

Cloud Deployment Models in Nepal's Real-World Scenarios

In each case below we match the most suitable **cloud deployment model** to the scenario's requirements (scalability, cost, security, collaboration) and explain **why**. This document also outlines the high-level architecture (availability zones, replication, failover, etc.), list the advantages, and note the limitations of that model. All claims are backed by industry sources.

1. eSewa / Khalti – Festival Transaction Surge

Chosen Model: Public Cloud (e.g. AWS, Azure, Google Cloud)

Why Public Cloud? During festivals (Dashain/Tihar) digital wallets see 2–3× spikes in transactions in short bursts. Public cloud's **rapid elasticity** and pay-as-you-go pricing make it ideal: the system can auto-scale on demand to meet peaks, then shrink back (no idle hardware cost). Public cloud providers also offer built-in high availability (HA) via multiple **Availability Zones (AZs)** or regions, reducing downtime risk. For example, a load-balanced cluster can span AZ-1, AZ-2, AZ-3; if one AZ fails, traffic shifts to others automatically. Security and maintenance are mostly handled by the provider, which lets eSewa focus on its application logic, not on data center ops.

Architecture & Scalability: A typical design on AWS/Azure might include:

- **Stateless App Servers (EC2/Azure VMs or containers)** behind a global *Load Balancer*. Auto Scaling Groups spin up new instances as CPU/queue thresholds are met. (Think “open more supermarket checkouts when queues form.”)
- **Microservices & Queues:** Services decouple via message queues (e.g. AWS SQS/Kafka) so that spikes queue up without dropping requests.
- **Distributed Databases:** A primary database (RDS/Azure SQL) in one AZ, with multi-AZ or Read-Replica instances in other AZs for HA and read scaling. If primary fails, an automated failover to a replica can occur in seconds.
- **Caching & CDN:** An in-memory cache (Redis or Memcached) handles frequent reads (e.g. user sessions), and CDNs (CloudFront/Azure CDN) serve static content. This offloads work from back-end servers.
- **Durable Storage:** Transaction logs and backups go into cloud object storage (S3/Azure Blob) replicated across zones or regions, ensuring no data loss if a zone dies.

This design lets the system **burst** to hundreds of servers instantly during Dashain. For instance, AWS's Auto Scaling can spin up new EC2 instances in minutes to handle surges[2]. After the festival, resources scale down automatically, saving cost. Industry experience (e.g. AliPay handling 544k TPS on Singles' Day) shows elastic cloud infrastructure is the key to surviving transient peaks.

Advantage: *Extreme Scalability & Cost Efficiency.* Pay-per-use means eSewa only pays extra during peaks. The public cloud's multi-AZ support ensures **high availability** (HA) – e.g. replicating VMs and databases across zones provides near-zero downtime. Built-in services (managed DB, caching, load balancers) speed up development and ops.

Limitation: *Security Concerns & Vendor Dependency.* In a public cloud, data resides on shared hardware managed by the provider. While cloud security is strong, regulators or users may worry about data sovereignty. Also, if the cloud provider has an outage (a rare AWS downtime or region failure), eSewa has no direct control to fix it – it must rely on the provider's SLAs. Finally, costs can balloon if scaling is not managed.

Fortinet – *“Public clouds provide advantages like scalability and cost efficiency” and “resources are shared across organizations...scale services on demand”.*

2. Nepal Rastra Bank – Secure Financial Data

Chosen Model: Private Cloud (or on-premises cloud)

Why Private Cloud? NRB manages highly sensitive, regulated financial data (interbank settlements, reserves, etc.). Security, compliance and data control are paramount. A private cloud – a dedicated infrastructure solely for NRB – gives *absolute control* over servers, networks and physical security. This ensures data never leaves government premises (data residency) and can meet strict mandates (central bank rules, PCI-DSS, etc.). Unlike public multi-tenant clouds, there is no “noisy neighbor” or sharing.

