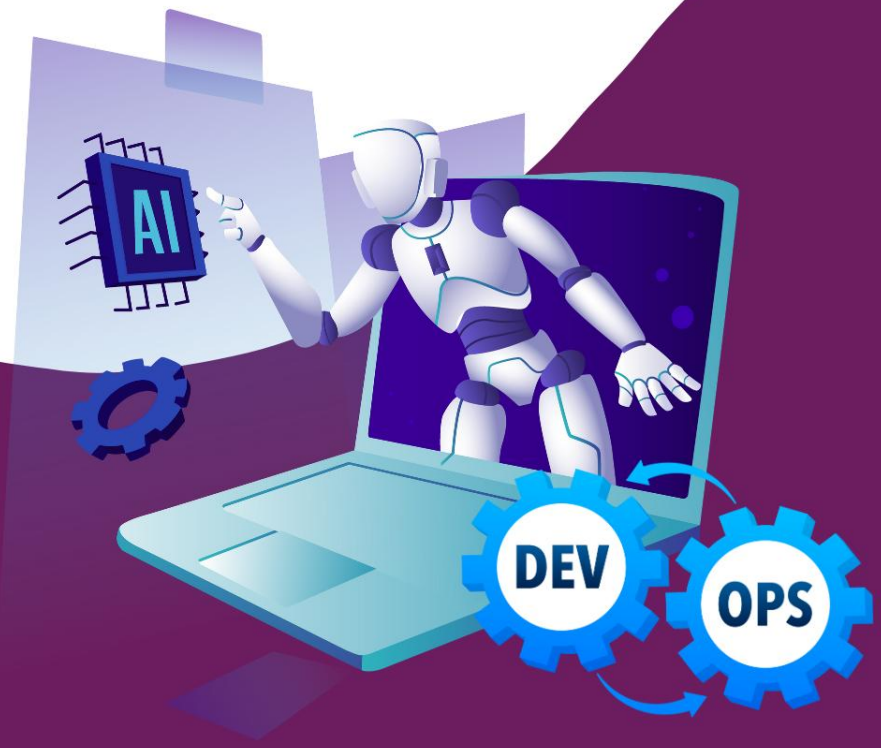
AI Driven DevOps



Xebia

AI Driven DevOps

Sno.	Subject	Lecture (L)	Practical (P)
1	DevOps Foundations	3	2
2	Source Code Management	3	2
3	Build & Release Management	3	2
4	Continuous Integration & Continuous Delivery	3	2
5	Aws - Cloud Engineering	3	2
6	Application Containerization	3	2
7	System Provisioning (On-Premise/On-Cloud)	3	2
8	Test Automation with Selenium	3	2
9	DevOps Automation With AI	3	2
10	AI-Driven Observability & Monitoring	3	2
11	AI-Powered Security & DevSecOps	3	2
12	AI for Platform Engineering	3	2

Total Credits: 60

Subject 1	Lecture: 3 Credits
Total Credits: 3	Lab: 2 Credits

DEVOPS OVERVIEW

Unit 1: Traditional SDLC Models, their problems, & their solutions overview

Software, History of Software Engineering and Software, Development Methodologies, Traditional Software Development Models, Waterfall Model, Classical Waterfall Model, Traditional IT Organizations, Developers vs IT Operations Conflict, Birth of Agile, Four Values of the Agile Manifesto, Agile and Lean

Unit 2: Introduction to DevOps

Definition of DevOps: Challenges of traditional IT systems & processes, History and emergence of DevOps, DevOps definition and principles governing DevOps, DevOps and Agile, The need for building a business use case for DevOps, Purpose of DevOps, Minimum Viable Product (MVP), Benefits of MVP, Application Deployment, Automated Application Deployment, Application Release Automation (ARA), Components of Application Release Automation (ARA), Continuous Integration, Best Practices of CI, Benefits of CI, Continuous Delivery, Process, CAMS – Culture, Cultural aspects of DevOps, CAMS – Automation, Delivering high value, CAMS – Measurement, Metrics used for tracking, CAMS – Sharing, Test-driven development.

Unit 3: Linux Basics and Admin

Introduction to Linux (Operating System), Importance of Linux in DevOps, Linux Basic Command Utilities, Linux Administration, Environment Variables, Networking, Linux Server Installation, RPM and YUM Installation

Unit 4: Introduction to DevOps Tools & Technologies

Git & GitHub (SCM), Docker (Containerization), Jenkins (CI/CD Pipelines), Terraform (Provisioning), Maven (Build & Release Management), Ansible (Configuration Management), Selenium (Test Automation), AWS (Cloud Computing), SonarQube (Code Quality Checking), Prometheus/Nagios (Monitoring), AI observability tools (Datadog AI, Dynatrace AI), Snyk AI

Unit 5: Introduction to AI in Modern Software Delivery

AI's role in DevOps today, AI is the natural evolution of DevOps, Predictive DevOps: anticipating failures before they happen, AI predicting deployment risks before pushing code to production, AI suggesting rollback automatically during anomalies, AI optimizing cloud costs based on usage patterns

DEVOPS OVERVIEW LAB

List of Objectives:

1. Connect to a Linux server and practice essential commands to navigate and manage the system.
2. Customize your shell environment with aliases and environment variables for productivity.
3. Initialize a local Git repository and track file changes using basic Git commands.
4. Collaborate on code by pushing to GitHub and managing pull requests.
5. Manually deploy a static website, then automate it using a simple shell script.
6. Write a shell script to automate software installation and basic system setup.
7. Containerize a basic app using Docker to ensure consistent environments.
8. Install Jenkins and create a job that runs a basic build script.
9. Use Maven to build a simple Java project and understand dependency management.
10. Run a static code analysis tool to identify quality issues in your codebase.
11. Use Git branches to simulate Agile-style feature development and merging.
12. Analyze a DevOps case study and map improvements using the CAMS model.

Subject 2	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

SOURCE CODE MANAGEMENT

Unit 1: Introduction to Source Code Management

Definition of Source Code Management: Understand the role of SCM in tracking and controlling changes to source code, Importance in DevOps: Explore how SCM is a fundamental aspect of the DevOps lifecycle, Key concepts: Introduce essential SCM concepts such as repositories, version control, branching, and merging, Centralized vs. Distributed SCM: Compare and contrast centralized and distributed SCM systems, Popular SCM tools: Explore widely-used SCM tools like Git, SVN, and Mercurial.

Unit 2: Git Basics

Git workflow: Learn the standard Git workflow and understand how changes move through the Git system, Basic commands: Cover fundamental Git commands (init, add, commit, status, log) for day-to-day operations, Branching and merging, Remote repositories, including cloning and setting up tracking branches, Fetching, pulling, pushing changes, Working with remote branches, Git Hooks: Implement pre-commit and post-commit hooks, customizing Git behavior, Branching Strategies, Feature branching: Explore the concept of feature branches and their role in collaborative development, GitFlow Workflow.

Unit 3: SCM in CI/CD

Integration with CI/CD Tools: Explore integration points between SCM and CI/CD tools like Jenkins, GitLab CI, and GitHub Actions, Automating builds and deployments: Learn to automate the build and deployment processes using SCM, Infrastructure as Code (IaC) and SCM: Understand the relationship between SCM and IaC tools for managing infrastructure.

Unit 4: Advanced SCM Strategies

Feature toggles and SCM: Implement feature toggles for controlled feature releases and explore their use in SCM, SCM for Microservices Architecture: Address challenges in versioning and organization in a microservices environment, SCM in DevSecOps: Explore integrating security practices into SCM, implementing secure coding practices and automated security checks, SCM for Continuous Improvement: Examine how SCM contributes to continuous improvement, implementing strategies for tracking and managing technical debt through SCM.

Unit 5: GitHub Copilot & AI Tools

AI-assisted coding, Benefits of using AI in repositories, Role of AI in code quality, collaboration, and security, GitHub Copilot, Types of code suggestions (completions, functions, documentation), Setting up GitHub Copilot in Visual Studio Code, Writing code with Copilot assistance, Accepting, modifying, or rejecting AI code suggestions, Auto-suggesting commit messages, Using GitHub's built-in CodeQL (AI-enhanced) to find vulnerabilities

SOURCE CODE MANAGEMENT LAB

List of Objectives:

1. Set up a new Git repository, configure project-specific settings, and push the initial codebase to a remote server.
2. Create isolated feature branches, collaborate through pull requests, and merge changes back into the main branch.
3. Quickly create hotfix branches to patch production issues and deploy urgent fixes without disrupting ongoing development.
4. Simulate conflicting code changes across branches and manually resolve merge conflicts during integration.
5. Configure Git pre-commit and post-commit hooks to enforce code quality and automate checks.
6. Trigger automated builds and deployments by integrating Git commits with Jenkins pipelines.
7. Version infrastructure code by managing Terraform files in a Git repository with collaborative workflows.
8. Implement feature toggles using Git to enable or disable features without deploying new code.
9. Organize Git branches to align with Agile sprints, tagging releases, and maintaining clean sprint delivery pipelines.
10. Conduct peer reviews, enforce approvals, and improve code quality using pull request workflows.
11. Simulate a distributed team environment by managing contributions, branching strategies, and asynchronous reviews.
12. Implement secure coding practices, automate security scans, and manage secrets through GitOps principles.
13. Use GitHub Copilot to generate boilerplate code and documentation, and compare it with manual efforts.

Subject 3	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

BUILD & RELEASE MANAGEMENT

Unit 1: Introduction to Build and Release Management

Overview, Definition and importance in DevOps, Key components: build, release, deployment, Infrastructure, Build Tools and Environments, Introduction to Build Tools, Overview of build tools: Maven, Gradle, Ant, Configuring build tools for projects, Build Environments, Local development environments, CI/CD build environments

Unit 2: Automated Builds

Continuous Integration (CI) Concepts, CI benefits and principles, Setting up CI pipelines, Building with Jenkins, Configuring Jenkins for automated builds, Integrating build tools with Jenkins, Introduction to Dependency Management, Managing project dependencies, Dependency management tools: Maven, npm, Dependency Caching and Versioning, Strategies for caching dependencies, Versioning best practices

Unit 3: Release Strategies

Release Management Overview, Definition and significance, Role in DevOps pipeline, Release Versioning and Tagging, Semantic versioning, Tagging and branching strategies, Configuration Management, Configuration as Code, Managing configuration in code, Infrastructure as Code (IaC), Configuration Management Tools, Ansible, Puppet, Chef, Integrating configuration management into the release process

Unit 4: Continuous Deployment

Introduction to Continuous Deployment, Principles and benefits, Automated deployment pipelines, Deployment Strategies, Blue-Green deployment, Canary releases, Environment Management, Environment Provisioning, Automated environment provisioning, Tools like Docker, Kubernetes, Managing Configuration Across Environments, Strategies for consistent configurations, Environment-specific configuration files

Unit 5: AI-Assisted Build Insights

Introduction to AI in Build and Release Pipelines, Harness platform introduction, Set up a Jenkins Pipeline that uses Build Failure Analyzer plugin, View predicted build failure causes from Jenkins, Use Harness trial (or sandbox) to build a small pipeline with AI-based release approval, Automatic green/red light based on risk analysis, Explore Harness to show build health scores

BUILD & RELEASE MANAGEMENT LAB

List of Objectives

1. Configure a local development environment and integrate a basic build tool like Maven to standardize project setups.
2. Set up a Jenkins pipeline to automate project builds and integrate it with a build tool for continuous integration.
3. Manage project dependencies using Maven, including handling caching and managing multiple versions efficiently.
4. Implement semantic versioning in a project and use tagging and branching strategies to manage release lifecycles.
5. Set up Ansible for configuration management and apply automated configuration changes across environments.
6. Build a continuous deployment pipeline and implement advanced deployment strategies like Blue-Green and Canary releases.
7. Create Docker containers for an application and integrate containerization into the deployment workflow.
8. Manage and automate environment-specific configuration files to maintain consistency across multiple deployment environments.
9. Implement rollback mechanisms in a deployment pipeline and practice safe rollbacks to previous stable releases.
10. Integrate automated testing into a deployment pipeline to ensure software quality and release reliability.
11. Implement monitoring and logging solutions in a production environment and practice incident response and troubleshooting.
12. Using Harness to build pipeline with AI based approvals.

Subject 4	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

Unit 1: Overview

Introduction to CI, Continuous Integration Workflow, Benefits of Continuous Integration, How CI Benefits Distributed Teams, Continuous Delivery, Steps Involved in CI/CD, Pipelines, Prerequisites, Checklist, Business Drivers for Continuous Deployment, Benefits of Continuous Deployment, CD – The HP Laserjet Case Study, Understanding the CI/CD pipeline and its benefits, Key principles of CI/CD, Popular CI/CD tools (Jenkins, GitLab CI/CD, CircleCI, etc.), CICD case studies

Unit 2: Stages of Continuous Integration and Continuous Delivery

Core CI Process, VCS, Merging Local Changes to Integration Branch, Fork & Pull, Code Review, Automated code builds – Key metrics, Git branching strategies (GitFlow, Trunk-Based Development), Continuous Integration with Git hooks and webhooks, Static Code Analysis, Snapshot, Sample Bug Report, Automated Unit Testing- JUNIT, Test Frameworks, Automated Unit Testing Process, Code formatting checks (e.g., Checkstyle, ESLint)

Unit 3: CICD with Jenkins

Initial Jenkins setup and user management, Jenkins global settings and Jenkins jobs, Jenkins pipelines, Jenkins pipeline scripts, Declarative vs. Scripted pipelines, Integrating Jenkins with version control systems (e.g., Git), Jenkins plugins and their usage, Integrating Jenkins with other tools (e.g., Docker, SonarQube), Jenkins pipeline library for reusable code, Integration of code quality tools in the CI/CD pipeline, Code Coverage analysis, Uploading build artifact to a repository, Advanced CI process, Automated Functional Testing, Publish Report to the Development Team

Unit 4: CICD with GitHub Actions

GitHub Actions and its benefits, Setting up GitHub Actions in a repository, GitHub Actions workflows, Building and testing applications with GitHub Actions, Artifacts and caching in GitHub Actions, Implementing code quality and security checks in Jenkins and GitHub Actions, Automated deployments with Jenkins, Using Jenkins and GitHub Actions for deployments to various environments, Managing secrets and environment variables in CI/CD, ArgoCD introduction, ArgoCD features, implementing CICD pipeline with Git and ArgoCD

Unit 5: Anatomy of a Continuous Delivery Pipeline

Simple Delivery Pipeline, Continuous Deployment Pipeline, Continuous Deployment with Feature Flags, Security testing in the CI/CD pipeline, Complying with security standards and regulations, Releasing an application to Production, Zero-Downtime Releases, Rolling back deployments, A/B testing and monitoring during deployments, Blue/Green and Canary deployments with Jenkins and GitHub Actions, Rolling updates and feature flags in CI/CD pipelines, Deploying and Promoting your Application, Modelling Your Release Process and Promoting Builds, Continuous Deployment to successive environments until before Production, Continuous monitoring for the delivery pipeline, Nagios sampler report, Continuous Feedback rules

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY LAB

List of Programs

1. Set up and configure Jenkins for continuous integration and delivery.
2. Install and configure Git, Java, and Maven on a Windows-based build server to support Jenkins builds.
3. Create Jenkins jobs with parameters, build steps, post-build actions, and automate them using pipelines.
4. Configure Jenkins agents/slaves on both Windows and Ubuntu hosts for distributed builds.
5. Integrate Jenkins with Git using the Git plugin for source control management.
6. Create and manage a new Jenkins pipeline to automate build and deployment workflows.
7. Merge local code changes into a Git repository and manage version control operations through Jenkins.
8. Install and configure Nexus Repository Manager to manage build artifacts.
9. Configure Jenkins as a Continuous Integration server to automate build, test, and integration tasks.
10. Deploy applications automatically to staging and production environments using Jenkins pipelines.
11. Merge feature branch code into the main application repository to manage new version releases.
12. Upload and manage Jenkins plugins manually to extend Jenkins capabilities.
13. Set up backup and recovery strategies for Jenkins server configurations and data.
14. Use the Docker plugin in Jenkins to build, test, and deploy containerized applications.
15. Integrate Kubernetes with Jenkins using the Kubernetes plugin to orchestrate container deployments.

Subject 5	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

AWS - CLOUD ENGINEERING

Unit 1: Overview

Origin & History of Cloud Computing, Cloud & Cloud Engineering, Evolution, Characteristics of Cloud, Cloud Computing Elements, Types of Cloud Computing, Trends in Cloud Computing , Cloud Service Providers, Traditional IT Providers, Cloud & DevOps, Benefits of Cloud Computing, Properties, Disadvantages, Amazon Web Services(AWS)

Unit 2: Cloud Computing Architecture

Traditional Vs Cloud Computing Architecture, Services(Level Based), How Cloud Computing Works, Networking in Cloud, We Services, Service Models(XaaS), Deployment Models, IaaS (Infrastructure as a Service) & its Resource Virtualization examples, PaaS (Platform as a Service), SaaS (Software as a service), STaaS (Storage as a Service), difference between IaaS, PaaS, & SaaS, Cloud Platform & Management, Service Management in Cloud Computing, Service Level Agreements (SLA's) , Scaling support in cloud / elastic nature, setting budget & notifications for budgets, Global Infrastructure of any cloud in general (Regions, Availability Zones, Data Centres)

Unit 3: AWS Basic Architecture

Elastic Compute Cloud (EC2), Elastic Block Storage (EBS), Elastic Load Balancing (ELB), Amazon Cloud Front, Security Groups, Security Management, Elastic Cache, Amazon RDS, Storage & Backups, Scaling, Amazon Virtual Private Cloud (VPC), Features of VPC, Subnets, Cloud Front, Content Delivery Network (CDN using Cloud Front), Features, Regions & zones, VM Instances on Cloud (EC2), IAM, ELB, Cloud Watch, Beanstalk, RDS, route 53, S3, Auto Scaling Groups

Unit 4: Cloud Security

Security Concepts, Security Planning, Boundaries, Cloud Security essentials, Cloud Security Alliance, Data Security, Encryption, Isolated Access to Data, Introduction to Key Management Service (KMS) of AWS

Unit 5: Role of Cloud in DevOps with AI

The Role of Cloud in DevOps, Cloud for successful Ops, Secure Cloud Platforms for DevOps, Cloud & IT Budgets, building a business case for cloud computing, Infrastructure as Code (IAC), AI-based cloud cost forecasting (AWS Cost Explorer with machine learning models), Enable and review AWS Cost Forecasting with Machine Learning.

AWS - CLOUD ENGINEERING LAB

List of Objectives

1. Set up an AWS account to establish a secure and scalable foundation for deploying cloud resources.
2. Create IAM users, organize them into groups, and assign policies to enforce fine-grained access control in AWS.
3. Launch EC2 instances with or without additional EBS volumes to manage compute resources and storage flexibility.
4. Create and modify security groups to control inbound and outbound traffic for cloud applications.
5. Provision encrypted EBS volumes to ensure secure data storage for sensitive information.
6. Set up a Classic Load Balancer (CLB) and attach instances to distribute incoming application traffic.
7. Create an Application Load Balancer (ALB) and configure target groups for advanced traffic routing.
8. Design and deploy a custom Virtual Private Cloud (VPC) to isolate and secure cloud infrastructure.
9. Create an S3 bucket, upload and download files to manage scalable object storage for applications.
10. Provision RDS instances with various database engines to store and manage relational data in the cloud.
11. Set up Elastic Cache instances to optimize application performance through caching strategies.
12. Create a CloudFront distribution to deliver content securely and quickly to users worldwide.

Subject 6	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

APPLICATION CONTAINERIZATION

Unit 1: Application Containerization

Understanding Containers: Transporting Goods Analogy, Problems in Shipping Industry before Containers, Shipping Industry Challenges, Container: The Savior, Solution by Containers in the Shipping Industry, How software function, and are deployed? Challenges in the Software Industry, Problems in Software Industry Before Containers, Put that in Container! Solution by containers in the Software Industry. Benefits and use cases.

Unit 2: Virtualization

Introduction, Hypervisor, Scope of Virtualization, Virtual Machines and its configurations, Applications of Virtualization, Choosing VM configuration for different use cases, Understanding Containers, Benefits and use cases of containerization, Containerization Platform, Runtime and Images, Container Platform, Container Runtime, Containers vs Virtual Machines, The Chroot System, FreeBSD Jails, Linux Containers (LXC), Docker

Unit 3: Introduction to Containerization

Docker architecture, Docker Daemon (Container Platform), Docker Rest API, Understanding Different environments: (Dev, QA and Prod). Installing Docker and basic docker commands, Dockerfile fundamentals, Docker Networking models, Exposing ports and container communication, Managing data persistence, Docker compose, writing compose files for multi container applications. Advance docker components, container security with docker, Best practices

Unit 4: Orchestration Tools

What is orchestration? Need of orchestration, Case study: Need of Orchestration, Need of Orchestration: Container and Microservices, Docker Swarm and Kubernetes, Kubernetes architecture and components, Working with Kubernetes pods, Rolling updates and rollbacks. Kubernetes service types, network policies, secure communication, Persistent storage in Kubernetes, Kubernetes configmaps and secrets, advance Kubernetes features

Unit 5: Application Deployment with AI

End to End Application deployment with containers, monitoring containerized applications, logging and log management. Using tools (Prometheus/Grafana) for container monitoring, Integrating containers into the CI/CD pipeline, AWS (ECS,EKS), AWS Elastic Container Services Architecture, Azure Kubernetes Services, OpenShift, KUBERNETES ON CLOUD

APPLICATION CONTAINERIZATION LAB

List of Objectives

1. Use Vagrant to quickly create and manage lightweight, reproducible virtualized development environments.
2. Understand and configure the Vagrantfile to automate the setup of isolated sandbox environments.
3. Install and configure Docker Machine to provision and manage Docker hosts on local and remote systems.
4. Work with Docker images and run Docker containers to package and deploy applications efficiently.
5. Create Dockerfiles to build custom containerized applications for consistent development and deployment.
6. Explore advanced Docker features like port binding, volumes, linking, and basic container monitoring for better container management.
7. Publish Docker images to Docker Hub and manage private repositories for sharing and deploying containers.
8. Use Docker Compose to define and manage multi-container applications with simple YAML configurations.
9. Set up a Docker Swarm cluster by spinning up three virtual machines and organizing them into one manager and two worker nodes.
10. Work with Kubernetes using Minikube to simulate a production-like container orchestration environment locally.
11. Deploy and manage Kubernetes pods and services on Minikube to orchestrate containerized applications.

Subject 7	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

SYSTEM PROVISIONING (ON-PREMISE/ON-CLOUD)

Unit 1: Introduction to System Provisioning and Configuration Management

Overview, Definition and importance of DevOps, Key concepts: provisioning, configuration management, Infrastructure as Code (IaC), Evolution of System Provisioning, Traditional vs. modern approaches, Role of automation in provisioning, Introduction to Provisioning Tools, Ansible, Chef, Puppet, Terraform, Use cases and comparison, Hands-on with Terraform, Basic Terraform commands, Provisioning infrastructure on a cloud provider.

Unit 2: Configuration Management Concepts & Tools

Understanding Configuration Management, Desired State Configuration (DSC), Idempotence and declarative vs. imperative models, Introduction to Configuration Management Tools, Ansible, Chef, Puppet, SaltStack, Use cases and comparison, IaC Principles, Codifying infrastructure, Benefits and challenges, IaC Tools, Terraform, AWS CloudFormation, Creating and managing infrastructure as code

Unit 3: Advanced Configuration Management

Managing Environment Configuration, Dynamic Inventories, Generating inventories dynamically with Ansible, Managing large-scale infrastructure, Configuration Management Best Practices, Designing modular and reusable configurations, Error handling and troubleshooting, Security and Compliance as Code, Security Best Practices, Securing provisioning and configuration processes, Compliance as code, Securing Secrets, Managing sensitive information securely, Integration with secret management tools

Unit 4: Orchestration and Automation

Orchestration vs. Automation, Understanding the difference, Use cases for orchestration, Workflow Automation, Integrating provisioning and configuration processes, Tools for workflow automation, Orchestration with Ansible Tower, Set up and configure Ansible Tower, Create and execute orchestrated workflows

Unit 5:

Unit 5: AI Tools for SPCM

AI-enhanced Infrastructure as Code – using AI to predict optimal provisioning parameters, Terraform with AI Plugins or Ansible AI extensions

SYSTEM PROVISIONING (ON-PREMISE/ON-CLOUD)

List of Objectives

1. Set up Terraform and provision a simple cloud infrastructure to automate resource creation.
2. Write and execute Ansible playbooks to automate the configuration of multiple servers.
3. Implement dynamic inventories in Ansible and apply best practices for scalable configuration management.
4. Securely manage secrets and enforce security best practices using Ansible Vault during automation.
5. Create and manage orchestrated Ansible workflows without using Ansible Tower, focusing only on open-source Ansible features.
6. Deploy an application across multiple cloud providers using Terraform and Ansible to handle multi-cloud infrastructure.
7. Implement continuous configuration updates using Ansible integrated with version control and CI pipelines.
8. Design rollback strategies and version infrastructure changes in a continuous configuration management setup.
9. Perform compliance checks automatically by embedding security rules and auditing policies into Ansible playbooks.
10. Integrate compliance validation into the configuration pipeline to meet auditing and regulatory requirements.
11. Scale automation processes with Ansible and Terraform to handle infrastructure growth efficiently.
12. Address performance and efficiency challenges in large-scale provisioning and configuration management environments.

