Cloud-native testing focuses on validating applications and services built specifically for cloud environments. This testing methodology is designed to take full advantage of cloud-native architectures, such as microservices, containers, and serverless platforms. Below are some key points and mindset principles to understand about cloud-native testing:

Duration: 40 hours

Day 1: Introduction to Cloud-Native Testing, Functional & Load Testing in AWS and Azure

1.1 Introduction to Cloud-Native Testing and Testing Types

- **Overview of Cloud-Native Testing**
  - Introduction to cloud-native testing principles.
  - Key differences from traditional testing.
  - Benefits: cost efficiency, scalability, and flexibility in cloud environments.
- **Types of Cloud-Native Testing**
  - Functional, load, performance, and database testing.
  - Key use cases in cloud environments: purpose, benefits, and unique opportunities.

1.2 Functional Testing on AWS and Azure

- **Use Case**: Testing the functionality of APIs, web apps, and mobile apps.
- **AWS Functional Testing Tools**
  - **AWS Device Farm**: Functional testing for mobile and web applications.
  - **Hands-On**: Setting up Device Farm and running a sample functional test on a mobile app.
- **Azure Functional Testing Tools**
  - **Azure App Center**: Functional testing for mobile and web apps.
  - **Hands-On**: Setting up App Center and running a sample functional test on a mobile app.
- **Open-Source Functional Testing Alternative**
  - **Selenium** (for web apps) and **Appium** (for mobile apps).
  - **Hands-On**: Running a basic functional test on a sample web application using Selenium.

1.3 Load Testing on AWS and Azure

- **Load Testing Use Case**: Assessing application performance under different user loads.
- **AWS Load Testing Tools**
    - **AWS CloudWatch** and **AWS X-Ray**: Monitoring, logging, and tracing requests to support load testing.
    - **Hands-On**: Setting up a load test using CloudWatch metrics and tracing requests with X-Ray.
- **Azure Load Testing Tools**
    - **Azure Monitor** and **Application Insights**: Telemetry for load testing and monitoring.
    - **Hands-On**: Configuring load test monitoring with Azure Monitor and analyzing the collected metrics.
- **Open-Source Load Testing Alternatives**
    - **Apache JMeter** and **Locust**: Distributed load testing tools.
    - **Hands-On**: Setting up and running a load test for a sample application using JMeter.

1.4 Practical Lab: Setting Up Functional and Load Testing Environments

- **Lab 1**: Configuring AWS Device Farm for functional testing of a mobile app.
- **Lab 2**: Setting up Azure App Center for functional testing of a mobile app.
- **Lab 3**: Configuring a load test on AWS with CloudWatch and X-Ray.
- **Lab 4**: Running a load test on Azure using Azure Monitor and Application Insights.

1.5 Recap and Q&A

- Review of tools, concepts, and hands-on exercises covered on Day 1.
- Q&A session on functional and load testing best practices, challenges, and real-world application.

Day 2: Database Performance / Serverless Testing, Comparison of AWS and Azure Tools, and Best Practices

Introduction to Serverless Architectures on AWS and Azure

- Overview of serverless computing concepts.
- Key components of serverless architectures in AWS and Azure:
    - AWS Lambda, API Gateway, DynamoDB, and S3.
    - Azure Functions, Logic Apps, and Cosmos DB.

Performance Testing Concepts in Serverless Environments

- Unique challenges in testing serverless applications (cold starts, resource limits, etc.).
- Metrics to measure: execution duration, concurrency, memory usage, and throughput.
- Tools and strategies for serverless performance testing.

Hands-On: Load Testing AWS Lambda with AWS Tools

- Setting up performance testing for AWS Lambda using AWS Step Functions and AWS CloudWatch.
- Using AWS CloudWatch for monitoring Lambda invocations, latency, and error rates.
- Hands-On: Creating a sample AWS Lambda function, configuring load tests, and analyzing performance metrics.

Hands-On: Load Testing Azure Functions with Azure Monitor and Application Insights

- Setting up a test for Azure Functions to monitor response times, resource usage, and invocations.
- Configuring Application Insights for deep-dive metrics on performance.
- Hands-On: Deploying a simple Azure Function, creating load tests using Azure Load Testing, and analyzing the data with Azure Monitor and Application Insights.

Comparative Analysis of Performance Testing on AWS and Azure (1 Hour)

- Key differences in serverless performance testing tools and capabilities between AWS and Azure.
- Analyzing data and creating reports on latency, cost, and scalability.
- Hands-On: Generating performance reports and comparing outcomes across both platforms.

Overview of IP Protocols (15 mins)

- Brief review of IP protocol versions (IPv4 vs. IPv6).
- Reasons behind the transition to IPv6: exhaustion of IPv4 addresses, security, and scalability.

