AWS EKS Architecture

Introduction

This PDF document provides a comprehensive overview of the proposed AWS EKS architecture, covering the assumptions, decisions, and proposed design for deploying an application on the AWS EKS. The document aims to concisely present the key aspects of the architecture, from end-user interactions to the various EKS components deployed on the AWS. With a focus on clarity and succinctness, this document is limited to three to four pages to provide a concise yet informative overview of the proposed AWS EKS architecture.

Assumptions for Amazon Elastic Kubernetes Service (EKS):

- Containerization: The application is containerized using Docker containers, following containerization best practices, and can be deployed and managed using Kubernetes.
- Kubernetes Expertise: There is a basic understanding of Kubernetes concepts and operations, including deployments, pods, services, and ingress.
- Cloud-Native Deployment: The application is designed to be deployed in a cloud-native manner, leveraging the benefits of Kubernetes and EKS for scalability, availability, and fault tolerance.
- Infrastructure-as-Code (IaC): The infrastructure for EKS is defined and managed using infrastructure-as-code (IaC) tools like AWS CloudFormation, Terraform, or other similar tools.
- AWS Account: An AWS account is available and properly configured with the necessary permissions to create and manage EKS clusters, worker nodes, and other required resources.
- Network Configuration: The VPC and networking components required for EKS, such as subnets, route tables, and security groups, are properly configured according to AWS best practices.
- Application Architecture: The application architecture is designed to be compatible with Kubernetes and EKS, following best practices for containerized applications, such as decoupling services, using stateless containers, and leveraging Kubernetes features like auto-scaling, self-healing, and rolling updates.
- Monitoring and Logging: Proper monitoring and logging mechanisms are in place to ensure observability of the EKS cluster, including logging of container logs, monitoring of performance metrics, and setting up alarms and notifications for critical events.

- Security Measures: Appropriate security measures are implemented, such as using secure communication channels, securing container images, securing access to EKS cluster, and implementing network security groups and other security best practices in the EKS environment.

It is essential to carefully consider these assumptions and ensure they are met to ensure a successful and secure deployment of EKS for containerized applications. Adhering to best practices and following AWS documentation and guidelines will help ensure a robust and secure EKS environment for running containerized applications in the AWS Cloud.

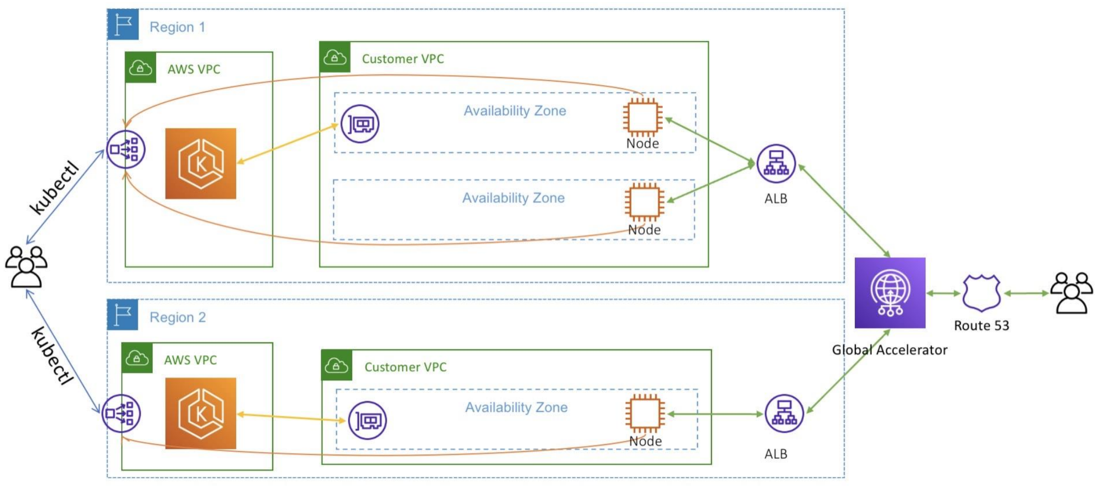
Decisions for Amazon Elastic Kubernetes Service (EKS):

- EKS as Kubernetes Orchestration Platform: Choose Amazon EKS as the Kubernetes orchestration platform for managing containerized applications due to its managed service offering, scalability, availability, and integrations with other AWS services.
- Multi-AZ Deployment: Deploy the EKS cluster across multiple Availability Zones (AZs) for high availability and fault tolerance, ensuring that the cluster remains resilient to AZ failures.
- Auto Scaling Groups for Worker Nodes: Use Amazon EC2 Auto Scaling groups to manage the worker nodes in the EKS cluster, automatically scaling the nodes based on demand, and maintaining the desired number of worker nodes for optimal performance.
- Amazon RDS or AWS managed databases for Persistence: Choose between Amazon RDS or other managed database services provided by AWS, such as Amazon DynamoDB or Amazon Aurora, for persistence and storage requirements of containerized applications running in EKS cluster, depending on the application requirements.
- Container Registry: Utilize an appropriate container registry, such as Amazon Elastic Container Registry (ECR) or other third-party container registries, for storing and managing container images used in the EKS cluster.
- Networking: Configure the Virtual Private Cloud (VPC) and networking components, such as subnets, route tables, and security groups, according to best practices for EKS, ensuring proper isolation and secure communication among EKS cluster components.
- Load Balancing and Ingress: Use Amazon Elastic Load Balancing (ELB) or other ingress controllers, such as NGINX, to distribute incoming traffic to containerized applications running in the EKS cluster and provide external access to the applications.
- Security Measures: Implement appropriate security measures, such as securing communication channels using SSL/TLS, securing container images using access control mechanisms, implementing proper

authentication and authorization for EKS cluster components, and following AWS security best practices for securing the EKS environment.

- Monitoring and Logging: Set up monitoring and logging mechanisms for EKS cluster components, containerized applications, and worker nodes using Amazon CloudWatch, AWS CloudTrail, and other logging and monitoring tools, to ensure observability and detect and resolve issues promptly.
- Infrastructure-as-Code (IaC): Use infrastructure-as-code (IaC) tools, such as AWS CloudFormation or Terraform, to define and manage the EKS cluster and associated resources, enabling reproducibility, versioning, and scalability of the infrastructure.

Proposed Architecture Diagram:



- Availability in 24 AWS Regions: Amazon EKS is available in all 24 AWS Regions, allowing for consistent, global Kubernetes-backed infrastructure deployment.

- GitOps and Infrastructure-as-Code (IaC): GitOps and IaC tools can be used to manage and operate Amazon EKS clusters across regions, ensuring consistency and reproducibility of infrastructure.

- Inter-Region VPC Peering: Applications hosted in Amazon EKS clusters in multiple regions can be connected over private network using inter-region VPC peering, enabling secure communication between regions.

- Data Replication Strategy: AWS Database and Storage services can be used to implement data replication strategies that align with recovery point (RPO) and recovery time (RTO) objectives, ensuring data durability and availability across regions.

AWS Global Accelerator: AWS Global Accelerator's routing capabilities simplify directing traffic to multiple AWS Regions, enabling implementation of active-failover, active-active, or other complex scenarios for high availability and fault tolerance.