THE SCALE FACTORY

EFFICIENT KUBERNETES SCALING **USING KARPENTER_**

Marko Bevc

WHSmith

TAXI
WW55 CAB

ABOUT
ME_

- Head of Consultancy at The Scale Factory (B2B SaaS consultancy, AWS Advanced consulting partner and K8s service provider)
- Ops background, wearing different hats, engaged with many different technologies
- Open source contributor, maintainer and supporter
- HashiCorp Ambassador, OpenUK Ambassador
- Certifications and competencies: AWS, CKA, RHEL, HCTA
- Fan of automation/simplifying things, hiking and travelling

@_MarkoB | @marko@hachyderm.io

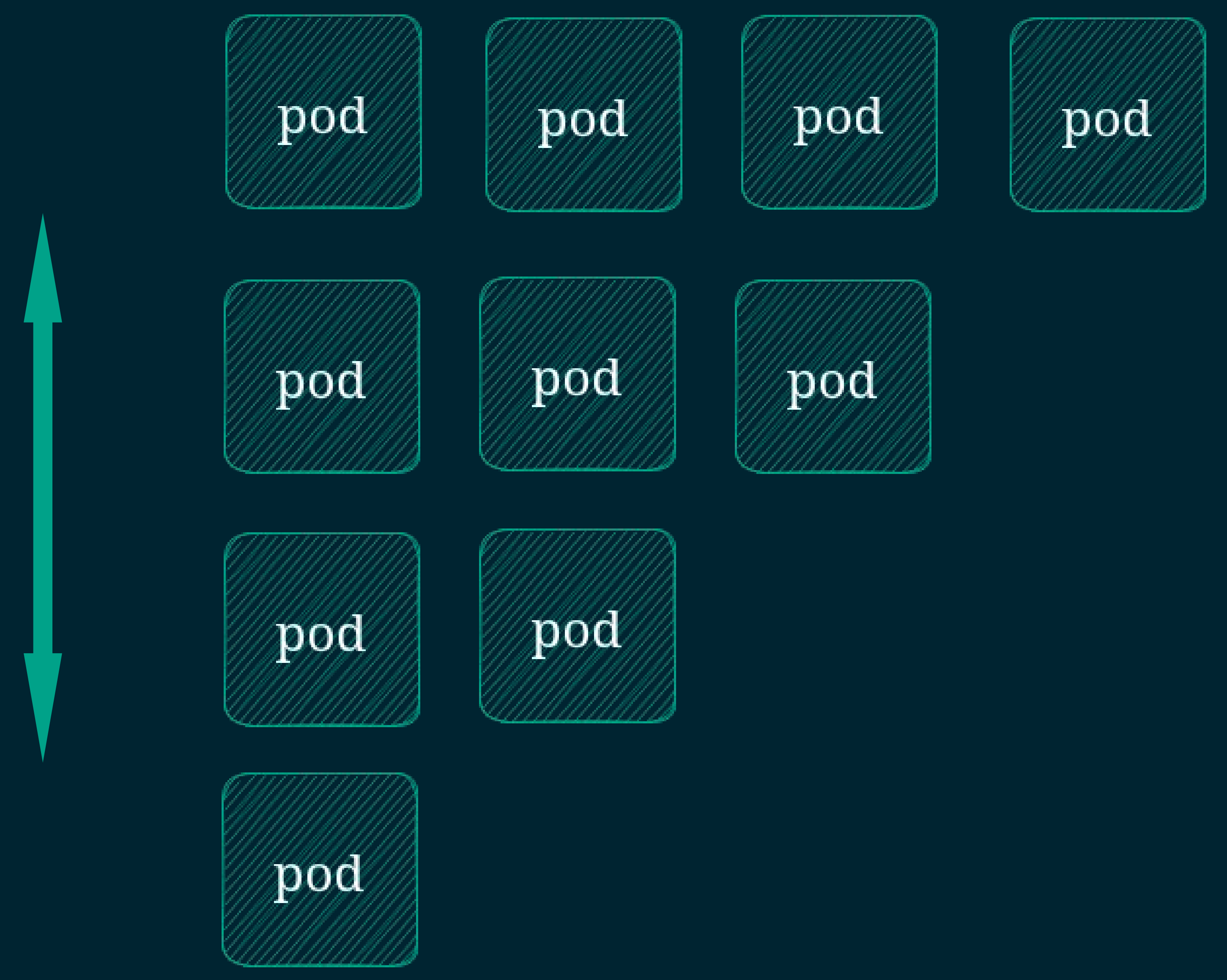
https://www.linkedin.com/in/marko-bevc/

KUBERNETES SCALING_

- None out of the box – manual
- Kubernetes resources:
  - Pods – the smallest execution unit
  - Nodes – compute/instances to run Pods on
  - Other: storage, network, etc.

