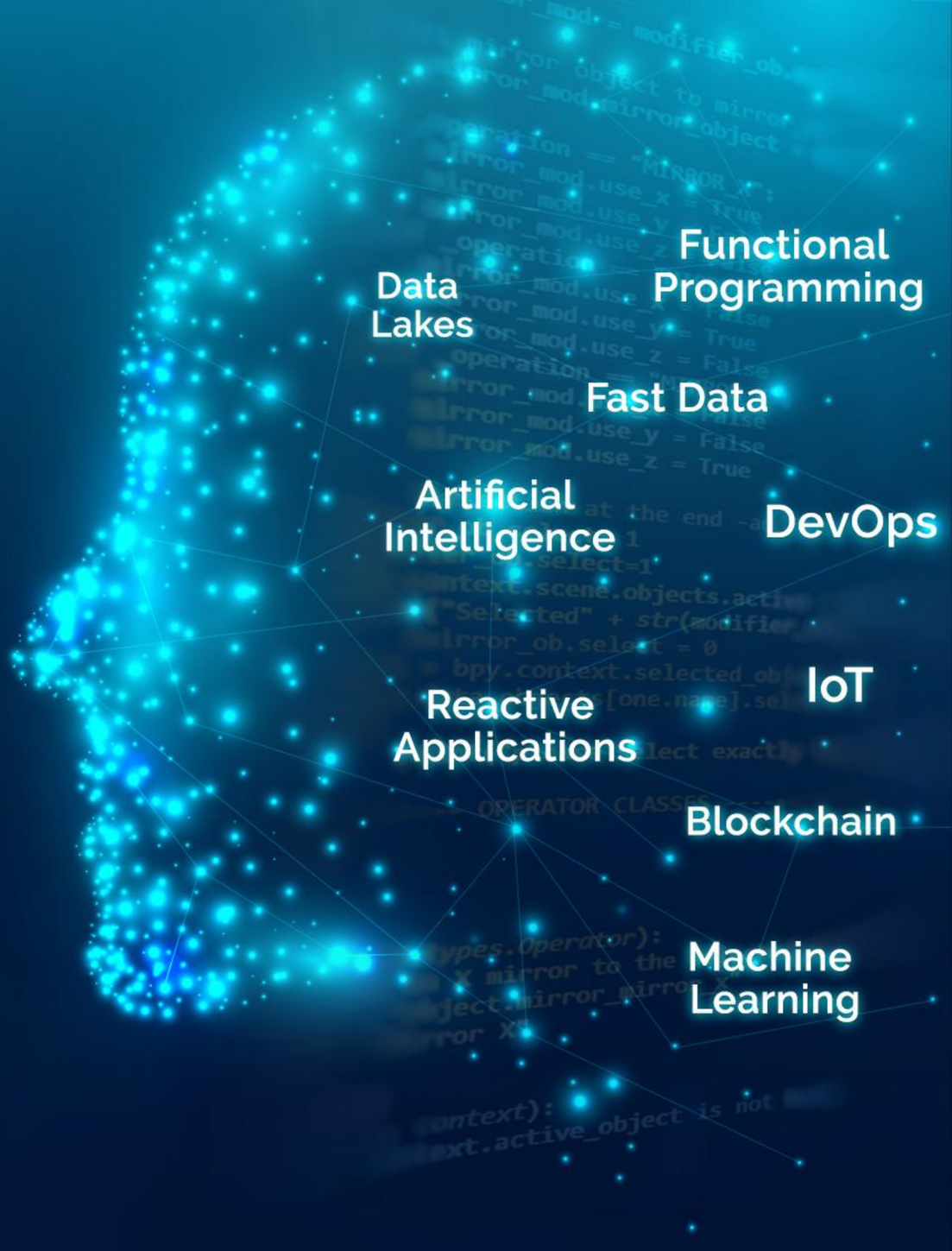
knóldus

knols . commitment . results

Karpenter



Presented By:
Vidushi Bansal
(Devops Studio)

Lack of etiquette and manners is a huge turn off.

