TECH WORLD WITH MURALI

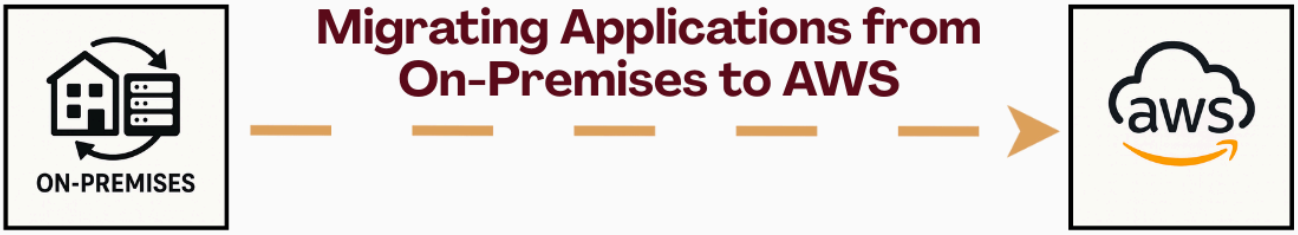
Courses Offered:

- DevOps
- AWS
- Kubernetes
- Terraform
- Helm
- Logging and Monitoring, etc.

Mail: techworldwithmurali@gmail.com

MOOLE MURALIDHARA REDDY
DevOps Consultant and a Solution Architect

Website: techworldwithmurali.com



End-to-End Cloud Migration Plan from On-Premises to AWS

TechWorld With Murali

- This plan outlines a **comprehensive enterprise cloud migration strategy** led by cross-functional collaboration across Dev, QA, DevOps, and Architecture teams.
- The objective is to migrate a mix of **microservices**, **monolithic applications**, and **databases** from on-premises infrastructure to the **AWS cloud** using industry best practices, automation tools, and a phased execution approach.

Example Scenario:

Your organization has multiple environments in an on-premises setup, and now, we are planning to migrate some applications and databases from on-premises to AWS.

Existing On-Premises Environments:

- Dev
- Test
- QA
- UAT
- Pre-PROD
- Prod

Applications & Databases to be Migrated:

- 10 microservices
- 4 monolithic applications
- 5 MySQL databases

Phase 1: Planning & Preparation

Step 1: Initial Discussion

- The Development (Dev) team, DevOps team, and QA VP/Director discuss the migration strategy.

Step 2: Informing Managers

- The respective VP/Director informs managers about the new project to migrate from on-premises to AWS.

Step 3: Manager/Architect Level Planning

- Managers and Architects discuss and plan the necessary actions required for the migration.

Step 4: Communicating the Plan to Teams

- The respective managers inform their teams about the migration project.
- Managers determine which team members will be involved in the migration.

Step 5: Scrum Master's Role

- The Scrum Master is responsible for:
    - Scheduling migration meetings

- Tracking status updates
- Managing migration-related activities
- Coordinating with other teams by reaching out or scheduling meetings when additional information is needed.

Step 6: Dev, DevOps & QA Collaboration

- The Development, DevOps, and QA teams meet to discuss the migration process in detail.

Step 7: Application Inventory Preparation

- Prepare an Excel sheet listing all applications and databases that need to be migrated, along with any dependencies.

Example:

- 10 microservices
- 4 monolithic applications
- 5 MySQL databases
- Other dependencies, if applicable

Step 8: Review & Task Assignment

- The Excel sheet is reviewed step by step.
- Tasks are assigned to teams/engineers responsible for different components.

Step 9: JIRA/Rally Ticket Creation

- The DevOps manager or Scrum Master creates JIRA/Rally tickets based on assigned tasks from the Excel sheet.
- Daily internal meetings for the DevOps, QA, and Development teams will be conducted separately to track progress.
- The Scrum Master will set up daily or bi-weekly migration meetings to monitor progress.

📦 Cloud Migration Strategies: The 6 R's

When migrating workloads to the cloud, organizations typically follow one or more of the following six strategies, commonly known as the 6 R’s of Cloud Migration:

---

1. 🚚 Lift and Shift (Rehost)

- Definition: Move applications to the cloud without modifying them.
- When to Use:
  - Quick migrations
  - Legacy systems
  - When minimal change is required

- **Example:** Migrating a virtual machine from on-premises to AWS EC2 without altering the application.

---

2. 🔧 Replatform (Lift, Tinker, and Shift)

- Definition: Move to the cloud with minor optimizations to take advantage of cloud-native capabilities.
- When to Use:
  - To improve performance or cost
  - When small changes are feasible
- **Example:** Migrating an app to AWS EC2 and replacing a local database with Amazon RDS.

---

3. 🧬 Refactor / Rearchitect

- Definition: Redesign the application to be cloud-native, often involving significant changes.
- When to Use:
  - For scalability, agility, or long-term cost savings

- To adopt microservices, containers, or serverless architectures

- **Example:** Refactoring a monolithic application into a microservices-based architecture using AWS Lambda for compute, API Gateway for routing, and DynamoDB as a scalable NoSQL database.

---

4. 🗑️ Retire

- Definition: Decommission applications that are no longer useful.
- When to Use:
  - During assessment, when apps are identified as obsolete or redundant
- **Example:** Shutting down an old internal reporting tool that is no longer used.

---

5. 🛑 Retain

- Definition: Keep certain applications on-premises or delay migration.
- When to Use:
  - Compliance or regulatory concerns
  - Low priority applications
  - Systems not yet ready for migration

- **Example:** Retaining an ERP system that requires on-prem hosting due to licensing constraints.

---

6. 🛒 Repurchase

- Definition: Move to a SaaS-based solution, effectively replacing the existing system.
- When to Use:
  - When a better SaaS alternative exists
  - For faster time to value and reduced management overhead
- **Example:** Moving from a self-hosted CRM to Salesforce or Microsoft 365.

Phase 2: Infrastructure Setup

Step 10: Architecture Design

- Respective Architects will design the implementation architecture for this migration.

Step 11: Task Execution Begins

- Based on assigned tasks in JIRA, DevOps engineers and the Development team start executing tasks based on priority.

Step 12: AWS Account Creation

- If AWS accounts are not already available, create AWS accounts as per the migration plan.
- Either set up environment-specific accounts (e.g., separate accounts for Dev, Test, QA) or consolidate all lower environments into a single AWS account.
- Production, Pre-Prod, and Infrastructure accounts should be separate for security and compliance.

Step 13: AWS Networking Setup via Terraform

- The DevOps engineer sets up AWS networking components, including:
    - VPC, Subnets, Route Tables, NAT Gateways, Internet Gateways, Security Groups, NACLs
    - Setting up the Transit Gateway (TGW) if required for multi-environment networking

Example: VPC CIDR Blocks

- DEV CIDR - 10.20.0.0/16
- TEST CIDR - 10.30.0.0/16
- UAT CIDR - 10.40.0.0/16
- PRE-PROD CIDR - 10.50.0.0/16
- PROD CIDR - 10.60.0.0/16

- INFRA CIDR - 10.70.0.0/16

  Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 14: Site-to-Site VPN Setup

- Work with the on-premises network team to establish a Site-to-Site VPN between on-premises and AWS.
- Test connectivity to ensure smooth integration with AWS resources.

  Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 15: AWS Client VPN / Cisco VPN Setup

- Set up AWS Client VPN or Cisco VPN to allow secure access to private AWS resources.
- Example use case: Accessing internal application URLs (e.g., https://user-registration-dev.techworldwithmurali.in).

Prerequisites for AWS Client VPN Setup:

1. AWS Directory Service configuration
2. Amazon Certificate Manager (ACM) for SSL/TLS certificates

Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 16: DevOps Tools Installation in Infra VPC

- Deploy essential DevOps tools in the Infra VPC for automation and CI/CD processes.

Tools to be installed:

- Jenkins - for CI/CD pipeline automation
- JFrog Artifactory - for artifact storage
- SonarQube - for code quality analysis

Example URLs:

- Jenkins - https://jenkins.techworldwithmurali.in
- JFrog Artifactory - https://artifactory.techworldwithmurali.in
- SonarQube - https://sonarqube.techworldwithmurali.in

Note: These tools can either be deployed on individual EC2 instances or hosted in an EKS (Kubernetes) cluster for better scalability.

Step 17: ECS or EKS setup for a microservice application via Terraform.

- Deploy microservices in either:
  - ECS (Elastic Container Service)
  - EKS (Elastic Kubernetes Service)
- The choice depends on application architecture and team discussions.

Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 18: Monolithic Applications Setup via Terraform

- For monolithic applications, set up:
    - Auto Scaling Groups (ASG)
    - Application Load Balancer (ALB) (internal/external) for traffic distribution
    - Create/update the records in Route 53 hosted zone etc.

Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 19: Route 53 DNS Configuration

- Create a Hosted Zone in Route 53. Example:
    - techworldwithmurali.in
- Migrate DNS records to the Route 53 hosted zone, if applicable.

Step 20: Database Setup

- Set up Amazon RDS MySQL Cluster as per the database requirements.
- Inform the DBA team once provisioning is complete.

  Note: Wherever applicable, we will use Terraform to automate infrastructure provisioning.

Step 21: Database Migration

- Migrate databases from on-premises to AWS using the appropriate strategy:
  - AWS DMS (Database Migration Service)
  - Log Shipping
  - Other database replication techniques

Responsibility: The DBA team will handle this migration process.