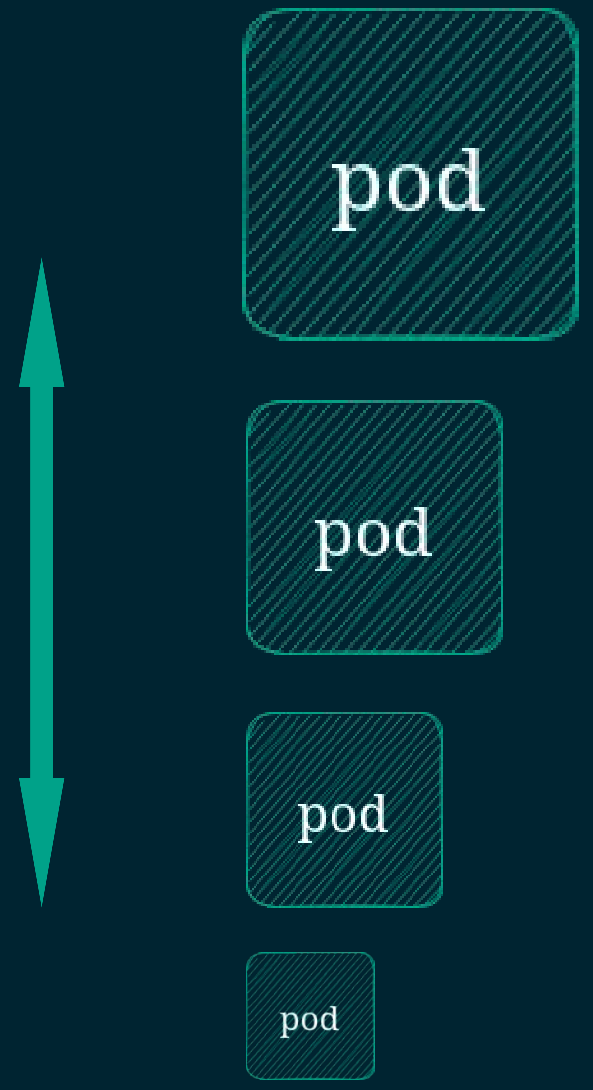
HPA
CONCEPT_

- Horizontal Pod Autoscaler
- Adding more instances(e.g. Pods)
- Doesn’t apply to non-scalable objects (e.g. DaemonSet)
- Target observed metrics (i.e. average CPU or memory utilization)
- Scaling **out**



VPA
CONCEPT_

- Vertical Pod Autoscaler
- Adjusting size/power (e.g. resources/limits)
- “Right-sizing” your workloads to actual usage
- Most commonly used on a Deployment objects
- Scaling **up**

