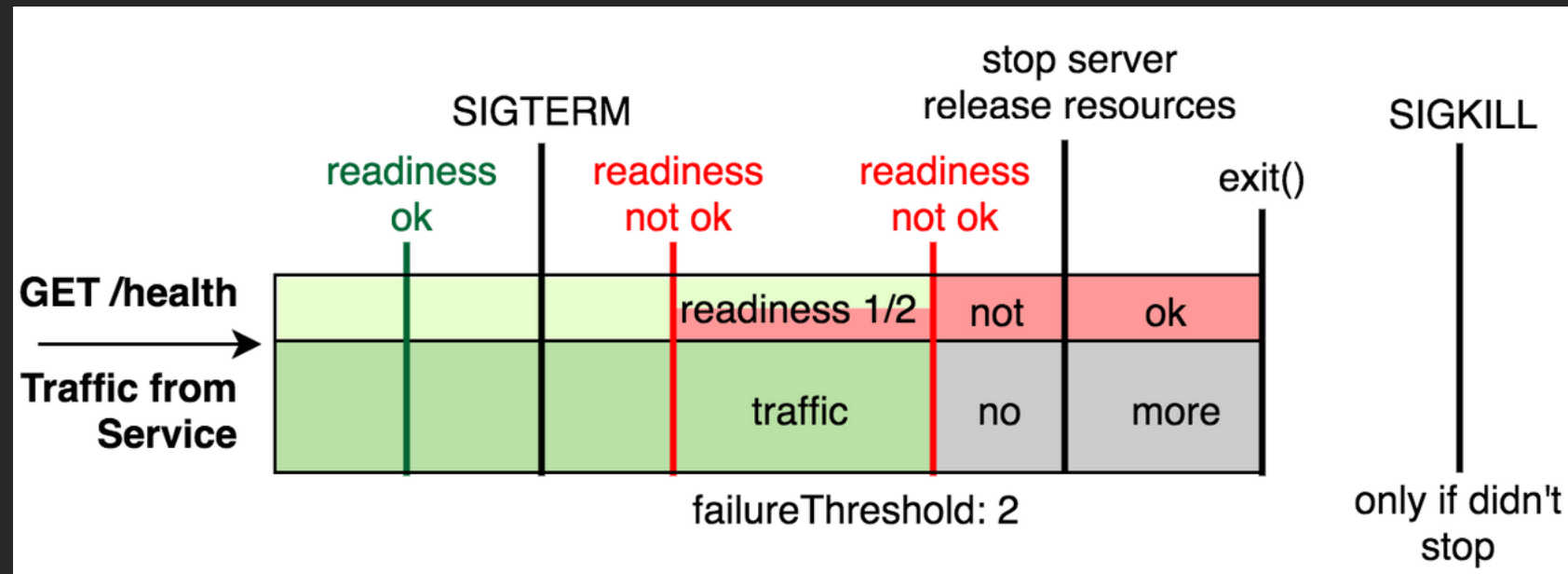
- CPU usage (Util) = $\text{Current CpuConsumption} / \text{PodCpuRequest}$
- Replica count = CPU usage sum of all pods in deployment / target
 - Pod 1 = 70%
 - Pod 2 = 80%
 - Sum = 150%
 - Target (desiredMetricValue) = 50%
 - Replica count = 3

HPA control loop

- HPA checks CPU usage in metric server every 15 seconds
 - Controller manager : `--horizontal-pod-autoscaler-sync-period` flag
 - Metric server should be installed
- HPA backs off for 5 mins. before taking another downscale
 - Kube-controller-manager : `--horizontal-pod-autoscaler-downscale-stabilization`

