**15-Day Daily Work Report – Wanderlust MERN Cloud Project**

**Day 1 – Master Machine Setup**

* Launched AWS EC2 Master Node (t2.large, 8GB RAM, 29GB storage).
* Installed Docker and added user to Docker group.
* Installed Java 17 and Jenkins.
* Configured Security Group ports (22, 8080, Docker).
* Accessed Jenkins Master in browser and completed initial setup.

**Day 2 – AWS CLI & Kubernetes Tools**

* Installed and configured **AWS CLI** with IAM credentials.
* Installed **kubectl** on Master node.
* Installed **eksctl** for EKS cluster management.
* Verified AWS CLI, kubectl, and eksctl versions.

**Day 3 – EKS Cluster Creation**

* Created EKS Cluster wanderlust in **us-west-1** region.
* Configured IAM OIDC Provider for the cluster.
* Verified cluster and contexts with kubectl get nodes.

**Day 4 – Nodegroup Setup**

* Created Nodegroup (wanderlust) with 2 t2.large nodes.
* Generated SSH key pairs on Master node.
* Added Nodegroup public key to AWS for EC2 access.
* Verified nodegroup readiness in cluster.

**Day 5 – Jenkins Worker Setup**

* Launched **Jenkins Worker Node** (t2.large, 8GB RAM).
* Installed Java 17 and AWS CLI on Worker.
* Configured SSH access from Master → Worker.
* Added Worker node to Jenkins (permanent agent).
* Installed Docker on Worker and configured permissions.

**Day 6 – SonarQube & Trivy Setup**

* Installed **SonarQube** (Docker container on Master).
* Verified SonarQube UI running on port 9000.
* Installed **Trivy** on Worker for vulnerability scanning.
* Installed **OWASP Dependency Check plugin** in Jenkins.

**Day 7 – ArgoCD Setup**

* Created **ArgoCD namespace** in EKS.
* Installed ArgoCD manifests.
* Patched ArgoCD service to NodePort.
* Installed ArgoCD CLI on Master node.
* Accessed ArgoCD Web UI via Worker Node Public IP.

**Day 8 – Jenkins Plugins & Configurations**

* Installed required Jenkins plugins:
  + Pipeline
  + Docker
  + SonarQube Scanner
  + OWASP
* Configured Jenkins credentials:
  + GitHub Personal Access Token
  + DockerHub Login
  + SonarQube Token

**Day 9 – CI Pipeline Setup**

* Created **Wanderlust-CI pipeline** in Jenkins.
* Configured stages: Git Checkout → Build → SonarQube → Trivy → Docker Build & Push.
* Verified CI pipeline pushes images to DockerHub.

**Day 10 – CD Pipeline Setup**

* Created **Wanderlust-CD pipeline** in Jenkins.
* Configured ArgoCD integration with EKS cluster.
* Added GitHub repo to ArgoCD.
* Deployed Wanderlust app (frontend + backend) via ArgoCD.

**Day 11 – Application Verification**

* Verified application deployment on EKS cluster.
* Opened NodePorts (31000 for frontend, 31100 for backend).
* Accessed Wanderlust MERN app in browser.
* Validated Redis caching integration.

**Day 12 – Notifications Setup**

* Configured Gmail SMTP in Jenkins for email alerts.
* Generated Gmail App Password for Jenkins authentication.
* Set up Extended Email Notification plugin.
* Verified email notifications for pipeline success/failure.

**Day 13 – Monitoring with Helm**

* Installed **Helm** on Master node.
* Added Prometheus & Grafana Helm repos.
* Installed Prometheus + Grafana stack on EKS.
* Exposed Grafana via NodePort and logged in (admin/admin).
* Configured dashboards for Kubernetes monitoring.

**Day 14 – Testing & Reports**

* End-to-end testing of CI/CD pipeline (GitHub → Jenkins → DockerHub → ArgoCD → EKS).
* Verified SonarQube quality gates and Trivy scan reports.
* Checked OWASP dependency scan.
* Validated monitoring in Grafana.
* Ensured Jenkins email alerts working correctly.

