U S
T ■

ELASTIC KUBERNETES SERVICE & GOOGLE KUBERNETES SERVICE

AMRUTHA M NAIR (245138)

AYSHA FATHIMA (245237)

ANIRUDH G (245099)

JENSON MATHEW (245048)

ELASTIC KUBERNETES SERVICE

What is EKS?

- Elastic Kubernetes Service (EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Kubernetes is an open-source system for automating the deployment, scaling and management of containerized applications.

Key Details About EKS

- ❑ Kubernetes Compatibility: Amazon EKS is fully compatible with the Kubernetes API, allowing you to use standard Kubernetes tooling and APIs to deploy and manage your applications.
- ❑ Cluster Management: With Amazon EKS, you can create and manage Kubernetes clusters using the AWS Management Console, AWS CLI, or AWS SDKs. The service handles the underlying infrastructure and control plane components, including scaling, patching, and updates.
- ❑ Multi-AZ High Availability: Amazon EKS automatically deploys and runs your Kubernetes control plane across multiple availability zones (AZs) for high availability and fault tolerance.

Key Details About EKS

- ❑ Worker Nodes: You can create a fleet of worker nodes in Amazon EKS using Amazon Elastic Compute Cloud (Amazon EC2) instances. These worker nodes run the containers and applications you deploy on your Kubernetes cluster.
- ❑ Integration with AWS Services: Amazon EKS integrates with various AWS services, such as Elastic Load Balancing, Amazon RDS, Amazon ECR, AWS Identity and Access Management (IAM), AWS CloudTrail, AWS CloudFormation, and more. This enables you to leverage the benefits of these services within your Kubernetes applications.
- ❑ Scaling and Auto Scaling: Amazon EKS allows you to scale your Kubernetes clusters manually or automatically using the Kubernetes Horizontal Pod Autoscaler or Amazon EC2 Auto Scaling. This helps ensure that your applications can handle varying levels of traffic and demand.

❑ Monitoring and Logging: You can monitor the health and performance of your EKS clusters and applications using AWS CloudWatch, and also utilize tools like Amazon CloudWatch Container Insights for deeper insights into containerized applications.

❑ Security: Amazon EKS provides security features such as IAM integration for fine-grained access control, encryption of data at rest and in transit, VPC networking, and integration with AWS PrivateLink for secure communication with EKS clusters.

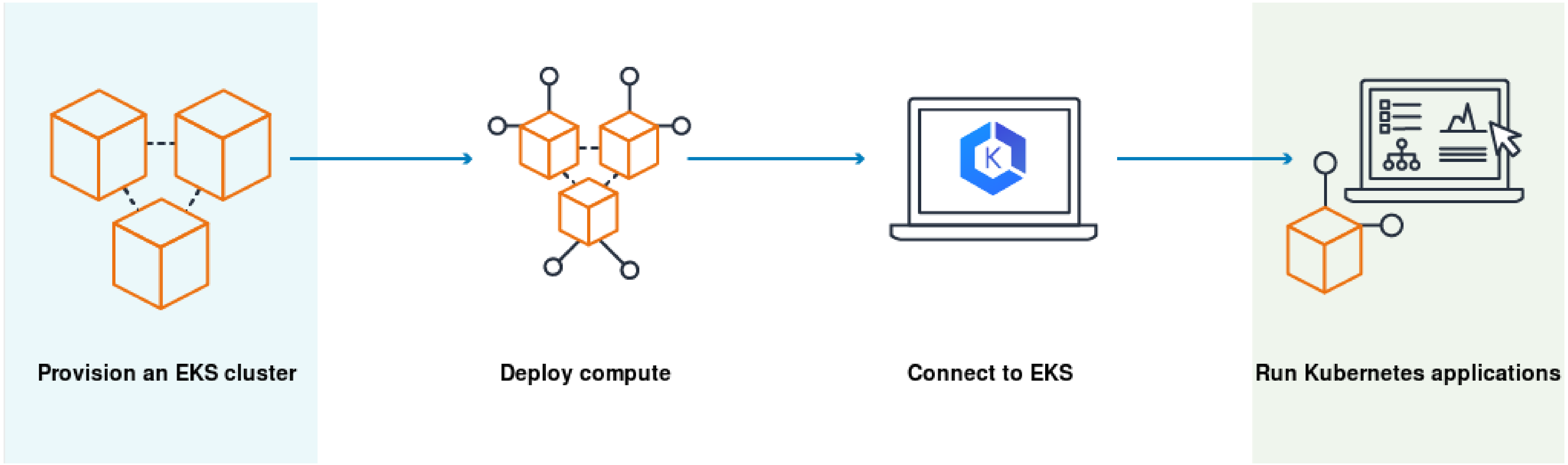
❑ Cost Optimization: With Amazon EKS, you pay for the underlying AWS resources (e.g., EC2 instances) used by your worker nodes and any additional services you utilize, such as load balancers or storage. EKS itself does not have any upfront costs or additional charges.