Step 21: Infrastructure Setup Completion Notification

- Once the infrastructure setup is complete:
  - The DevOps team will notify the Scrum Master, Development team, and QA team.
  - All corresponding JIRA/Rally tickets will be updated accordingly.
  - A final infrastructure validation checklist will be completed to ensure:
    - Networking components (VPC, subnets, security groups) are correctly configured.
    - VPN connectivity is established and tested.
    - DevOps tools are accessible and functional.
    - Databases are provisioned, and accessible.
    - Route 53 DNS records are correctly set up.

- Documentation, including architecture diagrams, and access credentials, will be shared in the designated **Confluence/JIRA **.

Phase 3: Build, Deployment & Testing

Step 22: Development Team Preparation

- Developers implement any necessary changes to their applications for deployment.

Step 23: Application Build

- Developers build the application using Jenkins jobs.

Step 24: Deployment of Microservices/Monolithic Applications

- Developers deploy microservices in the DEV environment using Jenkins jobs. After deployment:
  - Load Balancers (LB), listeners, and Route 53 records are automatically created using the Ingress Controller and ExternalDNS.
- Developers deploy monolithic applications in the DEV environment using Jenkins jobs. After deployment:
  - Load Balancers, listeners, and Route 53 records are automatically created via Terraform.

Step 25: Testing the Application

- Validate the application whether the functionality is working as expected.

Note: Once testing is complete, we will shut down the application on the AWS side. During the cutover, we will bring the application back up.

Phase 4: Final Cutover to AWS: Cutover Plan & Go-Live Checklist

Step 26: Schedule the Cutover Meeting

- Based on the final discussion, a cutover date will be finalized.
- The Scrum Master will schedule the meeting.
- The cutover process typically takes 3 to 4 hours, but depending on the complexity of the applications, it may take longer.

Step 27: Send Notification to Clients

- **Notify Clients:** Send a formal notification to all relevant clients/stakeholders regarding today's scheduled cutover plan.
- Include Key Details:
    - Cutover Date & Time
    - Expected Downtime (if any)

- New AWS endpoint or DNS URL (if it's changing)
- What actions (if any) are required from the client side
- Point of Contact in case of any issues or queries

- **Purpose of the Notification:**
  - Ensure clients are aware of the migration
  - Minimize surprises and set expectations regarding any temporary inaccessibility
  - Help clients validate functionality post-cutover

✅ **Example email subject:**

"[INFO] Scheduled Cutover to AWS – [Application Name] on [Date]"

✅ **Example content snippet:**

Dear Client,

As part of our cloud modernization initiative, we are migrating [Application Name] from our on-premises infrastructure to AWS today, [Date].

The expected cutover time is from [Start Time] to [End Time].

During this time, the application may be briefly inaccessible.

Post-migration, the application will be accessible via: [New URL].

Please contact [Support Contact Info] for any assistance.

Step 27.1: Stop the Application in On-Premises

- Stop the on-premises application after the final data sync.
- Ensure that no new transactions are being processed in the on-premises system.

Step 28: DNS Cutover & Go-Live

- Update DNS Records: Modify Route 53 records to point to the AWS Load Balancer endpoints.
- Enable AWS Application: Make the AWS-hosted application live.
- Monitor Traffic Flow: Ensure users are now connecting to AWS instead of the on-premises system.

Step 29: Application Validation & QA Testing

- Developers and QA teams validate that the application functions as expected.
- Ensure all integrations, services, and APIs are working correctly.
- The QA team performs regression testing and notifies all relevant teams of any issues.

Step 30: Monitoring & Stabilization

- Monitor the environment for a few days to detect any issues.
- Address and resolve issues as needed with the Development or DevOps teams.

Step 31: Rollback Plan (If Needed)

- If issues occur, attempt to fix them on demand.
- If unresolved, roll back the DNS endpoint to the on-premises infrastructure.

Step 32: Migration to Other Environments

- If everything is stable, replicate the same process for:
    - TEST
    - QA
    - UAT
    - Pre-Prod
    - PROD

About the Author:

MOOLE MURALIDHARA REDDY

Solution Architect / DevOps Consultant

- I am having rich experience in DevOps and Cloud technologies and have done many projects on all varieties of tools which are hot cake in the market.
- I am passionate about learning new technology and teaching.
- My courses focus on providing students with an interactive and hands-on experience in learning new technology that makes learning really interesting.
- I have a wide range of experience in Telecom, Banking, Healthcare, Retail domains.
- I have been training people in newer technologies, like DevOps, AWS, Kubernetes, Terraform, Rancher, etc. and they have settled in MNC's and drawing respectable salaries.
- I have undergone many challenges and changed the entire phase of the projects and mastered in DevOps implementation and many more to go.
- Certified in AWS, Kubernetes , Terraform, Linux and many to go.

Please check out my courses and join me with thousands of others who are learning the latest DevOps and Cloud tools!

Youtube : https://www.youtube.com/@TechWorldwithMural

Website : https://techworldwithmurali.com/