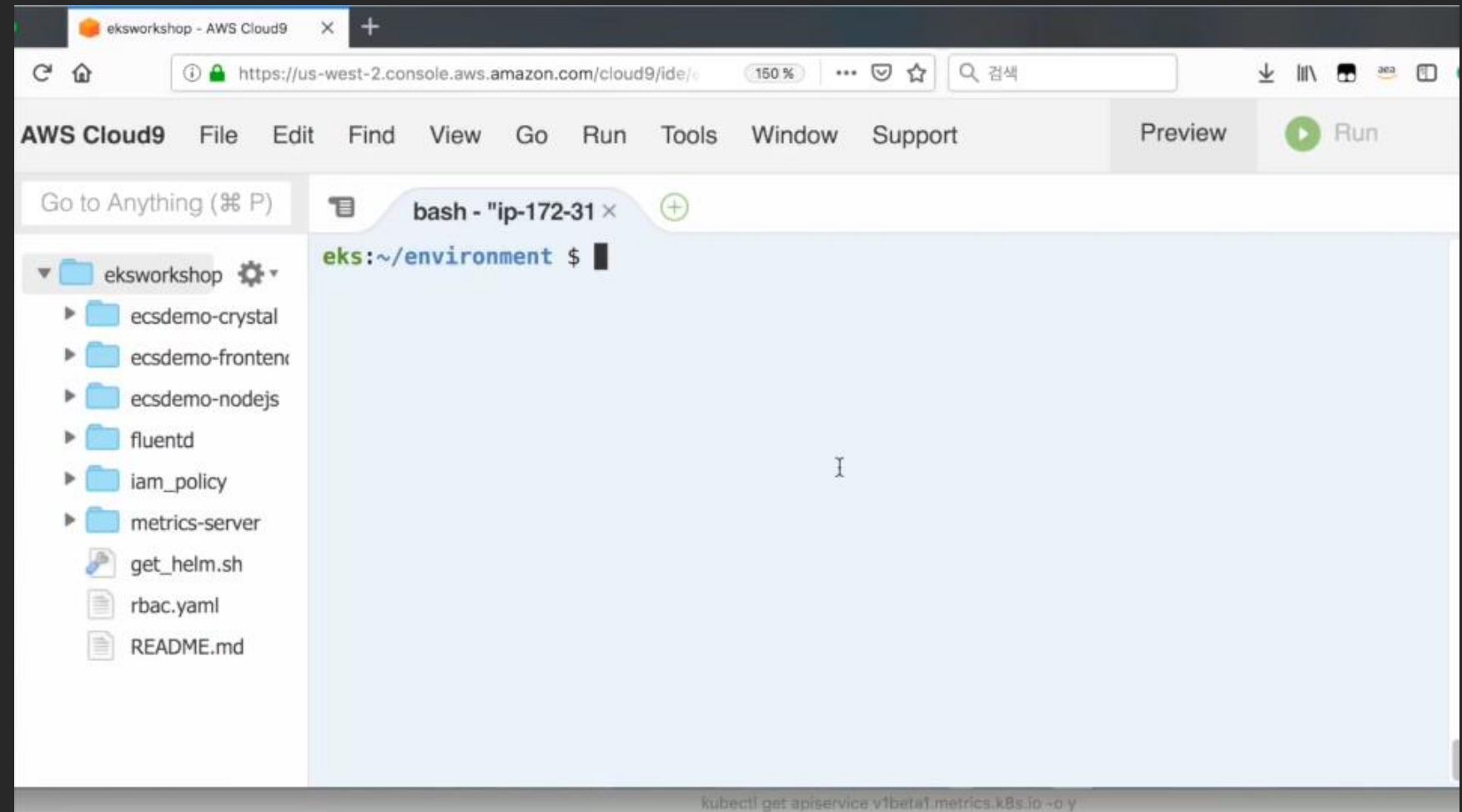
HPA graceful scale-down

- Handle pod shutdown (SIGTERM) for graceful shutdown
 - Finish all in-flight requests being currently processed
 - Wait for requests that may arrive after the pod termination begins
 - Define readiness, liveness probes