Architecture & Compliance: The private cloud would be built as follows:

- **Dedicated Data Centers:** NRB would run its own secure DCs (or a hosted private cloud) with redundant power, cooling, and network. They can define **logical AZs** by using separate racks or halls. Each core system (settlements, payments, anti-fraud) runs on VMs or containers under a hypervisor (e.g. VMware or OpenStack), all on private networks.
- **Active–Passive DR Setup:** One site (Kathmandu) is active, another (e.g. Pokhara) is a hot standby. Data is **synchronously replicated** from active to DR site over a private link so that failover yields minimal data loss. Disaster tests ensure RTO (Recovery Time Objective) and RPO (Recovery Point Objective) meet strict targets.
- **High Availability Controls:** Even on private hardware, NRB can implement HA patterns: clustered databases, redundant load balancers, and automated failover within its cloud. Core banking and settlement systems would mirror across zones in the private cloud.
- **Isolated Networks:** There is no public internet exposure. Banks connect via secure MPLS/VPN lines into NRB's cloud. Strict firewalls/DMZs control any external traffic. Only authorized personnel and services (e.g. RTGS system) have access.

Because NRB “owns” the stack, they can certify it under banking regulations. Data never touches a third-party’s multi-tenant platform, which meets the **data privacy and sovereignty** requirements.

Advantage: *Maximum Security and Compliance.* Private clouds allow NRB to implement the strictest security measures and audits. It ensures “military-grade” control – e.g. hardware encryption, custom IAM, direct vetting of all software. There’s no sharing of infrastructure, so risk of outside data breach is minimized. Uptime is still high via redundant setups across internal zones. NRB can meet local regulations easily (e.g. keep all transaction data in Nepal).

Limitation: *High Cost & Limited Elasticity.* Building and maintaining one’s own cloud is very expensive – capital expenses for servers, plus staff to run it. This model **does not scale instantly**. NRB must provision enough capacity for peak load, which sits idle off-peak. Unexpected surges (e.g. electronic bond auctions or payment crises) could overwhelm static capacity without rapid cloud-like elasticity. Upgrades also require lengthy procurement cycles.

3. Nagarik App – Sensitive & Public Data Mix

Chosen Model: Hybrid Cloud

Why Hybrid? Nagarik App hosts a mix of public content (service info, announcements) and highly sensitive citizen data (IDs, education records). A hybrid cloud allows splitting the workload: keep sensitive data in a **private** part of the cloud while exposing public-facing services on a **public** cloud. This achieves both **security** and **scalability**. Sensitive modules (national IDs, PAN data) stay on a government-controlled private infrastructure; scalable user-facing modules (login portal, dashboards, public info) run on AWS/Azure to handle millions of users (leveraging auto-scaling). This balance is often chosen by governments to “have their cake and eat it” – meeting compliance without giving up cloud agility.

Architecture & Data Flow:

- **Segmented Components:** All truly confidential databases (e.g. NID/PAN backends) reside on a private cloud or government data center. These services expose secure APIs only callable over a VPN/Direct-Connect link. The public side (hosted on AWS/Azure) contains the application logic, authentication front-end, and static content.
- **API Gateway and Auth:** A secure API gateway on the public side handles login and routing. Upon login, it issues a short-lived token and forwards requests to either the public (open data) or private (sensitive data) backend as needed. For example, “*Check NID*” calls travel through the gateway into the private cloud, fetch data, and return.
- **Private Connectivity:** A dedicated encrypted link (VPN or Azure ExpressRoute) connects the two clouds. Data flows only as needed; nothing is publicly exposed.
- **Caching and Edge Services:** To reduce load on private data stores, public cloud side can cache frequently accessed info (e.g. citizenship status flags, once verified). Content Delivery Networks can accelerate public content to citizens.

- **High Availability:** The public cloud portion still uses multi-AZ deployment for HA and scale. The private cloud portion can use an on-prem Active-Passive setup for resilience.

This hybrid design “federates” government services: Nagarik App effectively becomes a **central gateway** linking multiple agencies. It gains cloud’s elasticity for user load, yet keeps citizen records locked down. For example, AWS Outposts or Azure Stack could even be used to physically co-locate a mini-cloud in Nepal for low-latency private workloads while bursting to public cloud when needed.