Understanding IPv6 Address Structure (45 mins)

- IPv6 address format, length, and notation (hexadecimal and colon-separated).
- Differences between IPv4 and IPv6 addressing.
- Types of IPv6 addresses:
    - Unicast, Multicast, and Anycast.
- Hands-On: Practice identifying and interpreting IPv6 address types.

IPv6 Configuration and Addressing (1 Hour)

- Stateless and Stateful Address Autoconfiguration (SLAAC and DHCPv6).
- Neighbor Discovery Protocol (NDP) and its role in IPv6.
- IPv6 address assignment and prefix delegation.
- Hands-On: Setting up and configuring IPv6 on virtual machines.

2.1 Database Performance Testing on AWS and Azure

- **Database Performance Testing Use Case**: Evaluating database efficiency under load, query optimization, and resource management.
- **AWS Database Testing Tool**
    - **Amazon RDS Performance Insights**: Tool for monitoring and optimizing database performance.
    - **Hands-On**: Setting up RDS Performance Insights, running sample queries, and analyzing performance.
- **Azure Database Testing Tool**
    - **Azure SQL Database Insights**: Monitoring SQL database performance and optimization.
    - **Hands-On**: Configuring Azure SQL Database Insights, tracking metrics, and optimizing queries.
- **Open-Source Database Testing Alternatives**
    - **pgbench** (for PostgreSQL), **sysbench** (for MySQL), and **HammerDB** (multi-database support).
    - **Hands-On**: Running performance tests on sample databases and analyzing results using open-source tools.

Day 3 : Terraform, Ansible, Web Application Deployment, and Testing

Note: it may extend to 10 hours so can cover in day4 also I

1.1 Introduction to Infrastructure as Code (IaC) and Configuration Management

- **Concepts Covered**:
    - Overview of Infrastructure as Code (IaC) with Terraform.
    - Benefits of using Terraform for multi-cloud environments.
    - Introduction to configuration management with Ansible.
    - Importance of automating configuration for web application setup.
- **Objective**: Set a foundational understanding of why IaC and configuration management are essential for automated deployments.

1.2 Hands-On: Setting Up Infrastructure with Terraform

- **Infrastructure Setup**:
    - Define infrastructure resources in Terraform for AWS and Azure, including:
        - **Network Components**: VPC, subnets, security groups.
        - **Compute Resources**: EC2 instances in AWS, VMs in Azure.
    - **Hands-On Task**: Write Terraform scripts to provision the network and compute layers.
- **Deployment Execution**:
    - **Apply Terraform Configuration** on both AWS and Azure.
    - Track and compare **provisioning time** for infrastructure across AWS and Azure.

1.3 Configuring Applications Using Ansible Playbooks

- **Application Setup with Ansible**:
    - Overview of Ansible playbooks for automated configuration.
    - Writing a playbook to install and configure **Apache HTTPD** on provisioned instances.
- **Hands-On Task**:
    - Execute the Ansible playbook on both AWS and Azure instances.
    - Validate the HTTPD setup by accessing the deployed web application in a browser.

1.4 Load Testing the Deployed Web Application

- **Introduction to Load Testing Tools**:
    - Overview of tools such as **Apache JMeter** or **Locust** for load testing.
- **Hands-On Load Testing**:
    - Configure and run a **basic load test** against the deployed Apache HTTPD application.
    - Monitor response times, request handling, and performance metrics.

1.5 Comparative Analysis of Infrastructure and Application Setup on AWS vs. Azure

- **Key Metrics for Comparison**:
  - Infrastructure provisioning speed.
  - Time taken for full deployment (infrastructure + application setup).
  - Performance metrics from load testing.
- **Discussion and Insights**:
  - Analyze the speed, efficiency, and performance across AWS and Azure.
  - Identify key takeaways and potential optimizations for each environment.

Capturing Cost Estimates Using Terraform for AWS and Azure

- **Cost Estimation for AWS**:
  - Use the **Terraform AWS Pricing module** or AWS's `pricing` API to estimate costs based on the resources you plan to deploy.
  - There are external tools, like `Infracost`, that integrate with Terraform to provide cost estimates before applying changes.
- For example:
  bash
  Copy code
  `infracost breakdown --path /path/to/your/terraform/files`
  -
  - `Infracost` will provide a cost breakdown based on the resources defined in Terraform.
- **Cost Estimation for Azure**:
  - For Azure, you can also use `Infracost`, or refer to the **Azure Pricing Calculator** and API, or use built-in Azure cost analysis tools post-deployment.
  - Alternatively, configure **Azure Cost Management** to monitor the actual costs incurred.