HPA demo

- Check metric server status
- Create sample php-apache deployment and service
- Create HPA
 - CPU 50% target, min=1, max=10
- Run log-generator
- Check scaling status
- Stop log-generator
- Check scaling status



Custom metric

- The API aggregation layer should be enabled
- The corresponding APIs are registered
 - Metrics.k8s.io API, provided by metrics-server
 - Custom.metrics.k8s.io, provided by vendor's adapter API server
 - External.metrics.k8s.io provided by custom metrics adopter above

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/#autoscaling-on-multiple-metrics-and-custom-metrics>

Custom metric Amazon CloudWatch SQS log example

<https://aws.amazon.com/blogs/compute/scaling-kubernetes-deployments-with-amazon-cloudwatch-metrics/>

Vertical Pod Autoscaler

Kubernetes auto scaling intro

	Nodes	Pods
Horizontal	# of nodes in node group (CA)	# of pods (HPA)
Vertical	Right resources for a node (N/A)	Right resources for a pod (VPA)

Vertical Pod Autoscaler

Are you confident your resource requests are correct?

```
apiVersion: apps/v1
kind: Deployment
...
template:
  metadata:
    labels:
      app: worker
  spec:
    containers:
      - name: worker
        resources:
          requests:
            memory: "100M"
            cpu: "250m"
```

Vertical Pod Autoscaler

Are you confident your resource requests are correct **over time**?

```
apiVersion: apps/v1
kind: Deployment
...
template:
  metadata:
    labels:
      app: worker
  spec:
    containers:
      - name: worker
        resources:
          requests:
            memory: ??
            cpu: ??
```