EKS control plane architecture

- EKS runs a single tenant Kubernetes control plane for each cluster.
- The control plane infrastructure isn't shared across clusters.
- The control plane consists of at least two API server instances and three etc. instances that run across three Availability Zones within an AWS Region.

- Actively monitors the load on control plane instances and automatically scales them to ensure high performance.
- Automatically detects and replaces unhealthy control plane instances, restarting them across the Availability Zones within the AWS Region as needed.
- Leverages the architecture of AWS Regions in order to maintain high availability.

How does Amazon EKS work?



EKS WORKING

Getting started with Amazon EKS is easy:

Create an Amazon EKS cluster in the AWS Management Console or with the AWS CLI or one of the AWS SDKs.

Launch managed or self-managed Amazon EC2 nodes, or deploy your workloads to AWS Fargate.

When your cluster is ready, you can configure your favorite Kubernetes tools, such as kubectl, to communicate with your cluster.

Deploy and manage workloads on your Amazon EKS cluster the same way that you would with any other Kubernetes environment. You can also view information about your workloads using the AWS Management Console.

Amazon EKS with a Spring Boot application

- ❑ Set up an Amazon EKS Cluster: First, create an Amazon EKS cluster using the AWS Management Console, AWS CLI, or AWS SDKs. This involves provisioning the necessary AWS resources, such as EC2 instances for worker nodes, configuring networking, and creating the EKS cluster control plane.
- ❑ Install and Configure kubectl: Install the Kubernetes command-line tool, kubectl, on your local machine. You'll use kubectl to interact with the EKS cluster and deploy your Spring Boot application.
- ❑ Build the Spring Boot Application: Develop your Spring Boot application following the usual development practices. Ensure that the application is containerized using Docker, as Kubernetes manages containers.

Amazon EKS with a Spring Boot application

- Create Kubernetes Deployment and Service Configurations: Create Kubernetes deployment and service configuration files (usually in YAML format) for your Spring Boot application. These files define how your application should be deployed, including details like container images, resource allocation, ports, and scaling parameters. Refer to the Kubernetes documentation for writing these configuration files.
- Deploy the Spring Boot Application: Use kubectl to deploy your Spring Boot application to the EKS cluster. Run the following command to apply the deployment and service configuration files:

```
kubectl apply -f <your-deployment-config-file.yaml>
kubectl apply -f <your-service-config-file.yaml>
```

Amazon EKS with a Spring Boot application

- ❑ Access the Spring Boot Application: By default, Kubernetes creates an internal cluster IP for your service. To access the Spring Boot application externally, you can expose it using a Kubernetes LoadBalancer, NodePort, or Ingress resource. Choose the appropriate method based on your requirements and Kubernetes capabilities.

- ❑ Monitor and Scale: Utilize Kubernetes and AWS monitoring tools to monitor the health and performance of your Spring Boot application. You can use tools like Amazon CloudWatch Container Insights, Kubernetes Dashboard, or other third-party monitoring solutions. Additionally, configure horizontal pod autoscaling to automatically scale your application based on CPU or custom metrics.

