وزارة الاتصالات
وتكنولوجيا المعلومات

NTI
National Telecommunication Institute

iVolve

Cloud DevOps Accelerator Program

Where are we located?

5 global offices providing support round-the-clock

Our presence in 5 global regions gives us the agility to work and coordinate with you without the hassle of time zones and reachability utilizing “Follow-the-Sun” model.

Dubai	Karachi	Riyadh	San Jose	Cairo
UAE	Pakistan	Saudi Arabia	USA	Egypt

iVolve

Cloud DevOps Accelerator Hands-On Labs

- Build Tools Overview (Maven & Gradle)
- Containerization with Docker
- Kubernetes for container orchestration
- CI/CD Pipelines with Jenkins
- GitOps Workflow with ArgoCD
- Linux Automation with Ansible

Build Tools Overview

Build Tools Overview

Lab 1: Building and Packaging Java Applications with Gradle

- Install Gradle.
- Clone source code https://github.com/Ibrahim-Adel15/build1.git
- Run Unit test.
- Build App [generate artifact in build/libs/ivolve-app.jar].
- Run App.
- Verify App is working.

Lab 2: Building and Packaging Java Applications with maven

- Install maven.
- Clone source code https://github.com/Ibrahim-Adel15/build2.git
- Run Unit test.
- Build App [generate artifact in target/ hello-ivolve-1.0-SNAPSHOT.jar].
- Run App.
- Verify App is working.

Containerization with Docker

Containerization with Docker

Lab 3: Run Java Spring Boot App in a Container

- Clone the Application Code https://github.com/Ibrahim-Adel15/Docker-1.git
- Write Dockerfile.
  - ✓ Use Maven base image with java 17
  - ✓ Create work directory
  - ✓ Copy the application source code into the container
  - ✓ Build the app using mvn package
  - ✓ Run the app on jar file located in target/demo-0.0.1-SNAPSHOT.jar
  - ✓ Expose port 8080
- Build app1 Image.
- Run container1 from app1 Image.
- Test the application.
- Stop and delete the container.

Containerization with Docker

Lab 4: Run Java Spring Boot App in a Container

- Clone the Application Code https://github.com/Ibrahim-Adel15/Docker-1.git
- Build the application.
- Write Dockerfile.
  - ✓ Use Java 17 base image
  - ✓ Create work directory
  - ✓ Copy the JAR file into the container
  - ✓ Run the app on jar file located in target/demo-0.0.1-SNAPSHOT.jar
  - ✓ Expose port 8080
- Build app2 Image (Note the image size).
- Run container2 from app2 image.
- Test the application.
- Stop and delete the container.

Containerization with Docker

Lab 5: Multi-Stage Build for a Node.js App

- Clone the Application Code https://github.com/Ibrahim-Adel15/Docker-1.git
- Write Dockerfile with Multi-stage.
  - ✓ Use Maven base image for first stag
  - ✓ Copy the application code into the container
  - ✓ Build the app using mvn package
  - ✓ Use java base image for second stag
  - ✓ Copy JAR file from first stag
  - ✓ Expose port 8080
  - ✓ Run the app
- Build app3 Image (Note the image size).
- Run container3 from app3 image.
- Test the application.
- Stop and delete the container.

Containerization with Docker

Lab 6: Managing Docker Environment Variables Across Build and Runtime

- Clone the Application Code https://github.com/Ibrahim-Adel15/Docker-3.git
- Write Dockerfile
  - ✓ Use python image
  - ✓ Install flask
  - ✓ Expose port 8080
  - ✓ Run python command on app.py
- Build Docker Image
- Run 3 containers and set both environment variables (APP_MODE & APP_REGION) as following:
  1. (development, us-east) as variables in the command when run docker container
  2. (staging, us-west) in a separate file and pass the file name in the command
  3. (production, canada-west) in the Dockerfile

Containerization with Docker

Lab 7: Docker Volume and Bind Mount with Nginx

- Create **nginx_logs** volume to persist Nginx logs and verify it in the default volumes path.
- Create a directory nginx-bind/html to serve a custom HTML file from your host machine.
- Create index.html file with “Hello from Bind Mount” syntax in nginx-bind/html directory.
- Run Nginx container with the following:
  - ✓ Volume for /var/log/nginx
  - ✓ Bind Mount for /usr/share/nginx/html
- Verify Nginx page by running curl command from your local machine.
- Change in the index.html file in your local machine then verify Nginx page again.
- Verify logs are stored in the nginx_logs volume.
- Delete the volume.