KnolX Etiquettes

Punctuality

Respect Knolx session timings, you are requested not to join sessions after a 5 minutes threshold post the session start time.

Feedback

Make sure to submit a constructive feedback for all sessions as it is very helpful for the presenter.

Silent Mode

Keep your mobile devices in silent mode, feel free to move out of session in case you need to attend an urgent call.

Avoid Disturbance

Avoid unwanted chit chat during the session.

knóldus
knols . commitment . results

Our Agenda

01 What is Karpenter?

02 Why to go for Karpenter?

03 Provisioners

04 Karpenter on EKS

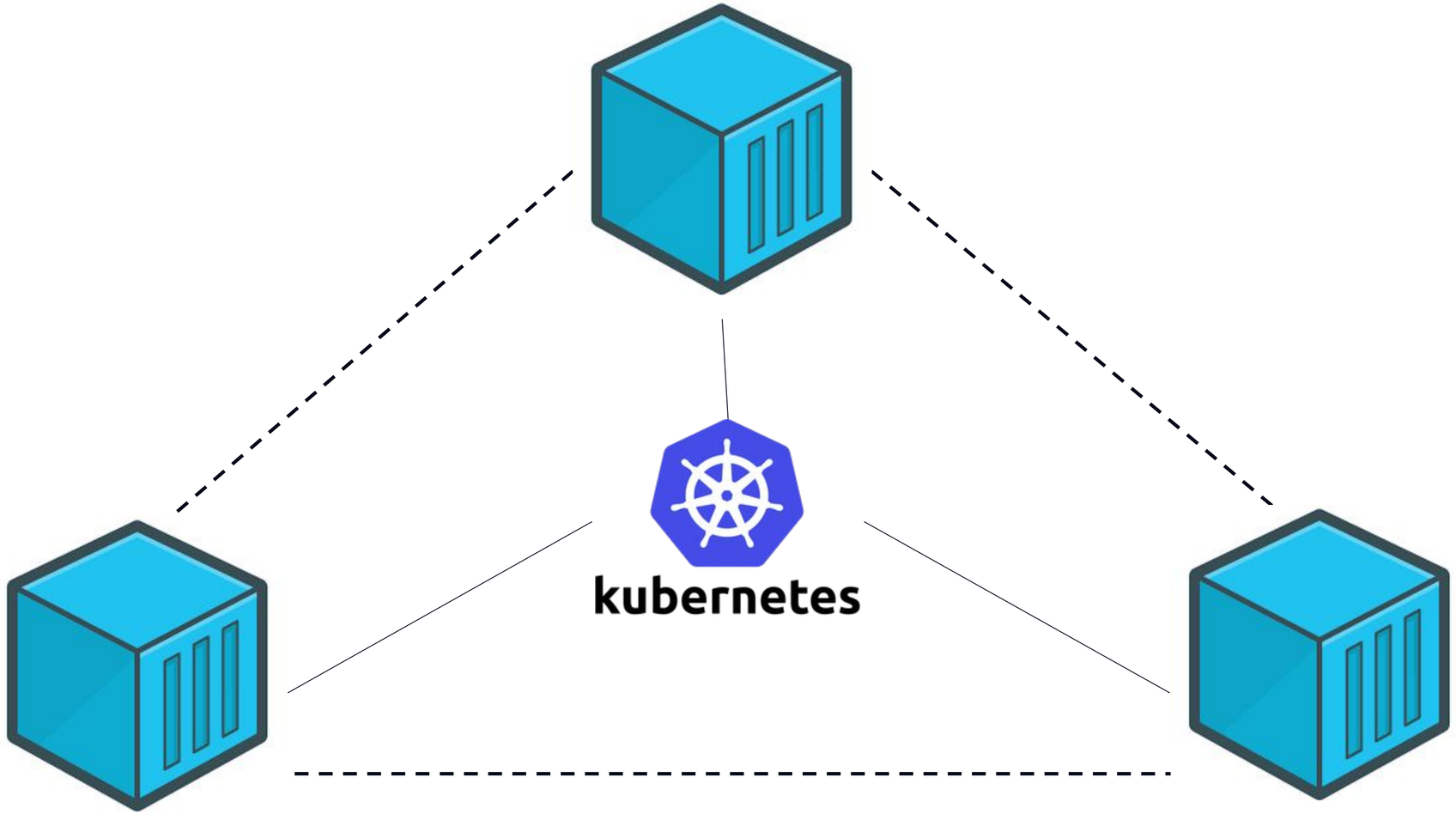
05 Demo

knólˣ

knóldus
knols . commitment . results

Prerequisites

MICROSERVICES



Linux

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications. It provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.

Elastic Kubernetes Service

