**📘 Module 1: Cloud 101 – Core Concepts**

**TPM-Focused Study Guide**

**🎯 Objective**

Help TPMs understand the foundational cloud concepts, their business relevance, and how to manage cloud-based programs with clarity, stakeholder alignment, and delivery precision.

**📍 Section 1: What is the Cloud?**

**🔹 Definition**

**Cloud computing** is the on-demand delivery of IT resources (compute, storage, databases, networking, etc.) over the internet with pay-as-you-go pricing.

**🔹 TPM Angle: Why It Matters**

* Helps you **scope** and **plan** projects accurately by understanding what "cloud-native" means.
* Allows **technical risk identification** early (e.g., latency, region availability, compliance).
* Aids in **communication** with engineering teams and stakeholders using correct terminology.

**🔹 Key Characteristics**

| **Feature** | **Description** | **TPM Relevance** |
| --- | --- | --- |
| On-demand self-service | Provision resources without human intervention | Speeds up prototyping and delivery |
| Broad network access | Services accessible via internet | Enables distributed teams, cross-region delivery |
| Resource pooling | Shared infrastructure | Must assess impact of multi-tenancy, security |
| Rapid elasticity | Scale up/down as needed | Plan for capacity spikes or variable demand |
| Measured service | Pay-as-you-go | Helps in cost tracking and budget management |

**📍 Section 2: Cloud Service Models – IaaS, PaaS, SaaS**

**🔹 Infrastructure as a Service (IaaS)**

* **What**: Provides virtual machines, networking, storage
* **Examples**: AWS EC2, Azure VM, GCP Compute Engine
* **TPM Angle**:
  + Works closely with infra and ops teams
  + Often required for **custom architectures** or lift-and-shift migrations
  + TPMs manage provisioning, configuration, and scale-up rollouts

**🔹 Platform as a Service (PaaS)**

* **What**: Provides platform to build/deploy apps without managing infrastructure
* **Examples**: AWS Elastic Beanstalk, Azure App Services, GCP App Engine
* **TPM Angle**:
  + Focus on **accelerating developer velocity**
  + TPMs facilitate environment setup, integration workflows, and CI/CD planning

**🔹 Software as a Service (SaaS)**

* **What**: Ready-to-use apps delivered over the web
* **Examples**: Google Workspace, Microsoft 365, Salesforce
* **TPM Angle**:
  + Leads **vendor onboarding**, **SLA tracking**, and **data privacy compliance**
  + Coordinates with legal/security for assessments (DPA, BAA, etc.)

**🔸 Visual Reference**

vbnet

CopyEdit

SaaS: "I use it" → Google Docs

PaaS: "I build on it" → Heroku

IaaS: "I manage it" → EC2

**📍 Section 3: Major Cloud Providers**

**🔹 AWS (Amazon Web Services)**

* Market leader
* Rich ecosystem (200+ services)
* **TPM Note**: Often default choice for startups and enterprises → TPMs must understand IAM, VPC, and billing dashboards.

**🔹 Microsoft Azure**

* Strong in hybrid cloud and enterprise services
* Tight integration with Microsoft ecosystem
* **TPM Note**: Likely to be chosen if client uses Active Directory or Microsoft stack

**🔹 Google Cloud Platform (GCP)**

* Strength in data, ML/AI, Kubernetes
* Focus on open-source and cloud-native tooling
* **TPM Note**: Great for data-intensive projects; know BigQuery, GKE

**🔸 TPM-Specific Evaluation Table**

| **Provider** | **Strength** | **TPM Program Focus** |
| --- | --- | --- |
| AWS | Breadth & maturity | Cost governance, VPC design, tagging |
| Azure | Enterprise alignment | Identity, compliance, hybrid solutions |
| GCP | Analytics, AI/ML | Data engineering programs, automation |

**📍 Section 4: Global Infrastructure – Regions, Availability Zones, Edge Locations**

**🔹 Region**

* A **geographic area** containing multiple isolated data centers (AZs)
* Example: us-east-1, asia-south1

**🔹 Availability Zone (AZ)**

* An **isolated DC** within a region; designed for fault tolerance
* Example: us-east-1a, us-east-1b

**🔹 Edge Locations / POPs**

* Content caching locations via CDN (CloudFront, etc.)
* Useful for **latency-sensitive applications**

**🔹 TPM Considerations**

| **Topic** | **TPM Focus** |
| --- | --- |
| **High Availability** | Use multi-AZ designs; track redundancy requirements |
| **Latency & Performance** | Select region closest to users; monitor SLAs |
| **Disaster Recovery** | Plan for cross-region failover setups |
| **Compliance & Data Residency** | Work with legal/security to choose correct regions (e.g., India, EU) |

**🔸 Visual Breakdown**

scss

CopyEdit

🌍 Region (India: Asia-South1)

├── AZ1 (Mumbai-1a)

├── AZ2 (Mumbai-1b)

├── AZ3 (Mumbai-1c)

**🧠 TPM Knowledge Nuggets**

| **Topic** | **TPM Checklist** |
| --- | --- |
| Cloud Selection | Have you evaluated fit vs. cost vs. capability? |
| Service Model | Are you aware which layer you're working at? |
| Region Selection | Is the region chosen optimal for users & legal constraints? |
| SLA & Resilience | Have you accounted for HA, backups, and RTO/RPO? |
| Stakeholder Clarity | Can you explain these terms in simple language to business stakeholders? |

**📌 Summary for TPMs**

| **Cloud Concept** | **TPM Role** |
| --- | --- |
| Understand cloud layers | Define program scope and risks |
| Choose right provider | Align with org architecture strategy |
| Navigate infra design | Communicate trade-offs to stakeholders |
| Support compliance & DR | Enable security/legal alignment |
| Track costs | Create guardrails and budgets |

**Module 2: Key Cloud Services – TPM-Focused Study Guide**

**🎯 Objective**

Equip TPMs with foundational knowledge of core cloud services in **compute, storage, and databases** across AWS, GCP, and Azure. Enable them to drive architecture discussions, ensure SLA compliance, and manage cost-performance trade-offs.

**📍 Section 1: Compute Services**

**🔹 Purpose**

Compute services provide the virtual resources (CPU, RAM) to run applications, workloads, APIs, and batch jobs.

**🟢 1.1 Google Compute Engine (GCE) – GCP**

* **What**: Infrastructure-as-a-Service (IaaS) that provides VMs in GCP
* **Use Cases**: Lift-and-shift apps, custom OS requirements, legacy apps
* **TPM Focus**:
  + Involved in **resource sizing** and **instance type decisions**
  + Ensure **auto-scaling** and **load balancing** policies are in place
  + Track **cost spikes** using GCP Billing + Quotas

**🟢 1.2 App Engine – GCP**

* **What**: Fully managed Platform-as-a-Service (PaaS) for app deployment
* **Use Cases**: Modern web/mobile apps, APIs, microservices
* **TPM Focus**:
  + Emphasize **developer agility** and quick delivery timelines
  + Handle **CI/CD pipeline integrations**
  + Track **SLA & rollback strategy** across releases

**🔵 1.3 Azure App Service – Azure (similar to App Engine)**

* **What**: PaaS for web apps and APIs
* **TPM Focus**:
  + Focus on **integration with Azure DevOps**
  + Check if **auto-deploy, staging slots, and monitoring hooks** are used
  + Validate **compliance scope** when apps handle PII or financial data

**✅ TPM Best Practices for Compute**

| **Best Practice** | **Description** |
| --- | --- |
| Use managed services where possible | Reduce operational overhead and simplify rollouts |
| Standardize instance types | Align with FinOps and automation teams for cost-efficiency |
| Define autoscaling thresholds | Prevent outages due to under-provisioning |
| Set quotas & alerting | Prevent unexpected cost overruns |
| Confirm HA & DR zones | Always deploy in multi-AZ or multi-region if critical |

**📍 Section 2: Cloud Storage Services**

**🔹 Purpose**

Object storage for unstructured data, backups, media, logs, and archives. Focused on durability, availability, and cost optimization.

**🟢 2.1 Google Cloud Storage (GCS) – GCP**

* **What**: Object storage system for blobs (files, images, archives)
* **Features**: Lifecycle rules, versioning, multiple classes (Standard, Nearline, Coldline)
* **TPM Focus**:
  + Drive **data classification** for cost efficiency
  + Ensure **versioning & encryption policies** are aligned with compliance
  + Coordinate with data governance on **retention policies**

**🔵 2.2 Azure Blob Storage – Azure**

* **What**: Microsoft's object storage platform
* **Use Cases**: Logs, images, backups, ML datasets
* **TPM Focus**:
  + Work with teams to tag and classify blobs correctly
  + Ensure **access tiers** (Hot/Cool/Archive) match usage patterns
  + Track **access control & integration with RBAC**

**✅ TPM Best Practices for Storage**

| **Best Practice** | **Description** |
| --- | --- |
| Define naming conventions | Helps in access control, logging, and lifecycle rules |
| Apply storage class policies | Match cost vs. frequency of access |
| Enforce encryption | Enable customer-managed keys if required |
| Enable lifecycle rules | Auto-archive or delete stale data |
| Monitor egress cost | TPMs must track data transfer costs across services/regions |

**📍 Section 3: Cloud Database Services**

**🔹 Purpose**

Managed database services eliminate the need to handle backups, patching, and scaling manually. Options include relational and NoSQL.

**🟢 3.1 Amazon RDS (Relational Database Service) – AWS**

* **What**: Managed SQL databases (MySQL, PostgreSQL, Oracle, etc.)
* **Use Cases**: OLTP apps, internal tools, ERP, CRM
* **TPM Focus**:
  + Align with DBAs on **RTO/RPO and snapshot policies**
  + Monitor **failover architecture** (multi-AZ or cross-region)
  + Ensure **SLA requirements are well documented**

**🟢 3.2 Amazon DynamoDB – AWS**

* **What**: Managed NoSQL database (key-value + document)
* **Use Cases**: Real-time data, serverless apps, IoT, gaming
* **TPM Focus**:
  + Validate **provisioned vs. on-demand throughput models**
  + Align with architects for **indexing strategy and data modeling**
  + Include **autoscaling policies** in delivery timelines

**🟢 3.3 Cloud SQL – GCP**

* **What**: Managed SQL service (MySQL/PostgreSQL/SQL Server)
* **Use Cases**: Line-of-business apps, internal portals
* **TPM Focus**:
  + Track **backups, replication setup**
  + Handle **data migration timelines** and tooling (e.g., DMS)
  + Ensure **IAM & VPC security rules** are followed

**✅ TPM Best Practices for Databases**

| **Best Practice** | **Description** |
| --- | --- |
| Choose right DB type | Work with architects to avoid mismatches (e.g., using SQL for a NoSQL use case) |
| Confirm backup frequency | Align with compliance and recovery policies |
| Enable multi-AZ or replicas | Critical for HA and DR |
| Plan migration windows | Downtime must be communicated early |
| Review IAM & data masking | TPMs must push for data protection and privacy compliance |

**📌 Summary Table – TPM Snapshot**

| **Service Category** | **Example Services** | **TPM Role** |
| --- | --- | --- |
| