JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

BTECH CSE 6TH SEMESTER

MINOR PROJECT-2 (15B19CI691)

WORK SUMMARY SHEET

TETRIS

(Deploying and automating two versions of a Dockerized Tetris Game simultaneously with scaling using ArgoCD, Terraform, Jenkins, Kubernetes)

Mentor:	**Submitted By:**
Dr. Amarjeet Prajapati	Mukund Sarda (21103105)
	Karanjot Singh (21103096)
	Priyanshu Jain (21103102)

1. Motivation Behind the Project:

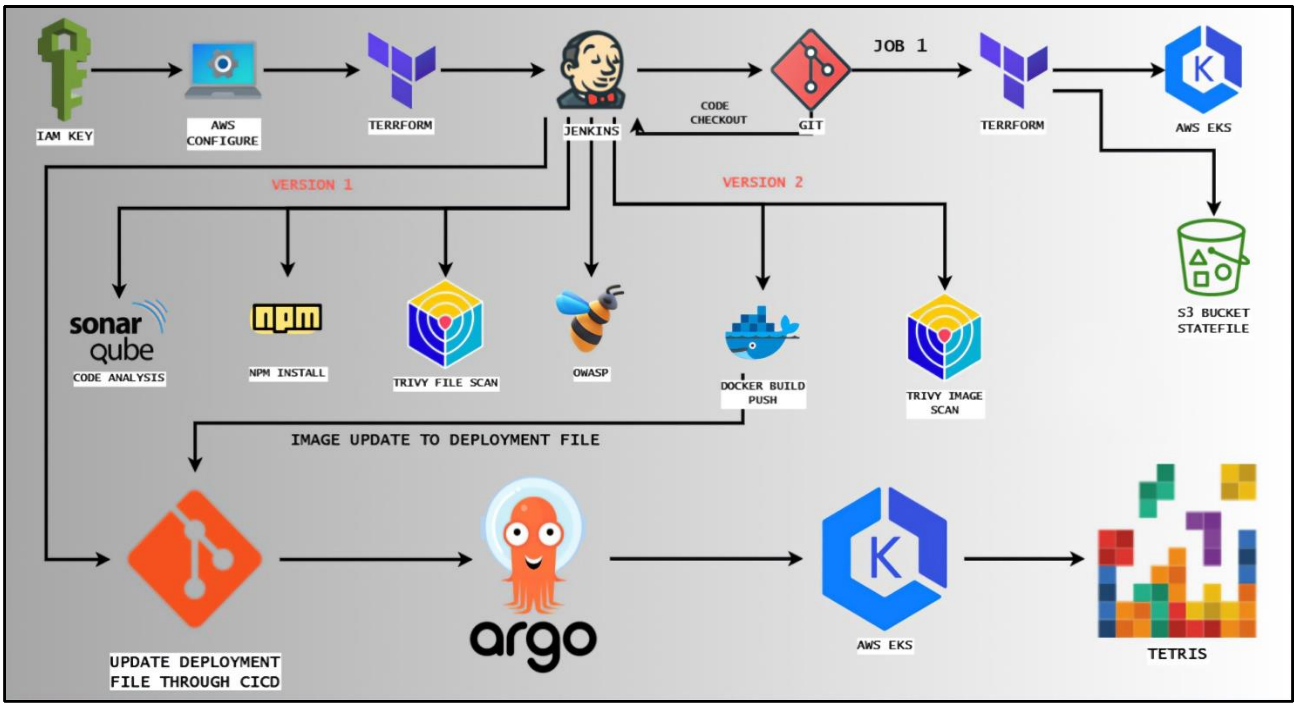
The aim of this project is to deploy and automate two versions of a Dockerized Tetris game simultaneously on Kubernetes clusters using ArgoCD for continuous deployment, Terraform for infrastructure provisioning, Jenkins for continuous integration, and Kubernetes for container orchestration. Additionally, the project aims to implement auto-scaling mechanisms to manage fluctuating demand efficiently.

The motivation behind this project stems from the desire to showcase the power of DevOps practices in modern software development. By automating the deployment process using tools like ArgoCD, Terraform, Jenkins, and Kubernetes, we aim to streamline the development lifecycle, reduce manual errors, and enhance deployment efficiency. Additionally, implementing auto-scaling mechanisms allows us to optimize resource utilization and meet varying demand effectively.

2. Type of Project:

Research cum Development Project

3. Overall Design of Project:



4. Languages and Technologies Used:

4.1. ReactJs: ReactJS is a JavaScript library for building user interfaces, particularly single-page applications. It allows developers to create reusable UI components, making the development process more efficient.

4.2. Docker: Docker is a containerization platform that allows developers to package applications and their dependencies into containers. These containers can then be deployed consistently across different environments.