Containerization with Docker

Lab 8: Custom Docker Network for Microservices

- Clone the frontend and backend Code https://github.com/Ibrahim-Adel15/Docker5.git
- Write Dockerfile for frontend and create image.
  - ✓ Use python image
  - ✓ Install packages in requirements.txt file
  - ✓ Expose port 5000
  - ✓ Run python command on app.py
- Write Dockerfile for backend and create image.
  - ✓ Use python image
  - ✓ Install flask
  - ✓ Expose port 5000
  - ✓ Run python command on app.py
- Create a new network called ivolve-network.
- Run backend container using ivolve-network.
- Run frontend container (frontend1) using ivolve-network.
- Run another frontend container (frontend2) using default network.
- Verify the communication between containers.
- Delete ivolve-network.

Containerization with Docker

Lab 9: Containerized Node.js and MySQL Stack Using Docker Compose

- Clone the application source code from https://github.com/Ibrahim-Adel15/kubernets-app.git
- The application requires a MySQL connection and must find a database named ivolve to start working.
- Create docker-compose.yml file with:

  App service
  - Build from the local Dockerfile
  - Map port 3000
  - Use environment variables:
    - ✓ DB_HOST
    - ✓ DB_USER
    - ✓ DB_PASSWORD

  db service
  - Use MySQL image
  - Set environment variable MYSQL_ROOT_PASSWORD

  db_data volume for /var/lib/mysql
- Verify app is working.
- Verify app /health & /ready.
- Verify app access logs at /app/logs/.
- Push the Docker image into your DockerHub.

K8S for
Container Orchestration

iVolve

K8S for Container Orchestration

Lab 10: Node Isolation Using Taints in Kubernetes

- Run Kubernetes cluster with 2 nodes.
- Taint one node with a specific key-value '**node=worker**' and effect **NoSchedule**.
- Describe all nodes to verify the taint.

Lab 11: Namespace Management and Resource Quota Enforcement

- Create a namespace called **ivolve**.
- apply resource quota to limit pods number to only **2 pods** within the namespace.

Lab 12: Managing Configuration and Sensitive Data with ConfigMaps and Secrets

- Define a ConfigMap to store non-sensitive MySQL configuration variables:
  - ➢ **DB_HOST** – The hostname of the MySQL StatefulSet service
  - ➢ **DB_USER** – The database user that the application will use to connect to the ivolve database.
- Define a Secret to store sensitive MySQL credentials securely:
  - ➢ **DB_PASSWORD** – The password for the DB_USER.
  - ➢ **MYSQL_ROOT_PASSWORD** – The root password for MySQL database.
- Use base64 encoding for the Secret data values.

K8S for Container Orchestration

Lab 13: Persistent Storage Setup for Application Logging

- Define a Persistent Volume (PV) with the following specifications:
  - **Size:** 1Gi
  - **Storage type:** hostPath
  - **Path:** /mnt/app-logs on the node file system (/mnt/app-logs must be created on app node with 777 permission)
  - **Access mode:** ReadWriteMany (to allow all replicas read/write access)
  - **Reclaim policy:** Retain
- Define a Persistent Volume Claim (PVC) that requests 1Gi storage.
- Ensure the PVC access mode matches the PV (ReadWriteMany).

