Cloud Tower: A Beginner’s Guide to Cloud Computing Layers

1. What Is the Cloud Tower Model?

In this guide, we use a “Cloud Tower” with nine floors to explain the main layers of modern computing.
You can imagine you are building and running an application on top of a stack of invisible layers.
Each layer has a specific job. In traditional on-premises IT, you (or your company) manage almost all of these layers. In the cloud, some layers are still your responsibility, and some are handled by the cloud provider (like AWS).

The nine layers we’ll use in this document and in the game are:

1. Networking
2. Servers
3. Virtualization
4. Operating System (OS)
5. Runtime
6. Middleware
7. Data
8. Storage
9. Applications

You can think of it like building a restaurant:

- Networking = the roads and delivery routes leading to your restaurant
- Servers = the building and physical kitchen equipment
- Virtualization = how you divide the kitchen into separate workstations
- OS = the basic rules and routines that keep the place running (open/close, cleaning)
- Runtime = the tools and environment chefs use (stoves set to a certain temperature, knives, etc.)
- Middleware = the waiters, order screens, and delivery apps that connect kitchen and customers
- Data = your recipes, orders, and customer records
- Storage = the pantry, fridge, and freezer where ingredients and supplies live
- Applications = the actual meals and experience you serve to customers

2. On-Prem vs Cloud: Who Manages What?

In an on-premises setup, your company is responsible for almost all of these layers:

- You buy, install, and maintain servers and storage.
- You configure networking equipment.
- You install and patch operating systems and runtimes.
- You build and run the applications, databases, and middleware on top.

In the cloud, you rent parts of this tower from a provider (for example, AWS). Depending on the
service type, the provider takes care of more or fewer layers:

- IaaS (Infrastructure as a Service): You still manage OS, runtime, data, and apps. The provider manages servers, storage, and physical networking.
- PaaS (Platform as a Service): The provider also manages the OS and runtime. You focus on apps and data.
- SaaS (Software as a Service): The provider manages almost everything. You mostly manage how you use the application and the data you put into it.

3. Layer-by-Layer Breakdown

3.1 Networking

What it is:
- All the connections that let users and systems reach your app: IP addresses, subnets, firewalls, load balancers, VPNs.
- In AWS terms: you can think of concepts like Virtual Private Cloud (VPC), subnets, security groups, and load balancers.

On-Prem:
- You buy and manage routers, switches, firewalls, and physical links.
- You design IP ranges, VLANs, and security rules.
- If something goes wrong, your team has to diagnose and fix it.

Cloud:
- The cloud provider runs the physical network.
- You define virtual networks and rules with software (for example, creating a VPC, setting inbound/outbound rules).
- It is easier to create isolated environments and connect services securely.

Key ideas:
- Latency (how fast traffic reaches your app).
- Security (who is allowed to connect).
- Global reach (how users from different regions are served).

How it shows up in the game:

- For a viral social app, you should pick cloud networking with things like VPC + load balancers + edge delivery.
- For a banking app, you should emphasize private networks, VPNs, and strict access.

3.2 Servers

What it is:
- Physical machines that provide CPU, memory (RAM), and disks where workloads run.

On-Prem:
- You purchase servers, rack them, power them, cool them, and replace them when they fail.
- You must guess capacity in advance. Buying too much is expensive; buying too little causes outages.

Cloud:
- You rent virtual servers (for example, instances similar to AWS EC2).
- You pay per hour/second instead of buying hardware.
- You can scale up (more power) or out (more instances) as needed.

Key ideas:
- Elasticity (ability to grow and shrink depending on load).
- Cost model (capital expense vs pay-as-you-go).
- Availability (what happens when a server fails).

How it shows up in the game:
- In the social app scenario, cloud servers with auto-scaling are usually the best choice.
- In banking, you may still use well-controlled dedicated servers, but with redundancy.

3.3 Virtualization

What it is:
- A way to run multiple “virtual” machines or containers on a single physical server.
- Common technologies: virtual machines (VMs), containers (like Docker), and orchestrators (like Kubernetes).

On-Prem:
- You run hypervisors and virtualization platforms on your servers.
- You must manage placement, capacity, and isolation.

Cloud:
- Virtualization is built into the platform.
- You can start and stop VMs or containers with APIs.
- Managed container services reduce operational work.

Key ideas:

- Isolation between services.
- Efficient use of hardware.
- Flexibility to run different apps on the same infrastructure.

How it shows up in the game:
- A good answer usually uses VMs/containers with a load balancer and avoids packing everything into one process.

3.4 Operating System (OS)

What it is:
- The base system software (such as Linux or Windows) that manages hardware and runs your programs.

On-Prem:
- You install, patch, and secure OSes yourself.
- You must track versions, updates, and security patches.

Cloud:
- You still choose OS images, but cloud tools help you patch and update at scale.
- In more managed services, the OS is completely hidden from you.

Key ideas:
- Security updates.
- Consistency across instances.
- Hardened images and controlled access.

How it shows up in the game:
- Good options use standard, hardened images and automated patching.
- Weak options rely on never patching or manually changing OS settings on each server.

3.5 Runtime

What it is:
- The language or platform environment where your code runs (Python, Java, Node.js, .NET, etc.).