Amazon Elastic Kubernetes Service (Amazon EKS) is a fully-managed, certified Kubernetes conformant service that simplifies the process of building, securing, operating, and maintaining Kubernetes clusters on AWS. Amazon EKS integrates with core AWS services such as CloudWatch, Auto Scaling Groups, and IAM to provide a seamless experience for monitoring, scaling and load balancing your containerized applications.

Amazon CloudWatch

Amazon EC2

AWS IAM

AWS VPC

Kubernetes Autoscaling

knóldus
knols . commitment . results

POD AUTOSCALING

There are two levels of pod autoscaling.

Horizontal Pod Autoscaling

Adds or removes more replicas of pods to the deployment as needed

Vertical Pod Autoscaling

resizes pod's CPU and memory requests and limits to match the load

NODE AUTOSCALING

Adding or removing nodes as and when needed

Cluster Autoscaler

scale up or down your managed node groups through EC2 Auto Scaling Groups.

Karpenter

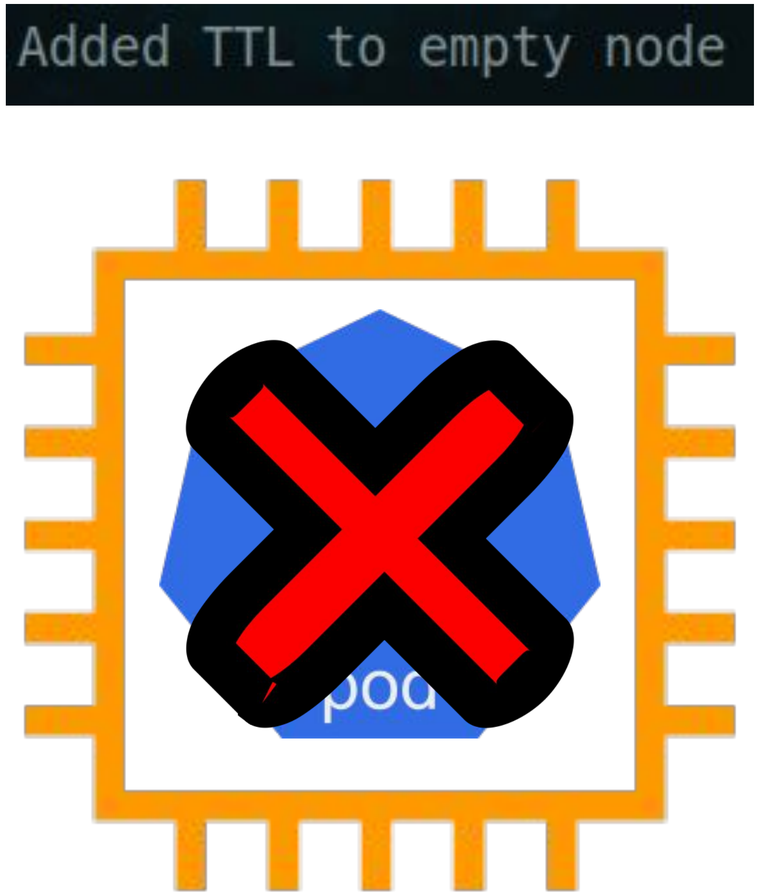
Karpenter manages each instance directly, without use of additional orchestration mechanisms like node groups.

knólˣ

knóldus
knols . commitment . results

What is Karpenter?