4.3. AWS EKS stack: AWS EKS (Amazon Elastic Kubernetes Service) is a managed Kubernetes service provided by Amazon Web Services (AWS). It simplifies the process of deploying, managing, and scaling containerized applications using Kubernetes.

4.4. ArgoCD: ArgoCD is a declarative, GitOps continuous delivery tool for Kubernetes. It automates the deployment of applications to Kubernetes clusters by syncing the desired state defined in Git repositories with the actual state in the clusters.

4.5. Terraform: Terraform is an infrastructure as code (IaC) tool used for provisioning and managing cloud infrastructure resources. It allows developers to define infrastructure configurations in code, enabling automated provisioning and versioning of infrastructure.

4.6. Jenkins: Jenkins is an open-source automation server used for continuous integration and continuous delivery (CI/CD). It automates the build, test, and deployment processes, enabling faster and more reliable software delivery.

4.7. Kubernetes: Kubernetes is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications. It provides features such as service discovery, load balancing, and auto-scaling.

5. Proposed Methodology:

5.1. Dockerizing Tetris:

- Both versions of Tetris are containerized using Docker to encapsulate the game along with its dependencies.

- Docker ensures consistency in deployment across different environments, enabling seamless deployment and scaling.

5.2. Continuous Integration with Jenkins:

- Jenkins is utilized for continuous integration, automating the build and testing process.

- Upon code commit, Jenkins triggers a build pipeline, running tests and generating Docker images for both Tetris versions.

5.3. Continuous Deployment with ArgoCD:

- ArgoCD is employed for continuous deployment, ensuring that the latest Docker images are deployed to the Kubernetes clusters.

- GitOps principles are followed, with ArgoCD monitoring the Git repository for changes and automatically synchronizing the desired state with the actual state in the clusters.

5.4. Infrastructure Provisioning with Terraform:

- Terraform is used for infrastructure provisioning, enabling infrastructure as code (IaC) practices.

- Terraform scripts define the desired infrastructure state, including Kubernetes clusters and associated resources, ensuring consistency and reproducibility.

5.5. Container Orchestration with Kubernetes:

- Kubernetes orchestrates the Tetris containers, providing features such as auto-scaling, load balancing, and service discovery.

- Kubernetes ensures high availability and fault tolerance by automatically managing container placement and health.

5.6. Auto-Scaling Mechanisms:

- Kubernetes Horizontal Pod Autoscaler (HPA) is configured to automatically scale the Tetris deployments based on CPU utilization or custom metrics.

- Auto-scaling ensures optimal resource utilization and responsiveness to fluctuating demand, enhancing user experience.

6. Description of The Work:

Tetris is a classic tile-matching puzzle video game that has been popular for decades. In this project, we leverage modern DevOps tools and techniques to deploy and manage two versions of Tetris concurrently. By containerizing the game using Docker, we ensure consistency in deployment across different environments. Kubernetes facilitates efficient container orchestration, ensuring scalability and high availability. In the realm of modern software development and deployment, automation and scalability are key factors for efficient and reliable operations. This project focuses on deploying two different versions of a Tetris game within Docker containers, leveraging container orchestration through Kubernetes, continuous integration and continuous deployment (CI/CD) with Jenkins, GitOps principles with ArgoCD, and infrastructure provisioning with Terraform. The Tetris game serves as a simple yet illustrative application for demonstrating the deployment and scaling capabilities of modern DevOps tools and practices.

7. Division of The Work Among Students

7.1. Karanjot:

- **DigitalOcean Setup and Configuration:** Set up a DigitalOcean account, generate an API token, and manage resources in projects. Use Terraform or DigitalOcean's control panel to provision Droplets, configure deployment pipeline components, and ensure SSH key management

- **Terraform Installation and Setup:** Download Terraform, set it up on Windows, add Terraform to system's PATH.

- **Terraform Provisioning:** Clone Tetris repositories, navigate to Terraform directory, initialize Terraform, validate code, plan resources, apply changes.

- **Jenkins Installation and Setup:** Use Terraform to provision EC2 instance with Jenkins, configure Jenkins, install necessary tools.

7.2. Priyanshu:

- **Jenkins Pipeline Job Configuration:** Assist Priyanshu in creating Jenkins pipeline jobs.

- **SonarQube Integration:** Set up SonarQube for code analysis, integrate SonarQube with Jenkins pipeline jobs.

- **Kubernetes Deployment Update:** Automate image updates in Kubernetes deployment using GitHub tokens and Jenkins.

7.3. Mukund:

- **ArgoCD Installation and Setup:** Install ArgoCD on Kubernetes, expose ArgoCD using a Load Balancer, configure repositories, manage applications.

