In my solution, I deployed a Docker container running the Apache web server on a CentOS base image. The AWS cloud platform was chosen for provisioning resources, utilising Terraform for Infrastructure as Code. I specifically created a VPC and an EKS resource group module. Helm charts were employed for Kubernetes deployments, and ArgoCD was used for GitOps.

Available Options:
Cloud Platform: AWS, GCP, Azure, etc.
IaC Tools: Terraform, AWS CloudFormation, Ansible, etc.
Containerization Tools: Docker, Kubernetes, etc.
GitOps Tools: ArgoCD, Flux, etc.

The Reasons Behind this Decisions:

AWS: It's a mature cloud platform with a comprehensive set of services, making it an ideal choice for this project.
Terraform: Enables version-controlling infrastructure and provides a simple way to create and manage resources on AWS.
Docker and CentOS: Docker for containerization simplifies deployment, and CentOS offers a stable and secure environment for the Apache web server.
Helm charts: Simplifies Kubernetes deployments and application management.
ArgoCD: Ideal for GitOps and ensures that the cluster state matches the configuration in the Git repository.

If I had More Time:
High Availability: Deploy the application across multiple availability zones.
Security: Introduce a Web Application Firewall (WAF) and implement SSL/TLS for the Apache server.
Monitoring and Logging: Add comprehensive monitoring and logging using tools like Prometheus and Grafana.
Secret Management: Use AWS Secrets Manager or a similar service for better secret management.

Optimize Costs: Utilize AWS Spot Instances for non-critical, fault-tolerant components of the application.