ONE IMMERSIVE

MONTHLY REPORT

Name: Neeraj Polavarapu
Contact: neerajchowdhary12@gmail.com
Github: https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE
(All files are uploaded to github)

CONTENTS:

- **WEBRTC REPORT**:
  I have prepared a comprehensive report on WebRTC, covering its components and functionalities. The report explores GetUserMedia() for accessing webcams and media devices, RTCPeerConnection() for secure peer-to-peer connections, and RTCDataChannel() for data sharing. It highlights the advantages of WebRTC, compares it with WebSockets, explains signaling using SDP, and discusses NAT traversal with STUN and TURN servers. The report also outlines the connection flow through a signaling server and emphasizes the role of GetUserMedia().
  Link for report:
  https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/Web_Real-Time_Communication.pdf

- **PIXEL STREAMING REPORT**:

I have prepared a report which focuses on the components involved in pixel streaming. It includes the Pixel Streaming plugin, which encodes rendered frames, and the Signalling and Web Server responsible for connection negotiation. The Pixel Streaming Application, Pixel Streaming Plugin, and Signalling Server play crucial roles in establishing WebRTC peer-to-peer connections. The Video Encoder compresses frames for streaming, and the Streaming Server handles transmission. Client devices display the streamed application, while the Streaming Client Application receives the video stream. The report also discusses address restricted NAT, port-restricted NAT, and symmetric NAT, and mentions the setup of a TURN server using the CoTURN open-source solution.
Link for report:
https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/PIXEL_STREAMING.pdf

- **KUBERNETES NOTES**:
  I have made a notes which contains all the important kubernetes commands and which can directly interact with coreweave kubernetes cluster from your own local machine itself.
  Link for Notes:
  https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/KUBERNETES.pdf

- **OMNIVERSE NUCLEUS REPORT**:
  I have made a small report on the Idea of Omniverse Nucleus. Nucleus is a server that facilitates collaborative work across different applications and users, such as Houdini, Maya, Blender, and Adobe. It utilizes a publisher-subscriber pattern and features components like Navigator, USD Composer, Deep Search, and Nucleus Connector.
  Link for report:
  https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/Omniverse_Nucleus.pdf

- **KUBERNETES**:
  The major part which I contributed for this company is Kubernetes. Following is the work and link for files I have done for setting up kubernetes and deploying sample apps and game. Which can be autoscaled and can perform loadbalancing.

- **Timeapp_v1**:
  In this i have made a flask time app which shows current time in port 5000, this is made to check asynchronous communication from kubernetes, made 1 app.py flask app and dockerfile for creating image of it and 1 deployment.yaml file for creating pods and

service.yaml file for creating cluster ip service for enabling pod level communication and acts as router and 1 ingress.yaml file for accessing the pod from anywhere with a domain and also loadbalancing.
Link :
https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/tree/main/time-app_V1

- **Timeapp_v2**:
  In this also made a flask time app which shows current time but with more modified webpage and added some html text to it. This will work on any port irrespective of 5000, works in 80 as well(standard port for any IP). This contains same 1 deployment yaml file for creating pods, 1 service yaml for creating Cluster Ip service, 1 ingress for creating domain with coreweave.
  Link:
  https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/tree/main/time-app_V2

- **Webapp_v1**:
  Here designed a html webpage and created image of it and deployed in creweave with domain created by ingress, in this autoscaling is also tested both horizontal autoscaling and vertical autoscaling.
  Link:
  https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/tree/main/web-app_V1

- **Test_yaml's:**
  1. Test_deployment_v1.yaml : This YAML file defines a Kubernetes Pod named 'nginx-pod1' with labels for integration and a 'todo' application. The Pod includes a container named 'nginx-container' running the latest version of the Nginx image. It exposes port 80 to allow communication with the Nginx web server.
     Link:
     https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/test_deployment_v1.yaml
  2. Test_deployment_v2.yaml: This deployment file is the updated version to v1 version and it has replicaset and creating 2 replicas, and also pulling image from private docker hub repository with port as 5000.
  3. Test_deployment_v3.yaml: This deployment file is the updated version to v2 version and it has environment variables feature added and also volumes are created and mounted.
  4. Test_ingress: This is the yaml file which creates ingress manifest and unables us to communicate the specified service and then pods.

