**🚀 Task 1: Create a Blog on Case Studies of Why Companies Use Kubernetes and the Benefits They Get**

**✅ Project Objective:**

To understand why organizations adopt Kubernetes by analyzing real-world case studies and how Kubernetes benefits their infrastructure, scalability, and deployment workflows.

**🛠️ Tools & Technologies Used:**

* Kubernetes
* Docker
* Google Cloud / AWS / Azure
* Helm (for package management)
* Jenkins/GitHub Actions (optional CI/CD)

**📸 Screenshots:**

*(Insert images of company case study diagrams or Kubernetes architecture illustrations)*

**🧠 Code Flow Explanation:**

This task is mostly research-based. The process:

1. Identify companies that have migrated to or use Kubernetes.
2. Understand their problems before Kubernetes.
3. Analyze how Kubernetes solved them.
4. Summarize findings in blog format.

**📈 Output/Results:**

**Case Study Highlights:**

1. **Spotify** – Uses Kubernetes to manage over 1,600 services. It improved release frequency and development speed.
2. **Airbnb** – Migrated to Kubernetes to scale services quickly and simplify infrastructure management.
3. **CERN** – Manages large scientific workloads with Kubernetes clusters.
4. **Reddit** – Transitioned to Kubernetes for better autoscaling during traffic spikes.
5. **BlaBlaCar** – Simplified deployment pipelines and improved fault isolation.

**Benefits Observed:**

* Improved scalability
* Self-healing infrastructure
* Easier rollbacks and updates
* Multi-cloud and hybrid-cloud flexibility

**🌐 Task 2: Run the Same Code in Your Environment and Try to Launch More Use Cases of Multi-Tier Websites**

**✅ Project Objective:**

To deploy a multi-tier web application (frontend + backend + database) on Kubernetes and explore variations.

**🛠️ Tools & Technologies Used:**

* Kubernetes (Minikube / GKE / EKS / AKS)
* Docker
* YAML (for K8s configs)
* Nginx / React (frontend)
* Flask / Node.js (backend)
* MongoDB / MySQL (database)

**📸 Screenshots:**

*(Insert screenshots of running pods, services, and application dashboard on browser)*

**🧠 Code Flow Explanation:**

1. **Dockerize** each component (frontend, backend, database).
2. Create Kubernetes YAML files:
   * deployment.yaml for each tier
   * service.yaml for exposing
3. Apply using:

bash

CopyEdit

kubectl apply -f .

1. Use kubectl get pods and kubectl get svc to check status.
2. Expose frontend via NodePort or Ingress.

**📈 Output/Results:**

* Multi-tier app deployed successfully.
* Frontend communicates with backend, which queries the database.
* Variations can include:
  + Adding caching layer (Redis)
  + Scaling backend replicas
  + Load balancing frontend

**📺 Task 3: How to Launch a Live Stream Website on Kubernetes**

**✅ Project Objective:**

To deploy a live streaming platform (e.g., OBS + Nginx RTMP + video player frontend) on Kubernetes.

**🛠️ Tools & Technologies Used:**

* Kubernetes
* Docker
* Nginx with RTMP module
* OBS Studio (for streaming)
* Video.js / React Player (frontend)

**📸 Screenshots:**

*(Include OBS broadcasting, stream visible in browser, pod/service logs)*

**🧠 Code Flow Explanation:**

1. **Dockerize Nginx with RTMP**:
   * Use a prebuilt Nginx-RTMP image or custom Dockerfile.
2. **Create K8s YAMLs**:
   * Nginx RTMP deployment.yaml
   * Frontend player deployment.yaml
   * Services for both
3. **Stream from OBS** to rtmp://<external-ip>/live/stream
4. **Access Player** via <external-ip>:port to view the stream

**Sample RTMP Dockerfile:**

Dockerfile

CopyEdit

FROM alfg/nginx-rtmp

COPY nginx.conf /etc/nginx/nginx.conf

**📈 Output/Results:**

* OBS pushes stream to Nginx in the pod.
* User can watch the stream using a web player.
* Real-time streaming app works reliably in Kubernetes.