**Task 1: Create a blog or case study on how companies are using Jenkins and what benefits they are getting**

* **Project Objective:**  
  Analyze real-world adoption of Jenkins in tech companies to understand how it automates CI/CD pipelines and improves deployment cycles.
* **Tools & Technologies Used:**  
  Jenkins, GitHub, Docker, Kubernetes (in case studies), Markdown/WordPress for blog writing.
* **Code Flow Explanation:**  
  No code, but follow these steps:
  1. Research case studies (e.g., Netflix, Facebook, Salesforce).
  2. Identify Jenkins' use in pipeline orchestration, testing, deployment.
  3. Document benefits: automation, reduced deployment time, error reduction.
* **Output/Results:**  
  A detailed blog or PDF case study highlighting the value Jenkins brings to DevOps workflows.

**✅ Task 2: Create a Jenkins job to install Docker and automatically launch a container**

* **Project Objective:**  
  Automate Docker installation and container launch on any node through Jenkins.
* **Tools & Technologies Used:**  
  Jenkins, Docker, Shell Script, GitHub (for pipeline script), Linux VM.
* **Code Flow Explanation:**
  1. Create a new freestyle/pipeline job.
  2. Use a shell script in the job like:

sh

CopyEdit

sudo apt update

sudo apt install -y docker.io

sudo systemctl start docker

sudo docker run -d -p 8080:80 nginx

* 1. Save and build the job.
* **Output/Results:**  
  Jenkins installs Docker and runs an Nginx container. You can access the web server on port 8080.

**✅ Task 3: Launch Kubernetes pods and expose them using Jenkins**

* **Project Objective:**  
  Automate Kubernetes pod deployment and service exposure using Jenkins pipeline.
* **Tools & Technologies Used:**  
  Jenkins, Kubernetes cluster, kubectl, YAML files.
* **Code Flow Explanation:**
  1. Jenkins pipeline uses shell steps to run:

sh

CopyEdit

kubectl apply -f pod.yaml

kubectl expose pod mypod --port=80 --target-port=80 --type=NodePort

* 1. Use Jenkins to monitor and display the NodePort or endpoint.
* **Output/Results:**  
  Jenkins deploys and exposes a pod, allowing external access to the application.

**✅ Task 4: Set up a multi-node Kubernetes cluster automatically with Jenkins**

* **Project Objective:**  
  Automatically provision a multi-node K8s cluster using Jenkins and infrastructure scripts.
* **Tools & Technologies Used:**  
  Jenkins, Terraform/Ansible, kubeadm, shell scripting.
* **Code Flow Explanation:**
  1. Use Jenkins pipeline with Terraform/Ansible scripts to:
     + Provision VMs
     + Initialize master node with kubeadm
     + Join worker nodes
  2. Configure kubeconfig for access.
* **Output/Results:**  
  Jenkins sets up a multi-node Kubernetes cluster ready for deployment tasks.

**✅ Task 5: Create a Jenkins job to send emails, SMS, download YouTube videos, and send WhatsApp messages**

* **Project Objective:**  
  Automate communication and media downloads via a Jenkins pipeline.
* **Tools & Technologies Used:**  
  Jenkins, YouTube-dl or pytube, Twilio (SMS), WhatsApp API, SMTP (email).
* **Code Flow Explanation:**
  + Sample Jenkins pipeline:

groovy

CopyEdit

pipeline {

agent any

stages {

stage('Download YouTube Video') {

steps {

sh 'youtube-dl https://youtube.com/xyz'

}

}

stage('Send Email') {

steps {

mail bcc: '', body: 'Download complete', subject: 'Status', to: 'you@example.com'

}

}

stage('Send WhatsApp & SMS') {

steps {

sh 'python3 send\_sms.py'

sh 'python3 send\_whatsapp.py'

}

}

}

}

* **Output/Results:**  
  On job execution, Jenkins downloads a video and sends notifications via email, SMS, and WhatsApp.

**✅ Task 6: Create a job for automatically retrieving views and likes of YouTube videos**

* **Project Objective:**  
  Use YouTube Data API to extract video analytics via Jenkins.
* **Tools & Technologies Used:**  
  Jenkins, Python, YouTube Data API, Google Developer Console, JSON.
* **Code Flow Explanation:**
  1. Python script uses API key:

python

CopyEdit

from googleapiclient.discovery import build

youtube = build('youtube', 'v3', developerKey='YOUR\_API\_KEY')

req = youtube.videos().list(part='statistics', id='VIDEO\_ID')

res = req.execute()

print(res['items'][0]['statistics'])

* 1. Jenkins runs this Python script in a scheduled job.
* **Output/Results:**  
  Jenkins periodically retrieves and prints views/likes data from YouTube.