

UNIT 2

Cloud Computing Architecture

(16 Hours) — Cloud Computing (CACS402)

Disclaimer: These notes were prepared as a study aid and may contain errors or omissions. Please cross-check important definitions, figures and product details (especially vendor services such as Google App Engine, Microsoft Azure and Aneka, which change often) against your prescribed textbook, class notes and official documentation before relying on them in an exam.

2.1 Cloud Reference Model

The **cloud reference model** (NIST / Cloud Security Alliance view) organises cloud offerings into three layered service models. Each higher layer builds on the one below and hides more complexity from the user.

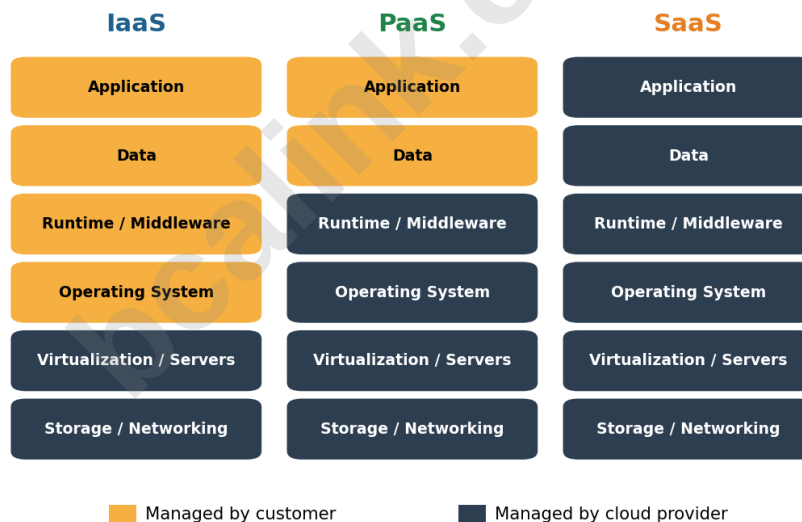


Figure 2.1 — Layered cloud reference model (IaaS, PaaS, SaaS)

2.1.1 Platform as a Service (PaaS)

PaaS provides a ready platform — OS, runtime, middleware, databases and development tools — on which developers build, test and deploy applications without managing servers.

- **User controls:** application code and its data/configuration.
- **Provider controls:** servers, storage, network, OS, runtime patches, scaling.
- **Examples:** Google App Engine, Microsoft Azure App Service, Heroku, AWS Elastic Beanstalk.

- **Advantages:** faster development, automatic scaling, lower admin effort.
- **Limitations:** restricted languages/frameworks, vendor lock-in, less low-level control.

2.1.2 Software as a Service (SaaS)

SaaS delivers a complete application over the Internet, usually through a web browser, on a subscription basis. The provider manages everything from infrastructure to application features and updates.

- **Examples:** Gmail, Google Workspace, Microsoft 365, Salesforce, Zoom, Dropbox.
- **Advantages:** no installation, automatic updates, accessible anywhere, predictable subscription cost.
- **Limitations:** limited customisation, data held by provider, needs Internet, integration issues.

2.1.3 Infrastructure as a Service (IaaS)

IaaS offers fundamental computing resources — virtual machines, block/object storage, virtual networks and IP addresses — on demand. The user installs and manages the OS and everything above it.

- **Examples:** Amazon EC2, Google Compute Engine, Azure Virtual Machines, DigitalOcean.
- **Advantages:** maximum control and flexibility, no hardware purchase, pay per use.
- **Limitations:** customer must patch and secure the OS and apps; needs sysadmin skills.

Aspect	IaaS	PaaS	SaaS
Target user	System administrators / IT architects	Developers	End users
Control	High	Medium	Low
Management effort	High	Medium	Minimal
Flexibility	Highest	Moderate	Lowest
Example	Amazon EC2	Google App Engine	Gmail

Example 2.1 — Choosing a service model

- A start-up wants to host a custom-built web app but does not want to patch servers.
- A school wants email and document editing for staff with no IT team.
- A research lab needs a Linux VM with a custom kernel and GPUs.