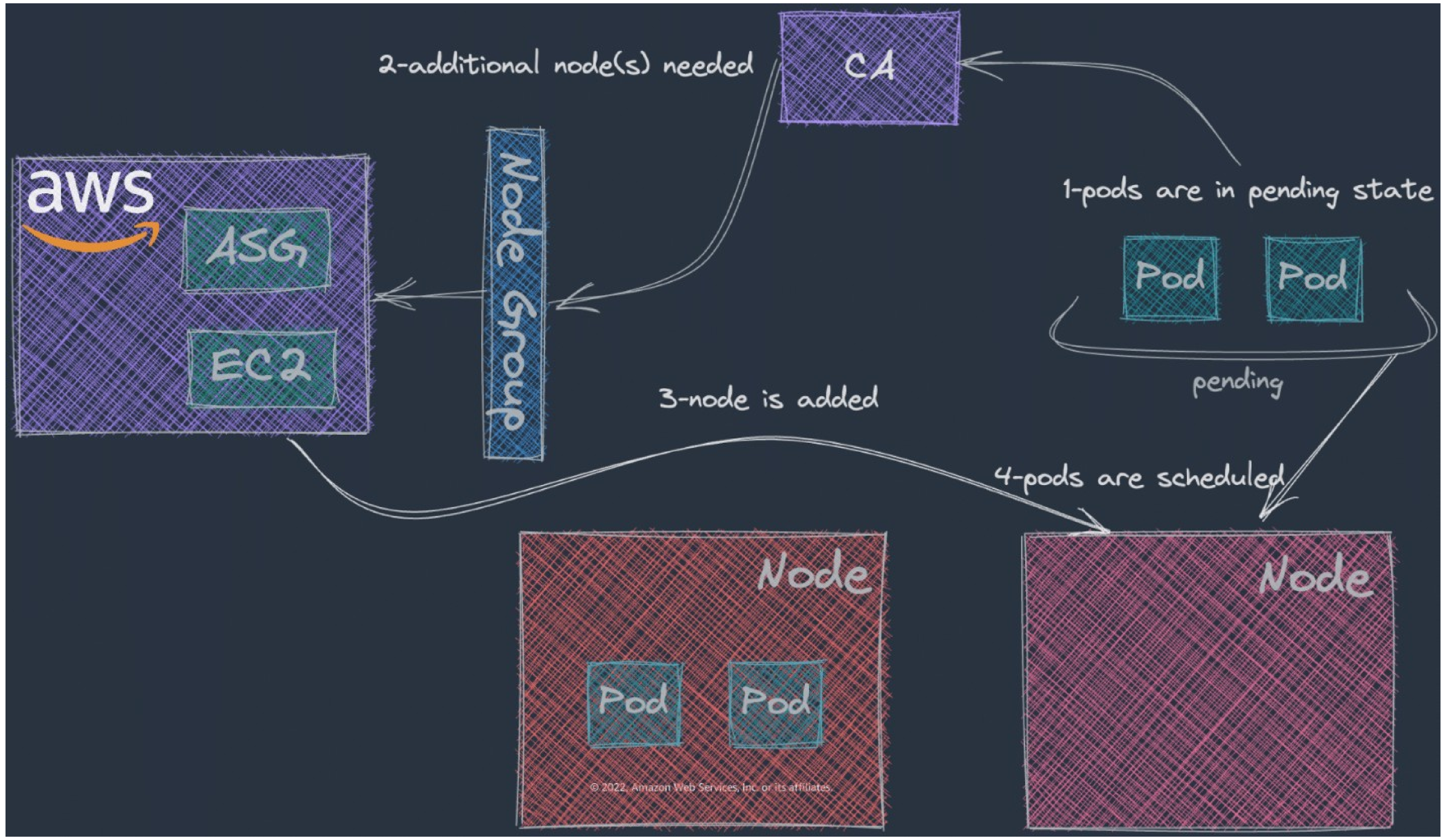


PODS SCALING_

- Other approaches:
  - HPA | VPA* (*HorizontalPodAutoscaler* | *VerticalPodAutoscaler*)
  - GCP: *MultidimPodAutoscaler*
  - KEDA (K8s Event Driven Autoscaling)
  - Knative (K8s based serverless platform)

CLUSTER **AUTOSCALER_**

- Industry ‘de-facto’ auto-scaling standard
- Cost efficiency – automatically adjusts cluster: **scale up/down**
- Leaning on existing Cloud building blocks

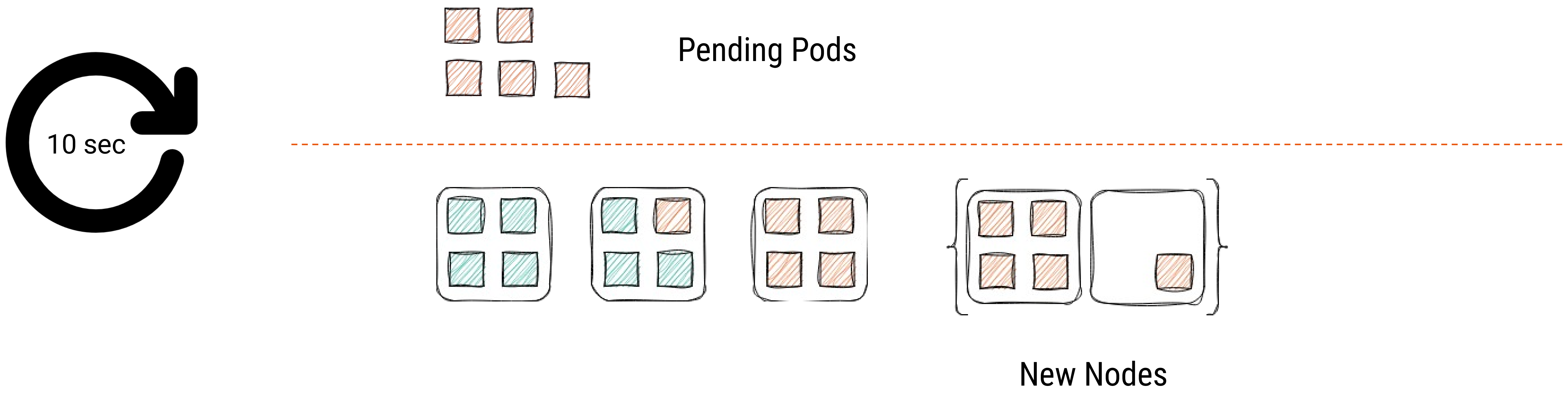


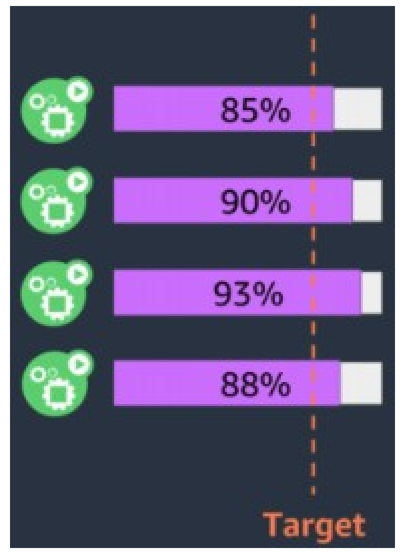
2-additional node(s) needed
CA
aws
ASG
EC2
Node Group
1-pods are in pending state
Pod
Pod
pending
3-node is added
4-pods are scheduled
Node
Pod
Pod
© 2022, Amazon Web Services, Inc. or its affiliates
Node

- Challenges: Node Group limitations (AZ, instance type, labels), complex to use, tightly bound to the scheduler, global controller

