Page 1: Introduction & Deployment Basics

1. What is AWS Deployment?

- Process of **launching applications or services** on AWS infrastructure.
- Can include **web apps, databases, serverless functions, containers, or entire infrastructures**.

2. Key Concepts

- **Regions & Availability Zones (AZs)**: Geographical locations to deploy resources.
- **Elasticity**: Automatically scale resources up/down based on demand.
- **High Availability**: Deploy across multiple AZs for redundancy.
- **Fault Tolerance**: Automatic failover in case of resource failure.

3. Deployment Types

- **Manual Deployment**: Using AWS console.
- **Automated Deployment**: Using scripts, CI/CD pipelines, or Infrastructure as Code (IaC).

⚙️ **Page 2: AWS Deployment Services**

1. Compute Services

Service	**Use Case**
EC2	Deploy virtual servers
Lambda	Serverless deployment
Elastic Beanstalk	Managed app deployment (web apps)
ECS / EKS	Containerized apps

2. Storage & Databases

- **S3**: Store and deploy static assets.
- **RDS / DynamoDB**: Backend database deployment.

3. Deployment & CI/CD Tools

- **CodeCommit**: Source code repository
- **CodeBuild**: Build and test application

- **CodeDeploy**: Automate deployment to EC2, Lambda, or on-premises
- **CodePipeline**: Orchestrate CI/CD workflow

4. Infrastructure as Code (IaC)

- **CloudFormation**: Define infrastructure using templates.
- **Terraform (optional)**: Cross-cloud IaC solution.

💾 Page 3: Best Practices & Advanced Topics

1. Deployment Strategies

- **Blue/Green Deployment**: Two identical environments; switch traffic from old to new version.
- **Rolling Deployment**: Gradually update instances to reduce downtime.
- **Canary Deployment**: Deploy to small subset of users first.

2. Monitoring & Logging

- **CloudWatch**: Track metrics, logs, alarms.
- **CloudTrail**: Audit API calls and deployment activities.

3. Security Considerations

- Use **IAM roles** for deployment permissions.
- Enable **encryption** for data in transit and at rest.
- Follow **least privilege principle** for deployment accounts.

4. Cost Optimization

- Use **Auto Scaling** to adjust resources.
- Leverage **Spot Instances** for non-critical workloads.
- Clean up unused resources after deployment.

5. Tips for Successful Deployment

- Test in a **staging environment** before production.
- Automate rollback mechanisms.
- Document the deployment process for reproducibility.