PROJECT DOCUMENTATION

1. TITLE :

GitOps-Based CI/CD Pipeline for TETRIS Gaming Application by using Node.js, ArgoCD and Kubernetes(Kops Cluster) .

2. INTRODUCTION :

Modern DevOps practices demand continuous, reliable, and scalable deployment mechanisms. GitOps simplifies this by using Git as the single source of truth for Kubernetes cluster configuration and application deployment. This project demonstrates a practical implementation of GitOps using ArgoCD to automatically sync and deploy Kubernetes manifests files from a GitHub repository.

3.ABSTRACT :

This project aims to showcase how GitOps principles can be applied to automate the deployment and lifecycle management of applications on Kubernetes using ARGOCD. So am creating CI/CD pipeline for automate purpose. Here in this project am creating KOPS Cluster because of my cluster is high availability so it is used to host applications whose manifests are stored in a version-controlled GitHub repository. ArgoCD is configured to continuously monitor the repository and apply changes in manifest files by changing the update versions and visualize the real-time synchronization via the ArgoCD dashboard.

4. TOOLS USED :

Git, Jenkins, Docker, GitHub, DockerHub, kubernetes(Kops Cluster), kubectl, ArgoCD.

5. STEPS INVOLVED IN BUILDING THE PROJECT :

#STEP-1 : First of all taking a New Server with Server configurations are server_name : project-10 AMI : Amazon linux kernel 5.10, Instance_type : t2.medium, Key_Pair : sai-kp, Security_Group : All-Traffic, EBS : 20GIB . After connecting to Server with SSH .
#STEP-2 : Now To Set-Up the Tools Git, Docker, Jenkins, kubernetes(Kops Cluster), kubectl, ArgoCD.
#STEP-3 : Now am creating a New Repository in GitHub and the Repository_Name is project-10 .
#STEP-4 : After am accessing the Jenkins dash board with public-ip:8080 . 8080 it is a port number . Now am creating a New Job for the pipeline running purpose .
- So Here am using CI/CD pipeline for some advanced stages because of automate purpose .
#Step-5 : Now am installed some plugins
PLUGIN_NAME : pipeline stage view , node.js, docker pipeline, eclipse temurin installer, sonarqube scanner, OWASP Dependency Check .
- Now writing and implementing CI/CD WorkFlow with the stages .
    stage-1 : It is a clean workspace .
    stage-2 : To get the source code from github to ci server .
    stage-3 : Here am using Sonarqube for the code quality analysis purpose because of to scan the source code .
    stage-4 : Next am using quality gates because the purpose of code must pass before it can move forward in the build, test, or deployment process . if the code is incorrect then the pipeline is aborted .
    stage-5 : To build the source code .
    stage-6 : OWASP Dependency-Check into a CI/CD pipeline helps checks known vulnerabilities in your project dependencies.
    stage-7 : To build the dockerfile .then a New docker-image has to come and also To Rename that docker-image .
    stage-8 : So here am using Trivy Tool because to scan the New docker-image .
    stage-9 : Last stage is new docker-image to push the docker-hub .
#STEP-6 : After my CI/CD pipeline is successfully executed . Then a New Docker Image is created .
NOTE : am taking the those screenshots of CI/CD pipeline and docker image are pasted in my GitHub Repository .
- There are some command are used in my project .
    - COMMAND : docker build -t image_name - To build the dockerfile .
    - COMMAND : docker push image_name - To push image to docker-hub .
#STEP-7 : Now am deploying my application in ARGOCD. So Here am creating KOPS cluster because of my cluster high is availability .
Now to set-up a Kubernetes Kops Cluster in server by using ,
- awscli commands . (collecting commands from via browser)
- kubectl commands . (collecting commands from via browser)
- kops cluster commands . (collecting commands from via browser)
- Now am creating kops cluster by using command .

- COMMAND : kops create cluster --name=saikumar.k8s.local --zones=us-east-1a,us-east-1b --master-size=t2.medium --master-count=1 --master-volume-size=30 –node-size=t2.medium –node-count=3 –node-volume-size=30 .

- COMMAND : kops get cluster - To check the list of clusters .

- After KOPS cluster set-up is completed . Now am creating two files for depoying my application .
- 1. one is deployment YAML file and the name is deployment.yaml . 2. second is service YAML file and the name is service.yaml .
- Then After Push application deployment manifests files to a GitHub Repository by using commands .