Lab 14: StatefulSet with Headless Service

- Create a StatefulSet with 1 replica running MySQL.
- Configure the StatefulSet pods to consume the root password from the secret.
- Add a toleration to the StatefulSet pod spec for the taint key node=worker with effect NoSchedule.
- Configure a Persistent Volume Claim (PVC) and mount it to /var/lib/mysql in the StatefulSet
- Write a YAML manifest for a headless service (clusterIP: None) targeting the MySQL StatefulSet pods.
- Confirm the database is operational by connecting with a MySQL client.

K8S for Container Orchestration

Lab 15: Node.js Application Deployment with ClusterIP service

- Create a Deployment named nodejs-app with 2 replicas (note 1 pod only will be running).
- Use your custom Docker image from Docker Hub.
- Configure pods to use environment variables from configmap/secret.
- Add a toleration to the pod spec with key node=worker with effect NoSchedule.
- Configure pod to use the static created PV.
- Create a ClusterIP service named nodejs-service to balance traffic across all deployment replicas.

Lab 16: Kubernetes Init Container for Pre-Deployment Database Setup

- Modify the existing Node.js Deployment to include an init container.
- Use a MySQL client image (e.g., mysql:5.7) for the init container.
- Pass necessary DB connection parameters using environment variables from configmap/secret.
- The init container should creates ‘ivolve’ database and a user with all access on ivolve database.
- Connect to MySQL manually to verify ivolve DB and user exist with the expected privileges.

K8S for Container Orchestration

Lab 17: Pod Resource Management with CPU and Memory Requests and Limits

- Update the existing Node.js Deployment to include resource requests and limits.
- Resource Requests:
  - ✓ cpu: 1 (1 vCPU)
  - ✓ memory: 1Gi
- Resource Limits:
  - ✓ cpu: 2 (2 vCPUs)
  - ✓ memory: 2Gi
- Use kubectl describe pod to verify the applied resource requests and limits.
- Use kubectl top pod to monitor real-time resource usage.

Lab 18: Control Pod-to-Pod Traffic via Network Policy

- Define a NetworkPolicy resource with the following specifications:
  - Name: allow-app-to-mysql
  - Pod Selector: Targets pods with label app=mysql.
  - Policy Types: Only Ingress
  - Ingress Rule:
    - ✓ Allows traffic only from the application pods .
    - ✓ Restrict access to port 3306 (MySQL default port).

K8S for Container Orchestration

Lab 19: Node-Wide Pod Management with DaemonSet

- Create **monitoring** namespace
- Deploy a **DaemonSet** for Prometheus node-exporter that tolerates all existing taints.
- Validate that a node-exporter pod is scheduled on each node.
- Confirm correct metrics exposure by accessing :9100/metrics on any node.

Lab 20: Securing Kubernetes with RBAC and Service Accounts

- Create jenkins-sa Service Account in the ivolve namespace.
- Create a secret and retrieve the service account token.
- Define a Role named pod-reader that grants read-only permissions (get, list) on Pods in the ivolve namespace.
- Create a RoleBinding to bind the pod-reader Role to the jenkins-sa ServiceAccount.
- Validate the role can only list pods.

Continues Integration / Continues Delivery

Continues Integration / Continues Delivery

Lab 21: Role-based Authorization

- Create user1 and user2.
- Assign admin role for user1 & read-only role for user2.

Lab 22: Jenkins Pipeline for Application Deployment

- Clone source code and Dockerfile from: https://github.com/Ibrahim-Adel15/Jenkins_App.git
- Create a pipeline that automates the following processes:
  1. Run Unit Test
  2. Build App
  3. Build Docker image from Dockerfile in GitHub.
  4. Push image to Docker hub.
  5. Delete image locally.
  6. Edit new image in deployment.yaml file.
  7. Deploy to k8s cluster.
- Set pipeline post action (always, success, failure)