Advantage: *Security + Scalability.* Nagarik keeps sensitive data off the internet (private cloud) while still serving millions of users on demand (public cloud). It also supports cross-government collaboration: different agencies connect into the common hybrid framework. The model is well-suited for digital-ID projects worldwide.

Limitation: *Operational Complexity.* Hybrid clouds are complex to manage. Nagarik’s IT team must orchestrate two environments, ensure synchronization, and maintain the secure tunnel. There is more DevOps overhead (e.g. managing VPN links, monitoring cross-cloud APIs, ensuring data consistency). Latency and troubleshooting are harder than single-cloud solutions.

4. Universities (TU, KU, PU) – Shared Library & LMS

Chosen Model: Community Cloud (Education sector cloud)

Why Community Cloud? Multiple universities want to **collaborate and share costs** on a common e-learning platform. A community cloud – a cloud shared by organizations with similar goals – is ideal. They can pool resources and split costs, yet restrict access to only students/staff (unlike a public cloud). This is more secure (and often cheaper) than each uni building its own private cloud, while still catering to their academic and regulatory needs. Essentially, it’s like a consortium-run cloud: owned or governed by the group of universities (or a trusted third party on their behalf).

Architecture & Collaboration:

- **Shared Infrastructure:** The universities jointly provision a cloud (could be on premises or with a specialized provider) that hosts common services: a central LMS (e.g. Moodle/Canvas), digital libraries, research databases, etc. Each institution has a virtual “slice” (tenant) with isolated networks and storage.
- **Access Controls:** Single Sign-On (SSO) federates login across all institutions. Students log in with their university ID but land on the community platform. Role-based access ensures each uni’s data is kept separate if needed.
- **Workload Management:** During exam periods, the LMS usage spikes. The cloud scales (vertically/horizontally) within its fixed capacity limits. Administrators can set usage quotas per uni to prevent one university’s load (e.g. heavy exam traffic) from crippling others.
- **Cost and Governance:** All members agree on a cost-sharing and governance model. For example, TU, KU, PU each contribute hardware and share maintenance. This spreads capital and operational expense.

During COVID-19, many universities worldwide leveraged such community-like clouds for online learning. By collaborating, Nepali universities can also improve security (limited to accredited users) and share best practices. In effect, they get “better than public cloud security” since it’s a restricted community, yet much lower cost than building three separate data centers.

Advantage: *Collaboration & Cost-Sharing.* Community clouds are explicitly designed for shared goals. This model allows TU/KU/PU to pool budgets and expertise, getting economies of scale. It also provides more security than public clouds (restricted membership), and standardized compliance (like academic data privacy).

Limitation: *Resource Contention.* Since all institutions share the same pool of servers/storage, peak demands by one (e.g. exam result release) can affect others if not managed. Capacity is finite – if growth outpaces the pooled investment, performance may suffer unless the community upgrades. Also, decision-making (governance) can be slower, as all institutions must agree on changes.

5. Pathao / Tootle – Real-Time GPS & User Growth

Chosen Model: Public Cloud

Why Public Cloud? Ride-hailing apps must handle **real-time demands** (GPS tracking, rapid matching) and **rapidly growing user bases**. A public cloud’s elastic infrastructure is perfect for spiky, latency-sensitive workloads. Startups benefit from no/low upfront cost and the ability to scale literally from tens of servers to thousands in minutes. Unlike fixed in-house hardware (which could be underutilized most of the time), a public cloud lets Pathao/Tootle pay only for what they use – critical for cost-sensitive startups.