Everything is changing

Daily/weekly traffic patterns

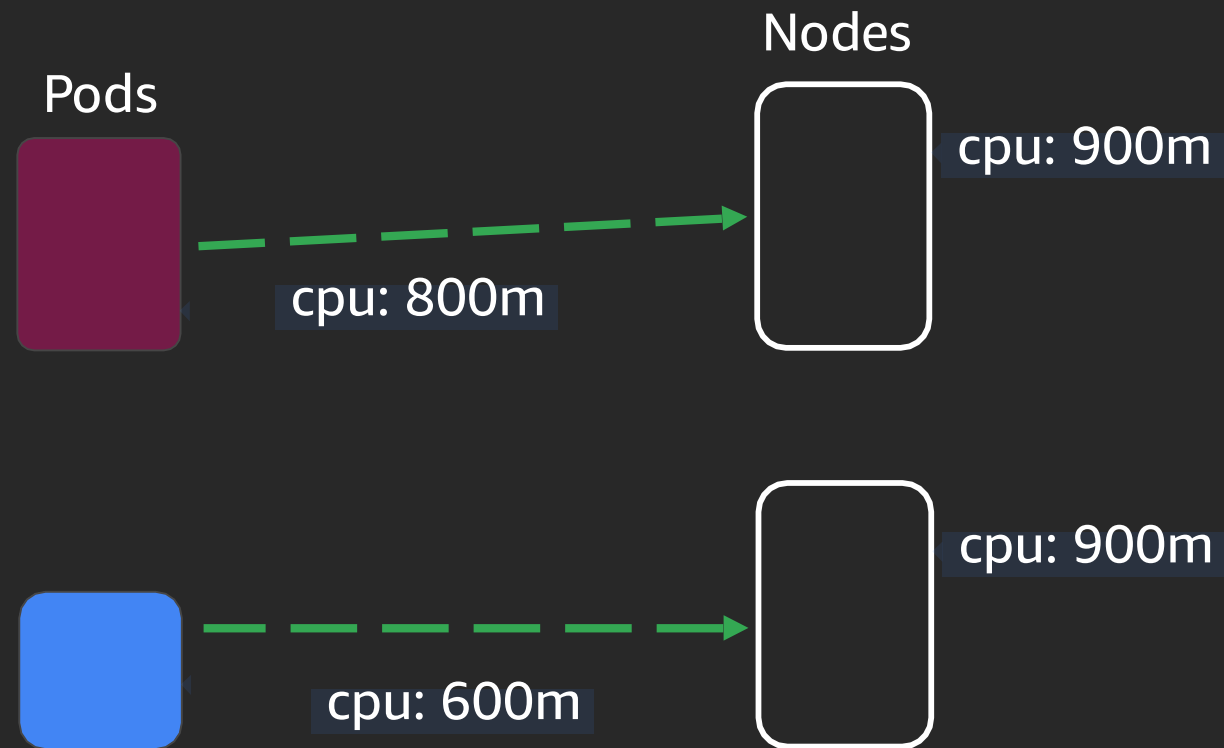
User base growing over time

App lifecycle phases with different resources needs

Will it fit?