- COMMAND : git push -u origin main - To push the files from local to central github repository .

#STEP-8 : Now am install ARGOCD by using commands .

COMMANDS : kubectl create namespace argocd .

COMMANDS : kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml .

COMMANDS : kubectl get all -n argocd .

EXPOSE ARGOCD SERVER:
COMMANDS : kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
COMMANDS : yum install jq -y
COMMANDS : export ARGOCD_SERVER='kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.
.status.loadBalancer.ingress[0].hostname''
COMMANDS : echo $ARGOCD_SERVER
COMMANDS : kubectl get svc argocd-server -n argocd -o json | jq --raw-output .status.loadBalancer.ingress[0].hostname
The above command will provide load balancer URL to access ARGO CD .

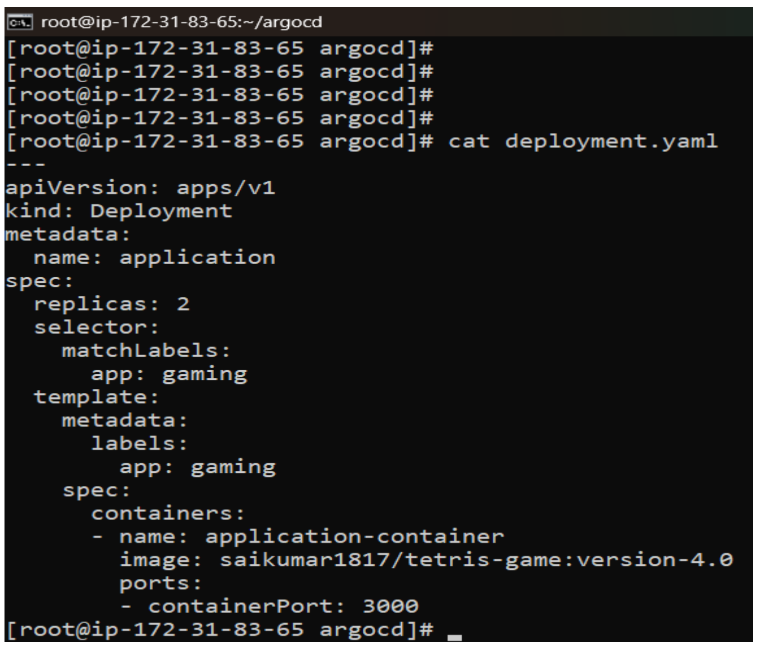
- Now am accessing ARGOCD Dashboard after am login by giving the credentials .
- In ArgoCD UI, enable auto-sync and self-heal options. Any changes pushed to the GitHub repo will now reflect in the cluster automatically.

#STEP-9 : Configure ArgoCD to auto-sync and deploy from Git

- In Argocd am creating a New application for deploying my tetris gaming app and am integrating Argocd to GitHub by giving a GitHub Repository URL . After my tetris gaming app is deployed then am accessing with public-ip:31574 .

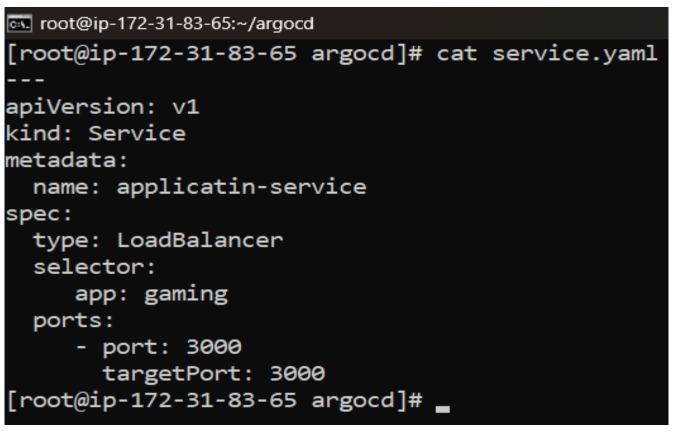
NOTE : am taking the screenshots of deployed Tetris Gaming Application and pasted in GitHub Repository into Delivariables folder .