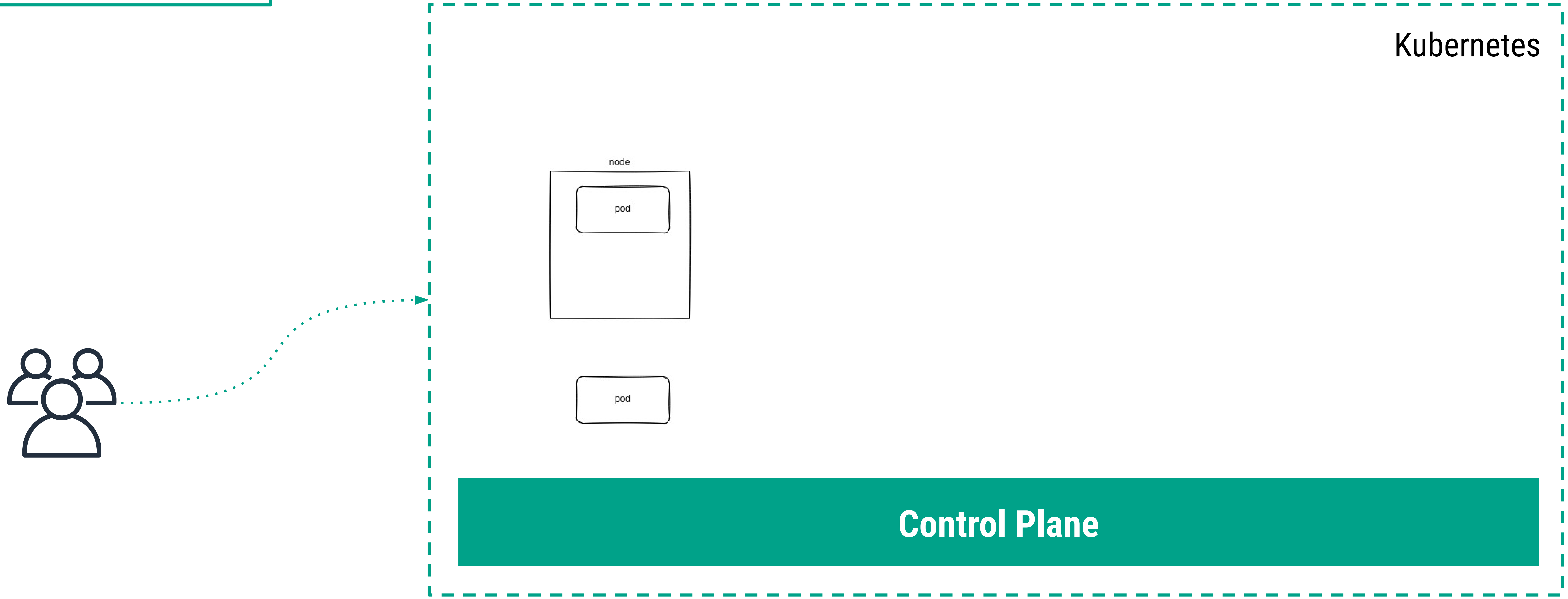
CLUSTER AUTOSCALER **SCALE-UP_**

- Reconciliation and filtering
- Scale up (in-memory simulation, <10sec)
- Expanders: random, most/least pods, price, priority
- Scale down (<10min)