Subject 8	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

TEST AUTOMATION

Unit 1: Foundations Of Test Automation

Testing Foundations, Types of Testing (Unit Testing, Integration Testing, System Testing, Acceptance Testing), Benefits and Challenges: Advantages of Automation over Manual Testing, common challenges and how to overcome them, Overview of Popular Tools (Selenium, Appium, Cypress, etc.) Fundamentals of test automation – benefits and limitations, comparison of software testing methods – manual and automation testing, when and how to use automation testing, automation types – unit testing, API testing, GUI testing, regression testing, tool evaluation / POC, automation framework, tools and process, automation analysis, automation test generation. Setting Up a Test Automation Project- Key Considerations, Building a Test Automation Strategy

Unit 2: Core Concepts And Tools In Test Automation

Scripting Languages for Test Automation: Python, Java, JavaScript: Pros and Cons, Selecting the Right Language for Your Needs, Introduction to Selenium WebDriver-Setup, Configuration and basic operations and commands, API Testing Tools-Postman and REST-Assured, Writing and Running API Tests, Mobile Test Automation- Introduction to Appium, Writing Tests for Mobile Applications, Advanced Test Automation Techniques: BDD, DDT, KDT, HDD. Continuous Integration and Continuous Delivery (CI/CD)-Integrating Test Automation in CI/CD Pipelines, Tools like Jenkins and GitLab CI

Unit 3: Mastering Selenium 4.X

Describe Selenium 4.x advantages and implementation, understanding selenium 4.x grid architecture, selenium commands, actions, asserts, assessors, developing test cases and test suites, WebDriver API, cross browser testing – Firefox, IE, Chrome, Edge, UI elements locator, identifying webElements, XPath, CssSelector, TagName, handling web Elements using webDriver, handling dynamic events, validation commands, synchronization commands.

Unit 4: Framework Implementation – Selenium

Automation frameworks, introduction to selenium and TestNg, selenium components, selenium architecture, TestNg, features, comparison with Junit, configuring TestNg with Eclipse, writing selenium scripts from scratch, test runs and reports, TestNG annotations, creating and running test suits, Hard Assertion, Soft Assertion, prioritizing of execution for test cases, TestNG Reports, configuring ANT, XSLT report generation using TestNg and ANT.

Unit 5: Best Practices And Design Patterns

Code Quality and Maintenance-Writing Maintainable Test Code, Code Reviews and Refactoring Techniques, Design Patterns in Test Automation: Page Object Model (POM), Screenplay Pattern, Test Reporting and Metrics-Effective Reporting Strategies, Key Metrics for Test Automation, AI-generated Test Cases, explore research/tools, Diffblue Cover for Java, Use AI (e.g., Diffblue Cover) to generate unit tests and validate them against manual tests.

TEST AUTOMATION LAB

List of Objectives

1. Design and document test scenarios to validate the core functionalities of an e-commerce web application.
2. Build an automation script to automate user registration and login processes on an e-commerce platform.
3. Write the registration page automation script using Java to enhance cross-language automation skills.
4. Integrate Extent Reports into automation scripts to generate detailed test execution reports.
5. Build an automation script to search for articles on the e-commerce platform and validate search functionality.
6. Build an automation script to automate interactions with the article details page on the e-commerce platform.
7. Develop an automation script to search for an article and validate the accuracy of search results.
8. Automate navigation to the article details page and validate that all displayed information is accurate.
9. Implement parameterized test data into automation scripts to support flexible and scalable testing.
10. Create a data-driven automation script for end-to-end testing on the e-commerce platform.
11. Refactor and modularize test code for the login, search page, and article details page to align with best practices in framework design.
12. Execute the developed automation scripts across Chrome and Firefox browsers for cross-browser validation.
13. Integrate automation scripts with Jenkins to automate test execution during build deployments.

Subject 9	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

DEVOPS AUTOMATION & AI

Unit 1: Introduction to Automation

The Software Delivery Pipeline, Overview of the Continuous Delivery Pipeline, Fully Automated Software Delivery Process, The Build Process, Automated build, Automated Test, Automated Deployment, Benefits of Automated Deployment, Automated Deployment and DevOps Adoption, Automated Deployment and DevOps Adoption, Overview of Rapid Application Development (RAD), Phases in RAD, Essential Aspects of RAD, Code generation, Categories of Code Generators, Common

Unit 2: Advantages of Automation

Advantages of Automation, Automation Scenarios, Archiving Logs, Auto-Discard Old Archives, MySQL (RDBMS) Backups, Email Web Server Summary, Ensure Web Server is Running, User Command Validation, Disk Usage Alarm, Sending Files to Recycle Bin, Restoring Files from Recycle Bin, Logging Delete Actions, File Formatter, Decrypting Files, Bulk File Downloader, System Information, Install LAMP Stack, Get NIC's IP, Scenarios Where Automation Prevents Errors