What is Karpenter



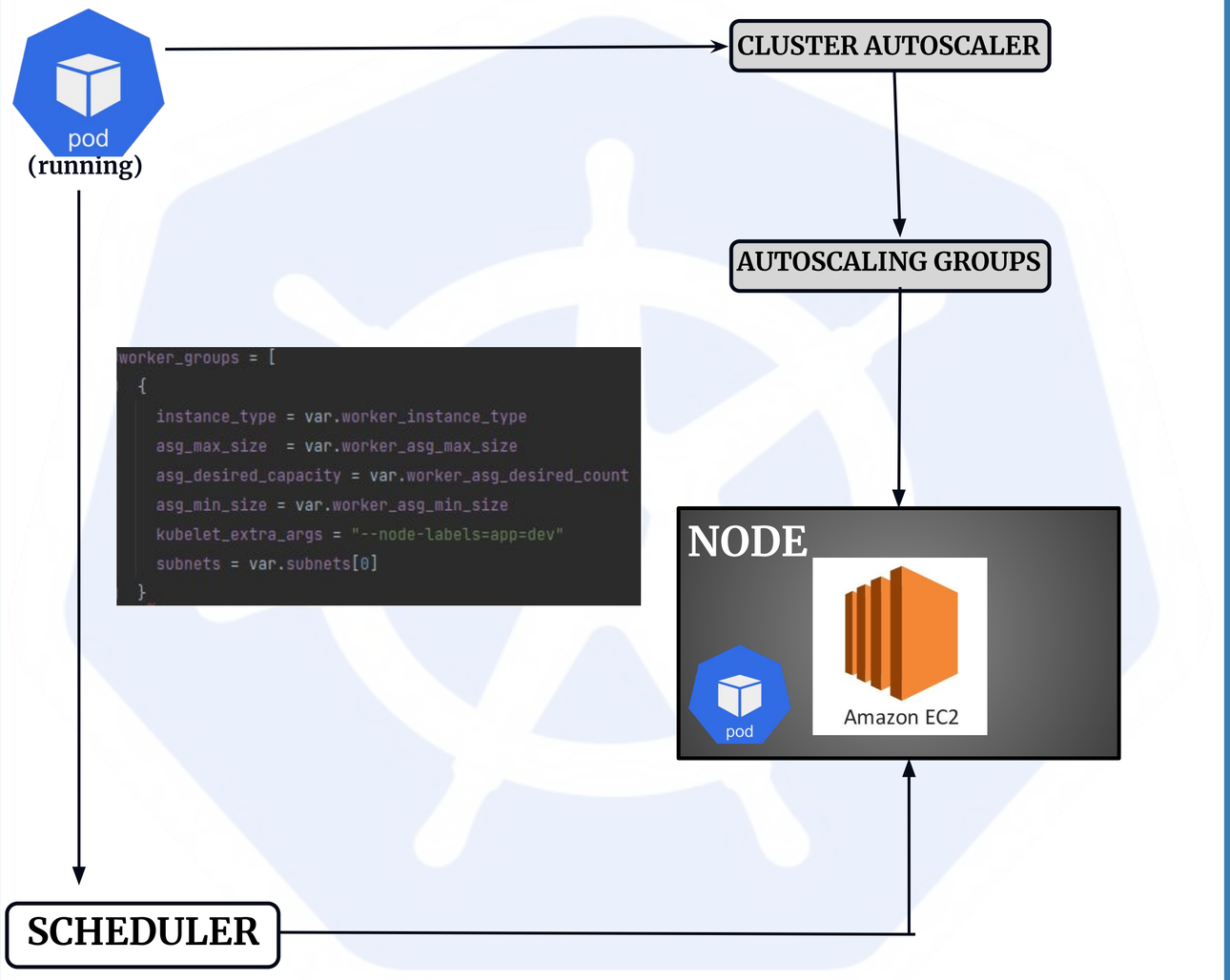
Karpenter is an open-source node provisioning project built for Kubernetes.It is tightly integrated with Kubernetes features to make sure that the right types and amounts of compute resources are available to pods as they are needed. Karpenter works by:

- **Watching for pods** that the Kubernetes scheduler has marked as unschedulable
- **Evaluating scheduling constraints** (resource requests, nodeselectors, affinities, tolerations, and topology spread constraints) requested by the pods
- **Provisioning nodes** that meet the requirements of the pods
- **Scheduling the pods** to run on the new nodes
- **Removing the nodes** when the nodes are no longer needed

knóldus
knols . commitment . results

Why Karpenter?

Why to go for Karpenter?



Users needed to dynamically adjust the compute capacity of their clusters to support applications using Amazon EC2 Auto Scaling groups and the Kubernetes Cluster Autoscaler. Cluster Autoscaler does not have direct control over EC2 instances, it control them through Auto Scaling Group. Whenever a pod is in pending state or in search of a suitable node to get mapped to, the cluster autoscaler asks AWS autoscaling group to scale up the nodes. Karpenter manages the node directly which enables it to retry in milliseconds instead of minutes when capacity is unavailable.It also allows Karpenter to leverage diverse instance types, availability zones, and purchase options without the creation of hundreds of node groups. Cluster autoscaler doesn't bind pods to the nodes it creates. Instead, it relies on the kube-scheduler to make the same scheduling decision after the node has come online. A node that Karpenter launches has its pods bound immediately. The kubelet doesn't have to wait for the scheduler or for the node to become ready. It can start preparing the container runtime immediately, including pre-pulling the image. This can shave seconds off of node startup latency.

knóldus
knols . commitment . results

Provisioners

PROVISIONERS

knóldus
knols . commitment . results

```yaml
apiVersion: karpenter.sh/v1alpha5
kind: Provisioner
metadata:
  name: default
spec:
  ttlSecondsUntilExpired: 2592000 # 30 Days = 60 * 60 * 24 * 30 Seconds;
  ttlSecondsAfterEmpty: 30
  # Provisioned nodes will have these taints
  # Taints may prevent pods from scheduling if they are not tolerated
  taints:
    - key: example.com/special-taint
      effect: NoSchedule
  # Labels are arbitrary key-values that are applied to all nodes
  labels:
    billing-team: my-team
  requirements:
    - key: "node.kubernetes.io/instance-type"
      operator: In
      values: ["m5.large", "m5.2xlarge"]
    - key: "topology.kubernetes.io/zone"
      operator: In
      values: ["us-west-2a", "us-west-2b"]
    - key: "kubernetes.io/arch"
      operator: In
      values: ["arm64", "amd64"]
    - key: "karpenter.sh/capacity-type"
      operator: In
      values: ["spot", "on-demand"]

  limits:
    resources:
      cpu: "1000"
      memory: 1000Gi
```

Provisioners define how Karpenter will manage unschedulable pods and expires nodes.