This is the SCREENSHOT OF deployment YAML file .



```
root@ip-172-31-83-65:~/argocd
[root@ip-172-31-83-65 argocd]#
[root@ip-172-31-83-65 argocd]#
[root@ip-172-31-83-65 argocd]#
[root@ip-172-31-83-65 argocd]#
[root@ip-172-31-83-65 argocd]# cat deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: application
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gaming
  template:
    metadata:
      labels:
        app: gaming
    spec:
      containers:
      - name: application-container
        image: saikumar1817/tetris-game:version-4.0
        ports:
        - containerPort: 3000
[root@ip-172-31-83-65 argocd]#
```

This is the SCREENSHOT of service YAML file .



```
root@ip-172-31-83-65:~/argocd
[root@ip-172-31-83-65 argocd]# cat service.yaml
---
apiVersion: v1
kind: Service
metadata:
  name: applicatin-service
spec:
  type: LoadBalancer
  selector:
     app: gaming
  ports:
     - port: 3000
       targetPort: 3000
[root@ip-172-31-83-65 argocd]#
```

- If also checks by using commands .

  - COMMAND : kubectl get po - To Check the list of pods .
  - COMMAND : kubectl get deploy - To Check the list of Deployments .
  - COMMAND : kubectl get svc - To check the list of services .

NOTE : am taking the SCREENSHOTS of list of pods, services, deployments and sending to my GitHub Repository into Deliveriables folder .

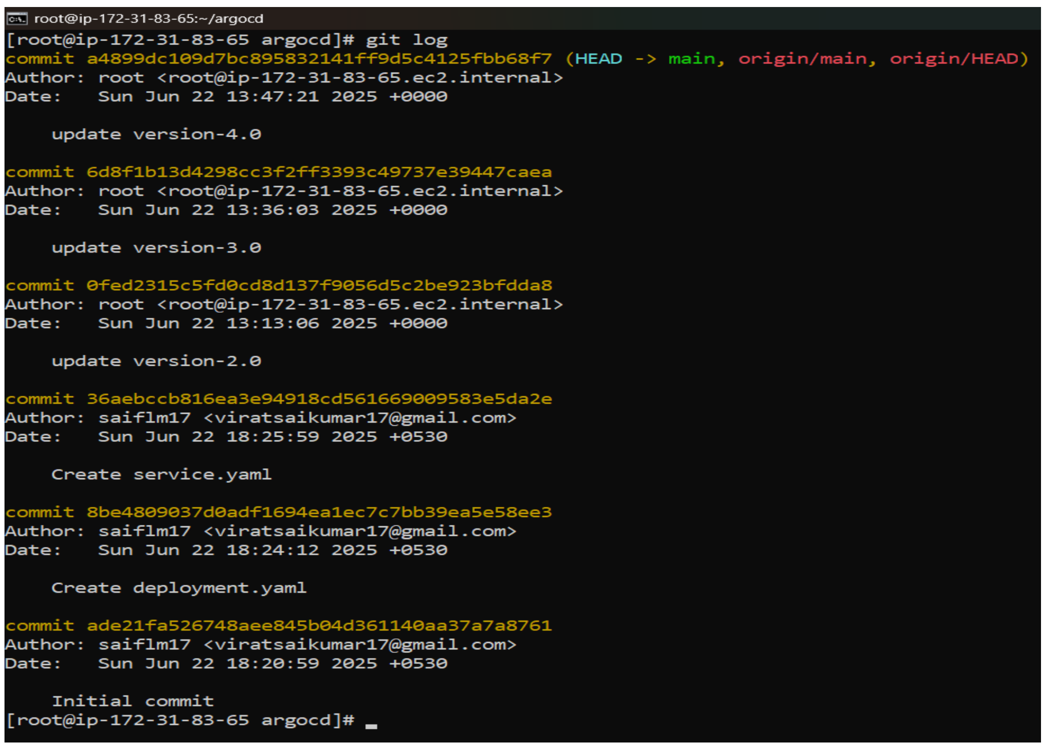
#STEP-10 : Now am Updating versions via Git commits and observing changes.

- Now cloning the repository into my server . After am updating the versions version-1.0, version-2.0, version-3.0, version-4.0 .

- And tracking and committing the files by using commands .

  - COMMAND : git add - To track the files .
  - COMMAND : git commit -m "commit_message" file_name - To commit the file .
  - COMMAND : git log - To see the commits .

- This is the SCREENSHOT of Git Commits by updating the versions .