Unit 3: Infrastructure Automation with AI

Infrastructure as Code (IaC) with AI: Terraform, Ansible AI, AI-powered configuration management: Chef Automate, Puppet Bolt, Self-healing Infrastructure: AI-based auto-scaling & self-recovery, AI-driven Cloud Cost Optimization & Auto-Healing Workloads, Ansible AI/Autopilot for infrastructure config

Unit 4: Intelligent Monitoring & AIOps

AI for log analysis, observability, and performance monitoring, AI-powered incident detection & root cause analysis, AIOps Tools: Datadog AI, Dynatrace AI, ELK Stack with AI-based log insights, AI for Automated Alerts & Remediation, AI-based predictive maintenance tasks for infra health

Unit 5: Future of AI in DevOps & Security

AI-powered DevSecOps & Threat Detection, Automated Compliance & AI-based security scans, AI in DevOps Trends: Generative AI for DevOps automation, Case Study: AI-powered incident response & anomaly detection

DEVOPS AUTOMATION & AI LAB

List of Objectives

1. Automate the code build, testing, and deployment process to streamline software delivery workflows.
2. Install and configure Terraform to enable infrastructure as code for cloud resource management.
3. Write a simple Terraform script to automatically deploy a virtual machine on AWS or Azure.
4. Implement AI-assisted code reviews using GitHub Copilot and Codacy to enhance code quality and reduce review time.
5. Build a CI/CD pipeline with AI-driven failure prediction using Harness.io to proactively address deployment risks.
6. Deploy a self-healing infrastructure using Terraform combined with AI-driven auto-scaling strategies.
7. Utilize AI-driven log analysis with Datadog AI to detect incidents and anomalies automatically.
8. Automate security compliance checks using AI-powered tools like Snyk and AWS GuardDuty AI.
9. Develop intelligent monitoring dashboards using Prometheus integrated with AIOps tools for proactive system visibility.
10. Experiment with AI-driven auto-remediation solutions like PagerDuty AI and Opsgenie AI to automate incident response.
11. Install and configure Snyk AI or GitHub Advanced Security to automate vulnerability scanning in codebases.
12. Create a self-healing security system by integrating multiple AI tools for detection, alerting, and automated mitigation.

Subject 10	Lecture: 3 Credits
Total Credits: 4	Lab: 2 Credits

AI-DRIVEN OBSERVABILITY & MONITORING

Unit 1: Introduction to AI-Driven Observability

Difference between Monitoring vs Observability, AI-powered Observability: How AI enhances traditional monitoring, Key Observability Components: Logs, Metrics, Traces, AI & Machine Learning in Observability: Self-healing & anomaly detection

Unit 2: AI-Powered Metrics & Performance Monitoring

AI-driven real-time metrics collection & processing, AI-powered performance monitoring & anomaly detection, Predictive analytics for proactive issue resolution, Tools: Datadog AI, Dynatrace, New Relic AI, AWS DevOps Guru

Unit 3: AI-Enhanced Log Monitoring & Analysis

Traditional vs AI-driven log analysis, AI-powered log ingestion & intelligent correlation, AIOps for automated log insights & anomaly detection, Tools: Elastic Stack (ELK) with AI, Splunk AI, Google Cloud Operations Suite

Unit 4: AI for Distributed Tracing & Incident Management

AI-powered distributed tracing for microservices & cloud-native apps, AI-based root cause analysis & automated troubleshooting, AI-powered incident prediction & auto-remediation, Tools: OpenTelemetry AI, Honeycomb, Lightstep

Unit 5: Intelligent Dashboards, Alerts & Automated Remediation

AI-powered alerting systems to reduce noise & false positives, Automated remediation workflows using AI, Case Study: AIOps implementation in a large-scale DevOps setup, Tools: PagerDuty AI, Opsgenie AI, Moogsoft

Unit 6: Monitoring using Nagios

What is Nagios, Nagios Installation, Configuring Nagios, Infrastructure monitoring using Nagios, Managing the Nagios server, Nagios Notifications, Nagios Core Backups, Nagios Managing Time Options, Nagios Event Handlers, Introduction to AI-assisted anomaly detection in logs before moving into AIOps

AI-DRIVEN OBSERVABILITY & MONITORING LAB

List of Objectives:

1. Set up AI-driven observability tools to monitor and improve application and infrastructure visibility.
2. Install and configure Datadog or Dynatrace agents to collect real-time metrics and telemetry data.
3. Enable AI-powered monitoring features to automatically detect performance anomalies and patterns.
4. Deploy and configure the ELK stack to centralize and visualize application logs.
5. Integrate application logs into the ELK stack for better traceability and real-time troubleshooting.
6. Implement AI-based log anomaly detection within the ELK stack to proactively identify issues.
7. Deploy OpenTelemetry to enable distributed tracing across microservices and application components.
8. Integrate OpenTelemetry with application services to collect trace data for performance insights.
9. Enable AI-driven root cause analysis using OpenTelemetry traces to reduce time to resolution.
10. Set up Moogsoft AIOps to automatically correlate alerts and reduce noise through AI-based alerting.
11. Automate anomaly detection and incident response using PagerDuty AI or Opsgenie AI integrations.
12. Build end-to-end automated remediation workflows leveraging AIOps tools to resolve incidents faster.