Features of EKS

- **Managed Kubernetes Control Plane:** EKS takes care of managing the Kubernetes control plane, which includes the API server, scheduler, and etcd. This eliminates the operational overhead of managing these components yourself.
- **Scalability and Availability:** EKS allows you to easily scale your Kubernetes clusters based on your application needs. It automatically manages the underlying infrastructure to ensure high availability and fault tolerance.
- **Integration with AWS Services:** EKS seamlessly integrates with other AWS services such as Elastic Load Balancer, Identity and Access Management (IAM), Amazon VPC, and more. This enables you to leverage the full capabilities of the AWS ecosystem within your Kubernetes clusters.

Features of EKS

- **Security and Compliance:** EKS provides secure cluster access with AWS Identity and Access Management (IAM) and integrates with AWS CloudTrail for auditing and compliance purposes. You can also encrypt your cluster data using AWS Key Management Service (KMS) for enhanced security.
- **Automated Updates:** EKS handles Kubernetes version upgrades and patching of the control plane automatically. This ensures that your clusters are always up-to-date with the latest security patches and feature releases.
- **Application Scaling:** EKS allows you to scale your applications horizontally using features like Kubernetes Horizontal Pod Autoscaler and Cluster Autoscaler. This ensures that your applications can handle increased traffic and workload demands.

Features of EKS

- Monitoring and Logging: EKS integrates with AWS CloudWatch for monitoring your Kubernetes clusters, providing insights into cluster performance, resource utilization, and application logs.
- Multi-Region and Multi-AZ Deployments: EKS supports deploying Kubernetes clusters across multiple AWS regions and availability zones (AZs), providing high availability and disaster recovery capabilities for your applications.
- Third-Party Integrations: EKS supports various third-party tools and services in the Kubernetes ecosystem. This includes popular tools for monitoring, logging, networking, and application deployment, allowing you to leverage your existing tools and workflows.

Disadvantages of EKS

- Complexity: Kubernetes itself is a complex technology, and EKS doesn't eliminate that complexity entirely. You still need to have knowledge of Kubernetes concepts and best practices to effectively manage and operate your clusters on EKS.
- Cost: While EKS simplifies the management of Kubernetes clusters, it comes at a cost. You pay for the underlying infrastructure, cluster resources, and any additional AWS services you use in conjunction with EKS. Costs can vary based on the size and scale of your clusters.

Disadvantages of EKS

- Learning Curve: If you are new to Kubernetes, there may be a learning curve involved in understanding and effectively using EKS. It requires familiarity with Kubernetes concepts, YAML manifests, and other related technologies.
- Vendor Lock-In: While EKS is compatible with the Kubernetes API, it is still a managed service provided by AWS. This means you may face some level of vendor lock-in, making it more difficult to migrate your workloads to a different Kubernetes provider in the future.

GOOGLE KUBERNETES SERVICE

What is GKS?

- Google Kubernetes Engine (GKE) is a managed Kubernetes service for containers and container clusters running on Google Cloud infrastructure. GKE is based on Kubernetes, an open source container management and orchestration platform developed by Google.

- Users can interact with GKE using the Google Cloud command-line interface or the Google Cloud Console.

- Software developers frequently use GKE to create and test enterprise applications. IT administrators also use GKE to meet the scalability and performance demands of enterprise software and hardware, such as web servers

WHY GKS?

There are several reasons why you might choose Google Kubernetes Service (GKS) for your container orchestration needs:

Managed Kubernetes: GKS is a fully managed Kubernetes service provided by Google Cloud. This means Google takes care of the underlying infrastructure, including cluster management, scaling, and upgrades, allowing you to focus on deploying and managing your applications.

Scalability and Reliability: GKS provides seamless scalability for your applications. It can automatically scale the number of containers or nodes based on resource utilization or custom-defined metrics.

Monitoring and Debugging

Ultimately, choosing GKS provides you with a managed Kubernetes solution that combines the power and flexibility of Kubernetes with the scalability, reliability, and security features of Google Cloud. It simplifies the deployment, management, and scaling of your containerized applications, allowing you to focus on delivering value to your customers.

How does Google Kubernetes Engine work?

- GKE is comprised of a group of Google Compute Engine instances running Kubernetes.
- Within a Kubernetes cluster, a control plane manages one or more worker nodes, each running a container runtime and kubelet agent needed to manage containers. The control plane also includes a Kubernetes API server, which interacts with the cluster and performs tasks such as servicing API requests and scheduling containers

How does Google Kubernetes Engine work?

