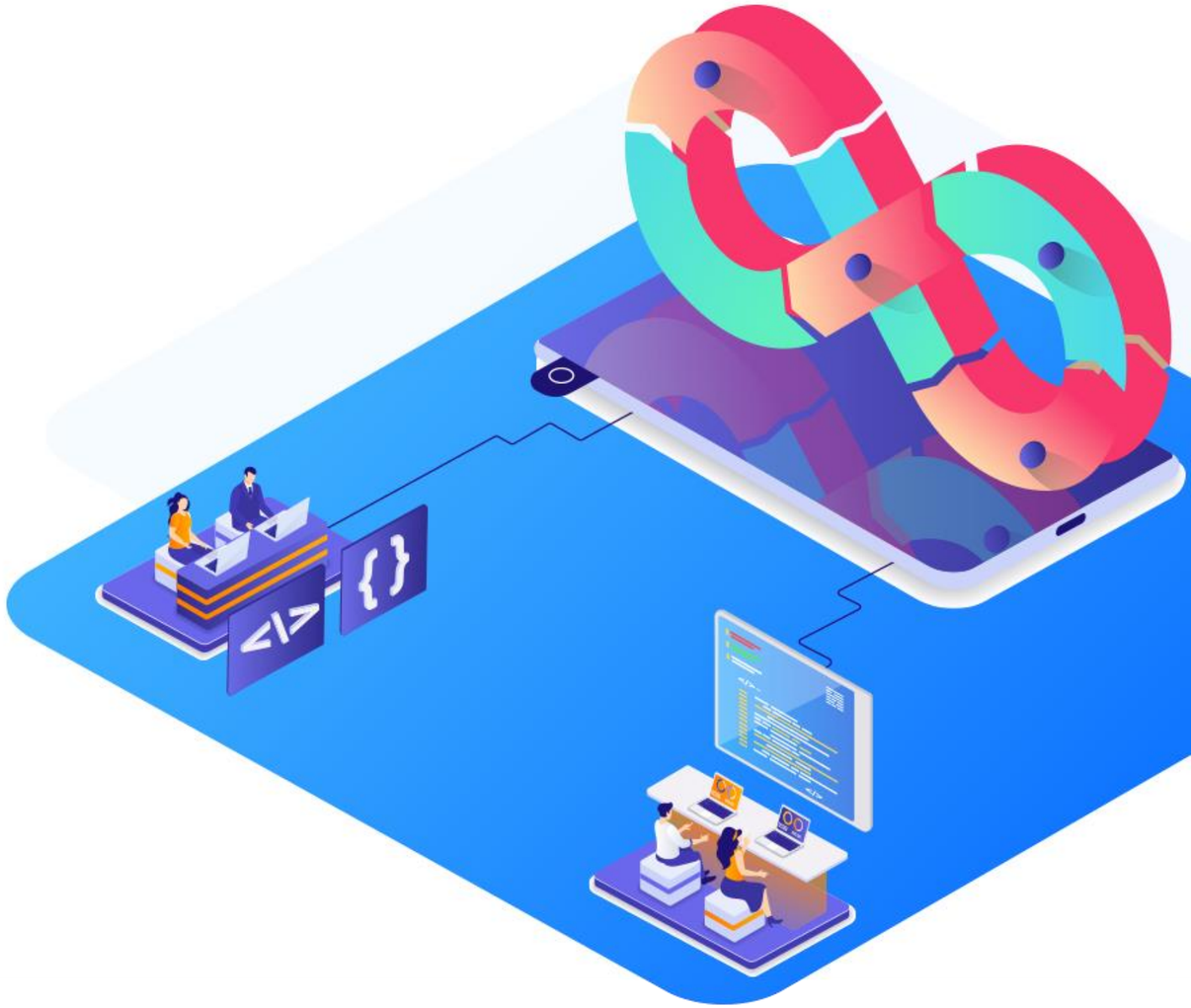
Capstone Project



Super Mario Game Deployment

Objective

Super Mario game deployment faces challenges in setting up a scalable and efficient infrastructure to host the game application. The project requires transforming manual deployment processes into automated workflows, leveraging modern DevOps tools and cloud technologies. This transformation is critical for ensuring high availability, seamless updates, and robust performance of the application.

The project aims to:

1. Set up a secure and scalable AWS infrastructure using Terraform
2. Automate the deployment and configuration process using tools like Docker and Kubernetes
3. Host the game application on an EKS cluster with load balancer integration

COURSE-END PROJECT

Problem Statement and Motivation

Real-time scenario:

A gaming company, PixelPlay Studios, has developed an online multiplayer version of the Super Mario game. With the game's growing popularity, the company faces challenges in managing its deployment infrastructure.

They want to ensure that:

- The application scales dynamically to handle fluctuating user traffic during peak gaming hours.
- Deployment is seamless, minimizing downtime during updates to keep users engaged.
- Infrastructure is cost-efficient while maintaining high availability across regions.

To address these challenges, PixelPlay Studios adopts **DevOps practices** to deploy and manage their game.

Industry Relevance

The following tools used in this project serve specific purposes within the industry:

- **AWS (Amazon Web Services):** Powers scalable and reliable cloud infrastructure for industries like e-commerce, gaming, and media streaming
- **Terraform:** Automates infrastructure management through code, ensuring consistency across multi-cloud and hybrid environments
- **Docker:** Standardizes application environments, enabling seamless deployment and scalability across platforms
- **Kubernetes (Kubectl):** Orchestrates containerized applications, managing scaling, load balancing, and fault tolerance in distributed systems
- **Jenkins:** Automates CI/CD pipelines for faster feature delivery and seamless updates in software development and IT operations
- **AWS CLI:** Provides command-line control over AWS services, enabling automation, scripting, and efficient resource management

Business Impact of the Tools

- **AWS (Amazon Web Services):** Provides a scalable and reliable cloud infrastructure, ensuring the Super Mario game can handle high traffic during peak gaming periods
- **Terraform:** Automates infrastructure provisioning, reducing deployment setup time and ensuring consistent environments for the game's backend
- **Docker:** Enables portable and consistent deployment of the game application, reducing errors and improving the speed of updates
- **Kubernetes (Kubectl):** Orchestrates containers for seamless scaling and load balancing, ensuring uninterrupted gaming experiences for users worldwide

Business Impact of the Tools

- **Jenkins:** Automates the CI/CD pipeline for rapid updates and bug fixes, minimizing downtime and maintaining player engagement
- **AWS CLI:** Facilitates efficient management of AWS resources, accelerating configuration changes and deployment for the game application

Tasks

The following tasks outline the process of implementing Super Mario game deployment:

1. **Infrastructure provisioning with Terraform:**
    I. The team uses Terraform to define and automate the setup of AWS resources, including EC2 instances, S3 buckets, and EKS clusters.
    II. The automated provisioning ensures quick scalability during traffic surges, such as weekend tournaments.
2. **Containerization with Docker:**
    I. The game application is containerized using Docker, making it lightweight and portable.
    II. Docker containers ensure the application runs consistently across environments, from development to production.
3. **Deployment on EKS using Kubernetes:**
    I. Kubernetes orchestrates the deployment of Docker containers in an Elastic Kubernetes Service (EKS) cluster.
    II. Auto-scaling and load balancing ensure users experience minimal latency and optimal game performance.

Tasks

The following tasks outline the process of implementing Super Mario game deployment:

4. **Continuous deployment Pipeline with Jenkins:**
    I. A CI/CD pipeline is set up using Jenkins to automate updates and bug fixes.
    II. This minimizes downtime during feature rollouts or performance optimizations.
5. **Monitoring and logging:**
    I. AWS CloudWatch and Prometheus are integrated to monitor application performance and log errors in real-time.
    II. Alerts are set up for quick incident response, ensuring minimal disruption to players.
    III. Monitor application capacity and runtime performance with Prometheus, and design real-time graphs using Grafana for visual analysis

Project References

- **Task 1: Refer to the course:** Configuration Management with Ansible and Terraform
- **Task 2: Refer to the course:** Containerization with Docker
- **Task 3: Refer to the course:** DevSecOps - Principles and AWS Cloud Security
- **Task 4: Refer to the course:** DevOps Foundations Version Control and CICD with Jenkins
- **Task 5: Refer to the course:** Monitoring and Logging in DevOps

COURSE-END PROJECT

Output Screenshots

Interface of Super Mario game:

Thank you