5. Test_pod: This is the yaml file which creates individual pods without any replicaset or any deployment. And i also added a feature in this called secretes. Through which only we can access private and secured image registries.
6. Test_service_v1: This is a cluster IP service manifest file created for acessing flask-app pod through 5000 port and accessible at 6000 port.
7. Test_service_v2: This is a cluster IP service manifest file created for acessing time-app pod through 5000 port and accessible at 6000 port.

- **Main_yaml's:**
  1. Main_deployment.yaml: This is the deployment manifest file created for initializing metahuman pods in coreweave kubernetes cluster.
     Link:
     https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/blob/main/main_deployment.yaml
  2. Main_hpa.yaml: This is the yaml manifest used for horizontal pod autoscaling and was set to minimum as 1 and maximum as 5.
  3. Main_cpa.yaml: This is the yaml manifest file responsible for cluster autoscaling, in this nodes will get autoscaled instead of pods, because in some cases we need excess resources and only nodes will serve the purpose.
  4. Main_ingress.yaml: Ingress file with domain mentioned so that it will be communicated with desired Cluster IP service at desired port.
  5. Main_traffic-generator.yaml: This manifest file is used to generate traffic we needed to check whether autoscaling is working correctly.
  6. Main_node-port.yaml: This is yaml manifest file used to create NordPort service. Which makes all the nodes to communicate and share the resources with each other.
  7. Main_pod.yaml: This code can be used when you decided not to create deployment or replicaset and just go with pod level.
  8. Main_replicaset.yaml: This code helps to create replicaset for pods without deployment feature.
  9. Main_service.yaml: This is clusterIP service used to create contact between pods and acts as a bridge between pods and ingress.

- **Volumes(Mongodb server):**
  Link:
https://github.com/neeraj-369/KUBERNETES_INTERN_ONEIMMERSIVE/tree/main/volumes
  - Deployment.yaml: This deploys a mongodb server with also creating volumes and specifying the mounts to it.
  - Pv.yaml: Persistant volumes are used out of node level and can be utilized even when node is recreated.
  - Pvc.yaml: Persistent volume claims will allocate a portion of persistent volumes for specific pod or deployment.

- Service.yaml: This is the Cluster IP service for accessing mongo db (database) server at port 27017.

- **Game_development:**
    - **Ingress;**
        - **Game_ingress.yaml:** This is the ingress for making the game public through port 80. But through this port you cant access pixel streaming.
        - **Signallin_ingress.yaml:** This ingress is for making the pixel streaming public through 1025 port and which is playable.
    - **Pods:**
        - **Coturn_pod.yaml:** This pod is created for establishing coturn server and establish connection by getting ice packets.
        - **Game_pod.yaml:** pod which runs game in it at port 80 with necessary volumes and environmental variables.
        - **Instance_manager_pod.yaml:** This pod is for controlling instances and sending those instances ot signalling_server pod through port 8888.
        - **Signalling_pod.yaml:**pod which will create an instance for signalling server and enables pixel streaming it listens at port 8888 from game_server and sends to port 1025 for the gamer so that he can access pixel streaming.
    - **Services:**
      The following are the CLUSTER IP service manifest files for their respective pods which are mentioned above.
        - **Auth_service.yaml**
        - **Coturn_service.yaml**
        - **Game_service.yaml**
        - **Instance_manager_service.yaml**
        - **signalling_service.yaml**

- **Signalling_server and Image Builder:**
  Worked on signalling server and enabling pixel streaming. Working on dependencies necessary for image building and running unreal build game.