**Day 15 – Cleanup & Documentation**

* Documented all steps and commands used.
* Performed cleanup of EKS cluster (eksctl delete cluster).
* Terminated unused AWS resources to avoid extra costs.
* Prepared final project report with architecture and pipeline screenshots.
* Shared learnings and challenges faced during project.

**Wanderlust Project (हिंदी में समझाना)**

👉 *"Wanderlust एक MERN (MongoDB, Express, React, Node.js) स्टैक पर बना हुआ ट्रैवल ब्लॉग एप्लीकेशन है। मैंने इसे क्लाउड और DevOps टूल्स का इस्तेमाल करके एंड-टू-एंड डिप्लॉय किया है।"*

**प्रोजेक्ट का उद्देश्य**

* ये प्रोजेक्ट असली इंडस्ट्री जैसी CI/CD पाइपलाइन और क्लाउड डिप्लॉयमेंट सीखने के लिए बनाया गया है।
* इसमें मैंने कोड को GitHub से लिया, Docker में कंटेनर बनाया, Jenkins से CI/CD पाइपलाइन तैयार की और AWS EKS (Kubernetes) पर Deploy किया।

**टेक्नोलॉजी और टूल्स**

* **GitHub** – Code repository
* **Docker** – एप्लिकेशन को कंटेनराइज़ करने के लिए
* **Jenkins** – CI/CD पाइपलाइन बनाने के लिए
* **SonarQube** – कोड क्वालिटी चेक करने के लिए
* **OWASP & Trivy** – सिक्योरिटी स्कैनिंग के लिए
* **ArgoCD** – Continuous Deployment और Kubernetes पर deploy के लिए
* **AWS EKS (Kubernetes)** – एप्लिकेशन रन करने के लिए
* **Prometheus & Grafana (Helm)** – मॉनिटरिंग और डैशबोर्ड के लिए
* **Redis** – कैशिंग के लिए

**काम करने का तरीका (Flow)**

1. **CI Pipeline** – Jenkins को GitHub से कनेक्ट किया → कोड pull हुआ → Docker image बनी → SonarQube + Trivy + OWASP से scan हुआ → फिर DockerHub पर push हुआ।
2. **CD Pipeline** – Jenkins ने ArgoCD को trigger किया → ArgoCD ने Kubernetes (EKS) क्लस्टर पर नया Docker image deploy कर दिया।
3. **Monitoring** – Prometheus और Grafana से cluster और application की हेल्थ मॉनिटर की।
4. **Notification** – Jenkins से Gmail SMTP सेटअप किया ताकि pipeline success/failure की ईमेल आए।

**इस प्रोजेक्ट से मैंने क्या सीखा**

* AWS पर EKS cluster create और manage करना।
* CI/CD pipeline बनाना और integrate करना।
* Docker, Jenkins, SonarQube, Trivy, ArgoCD जैसे DevOps टूल्स को connect करना।
* Kubernetes पर MERN stack application deploy करना।
* Monitoring और alerting setup करना (Prometheus + Grafana)।

**Interviewer:** *आपका प्रोजेक्ट Wanderlust explain कीजिए।*

🙋‍♂️ **आप (Answer):**  
"Wanderlust एक MERN स्टैक पर बना हुआ ट्रैवल ब्लॉग एप्लीकेशन है। मैंने इसको end-to-end DevOps tools और AWS cloud पर deploy किया है।

इस प्रोजेक्ट में GitHub से code लिया जाता है, Jenkins CI/CD pipeline चलाता है, और Docker image बनती है। इसके बाद SonarQube और Trivy से code और image की quality और security check होती है। Image को DockerHub पर push किया जाता है।

Deployment के लिए मैंने ArgoCD का इस्तेमाल किया, जो application को AWS EKS cluster (Kubernetes) पर deploy करता है। Application का frontend और backend NodePort के through browser में access किया जा सकता है।

Monitoring के लिए Prometheus और Grafana setup किया है, और Jenkins email notification भी configure किया ताकि pipeline की status mail से मिल सके।

इस प्रोजेक्ट से मैंने AWS EKS, Jenkins pipeline, Docker, SonarQube, ArgoCD और monitoring tools practically use करना सीखा।"