5.2. Student Handout

Student Handout: Running a Web Application on GKE, EKS, and AKS

Introduction

This handout provides a concise overview of deploying web applications using Kubernetes on Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS). By the end, you'll understand the differences between these platforms, how to migrate applications, and best practices for multi-cloud deployments.

Differences Between GKE, EKS, and AKS

GKE (Google Kubernetes Engine)

- **Integration**: Deep integration with Google Cloud services like BigQuery and Cloud Storage.
- **Features**: Automatic upgrades, scaling, and monitoring.
- **Example**: Use GKE for applications that require seamless integration with Google Cloud's AI and machine learning services.

EKS (Amazon Elastic Kubernetes Service)

- **Integration**: Works well with AWS services like S3 and RDS.
- **Features**: High availability and security, with some infrastructure management required.
- **Example**: Deploy applications that need to leverage AWS's extensive global infrastructure.

AKS (Azure Kubernetes Service)

- **Integration**: Integrates with Azure services like Azure Blob Storage and Azure Active Directory.
- **Features**: Automatic patching and scaling.
- **Example**: Ideal for applications that need to integrate with Microsoft services like Office 365.

Key Architectural Differences and Pricing Models

GKE Architecture

- **Control Plane**: Managed by Google Cloud.
- **Pricing**: Pay for worker nodes based on specifications.
- **Example**: Cost-effective for applications with fluctuating workloads due to automatic scaling.

EKS Architecture

- **Control Plane**: Managed by AWS, with more control over worker nodes.
- **Pricing**: Pay for both control plane and worker nodes.
- **Example**: Suitable for applications requiring high availability across multiple regions.

AKS Architecture

- **Control Plane**: Managed by Azure.
- **Pricing**: Pay for worker nodes based on specifications.
- **Example**: Efficient for applications that benefit from Azure's hybrid cloud capabilities.

Migrating a Web Application Across Cloud Providers

Steps for Migration

1. **Containerize Your Application**: Use Docker to package your application.

- **Example**: Containerize a Node.js application for consistent deployment.

2. **Export Kubernetes Manifests**: Use YAML files to define application deployment.

- **Example**: Create a Deployment manifest for a web server.

3. **Deploy on the New Platform**: Use `kubectl` to deploy with minimal changes.

- **Example**: Deploy a Python Flask app on EKS after migrating from GKE.

Potential Gaps

- **Cloud-Specific Services**: Modify applications using services like Google Cloud Storage to use AWS S3.
- **Networking Differences**: Adjust network configurations for each provider.

Portability of Kubernetes Applications Across Platforms

Challenges

- **Cloud-Specific Services**: Modify applications using services like AWS RDS to use Azure SQL Database.
- **IAM and Security**: Update security policies for each cloud provider.

Examples

- **Example 1**: Use Kubernetes Secrets to manage sensitive data across platforms.
- **Example 2**: Implement a service mesh like Istio for consistent networking.
- **Example 3**: Use Helm charts for consistent application deployment.

Handling Cloud-Specific Services and Dependencies

Strategies

1. **Use Open-Source Alternatives**: Replace cloud-specific services with open-source options.

- **Example**: Use MinIO instead of AWS S3 for object storage.

2. **Abstract the Service Layer**: Create an abstraction layer for service switching.

- **Example**: Implement a messaging interface to switch between Google Cloud Pub/Sub and AWS SNS.

3. **Example**: Use Terraform to manage infrastructure as code across providers.

Best Practices for Multi-Cloud Kubernetes Deployment

Strategies

1. **Use a Centralized CI/CD Pipeline**: Manage deployments with a single pipeline.

- **Example**: Use Jenkins to deploy applications on GKE, EKS, and AKS.

2. **Standardize Kubernetes Manifests**: Ensure consistency across platforms.

- **Example**: Use Kustomize to manage environment-specific configurations.

3. **Monitor Across Platforms**: Use centralized monitoring tools.

- **Example**: Implement Prometheus and Grafana for cross-platform monitoring.

Strategies for Disaster Recovery and Failover

Strategies

1. **Active-Active Setup**: Run applications on multiple providers simultaneously.

- **Example**: Deploy a microservices architecture across GKE and AKS.

2. **Backup and Restore**: Regularly back up data in a cloud-agnostic format.

- **Example**: Use Velero for Kubernetes backup and restore.

3. **Automated Failover**: Use tools for automated failover.

- **Example**: Implement Kubernetes Federation for cross-cloud failover.

CI/CD Pipeline Integration with Kubernetes

Tools

1. **Jenkins**: Automate build, test, and deployment processes.

- **Example**: Use Jenkins X for Kubernetes-native CI/CD.

2. **GitLab**: Automate deployments with GitLab CI/CD.

- **Example**: Use GitLab Auto DevOps for Kubernetes deployments.

3. **Example**: Use Argo CD for declarative GitOps continuous delivery.

Final Project: Deploying a Scalable Web App on GKE, EKS, and AKS

Steps

1. **Containerize Your Application**: Use Docker for packaging.

- **Example**: Containerize a React.js frontend application.

2. **Create Kubernetes Manifests**: Define deployment, service, and scaling policies.

- **Example**: Write a HorizontalPodAutoscaler manifest for scaling.

3. **Deploy on GKE, EKS, and AKS**: Use `kubectl` for deployment.

- **Example**: Deploy a Java Spring Boot application across all platforms.

4. **Monitor and Scale**: Use monitoring tools for scaling.

- **Example**: Implement Grafana dashboards for real-time monitoring.

Performance and Monitoring Comparisons Across Platforms

Metrics

- **Latency**: Measure response time on each platform.
- **Example**: Use Apache JMeter for load testing.
- **Throughput**: Measure requests per second.
- **Example**: Use Locust for performance testing.
- **Cost**: Compare running costs on each platform.
- **Example**: Use cloud provider calculators for cost estimation.

Conclusion

This handout covered deploying web applications on GKE, EKS, and AKS, highlighting differences, migration strategies, and best practices for multi-cloud deployments. You should now have a solid understanding of managing web applications across different cloud platforms using Kubernetes.