- Karpenter schedule pods that have a status condition Unschedulable=True, which the kube scheduler sets when it fails to schedule the pod to existing capacity.
- Karpenter defines a Custom Resource called a Provisioner to specify provisioning configuration.
- Provisioner uses well-known Kubernetes labels to allow pods to request only certain instance types, architectures, operating systems, or other attributes when creating nodes.
- A provisioner can also include time-to-live values to indicate when nodes should be deprovisioned after a set amount of time from when they were created or after they becomes empty of deployed pods.
- Multiple provisioners can be configured on the same cluster.

knólˣ

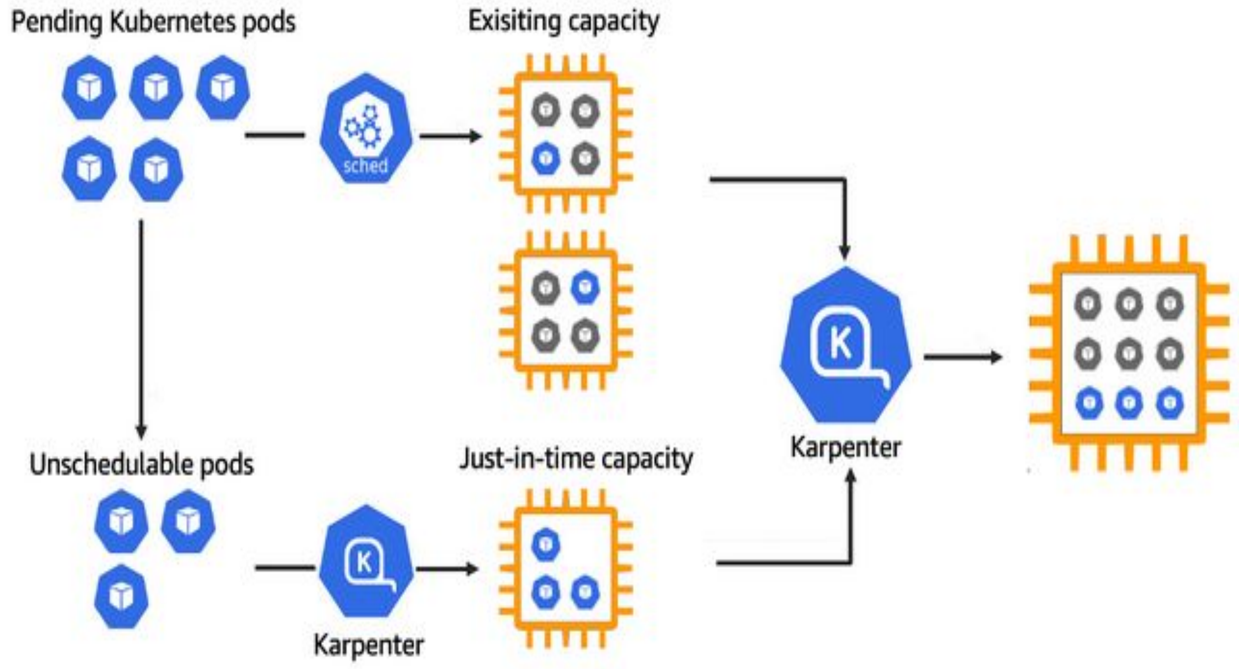
knóldus
knols . commitment . results

Karpenter on EKS

Karpenter on EKS

knóldus
knols . commitment . results

Scheduler schedules the pods following default Kubernetes behavior. Karpenter addresses unschedulable pods by creating just-in-time capacity of the most appropriate node type and size.



Karpenter is designed to be cloud provider agnostic, but currently only supports AWS. Karpenter itself can run anywhere, including on self-managed node groups, managed node groups, or AWS Fargate. Karpenter will provision EC2 instances in your account.

Karpenter has two control loops that maximize the availability and efficiency of your cluster.

Allocator — Fast-acting controller ensuring that pods are scheduled as quickly as possible

Reallocator — The Reallocator is a slow-acting cost-sensitive controller that ensures that excess node capacity is reallocated as pods are evicted.

Karpenter has an ability to launch right-sized compute resources in response to changing application load and provide just-in-time compute resources to meet your application's needs.

knólˣ

knóldus

knols . commitment . results

DEMO

knóldus

knols . commitment . results

Thank You !

Get in touch with us:

vidushi.bansal@knoldus.com