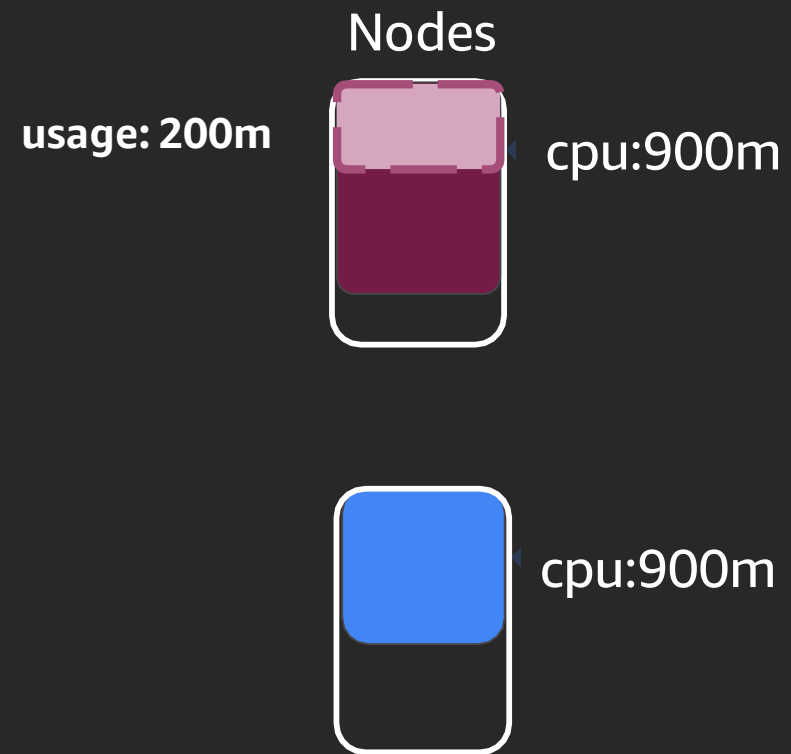
Kubernetes
scheduler



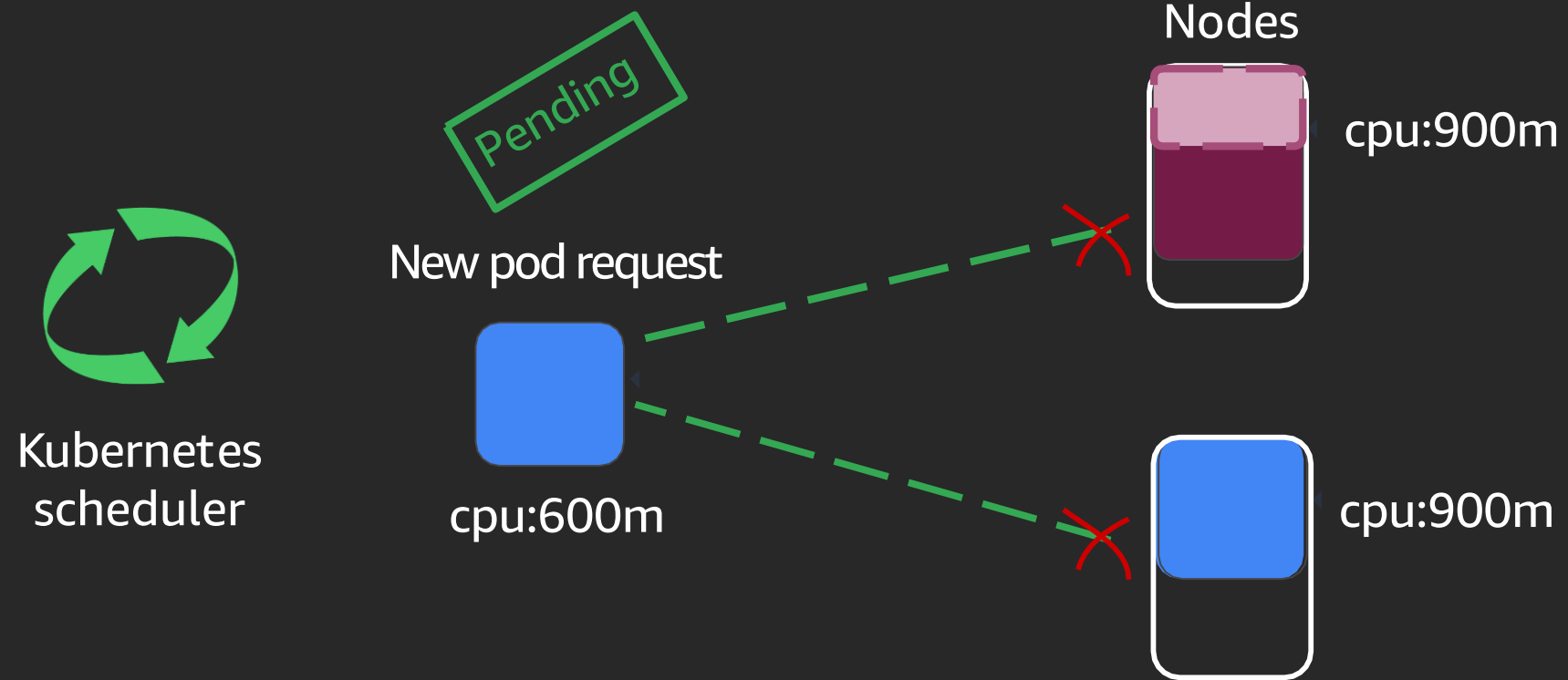
Will it fit?