Day 4: Jenkins CI/CD with Infrastructure Automation and Testing

1. **Introduction to Jenkins and CI/CD Pipelines**
   - Overview of Jenkins as a CI/CD tool.
   - Key concepts: Jobs, Pipelines, and Nodes.
   - Introduction to automation tools: Terraform for infrastructure and Ansible for configuration.

2. **Setting Up Jenkins and Integrating with Source Control**
    - Installing Jenkins and plugins for Terraform, Ansible, and Git.
    - Configuring Jenkins with GitHub or GitLab for version control.
    - Hands-on: Setting up a basic Jenkins Job to trigger on code commits.
3. **Automating Infrastructure Provisioning with Terraform in Jenkins**
    - Introduction to Terraform and its role in Infrastructure as Code (IaC).
    - Creating a Jenkins Pipeline to deploy cloud infrastructure (e.g., AWS EC2 instances) using Terraform.
    - Hands-on: Writing a Jenkins Pipeline that runs Terraform commands (`terraform init`, `terraform apply`, `terraform destroy`).
    - Adding post-deployment validation steps to ensure infrastructure is set up correctly.
4. **Configuration Management with Ansible in Jenkins**
    - Introduction to Ansible and its role in configuration management.
    - Configuring a Jenkins Job to run Ansible playbooks for setting up applications (e.g., deploying and configuring Apache HTTPD).
    - Hands-on: Writing a Jenkins Pipeline that triggers an Ansible playbook on provisioned servers.
    - Adding steps to validate application setup and configurations.
5. **Automated Testing with Selenium and JMeter**
    - Overview of Selenium for UI testing and JMeter for load testing.
    - Creating a Jenkins Pipeline to run Selenium tests on the deployed web application.
    - Integrating JMeter for load testing and performance benchmarking.
    - Hands-on: Setting up a Jenkins Job that runs both Selenium and JMeter tests, generating test reports.
6. **Container-Based Deployment with Jenkins**
    - Introduction to container-based deployment using Docker.
    - Building and pushing Docker images to a registry in Jenkins Pipeline.
    - Example: Writing a Pipeline to package the application as a Docker image and deploy to a containerized environment (local or cloud-based).
    - Hands-on: Deploying a sample application to a Kubernetes cluster using Jenkins.

Day 5: Security Scanning, Advanced Monitoring, and Containerized CI/CD

1. **Security Testing in Jenkins: SAST and DAST**
    - Overview of security testing in CI/CD: Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST).
    - Integrating SAST tools like SonarQube for code scanning.
    - Example: Adding a Jenkins stage to scan code for vulnerabilities and compliance with SAST.
    - Hands-on: Setting up SAST in a Jenkins pipeline.

2. **Implementing DAST Tools (ZAP and Nikto) in Jenkins Pipelines**
    - Introduction to OWASP ZAP and Nikto for DAST.
    - Creating a Jenkins Pipeline to perform DAST scans on the deployed application.
    - Configuring ZAP to identify vulnerabilities in web applications and generating security reports.
    - Hands-on: Configuring a Jenkins Job to run both ZAP and Nikto, and automatically fail builds on critical vulnerabilities.
3. **Container Security with Jenkins**
    - Overview of security best practices for containerized applications.
    - Implementing image scanning tools in Jenkins (e.g., Trivy or Aqua Security) for container security.
    - Example: Writing a Jenkins Pipeline that scans Docker images for vulnerabilities before deployment.
    - Hands-on: Setting up a container image scan and automated reporting in Jenkins.
4. **Monitoring with Dynatrace in Jenkins Pipelines**
    - Overview of application performance monitoring with Dynatrace.
    - Integrating Dynatrace with Jenkins to monitor application health post-deployment.
    - Hands-on: Configuring Dynatrace to send performance metrics to Jenkins and generate alerts on thresholds.
    - Adding Dynatrace monitoring stages to the Jenkins Pipeline for ongoing observability.
5. **Hands-On: End-to-End CI/CD with Jenkins, Containers, Security, and Monitoring**
    - Building a complete CI/CD pipeline that includes:
        - Provisioning infrastructure with Terraform.
        - Configuring applications with Ansible.
        - Running automated tests with Selenium and JMeter.
        - Performing security scans (SAST and DAST) with ZAP and Nikto.
        - Deploying the application as a Docker container to a Kubernetes cluster.
        - Monitoring with Dynatrace to verify application performance.
    - Reviewing best practices for Jenkinsfile management, security, and version control.
6. **Review and Q&A**
    - Summarize the topics covered and review key takeaways.
    - Open session for Q&A on advanced Jenkins CI/CD practices and troubleshooting.