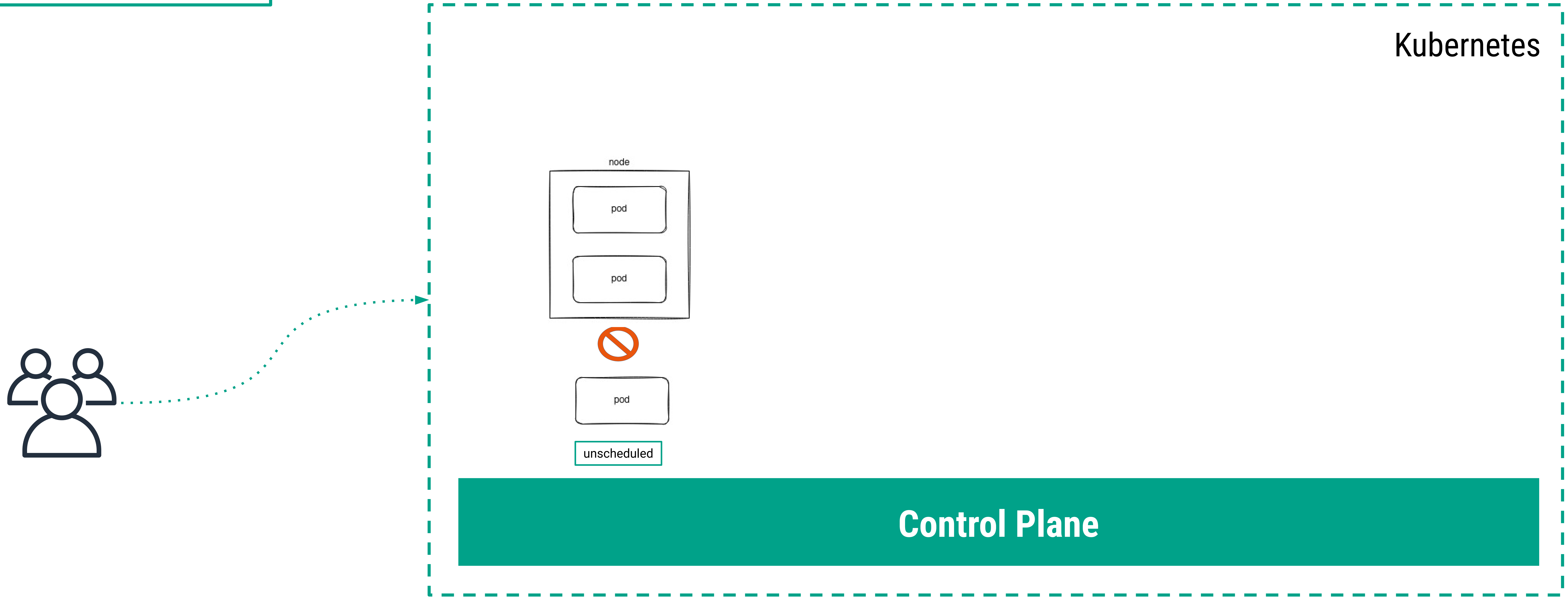


NODE **SCHEDULING_**



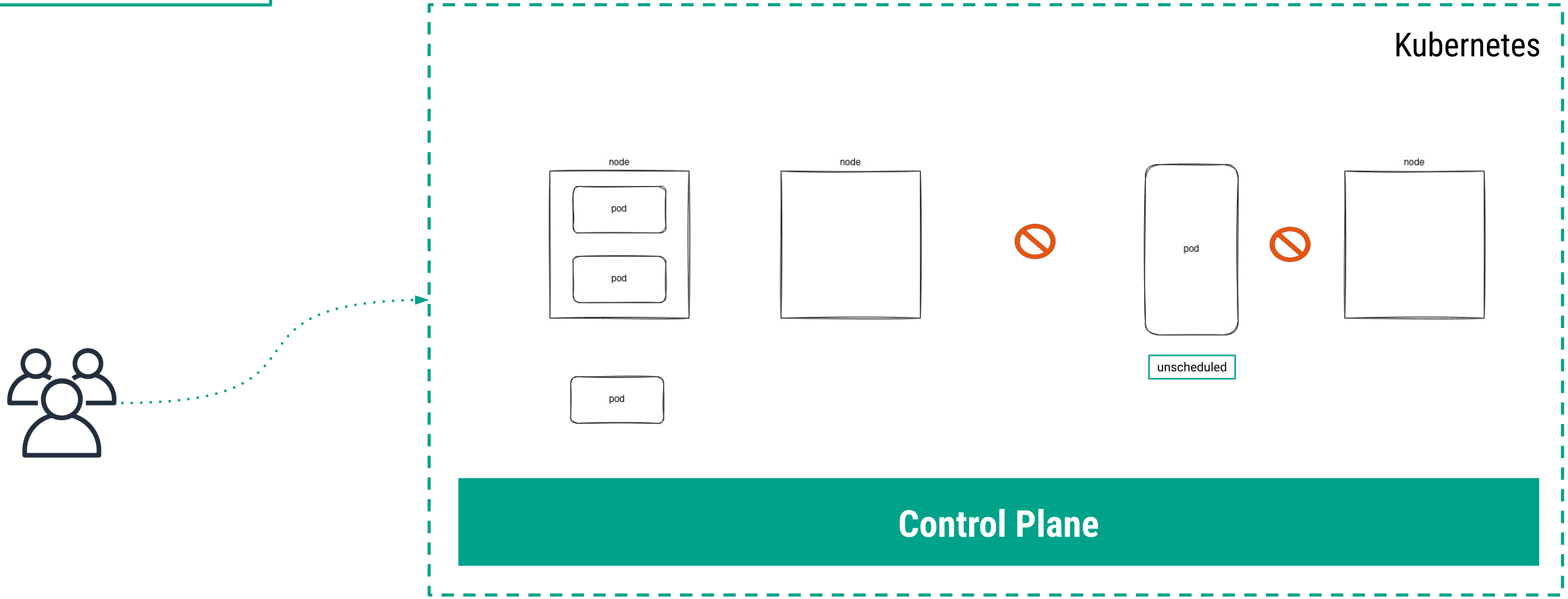
@_MarkoB

NODE
SCHEDULING_



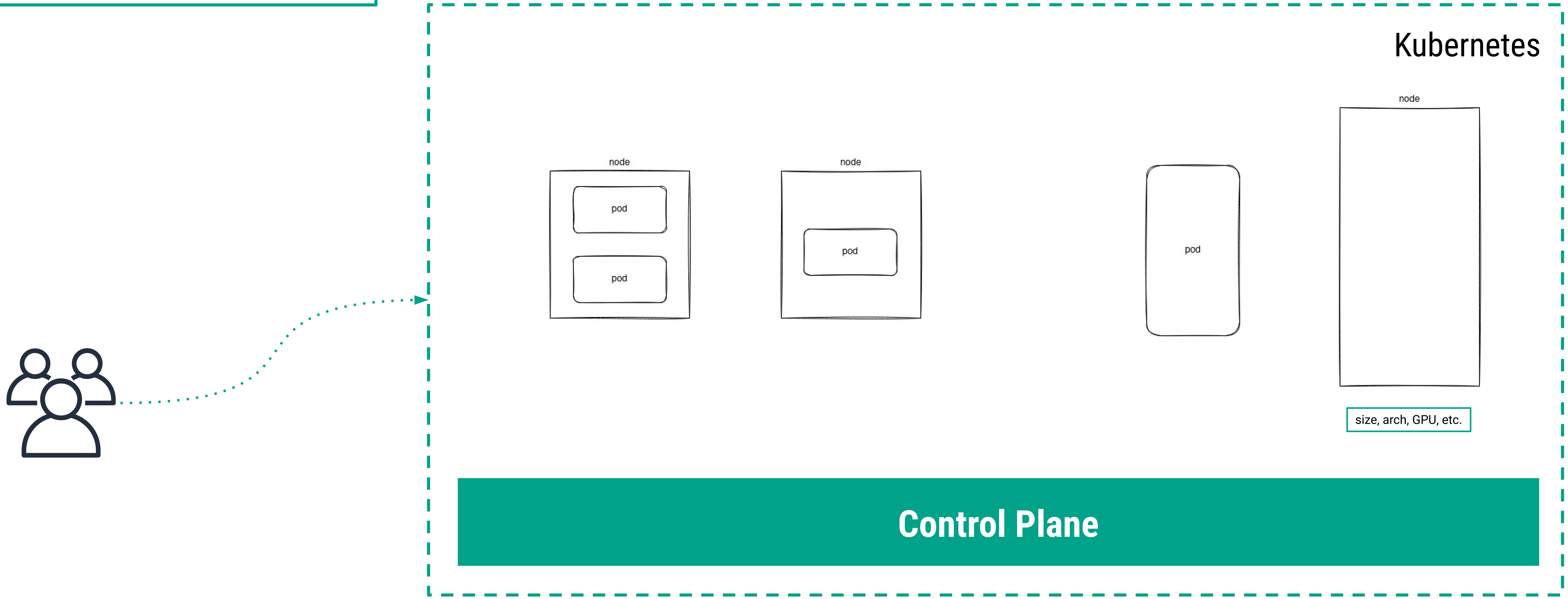
@_MarkoB

NODE CA **SCHEDULING_**

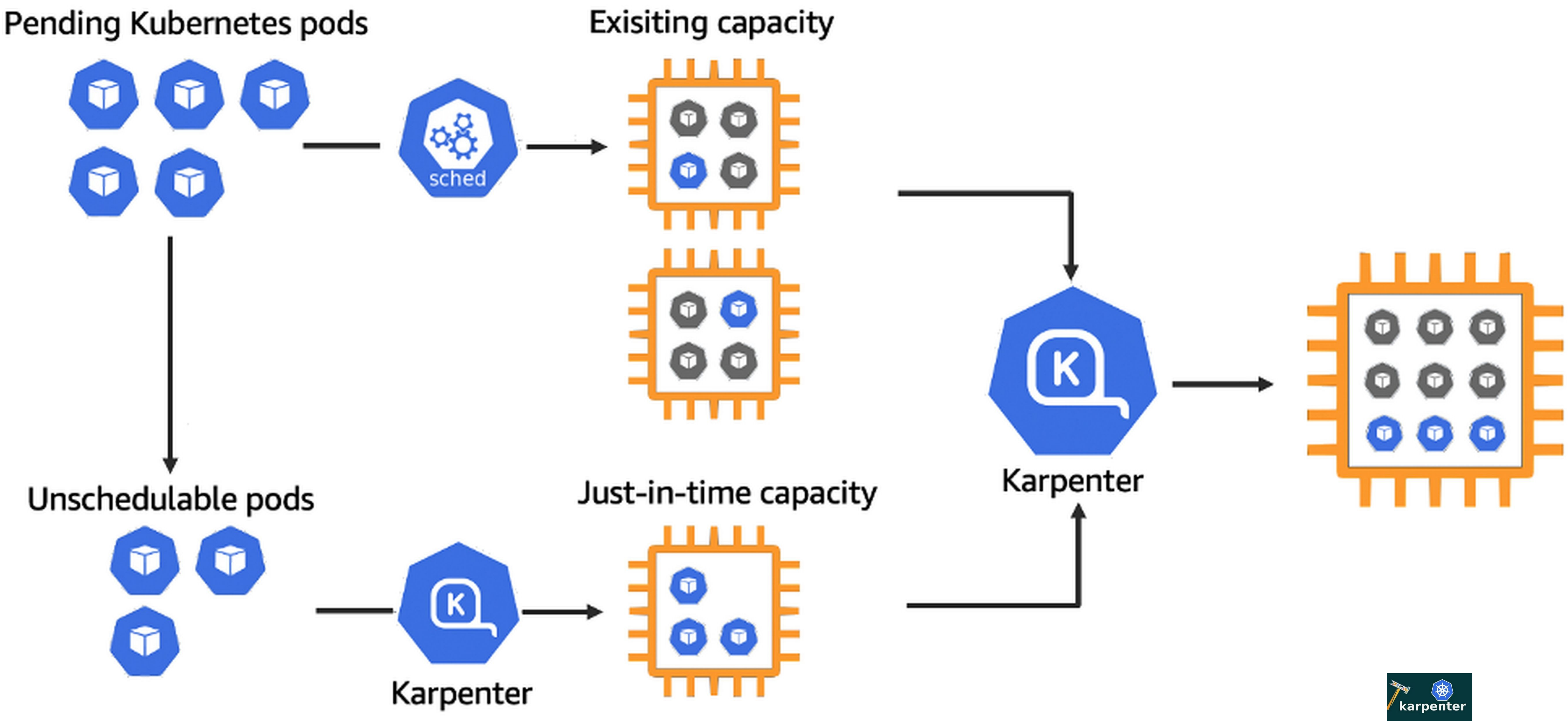


@_MarkoB

NODE KARPENTER **SCHEDULING_**



KARPENTER **ARCHITECTURE_**



karpenter

https://karpenter.sh