Solution

- (a) **PaaS** — upload code; provider manages OS and scaling.
- (b) **SaaS** — use Google Workspace or Microsoft 365 directly.
- (c) **IaaS** — full control over OS and hardware type is required.

Note: Some texts also mention XaaS (Anything as a Service); DBaaS, FaaS (serverless functions), CaaS (containers), etc.

2.2 Cloud Deployment Models

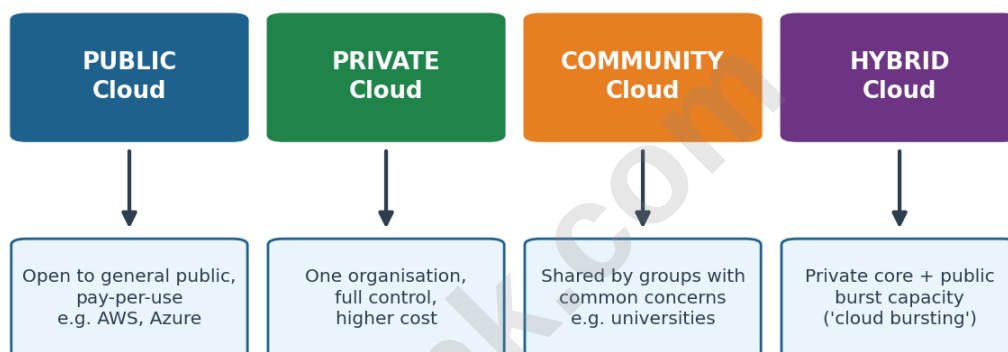


Figure 2.2 — Cloud deployment models

2.2.1 Public Clouds

Infrastructure is owned and operated by a third-party provider and made available to the general public over the Internet. Resources are shared among many tenants.

- **Pros:** no capital cost, huge scale, pay-per-use, no maintenance.
- **Cons:** less control, shared environment raises security/compliance concerns.
- **Examples:** AWS, Microsoft Azure, Google Cloud Platform.

2.2.2 Private Clouds

Cloud infrastructure dedicated to a single organisation. It may be hosted on-premises or by a third party, but is not shared with others.

- **Pros:** greater control, customisation, security and regulatory compliance.
- **Cons:** high capital and operating cost, limited scalability, in-house expertise needed.
- **Examples:** OpenStack or VMware-based enterprise clouds; bank or government clouds.

2.2.3 Community Cloud

Infrastructure shared by several organisations that have common concerns — mission, security requirements, policy or compliance. It may be managed by the organisations or a third party.

- **Pros:** cost shared among members; meets common compliance needs.
- **Cons:** fewer users to share cost than public; governance among members can be complex.
- **Examples:** a group of universities sharing a research cloud; government agencies sharing a secure cloud; healthcare consortia.

2.2.4 Hybrid Clouds

A composition of two or more distinct clouds (private, community or public) that remain unique entities but are bound together by standardised technology enabling data and application portability.

- **Pros:** keep sensitive data private while using public cloud for scale; **cloud bursting** handles peak load.
- **Cons:** complex to design, integrate and secure; network latency between clouds.
- **Example:** a bank stores customer records in a private cloud but runs its public website on a public cloud.

Feature	Public	Private	Community	Hybrid
Ownership	Provider	Single organisation	Group of organisations	Mixed
Cost	Low (pay-per-use)	High	Shared / medium	Medium
Control	Low	High	Medium	Medium–High
Security	Shared responsibility	Highest	High	Depends on split
Scalability	Very high	Limited	Moderate	High (via bursting)

2.3 Cloud Design and Implementation Using SOA

Service-Oriented Architecture (SOA) is a design style in which application functionality is packaged as loosely coupled, reusable **services** with well-defined interfaces that communicate over a network. Cloud computing extends SOA: every cloud resource is exposed as a service (“everything as a service”).

