# Kanpenter on AWS EKS POC

## 一、POC场景

客户使用EKS过程中发现auto scaling功能扩容速度较慢，并且没有充分利用集群资源，希望引入有效快速的弹性伸缩方案，本次POC目的测试了karpenter在EKS上的部署方案，并验证Kanpenter自动伸缩功能。

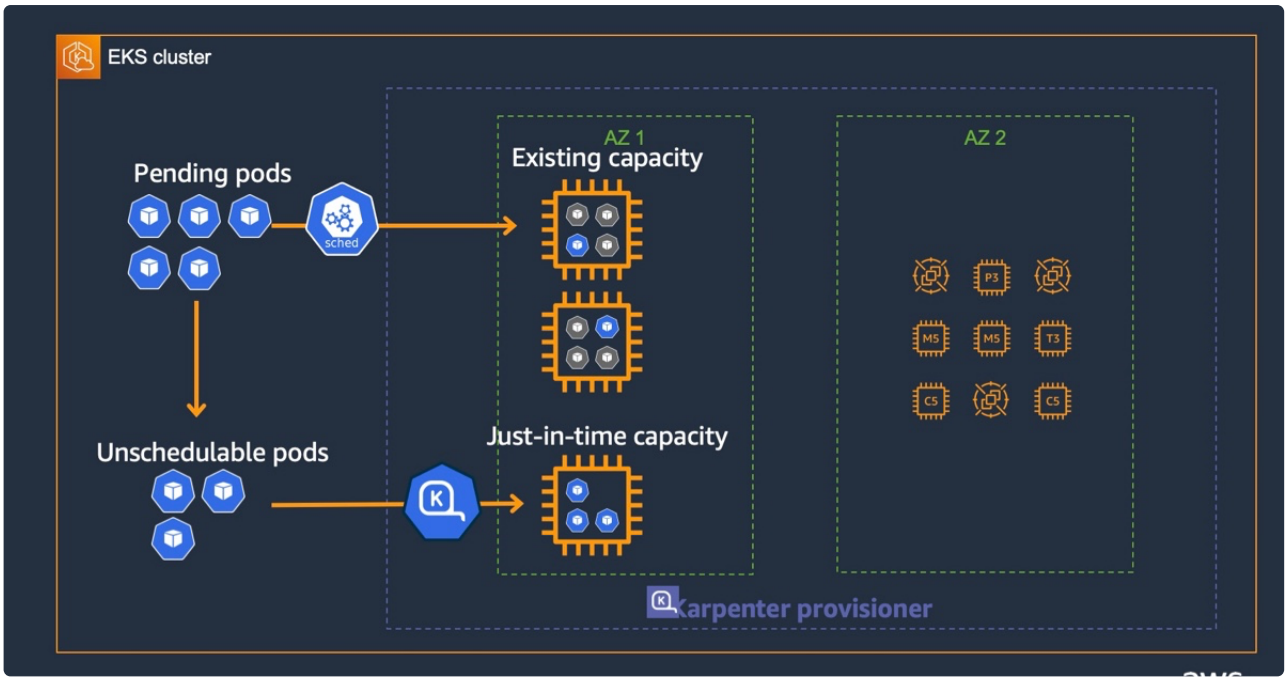
## 二、POC Scope

1、Karpenter在EKS上的部署方案

2、验证karpenter弹性伸缩方案

## 三、POC 方案

### 3.1 方案架构图




### 3.2 使用的主要服务和技术

#### EKS

Amazon Elastic Kubernetes Service (Amazon EKS) 是一项托管服务，可让您在AWS上轻松运行Kubernetes，而无需安装、操作或维护您自己的 Kubernetes 控制层面或节点。Kubernetes 是一个用于实现容器化应用程序的部署、扩缩和管理自动化的开源系统

### Karpenter

Karpenter是一个为 Kubernetes 构建的开源自动扩缩容项目。它提高了 Kubernetes 应用程序的可用性，而无需手动或过度配置计算资源。Karpenter 旨在通过观察不可调度的 Pod 的聚合资源请求并做出启动和终止节点的决策，以最大限度地减少调度延迟，从而在几秒钟内（而不是几分钟）提供合适的计算资源来满足您的应用程序的需求。

### Karpenter vs Cluster Autoscaler

Cluster Autoscaler 对节点组进行的自动扩缩容，是依赖于 launch template 和 Auto Scaling group 进行的;Cluster Autocaler 的操作也是比较复杂的，足有 78 个命令行参数

Karpenter 取消了节点组的概念，这是它与 Cluster Autoscaler 的根本区别，节点组通常是效率较低的原因之一;Karpenter 只在创建和删除容量时调用 API ，这种设计可以支持更高的 API 吞吐量。没有了节点组，调度的复杂程度也被降低;Karpenter 使用 Amazon EC2 Fleets;Karpenter 也优化了调度，一旦容量扩容的决定被做出，发出创建实例的请求，会立即获得实例 ID，不等实例创建完成就创建节点对象，将需要调度的 pod 绑定到节点

## 3.3 数据准备及数据来源

无

# 四、POC 操作指南

Amazon WorkDocs
https://amazon.awsapps.com/workdocs/index.html#/document/a5062a5954d7e366db0a356c43630780a81d18afd82e5f5c...
https://amazon.awsapps.com/workdocs/index.html#/document/a5062a5954d7e366db0a356c43630780a81d18afd82e5f5c...

# 五、测试结论

# 六、相关代码

GitHub - kylelikai/KarpenterPOC
Contribute to kylelikai/KarpenterPOC development by creating an account on GitHub.
https://github.com/kylelikai/KarpenterPOC