

### Experiment 1: Cloud Computing & Its Architecture

#### Aim:

- Understand what cloud computing is and its high-level architecture.

#### Theory:

- Definition: on-demand network access to shared configurable resources (NIST model).
- Origin: from 1960s ARPANET; term coined in 1996.
- Service Models: SaaS, PaaS, IaaS.
- Deployment Models: Public, Private, Community, Hybrid.
- Characteristics: on-demand self-service; broad network access; resource pooling; rapid elasticity; measured service.

#### Steps:

- Discuss and diagram the NIST reference architecture.
- List & classify examples of service/deployment models.
- Identify real-world cloud services matching each model.

#### Advantages:

- Easy backup/restore; improved collaboration; constant accessibility; lower maintenance; mobility.

#### Disadvantages:

- Internet dependency; vendor lock-in; limited control; security/privacy risks.

### Experiment 2: Hosted Virtualization (VirtualBox & KVM)

#### Aim:

- Study & implement hosted hypervisors.

#### Theory:

- Virtualization: multiple OS on one host.
- VM: software-emulated hardware.

#### Steps:

1. Download & install VirtualBox; create VM; install OS; configure network.
2. Verify CPU virtualization; install KVM packages; setup libvirt; use virt-manager to create VMs.

#### Advantages:

- Better resource utilization; isolation; rapid provisioning.

#### Disadvantages:

- Performance overhead; complexity; extra management layer.

### Experiment 3: Bare-metal Virtualization (Xen)

#### Aim:

- Install & configure Xen hypervisor.

#### Theory:

- Hypervisor: hardware abstraction for VMs.
- Bare-metal: runs directly on hardware.

#### Steps:

1. Install Xen Server from CD; configure network & storage.
2. Download & add server in XenCenter; create storage repository; create & install VMs.

#### Advantages:

- Near-native performance; strong isolation; enterprise features.

#### Disadvantages:

- Requires dedicated hardware; complex setup; tool lock-in.

### Experiment 4: Infrastructure as a Service (AWS EC2)

#### Aim:

- Launch & manage EC2 instances.

#### Theory:

- IaaS: on-demand compute, storage, networking.

#### Steps:

1. Launch EC2 instance using t2.micro; configure key pair & security group.
2. Connect via SSH/RDP.

#### Advantages:

- Fast provisioning; scalable; pay-as-you-go.

#### Disadvantages:

- User-managed OS; cost variability; vendor lock-in.

### Experiment 5: Platform as a Service (AWS Elastic Beanstalk)

#### Aim:

- Deploy web app via Elastic Beanstalk.

#### Theory:

- PaaS abstracts infrastructure; automated scaling & monitoring.

Steps:

1. Package app; create Beanstalk app; upload code; configure environment.

Advantages:

- No infra management; auto-scaling; monitoring.

Disadvantages:

- Limited infra control; customization limits.

#### Experiment 6: Storage as a Service (AWS S3)

Aim:

- Use S3 for object storage.

Theory:

- Object storage in buckets; versioning; policies; encryption.

Steps:

1. Create S3 bucket; upload objects; set permissions.

2. Enable versioning & lifecycle rules.

Advantages:

- High durability; scalability; fine-grained ACLs.

Disadvantages:

- Egress costs; eventual consistency; not ideal for block storage.

#### Experiment 7: Security as a Service (AWS Security Tools)

Aim:

- Explore AWS security (GuardDuty, Inspector, WAF, KMS).

Theory:

- Shared responsibility; key security services.

Steps:

1. Enable IAM best practices; MFA.

2. Activate GuardDuty & Inspector; configure WAF; create KMS key.

Advantages:

- Scalable security; pay-per-use.

Disadvantages:

- Configuration complexity; learning curve.

#### Experiment 8: Identity & Access Management (AWS IAM)

Aim:

- Implement users, roles & policies.

Theory:

- Authentication vs authorization; policy evaluation.

Steps:

1. Create IAM users/groups with least privilege; enable MFA.

2. Create EC2 role with S3 access; test via CLI.

Advantages:

- Fine-grained control; audit logs.

Disadvantages:

- Policy complexity; misconfiguration risk.

#### Experiment 9: Database as a Service (AWS RDS)

Aim:

- Launch & use RDS SQL Server.

Theory:

- DBaaS: managed DBs with automated backups/patching.

Steps:

1. Create DB via Easy create; note endpoint.

2. Connect via SSMS; run test query.

Advantages:

- Automated maintenance; high availability.

Disadvantages:

- Less OS control; cost; customization limits.

#### Experiment 10: Containerization (Docker)

Aim:

- Package & run apps in containers.

Theory:

- Docker containers vs images; Dockerfile builds images.

Steps:

1. Install Docker; pull base image.
2. Write Dockerfile; build & run container; manage with docker CLI.

Advantages:

- Fast startup; portability; consistency.

Disadvantages:

- Shared kernel risks; storage complexity.

Experiment 11: Container Orchestration (Kubernetes)

Aim:

- Deploy & manage apps on Kubernetes.

Theory:

- Pods, deployments, services; declarative desired state.

Steps:

1. Provision cluster; create YAML for Deployment & Service.
2. Apply manifests; scale & update; inspect resources.

Advantages:

- Self-healing; auto-scaling; rolling updates.

Disadvantages:

- Steep learning curve; overhead.

Experiment 12: Mini-project – Multi-tier Web App

Aim:

- Integrate IaaS, PaaS, DBaaS, Storage, Security services.

Theory:

- End-to-end cloud solution & best practices.

Steps:

1. Provision VMs; deploy PaaS app; configure S3 & RDS.
2. Secure with IAM, WAF, SSL; test autoscaling & backups.

Advantages:

- Real-world integration; full-stack practice.

Disadvantages:

- Complexity; cost management challenges.