Continues Integration / Continues Delivery

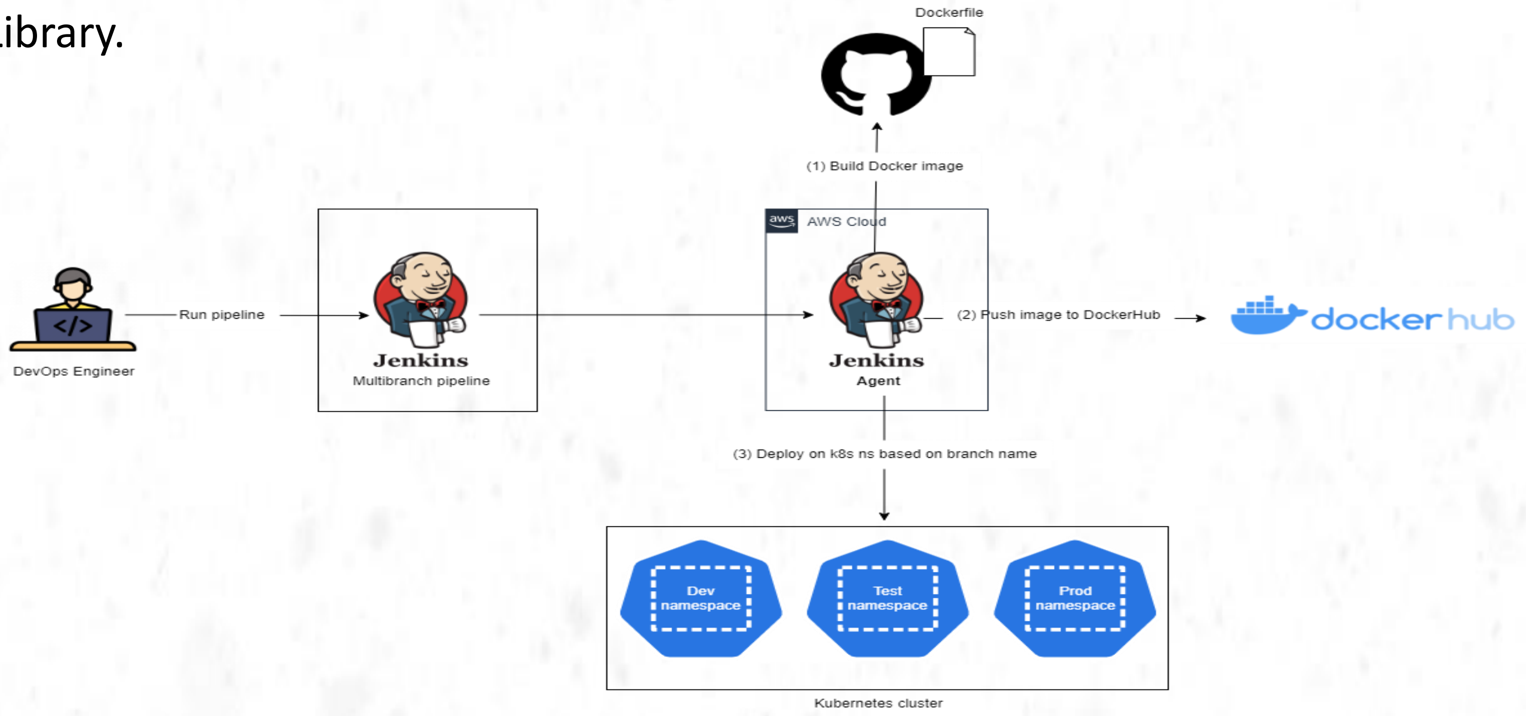
Lab 23: CI/CD Pipeline Implementation with Jenkins Agents and Shared Libraries

- Clone Dockerfile from: https://github.com/Ibrahim-Adel15/Jenkins_App.git
- Create a pipeline with the following stages:
    1. RunUnitTest
    2. BuildApp
    3. BuildImage
    4. ScanImage
    5. PushImage
    6. RemoveImageLocally
    7. DeployOnK8s
- Use shared library for these tasks and demonstrate its usage in different pipelines.
- Configure Jenkins slave to run the pipeline.

