Best Practices for Building

Cloud-Native Apps

Development

- Design applications with Microservices architecture
- Place business functions behind APIs
- Use stateless services and event-driven approach
- Automate tests - unit, API, acceptance

Infrastructure

- Utilize managed services such as RDS, Aurora, DynamoDB, and Redshift.
- Take advantage of autoscaling—automatically adjust resources
- Build resilient services to ensure auto-redundancy
- Use serverless technologies such as AWS Lambda and Azure Functions
- Benefit from multiple data centers to ensure business continuity

Operations

- Maintain infrastructure as code using tools such as AWS CloudFormation
- Plan for immutable infrastructure
- Automate code deployment pipeline
- Deploy services/applications in containers
- Use orchestration tools such as Kubernetes, Swarm

Storage

- Establish a storage lifecycle policy
- Organize data based on attributes such as frequency-of-access and planned retention period
- Enforce retention policies using code (and OS properties, where possible)
- Implement a cloud storage data ageing management mechanism
- Automate backup

Security

- Adopt DevSecOps approach
- Architect the solution based on applicable security standards
- Implement logic-based security solutions with custom scripting
- Encrypt sensitive data
- Harden servers and containers
- Use managed services such as web application firewall
- Integrate application security testing into CI/CD
- Define cloud-based backup and disaster recovery strategy

Monitoring

- Ensure continuous monitoring and threat prediction with stacks such as ELK and OSSEC
- Automate detection of environment/configuration drift
- Create compliance as code framework and automate audit checks
- Use white box monitoring methods in addition to external polling
- Adopt tools such as Prometheus to monitor a wide variety of custom metrics
- Track all related requests with request tracing tools such as Jaeger