•GKE users can organize one or more containers into pods that represent logical groups of related containers. Users create and manage these pods through jobs. Generally, identical containers are not organized into the same pod.

•Examples of pod groups include logfile system containers, checkpoint or snapshot system containers, and data compression containers. Similarly, network proxies, bridges and adapters might be organized into the same pod.

Features of GKS

- Can be configured to automatically scale node pool and clusters across multiple node pools based on changing workload requirements.

- Auto-repair can be enabled to do health checks on node

- Choose clusters tailored to your requirements based on:

  •Availability
  •Version Stability
  •Isolation
  •Pod Traffic requirements

Features of GKS

- ❑ Enable Cloud Logging and Cloud Monitoring via simple checkbox configurations.
- ❑ Kubernetes version can be enabled to auto-upgrade with the latest release patch.
- ❑ Supports Docker container format.
- ❑ Integrates with Google Container Registry so you can easily access your private Docker images.

GKS ARCHITECTURE

The architecture of Google Kubernetes Service (GKS), also known as Google Kubernetes Engine (GKE), follows the standard Kubernetes architecture.

GKS provides a managed Kubernetes environment that abstracts the underlying infrastructure and simplifies the management of Kubernetes clusters.

GKS simplifies the management of Kubernetes clusters by abstracting the underlying infrastructure and providing a managed environment.

This allows users to focus on deploying and managing their applications while benefiting from the scalability, reliability, and integration capabilities of Google Cloud Platform.

Control Plane:

- API Server: The API server is the central component of the control plane and exposes the Kubernetes API, allowing users to interact with the cluster. It receives and processes requests for cluster management operations.

- etcd: etcd is a distributed key-value store used for storing the cluster's configuration data and state. It provides the backend storage for Kubernetes.

Nodes:

- Node Components: Each node in the GKS cluster runs a set of components to support the execution of containers.

- These components include the Kubernetes kubelet, which communicates with the control plane, and the kube-proxy, responsible for network proxying and load balancing.

Networking:

- Cluster Network: GKS creates a cluster network that allows communication between pods and services running within the cluster. It assigns a unique IP address range to each cluster and provides connectivity for pod-to-pod and pod-to-service communication.

- Load Balancing: GKS integrates with Google Cloud Load Balancer to distribute traffic across services running in the cluster. This enables external access to the applications deployed on GKS.

Storage:

- Persistent Volumes: GKS supports persistent volumes, which are used for durable storage in the cluster.
- Persistent volumes can be provisioned dynamically or statically and are attached to pods as needed.

Google Cloud Storage Integration:

- GKS integrates with Google Cloud Storage, allowing you to use it as a storage solution for your applications.

Logging and Monitoring:

- Stackdriver: GKS integrates with Google Cloud's Stackdriver Logging and Monitoring services. These services provide logging, monitoring, and alerting capabilities for your GKS cluster, helping you gain insights into the health and performance of your applications.
- The GKS architecture abstracts away the underlying infrastructure management, allowing users to focus on deploying and managing their applications while Google handles the maintenance and scalability of the Kubernetes infrastructure.

GKS Cluster Management

- GKS (Google Kubernetes Service) provides robust cluster management capabilities that allow users to create, configure, and manage Kubernetes clusters efficiently. Here are some aspects of GKS cluster management:
- Cluster Creation
- Node Pools
- Autoscaling

- Cluster Upgrades
- Cluster Scaling
- Node Maintenance
- High Availability
- Cluster Deletion

- When creating a GKS cluster, you have control over various configuration options such as the number and type of nodes, node pool size, machine type, network settings, and more.
- GKS ensures that your Kubernetes control plane is kept up to date with the latest stable version.
- GKS allows you to scale your cluster's node pool dynamically. You can add nodes to handle increased workload or scale down to reduce costs during periods of low demand.

- To ensure the health and reliability of your cluster, GKS performs automated node maintenance tasks.

- When a node requires maintenance or replacement due to underlying infrastructure issues, GKS automatically schedules the evacuation of pods running on that node and migrates them to healthy nodes.