Subject 11	Lecture: 3 Credits
Total Credits: 3	Lab: 2 Credits

AI-POWERED SECURITY & DEVSECOPS

Unit 1: Introduction to AI-Powered Security & DevSecOps

Shift-left security approach with AI integration, AI-powered security automation in DevOps workflows, Introduction to AI-driven security tools for vulnerability scanning

Unit 2: AI-Powered Threat Detection & Prevention

AI in Cybersecurity: Attack surface analysis & anomaly detection, AI-driven malware detection & endpoint security, Tools: CrowdStrike AI, Darktrace, SentinelOne AI, Case Study: AI-driven zero-day attack prevention

Unit 3: AI in Application Security & Code Scanning

AI-based code security scanning & vulnerability detection, AI-enhanced secure coding best practices, Tools: GitHub Advanced Security AI, Snyk AI, Checkmarx AI, Case Study: AI-powered secure CI/CD pipeline

Unit 4: AI for Automated Compliance & Risk Management

AI-driven compliance enforcement in DevOps, Automated policy enforcement & risk analysis, Tools: AWS Security Hub AI, Google Chronicle, Microsoft Defender AI

Unit 5: AI-Driven Incident Response & Self-Healing Security

AI-powered real-time security analytics & auto-remediation, Self-healing security infrastructure using AI, AI-based threat intelligence & predictive security analytics, Tools: IBM Watson for Security, Splunk AI, SIEM with AI

AI POWERED SECURITY & DEVSECOPS LAB

List of Objectives:

1. Set up AI-powered security tools to enhance threat detection and vulnerability management.
2. Install and configure Snyk AI or GitHub Advanced Security to automate codebase scanning.
3. Run automated AI-based vulnerability scans and generate detailed remediation reports.
4. Deploy CrowdStrike AI or Darktrace to detect advanced threats using machine learning.
5. Configure real-time AI-driven threat alerts and dashboards to improve incident awareness.
6. Integrate and use Splunk AI or Google Chronicle for centralized security event analysis.
7. Build interactive dashboards to visualize and track AI-detected security incidents.
8. Apply AI-powered event correlation to detect complex attack chains or patterns.
9. Use AWS Security Hub AI to monitor compliance across cloud environments.
10. Automate compliance checks and remediation using policy-as-code and AI tools.
11. Deploy Microsoft Defender AI to enable adaptive threat response across cloud and endpoint.
12. Create a self-healing security system by integrating multiple AI tools for detection, alerting, and automatic mitigation.

Subject 12	Lecture: 3 Credits
Total Credits: 3	Lab: 2 Credits

AI FOR PLATFORM ENGINEERING

Unit 1: Introduction to Platform Engineering

Platform engineering vs traditional DevOps, Core concepts: developer self-service, golden paths, IDPs, Tools landscape: Backstage, Crossplane, Humanitec, Internal APIs

Unit 2: AI in Developer Experience

AI-assisted developer portals (GitHub Copilot in IDPs), Natural Language Interfaces for provisioning (Azure Dev CLI with AI), Auto-documentation and code snippet generation

Unit 3: AI-Powered Automation & Provisioning

Infrastructure as Code templates with AI-generated recommendations, Auto-tuning infrastructure using ML (e.g., K8s autoscaling), Predictive provisioning based on developer/team behavior

Unit 4: Observability and Intelligent Feedback

Integrating observability into IDPs (Grafana, Prometheus, Datadog), AI for log correlation and developer-facing error guidance, Feedback loops: AI-based build/test failure prediction and resolution

Unit 5: Governance, Compliance & Self-Healing Platforms

Policy-as-code with AI-based enforcement and risk scoring, Role-based access with intelligent provisioning limits, Self-healing workflows using AI for remediation (Moogsoft, PagerDuty AI)

AI FOR PLATFORM ENGINEERING LAB

List of Objectives:

1. Deploy a microservice using a Backstage template with GitHub Copilot for code scaffolding.
2. Auto-generate a CI/CD pipeline using OpenAI based on project metadata.
3. Provision cloud infrastructure with AI-suggested Terraform configurations.
4. Implement predictive auto-scaling in Kubernetes using Prometheus and ML models.
5. Create a golden path template to onboard new services with AI-generated YAML and Dockerfiles.
6. Scan and rotate secrets automatically using GitHub Actions and AI-based secret detection.
7. Generate Grafana dashboards from OpenTelemetry data using LLM prompts.
8. Detect log anomalies and failure patterns using AI in ELK or Datadog.
9. Enforce compliance by scanning Kubernetes manifests with OPA and an AI assistant.
10. Optimize cloud costs using AI-based rightsizing recommendations with Infracost.
11. Auto-generate service documentation using AI from code comments and Git metadata.
12. Simulate an incident and trigger automated AI-driven remediation via Opsgenie or PagerDuty.