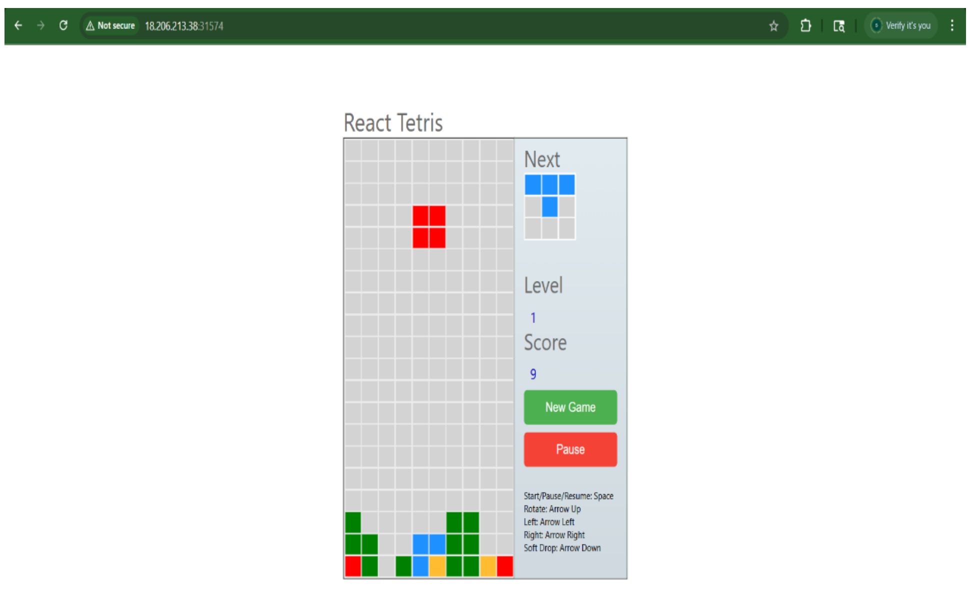
```
root@ip-172-31-83-65:~/argocd
[root@ip-172-31-83-65 argocd]# git log
commit a4899dc109d7bc895832141ff9d5c4125fbb68f7 (HEAD -> main, origin/main, origin/HEAD)
Author: root <root@ip-172-31-83-65.ec2.internal>
Date:   Sun Jun 22 13:47:21 2025 +0000

    update version-4.0

commit 6d8f1b13d4298cc3f2ff3393c49737e39447caea
Author: root <root@ip-172-31-83-65.ec2.internal>
Date:   Sun Jun 22 13:36:03 2025 +0000

    update version-3.0

commit 0fed2315c5fd0cd8d137f9056d5c2be923bfdda8
Author: root <root@ip-172-31-83-65.ec2.internal>
Date:   Sun Jun 22 13:13:06 2025 +0000

    update version-2.0

commit 36aebccb816ea3e94918cd561669009583e5da2e
Author: saiflm17 <viratsaikumar17@gmail.com>
Date:   Sun Jun 22 18:25:59 2025 +0530

    Create service.yaml

commit 8be4809037d0adf1694ea1ec7c7bb39ea5e58ee3
Author: saiflm17 <viratsaikumar17@gmail.com>
Date:   Sun Jun 22 18:24:12 2025 +0530

    Create deployment.yaml

commit ade21fa526748aee845b04d361140aa37a7a8761
Author: saiflm17 <viratsaikumar17@gmail.com>
Date:   Sun Jun 22 18:20:59 2025 +0530

    Initial commit
[root@ip-172-31-83-65 argocd]# _
```

6. CONCLUSION :

This project successfully demonstrates a GitOps-based deployment strategy using ArgoCD and Kubernetes. By version-controlling Kubernetes manifests in GitHub, and allowing ArgoCD to monitor and sync changes, we achieved a fully automated and reliable CI/CD system. The project highlights GitOps as an essential modern DevOps practice.

NOTE : ALL SCREENSHOTS am taken and sending to GitHub Repository into Deliverables folder .

This is the screenshot of deployed my tetris gaming application .


Not secure 18.206.213.38:31574
Verify it's you
React Tetris
Next
Level
1
Score
9
New Game
Pause
Start/Pause/Resume: Space
Rotate: Arrow Up
Left: Arrow Left
Right: Arrow Right
Soft Drop: Arrow Down