- GKS integrates with the Kubernetes Cluster Autoscaler, which automatically adjusts the size of your node pool based on the resource demands of your applications.

- The Cluster Autoscaler monitors resource utilization and scales the node pool up or down to match the workload requirements. This feature ensures efficient resource utilization and cost optimization.

Security and Access Control in GKS

- Security and access control are crucial aspects of managing Google Kubernetes Service (GKS) clusters to ensure the confidentiality, integrity, and availability of your applications and data.
- Authentication and Authorization
- Network Security
- Pod Security Policies
- Secrets Management
- Security Scanning and Vulnerability Management

- Google Kubernetes Service (GKS) provides robust security features and access control mechanisms to help protect your containerized applications.

- Authentication and Authorization: GKS supports multiple authentication mechanisms, including Google Accounts, service accounts, and external identity providers like OpenID Connect.

- Network Policies: GKS supports Kubernetes Network Policies, which allow you to define rules to control network traffic between pods and namespaces. With network policies, you can enforce traffic isolation, segmentation, and control ingress and egress traffic based on criteria like IP addresses, ports, or labels.
- Encryption: GKS provides encryption at rest and in transit to protect your data. At rest, GKS uses Google Cloud's encryption capabilities, such as Cloud KMS (Key Management Service) to encrypt data on persistent disks.

- Pod Security Policies: GKS allows you to use Pod Security Policies to enforce security standards and best practices for pods running in your clusters. Pod Security Policies enable you to define constraints on pod creation and specify security-related configurations like privileged access, host namespaces, and volume mount permissions.

ADVANTAGES

- Managed Kubernetes Environment: GKS provides a fully managed Kubernetes environment, abstracting the underlying infrastructure and cluster management complexities.
- Scalability and High Availability: GKS leverages the scalability and high availability features of Kubernetes, ensuring that your applications can scale seamlessly to meet demand and are highly available even in the face of failures.

- Integration with Google Cloud Platform (GCP): GKS seamlessly integrates with other Google Cloud services, providing a unified and comprehensive platform for your applications.
- High Availability: GKS is designed to provide high availability for applications. It ensures that your containers are always up and running by automatically replacing failed containers or nodes.
- Security and Compliance: GKS offers robust security features to protect your containerized applications. It provides granular access control, allowing you to define who can access and manage your clusters.

- Monitoring and Logging: GKS integrates with Google Cloud's monitoring and logging services, such as stackdriver Monitoring and Logging. This enables you to gain insights into the performance and health of your applications, as well as track and analyze logs for troubleshooting and auditing purposes.
- Community and Ecosystem: Kubernetes has a large and active community of developers and users, which means there is extensive support, resources, and knowledge available.

DISADVANTAGES

- Learning Curve: Working with Kubernetes, including GKS, can have a steep learning curve, especially for users who are new to container orchestration.
- Complexity: Kubernetes, including GKS, is a complex system with various components and configurations.
- Cost: While GKS offers cost optimization features, managing Kubernetes clusters can still be costly, particularly for resource-intensive or high-traffic workloads.

- Dependence on Internet Connectivity: GKS operates in the cloud, which means it relies on a stable and robust internet connection. If there are network connectivity issues or if your organization's internet connection is unreliable, it can impact the accessibility and performance of GKS.
- Complexity of Networking and Load Balancing: Kubernetes networking can be complex, especially when dealing with services, load balancers, and ingress controllers. Configuring and managing network policies, load balancing, and routing rules can be challenging, particularly for those unfamiliar with Kubernetes networking concepts.

- Limited Control: With a managed service like GKS, some level of control over the underlying infrastructure is relinquished.
- Version Compatibility: Kubernetes evolves rapidly, with new features and versions released regularly. It's essential to ensure compatibility between the Kubernetes version used by GKS and your applications or third-party tools.

- Vendor Lock-in: Choosing GKS as a managed Kubernetes service ties you to the Google Cloud ecosystem. Moving away from GKS to another cloud provider's Kubernetes service or managing Kubernetes on-premises may involve significant effort and potential compatibility challenges.

U S
T ■

Thank you

ust.com