- **Verification and Testing:** Verify installations, test deployments, ensure access to Tetris versions through load balancer URLs.

- **Documentation and Presentation:** Document entire setup process, prepare presentation summarizing deployment pipeline and its components.

8. Results:

After deploying both Tetris versions using ArgoCD, Terraform, and Jenkins, here are the outcomes:

8.1. Tetris Applications Deployment:

- Version 1: Successfully deployed on AWS EC2 with Jenkins managing CI/CD and SonarQube for code analysis.

- Version 2: Deployed alongside Version 1, with automated image updates ensuring synchronization.

8.2. Pipeline Efficiency:

- Streamlined deployment with reduced manual intervention.

- Automated testing and CI/CD facilitated rapid updates.

- ArgoCD integration enhanced deployment management.

8.3. Team Collaboration:

- Collaborative efforts led to successful deployment.

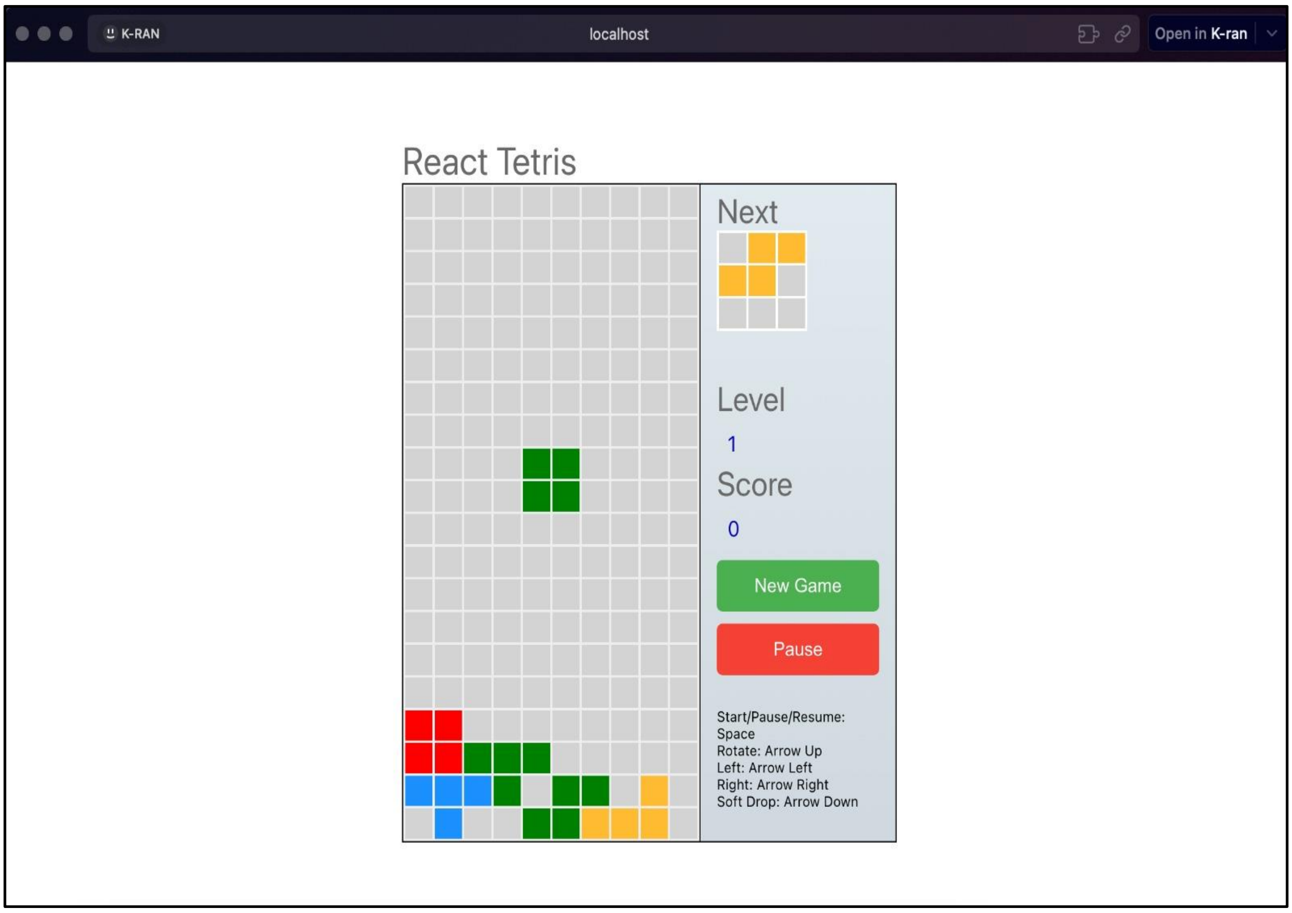
- Hands-on experience improved skills in DevSecOps tools.