On-Prem:
- You install runtimes manually and must keep them consistent across servers.
- You are responsible for updating them and ensuring compatibility.

Cloud:
- You can still manage runtimes yourself, or use managed runtimes/serverless platforms.
- In serverless platforms, the provider runs and patches the runtime; you only upload code.

Key ideas:
- Consistency: same runtime version everywhere.
- Security: patched and supported versions.
- Scalability: easier to add more instances with the same runtime.

How it shows up in the game:
- The best answers often use managed runtimes or containers with pinned versions.
- Bad answers use random versions or unmaintained runtimes.

3.6 Middleware

What it is:
- The “glue” that connects parts of your system: APIs, message queues, event buses, API gateways, and service meshes.
- Helps services talk to each other in a reliable, secure, and decoupled way.

On-Prem:
- You host and manage message brokers, API gateways, and integration tools.
- Scaling and keeping them reliable is your job.

Cloud:
- Managed messaging services and API gateways are common.
- You get dashboards, metrics, and scaling help from the provider.

Key ideas:
- Decoupling (services don’t need to know everything about each other).
- Reliability (dampening spikes, retrying).
- Observability (logs, metrics, traces of interactions).

How it shows up in the game:
- Good choices use APIs and queues/streams to buffer spikes and avoid tight coupling.
- Poor choices tie every service together directly or rely on manual data passing.

3.7 Data

What it is:
- Structured information such as customer records, account balances, posts, and transactions.
- Represented in databases (SQL, NoSQL) with schemas, relations, and queries.

On-Prem:
- You install and manage database servers.
- You handle backups, replication, and tuning.

Cloud:
- Managed database services handle backups, replication, and many operational tasks.
- You still design schema, queries, and data access patterns.

Key ideas:
- Consistency and correctness.
- Availability and performance.
- Backups, disaster recovery, and compliance.

How it shows up in the game:
- Good answers use managed databases with replication and backups.
- Bad answers use fragile setups like single on-prem DBs without redundancy or ad-hoc files.

3.8 Storage

What it is:
- Where large files and unstructured content live: logs, images, videos, documents, backups.

On-Prem:
- You handle NAS devices, SANs, local disks, and backup systems.
- You must plan capacity, redundancy, and offsite copies.

Cloud:
- Object storage services store massive amounts of data with high durability.
- Lifecycle rules can automatically move older data to cheaper storage tiers.

Key ideas:
- Durability (how likely you are to lose data).
- Scalability (can it grow with your needs).
- Cost over time (tiering and lifecycle).

How it shows up in the game:
- Correct options use scalable cloud storage for large content.
- Wrong options store everything on a single disk or in a single database.

3.9 Application Layer

What it is:
- The code and experience you deliver to users: mobile apps, web apps, APIs, and internal tools.
- The “top floor” of the tower that sits on all other layers.

On-Prem and Cloud:

- You almost always own this layer.
- You design architecture (monolith, microservices, serverless), UX, and business logic.
- In cloud, your app can take advantage of elasticity, managed services, and global reach.

Key ideas:
- Structure: monolith vs microservices vs serverless functions.
- Deployment strategy: rolling updates, blue/green deployments.
- Security and access control: authentication, authorization, session management.

How it shows up in the game:
- Good answers design apps that can scale, update safely, and enforce security rules.
- Poor answers create single points of failure or unsafe access patterns.

4. The Shared Responsibility Model

Cloud computing is not “the provider does everything.” Instead, it is a shared responsibility model:

As a simple rule of thumb:

- You always own:
  - Application code
  - Business logic
  - Data correctness and classification
  - Who can access what (permissions and roles)

- The cloud provider increasingly owns:
  - Physical data centers and networking
  - Physical servers and hardware
  - Virtualization layer
  - Parts of OS, runtime, and middleware (depending on the service type)

In our Cloud Tower game:

- Your choices represent how much of each layer you manage yourself.
- Cloud-heavy choices offload more layers to the provider.
- On-prem and hybrid choices keep more control (and more work) with you.

5. How to Use This Guide with the Game

When you play the Cloud Tower game:

- Each floor corresponds to one of the nine layers in this guide.
- Each question presents a small real-world decision at that layer.
- A “crash” is not a failure; it is a teaching moment that shows what can go wrong.

Suggested learning approach:

1. Read through this guide once to get an overview.
2. Play the game as the **Viral Social App** scenario.
   - Focus on scalability, global networking, and elastic storage.
3. Play again as the **Secure Banking Portal** scenario.
   - Focus on security, compliance, and controlled environments.
4. After each crash in the game, come back to the corresponding section:
   - Networking crash → re-read Networking section.
   - Data crash → re-read Data section.
   - Apps crash → re-read Application Layer section.

6. Summary: Why the Cloud Tower Matters

The Cloud Tower model helps you:

- Break down “cloud computing” into understandable layers.
- See that cloud is more than just “someone else’s server.”
- Understand which decisions belong to you and which belong to the provider.
- Practice making trade-offs in different scenarios (social media vs banking).

If you understand the nine layers and how they differ between on-prem, cloud, and hybrid environments, you already have a strong foundation in cloud computing fundamentals.

This is exactly what the Cloud Tower: Architect’s Quest game is designed to teach you in an interactive, scenario-driven way.