Continues Integration / Continues Delivery

Lab 24: Multi branch CI/CD Workflow

- Clone docker file from: https://github.com/Ibrahim-Adel15/Jenkins_App.git
- push the docker file to your repo and create 3 branches (prod/stag/dev).
- Create 3 namespaces in you K8S environment (prod/stag/dev).
- Create a Multibranch pipeline to automate the deployment in the namespace based on GitHub branch.
- Create Jenkins slave to run this pipeline.
- Use Shared Library.

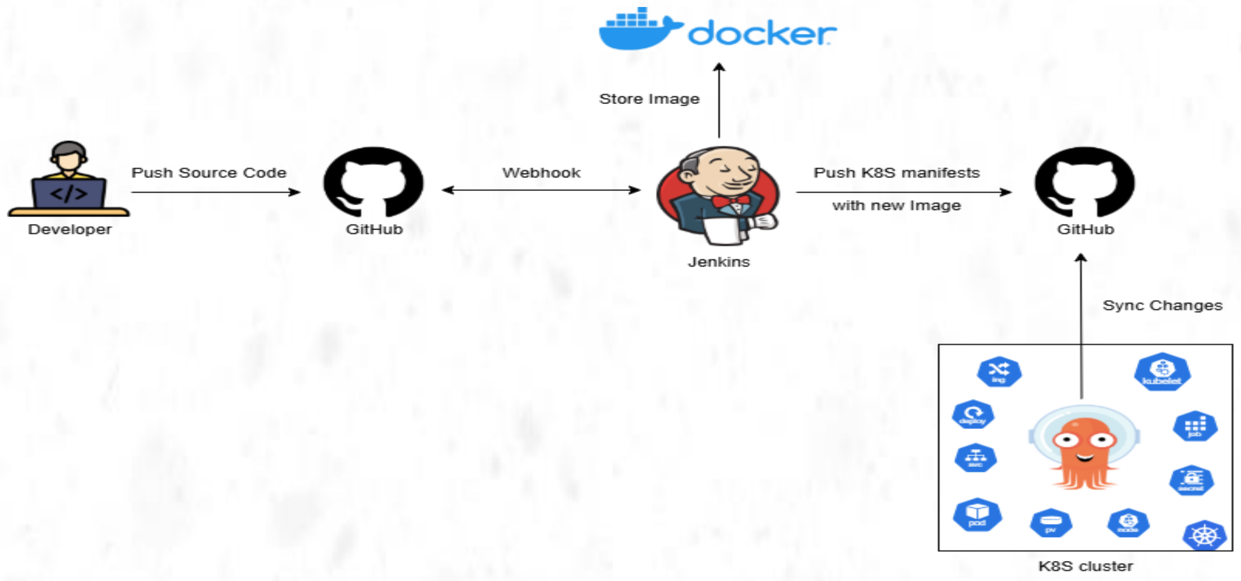


GitOps Workflow with ArgoCD

GitOps Workflow with ArgoCD

Lab 25: GitOps workflow

- Configure ArgoCD in your kubernetes cluster.
- Clone docker file from: https://github.com/Ibrahim-Adel15/Jenkins_App.git
- Create a pipeline that automates the following processes:
  1. Build App
  2. Build Docker image.
  3. Push image to Docker hub.
  4. Delete image locally.
  5. Edit new image in deployment.yaml file.
  6. Push new update files into Github repo.
- Validate ArgoCD deploy new app into the cluster.



Ansible Automation

iVolve

Red Hat Enterprise Linux Automation with Ansible (RH294)

Lab 26: Initial Ansible Configuration and Ad-Hoc Execution

- Install and configure Ansible Automation Platform on control node.
- Generate new key on control node.
- Transfer the public key into managed node using ssh-copy-id command.
- Create inventory of a managed node.
- Perform ad-hoc command (check disk space).

Lab 27: Automated Web Server Configuration Using Ansible Playbooks

- Write an Ansible playbook to automate the configuration of a web server
  - ✓ Install Nginx.
  - ✓ Customize the web page.
- Verify the configuration on managed node.