@_MarkoB

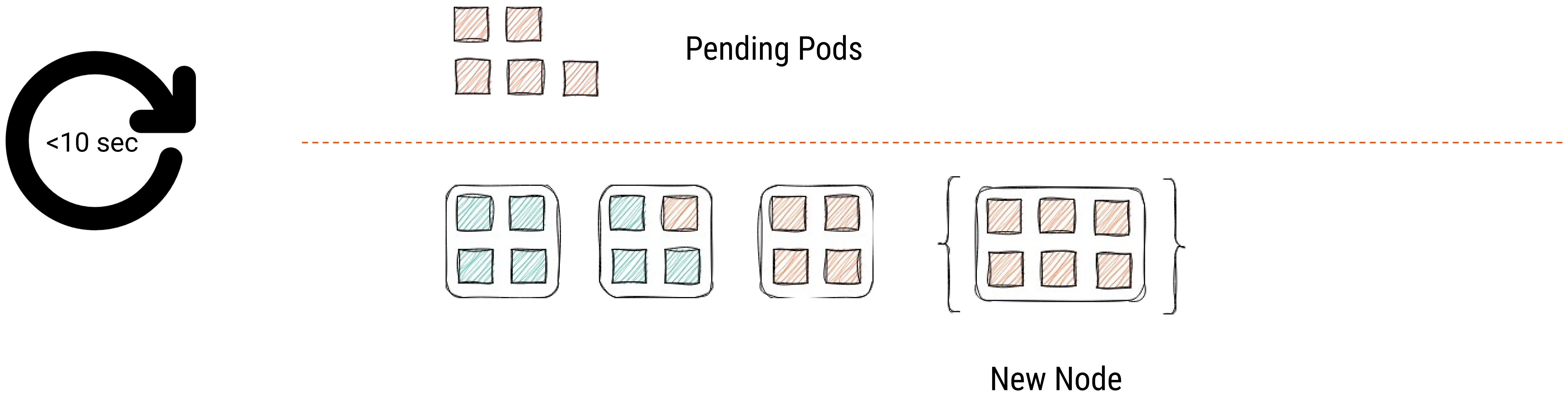
KEY **CONCEPTS_**

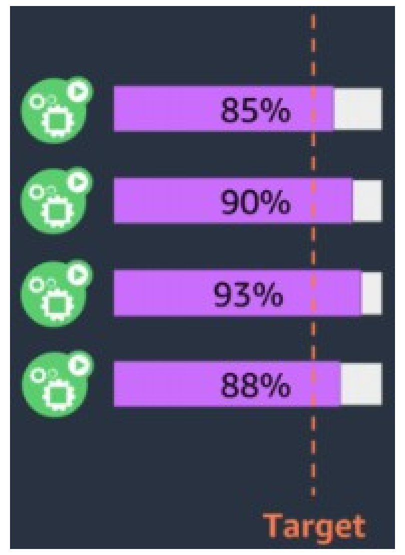
- Straightforward setup:
  - Provision AWS IAM Roles for Service Accounts (IRSA)
  - Install **controllers** (leader elect HA)
  - Apply **Provisioner CRD** (configuration) – one or more!
  - Deploy workloads
- Capacity life-cycle loop: *watch → evaluate → provision → remove*
- Well-known labels as *Provisioner* constraints:
  - *kubernetes.io/arch = amd64*
  - *kubernetes.io/os = linux*
  - *node.kubernetes.io/instance-type = m5.large*
  - *topology.kubernetes.io/zone = eu-west-1*
  - *karpenter.sh/capacity-type = on-demand | spot*
- Multi-dimension scaling (**up/down** and **in/out**)!