Kubernetes
scheduler



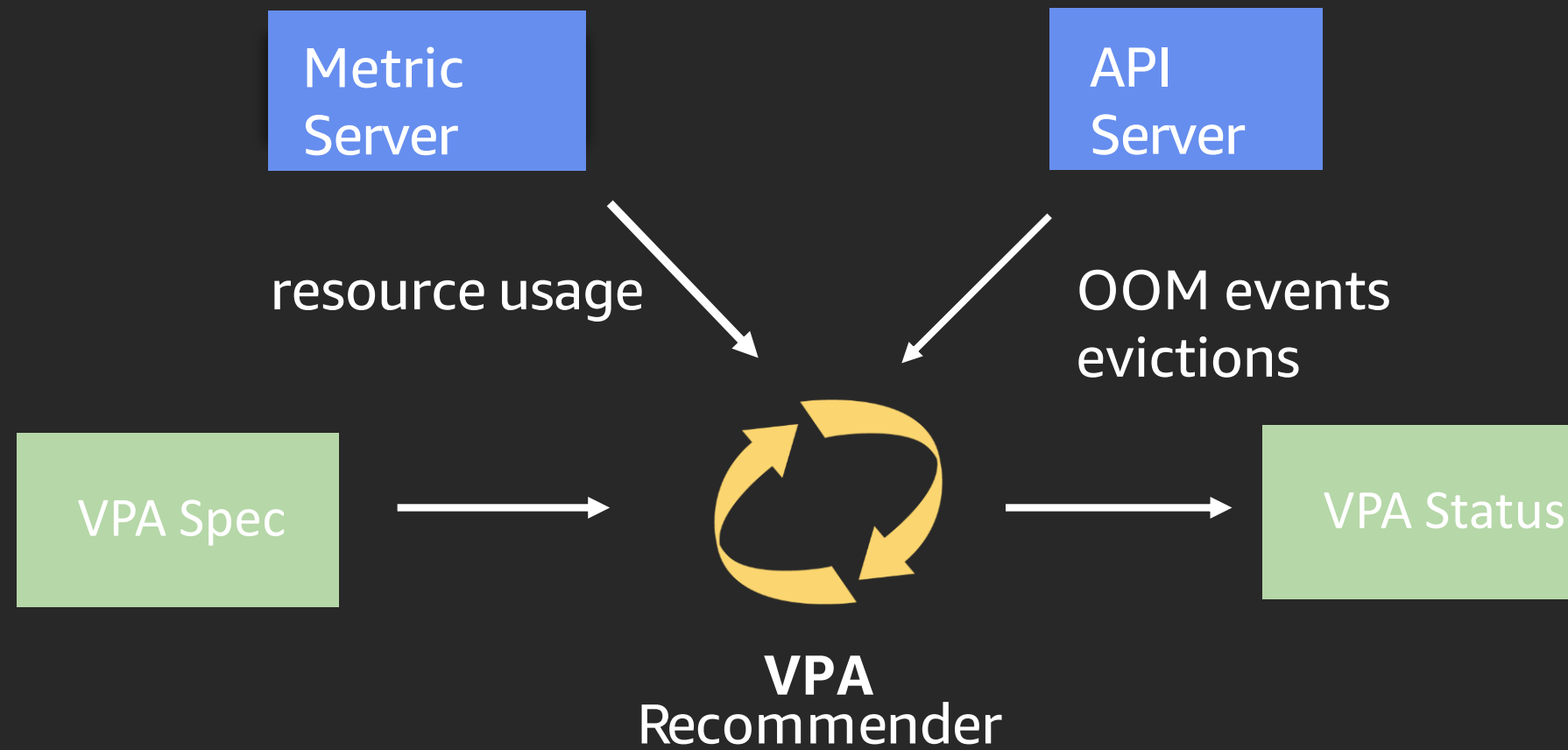
Will it fit?



Vertical Pod Autoscaler

- Observe usage
- Recommends resources
- Update resources
- Target
 - Deployment
 - DaemonSet
 - StatefulSet

Watch and recommend



Components

- **Recommender**
 - Gets pod's utilization and OOM events from metric & API server
 - Provides recommended CPU & memory and store the recommendation on VPA object
- **Updater**
 - Decide which pods should be restarted based on recommendation by recommender
 - Respects the pod disruption budget
- **Admission controller**
 - Apply recommendation on pod admission

VPA mode

- Off (recommendation only)
 - VPA publishes only recommended pod size, doesn't actually change anything
- Initial (initialization only)
 - VPA resizes pods only when pods are created
- Auto
 - VPA resizes pods by restarting existing pods
 - When your pods can be restarted and you gained enough confidence

VPA in production

- Start with recommendation-only mode
 - `Kubectl get vpa myvpa --output yaml`
- Define Pod Disruption Budgets when using AUTO mode
 - VPA resizes pods only when pods are created
- Set minimum and maximum container sizes in VPA
- Update recommended container size in deployment spec
- Be aware known limitations of VPA

References

- <https://eksworkshop.com>
- <https://aws.amazon.com/blogs/containers/cost-optimization-for-kubernetes-on-aws/>
- <https://github.com/kubernetes/autoscaler>
- <https://kubernetes.io/docs/setup/release/notes/>
- <https://github.com/kubernetes/community/tree/master/sig-autoscaling>

Thank you!

Chance Lee

changsul@amazon.com



Please complete the session survey in the mobile app.