Lab 28: Structured Configuration Management with Ansible Roles

- Create Ansible roles for configuring Docker, Kubernetes CLI ‘kubectl’, and Jenkins.
- Write Ansible playbook to run the created roles.
- Verify the installation on managed node.

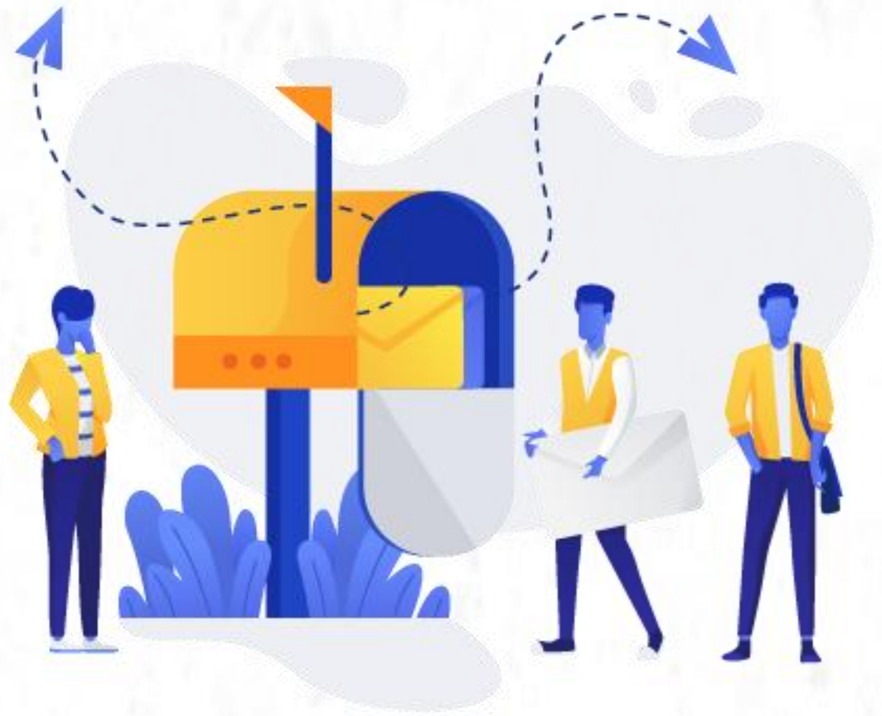
Red Hat Enterprise Linux Automation with Ansible (RH294)

Lab 29: Securing Sensitive Data with Ansible Vault

- Write Ansible playbook to automate the following tasks:
  - ✓ Install MySQL.
  - ✓ Create iVovle database.
  - ✓ Create user with all privileges on iVolve DB.
- Use Ansible Vault to encrypt sensitive information such as database user password.
- Validate DB on managed node by connect to database using the created user and listing databases.

Lab 30: Automated Host Discovery with Ansible Dynamic Inventory

- Create AWS EC2 with tag service:db.
- Set up Ansible dynamic inventory to automatically discover and manage running EC2.
- list the target hosts using ansible inventory command.
- Run MySQL role.



Y o u r   p a r t n e r   i n   t h e   c l o u d .

Have Any Questions?
Feel Free to Contact Us.

SAUDI ARABIA

18th Floor, Faisaliah Tower,
Olaya Street, Riyadh, 12212
Kingdom of Saudi Arabia

UAE

Dubai, Silicon Oasis, Techno Hub. 01,
Technology Entrepreneurship Center (DTEC)
P.O. Box: 6009, Dubai, UAE

Egypt

Westown Hub Building 3, Sodic, Beverly Hills, Cairo-Alexandria Desert Road, Unit C41

Pakistan

NGC Tower, 3rd Floor, Plot # A-1, Block-3 7/8 KECHS, Jinnah Housing Society, PECHS, Karachi, 75000

USA

123 E San Carlos St. PMB 4575 San Jose, CA 95112, USA

> sales@ivolve.io

> info@ivolve.io

Talk To Our Representatives

iVolve

THANK YOU!

Evolve your CLOUD&DIGITAL Transformation journey

with

iVolve