Architecture & Real-Time Flow:

- **Geo-Distributed Services:** Riders and drivers in Kathmandu connect to regional edge servers (maybe multiple AZs). For example, Eastern Kathmandu traffic could go to AZ-1, Western to AZ-2, reducing latency.
- **Event-Driven Backend:** Driver GPS updates stream into a message queue (Kafka/Kinesis). Microservices consume these streams: one computes ride-matches, another updates live map positions, another handles payments.
- **In-Memory Databases:** A fast in-memory store (e.g. Redis Cluster) keeps current locations and active ride data. This ensures sub-second responses. The main database (for trip history, billing) can be a managed DB scaled by replica.
- **Auto-Scaling:** The rider-matching and background services auto-scale: during rush hour or rain, incoming requests jump and trigger more service instances. When traffic subsides, instances terminate.
- **Real-Time Monitoring:** Cloud monitoring (CloudWatch/Stackdriver) and logging alert engineers if latency or error rates spike.

This enables Pathao to handle surges – e.g. during morning peak, demand may 5× normal. Public cloud can spin up needed containers/VMs instantly. Many ridesharing platforms globally use this approach (Uber’s backend on AWS, for example). The **result** is minimal wait times and no user-visible downtime, even during rapid growth.

Advantage: *On-Demand Scalability for Real-Time Workloads.* Public cloud auto-scaling matches Pathao's need to handle sudden ride request spikes (like during rains or festivals) without upfront investment. It also offers global CDN and map/data APIs. The low entry cost is crucial for startups to survive early stages.

Limitation: *Cost Overrun & Privacy Risks.* If traffic keeps growing, monthly bills can balloon unexpectedly. The pay-as-you-go model can become costly if not monitored – e.g. auto-scaling triggers many instances and data egress fees accumulate. Moreover, storing live location data on a third-party cloud raises privacy concerns: sensitive user GPS traces sit on someone else's servers, which might worry users/regulators about tracking abuse (this is an industry-wide concern for all ride-hail apps).

6. Ncell / NTC – Telecom DR & Uptime

Chosen Model: Hybrid/Multi-Cloud (Public Cloud DR and Multi-Region)

Why Hybrid/Multi-Cloud? Telecom networks (Ncell, NTC) are **critical infrastructure**: any outage affects millions, so they require **continuous availability and disaster recovery**. A pure on-premises DR solution (multiple data centers) is extremely costly. Instead, telecoms often use a **hybrid or multi-cloud architecture** for resilience. For example, core telecom services can run on private network, but backups and failover can leverage one or more public clouds (or different public clouds). This multi-cloud redundancy means if one provider or region fails (due to earthquake or outage), operations shift to another cloud instantly.

Architecture & Continuity:

- **Active-Active Multi-Cloud:** Critical systems are deployed simultaneously in two or more clouds (e.g. AWS + GCP). Data is geo-replicated among them. If one cloud's region goes down, traffic is rerouted to the other.
- **DR-as-a-Service:** Telecoms use cloud-based DR services (DRaaS) which continuously replicate on-prem databases (billing, user data) to cloud storage. In a failure, they spin up VMs in the cloud to take over with minimal delay.
- **Edge Continuity:** Edge functions (like SMS, call routing) can also mirror in multiple locations. DNS-based load balancing sends users to healthy networks.
- **Hybrid Setup:** They may keep latency-sensitive traffic local (private) but mirror backups in cloud. Hybrid links ensure seamless failover between on-prem and cloud.

Industry reports on telecom DR highlight that **multi-cloud/distributed architectures** greatly improve resilience. For example, multi-cloud DR ensures “operations continue regardless of location failures”. Telecom providers specifically build redundancy at multiple levels (diverse fiber paths, multiple datacenters, plus cloud backups) to meet 24/7 targets.

Advantage: *Maximum Uptime via Diversification.* Using multiple clouds and regions virtually eliminates single points of failure. It also shifts heavy DR costs from capital expenses (own DC) to operational (cloud usage). Telecoms can meet very low RTO/RPO by spinning up services in the cloud instantly after a disaster.

Limitation: *Complexity & Vendor Coordination.* Multi-cloud setups are complex: requiring specialized tools and expertise to synchronize data across providers and

switch over smoothly. Network latency between clouds can complicate real-time state replication. Additionally, costs can be unpredictable due to data transfer between clouds. And achieving compliance in multiple cloud jurisdictions needs careful planning.