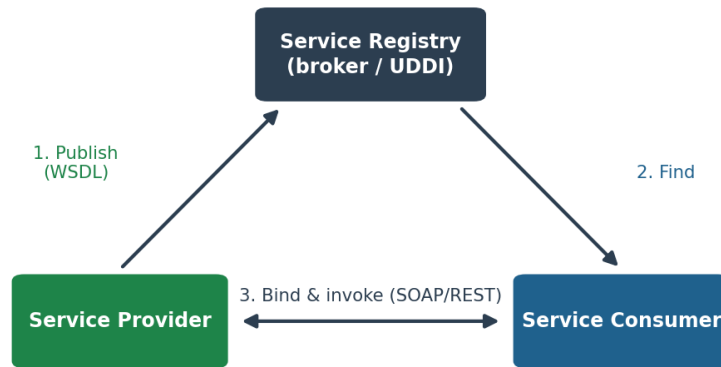


Figure 2.3 — SOA: publish–find–bind interaction

SOA principles

- **Loose coupling** — services depend only on interface contracts.
- **Reusability** — one service can be used by many applications.
- **Statelessness** — each request carries all needed information.
- **Discoverability** — services are described (WSDL) and listed in a registry (UDDI).
- **Composability** — small services combine into larger business processes (orchestration).

Technologies used

Technology	Role
SOAP	XML-based messaging protocol for web services.
WSDL	XML description of a service's operations and endpoints.
UDDI	Registry for publishing and discovering services.
REST	Lightweight style using HTTP verbs (GET, POST, PUT, DELETE) and JSON/XML.

Design and implementation steps

- Identify business processes and decompose them into services.
- Define service interfaces and contracts (WSDL / OpenAPI).
- Implement services and deploy them on IaaS/PaaS resources.
- Publish to a registry / API gateway; secure with authentication and encryption.
- Compose services using orchestration (e.g., BPEL) and monitor them against SLAs.

Note: Cloud layers map naturally to SOA: IaaS services expose infrastructure, PaaS services expose platform functions, SaaS services expose business functions.

2.4 Security, Trust and Privacy

Because customers hand data and workloads to a provider, **security, trust and privacy** are central architectural concerns.

Security

Protection of confidentiality, integrity and availability (the CIA triad) of data and services.

- Encryption of data at rest and in transit (TLS, AES).
- Strong identity and access management, multi-factor authentication.
- Network isolation (VLANs, firewalls, virtual private clouds).
- Monitoring, logging and incident response; regular audits.

Trust

Trust is the customer's confidence that the provider will behave as promised. It is built through transparency, SLAs, independent certifications (ISO 27001, SOC 2), reputation, and audit rights.

Privacy

Privacy concerns how personal data is collected, stored, shared and deleted. Issues include data location (jurisdiction), unauthorised access by provider staff or governments, data remnants after deletion, and regulations such as GDPR.

Example 2.2 — Privacy issue

A Nepali company stores customer data in a cloud region located in another country. Which concerns arise?

Solution

Data residency / jurisdiction — foreign laws may allow access to the data.

Compliance — national data-protection rules may restrict cross-border transfer.

Mitigation: choose an approved region, encrypt with customer-managed keys, sign a data-processing agreement.

Threat	Countermeasure
Data breach	Encryption, access control, key management
Insecure APIs	API authentication, rate limiting, input validation
Insider threat	Least privilege, logging, separation of duties
Multi-tenant leakage	Strong isolation, hardened hypervisors
Denial of Service	Load balancing, traffic filtering, auto-scaling

Key Takeaways — Unit 2

- ✓ Reference model: IaaS (infrastructure) → PaaS (platform) → SaaS (software); control decreases as you move up.
- ✓ Deployment models: public (shared, provider-owned), private (single organisation), community (shared by a group), hybrid (mix, cloud bursting).
- ✓ SOA builds cloud systems from loosely coupled, reusable services described with WSDL, discovered via UDDI and invoked using SOAP/REST.
- ✓ Security uses the CIA triad; trust comes from SLAs, audits and certifications; privacy concerns data location and regulation.
- ✓ Security in the cloud is a shared responsibility between provider and customer.

— End of Unit 2 —