@_MarkoB

SCALING UP_

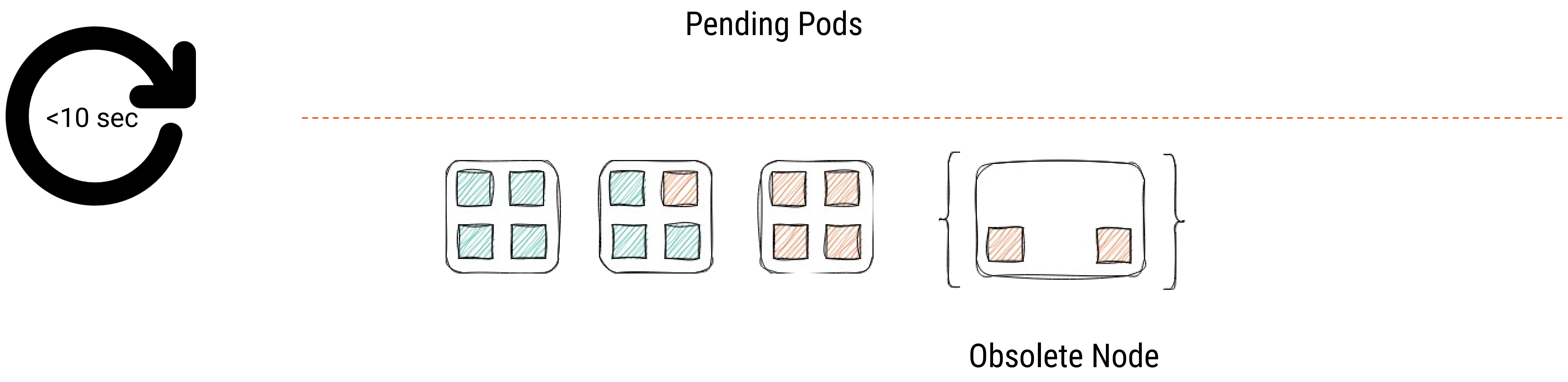
- Provisioning and scaling
- Adding more just-in-time capacity to meet demand
- Early binding to nodes
- Scheduling constraints: *resource.requests, nodeAffinity, nodeSelector, PodDisruptionBudget, topologySpreadConstraints, inter-pod (anti-)affinity*
- Removing scheduler tight coupling

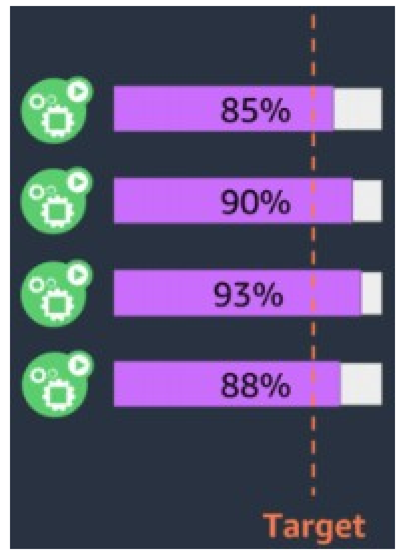




SCALING **IN_**

- Terminate obsolete capacity → reducing costs
- Removing underutilised or empty nodes
- Node TTLs (emptiness & expiration)
- Consolidation
- Interruption
- Drift





CAPACITY **CONSOLIDATION_**

- Consolidation, a.k.a off-line bin packing
- Rebalancing Node workloads based on utilisation (CPU, memory)
- Mechanisms for cluster consolidation:
  - **Delete** (on-demand | spot)
  - **Replace** (on-demand)
- Optimises for cost, minimising disruption obeying:
  - Scheduling constraints (PDBs, AZ affinity, topology spread constraints)
  - Termination grace period and expiration TTL
  - Instance unhealthy events and spot events (termination)
- Using least disruption when multiple Nodes that could be consolidated:
  - Nodes running fewer pods
  - Nodes that will expire soon
  - Nodes with lower priority Pods

OTHER **OPTIONS_**

- Custom User Data and AMI (i.e. Bottlerocket)
- Kubelet configuration (containerRuntime, systemReserved)
- Taints (or startupTaints)
- Control Pod Density
  - Network limitations
    - Number of ENIs
    - Number of IP addresses that can be assigned to ENI
  - Static Pod Density (podsPerCore)
  - Dynamic Pod Density (maxPods)
  - Limit Pod Density: topology spread, restrict instance types

TIME FOR
A DEMO!_

- Capacity planning is **hard**!
- Key advantages:
  - Flexible, lowers complexity & portable
  - Fast: provisioning latency <1min → down to 15sec (group-less)
  - Efficient: multi-dimension scaling, consolidation (delete or replace)
  - Adaptive: right-sizing, interruption events
  - Compliance (TTL)
- To keep in mind:
  - Currently supported provider is AWS (adoption in the future?*)
  - Not supporting Spot Rebalance Recommendations
  - Careful with non-interruptable workloads, edge case of 1 replica
  - https://github.com/aws/karpenter/issues

CONCLUSIONS_

& TAKEAWAYS

FURTHER **READING_**

- Resources:
  - https://github.com/mbevc1/public-speaking/
  - https://github.com/aws/karpenter/
  - https://kubernetes.io/docs/reference/labels-annotations-taints/
  - https://github.com/kubernetes/autoscaler
  - https://docs.aws.amazon.com/eks/latest/userguide/cluster-autoscaler.html
  - https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/proposals/scalability_tests.md
  - https://blog.kloia.com/karpenter-cluster-autoscaler-76d7f7ec0d0e
  - https://blog.scaleway.com/understanding-kubernetes-autoscaling/
  - https://aws.amazon.com/blogs/aws/introducing-karpenter-an-open-source-high-performance-kubernetes-cluster-autoscaler/

KEEP IN TOUCH_

Web:	https://www.scalefactory.com/
Twitter:	@_MarkoB
GitHub:	@mbevc1
GitLab:	@mbevc1
LinkedIn:	https://www.linkedin.com/in/marko-bevc/