- Documentation provided insights into pipeline architecture.

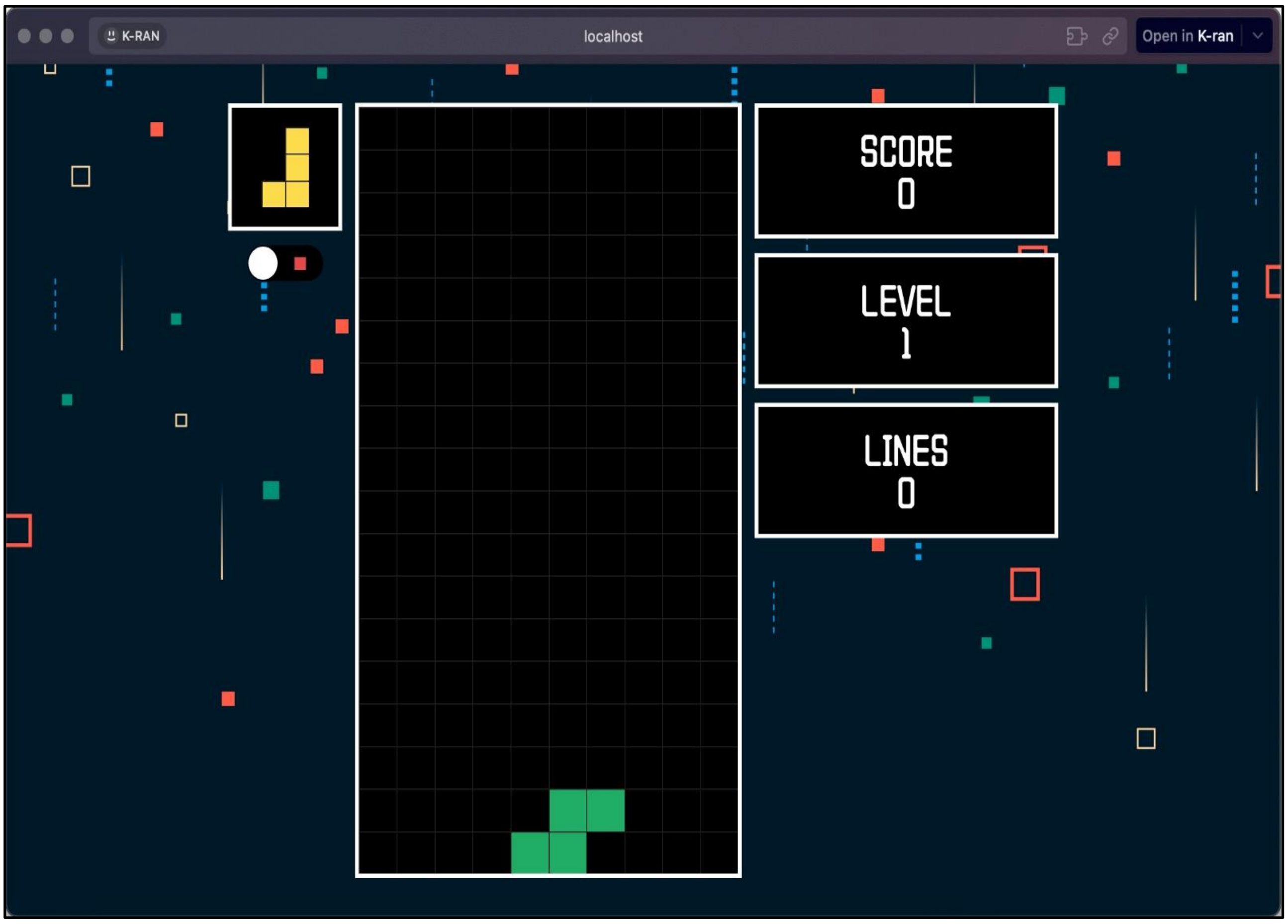
Overall, the project showcased efficient automation, collaboration, and learning opportunities in modern DevSecOps practices.

9. Conclusion:

In conclusion, this project demonstrates the seamless deployment and management of two versions of Tetris using modern DevOps tools and practices. By leveraging Docker, Jenkins, ArgoCD, Terraform, and Kubernetes, we achieve automation, scalability, and reliability in the deployment process. The implementation of auto-scaling mechanisms further enhances the efficiency and responsiveness of the Tetris deployments, showcasing the benefits of DevOps in modern software development.



Start/Pause/Resume: Space
Rotate: Arrow Up
Left: Arrow Left
Right: Arrow Right
Soft Drop: Arrow Down


K-RAN
localhost
Open in K-ran
SCORE
0
LEVEL
1
LINES
0