[use this URL *https://www.nclouds.com/blog/scalable-kubernetes-infrastructure-amazon-eks-managed-nodes]*
[see meta sheet, row 348 on nClouds tab, for meta title/description and keywords]
Posted by: Rishav Bagga
Blog categories: Amazon EKS, Application Modernization, Containers, Kubernetes
[short description for web tile, ideally up to 20 words; use header image for web tile: How to create a highly scalable Kubernetes infrastructure with Amazon EKS managed node groups. → Read blog post*]*
[see nClouds Social Media Content Calendar, row xxx, for social message and social tile message]

[Header image: Select from this board
https://www.istockphoto.com/collaboration/boards/3j__LIBpm0yde_7t1sKzYQ]

I like this one (Randy):
https://www.istockphoto.com/photo/kubernetes-helm-software-cloud-concept-in-the-sky-gm1350321208-426473157

How to create a highly scalable Kubernetes infrastructure with Amazon EKS managed node groups

Technological advances and astronomical growth in user access to the cloud demand the most innovative, reliable, scalable, available, and cost-optimized compute architectures and infrastructures that DevOps teams can build. Containerization offers a superior architecture to meet these specifications for many use cases. The most popular containerization platform is Kubernetes. Kubernetes automates the deployment, scaling, and management of containerized applications. This open-source container infrastructure has widely available tools, services, and support. Because of these attributes, AWS developed a fully-managed, certified Kubernetes-compatible platform, Amazon Elastic Kubernetes Service (Amazon EKS).

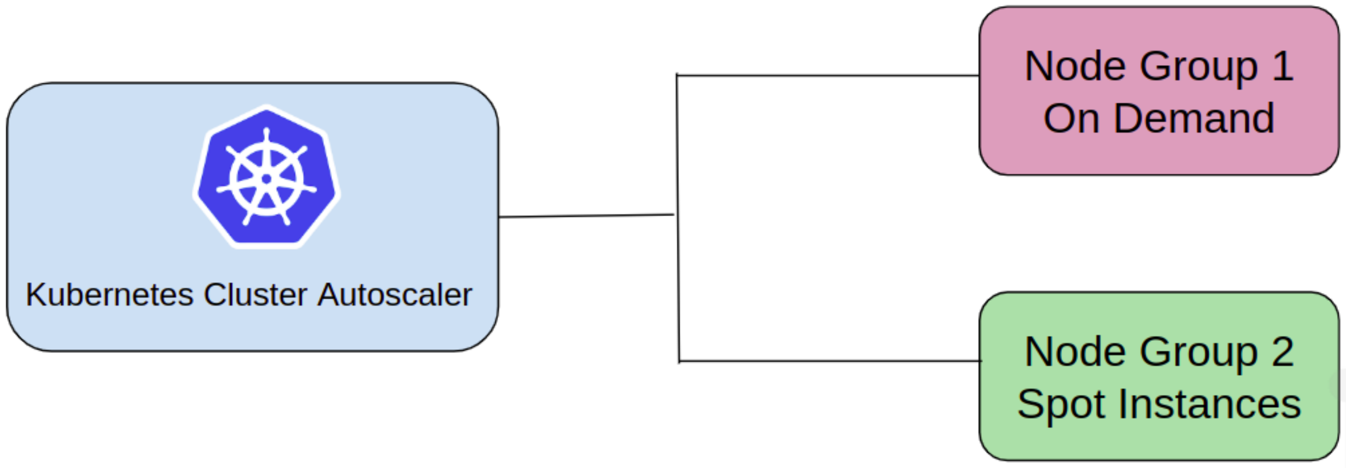
Amazon EKS simplifies the process of building, securing, operating, and maintaining Kubernetes clusters on AWS. Amazon EKS ensures high availability by automatically scaling across multiple Availability Zones within an AWS Region. Amazon EKS automatically detects and replaces unhealthy control plane instances while providing automated version updates and patching. Because Amazon EKS is fully conformant with any standard Kubernetes environment, any standard Kubernetes application can be migrated to Amazon EKS without code modification.

Also, unlike some platforms, Amazon EKS runs a single-tenant Kubernetes control plane for each cluster, which means that each AWS customer's infrastructure serves only one customer. Additionally, Amazon EKS uses Amazon VPC network policies to restrict traffic, giving its customers additional security. Further, Amazon EKS runs managed node groups, so the node groups can be configured to autoscale to meet the expectations of highly scalable, available, and managed infrastructures.

Amazon EKS has plenty of options for provisioning compute resources, like On-Demand, Reserved, and Spot Instances. However, since Spot Instances are one of the cheapest instances available, nClouds configures node groups and Kubernetes deployment to use Spot Instance lifecycles. By implementing this strategy, nClouds provides more resources to run applications at a lower cost.

By enabling Capacity Rebalancing for an Auto Scaling group, there is no need to worry about Spot Instance interruptions. Amazon EC2 Auto Scaling proactively replaces the Spot Instances in a group that receives a rebalance recommendation. If any Spot Instance in a node group is at risk of termination, it automatically generates a new replacement node. For previously existing clusters, you enable automation by reloading specified deployments and adding one to two lines of changes in the deployment file.

So, to create a highly scalable Kubernetes infrastructure, nClouds configures node groups for autoscaling and enables autoscaling on the Pod level. With horizontal autoscaling, Kubernetes ReplicaSets are automatically increased to ensure a match for the load on an application's resource requirements, even with previously deployed Pods. These two steps optimize costs and ensure a highly modern, scalable, and available infrastructure supporting an application.



Prerequisites:-

1. Amazon EC2 instance Type T2.medium is recommended to handle the load of the Kubernetes controller, API server and workloads.

2. Install Docker, AWS CLI, eksctl, kubectl.
3. Configure AWS: Confirm which account/profile is being currently configured or selected using this command: `aws sts get-caller-identity`

View this following tutorial video to learn how to create Amazon EKS Managed Node Groups to automate the provisioning and lifecycle management of nodes (Amazon EC2 instances) for Amazon EKS Kubernetes clusters. Also, learn how to implement Amazon EC2 Spot Instances and autoscaling for Amazon EKS.

[embed YouTube video https://www.youtube.com/watch?v=xL-uAPn9znw&t=3s]

Need help with Kubernetes on AWS? The nClouds team is here to help with that and all your AWS infrastructure requirements.

Contact Us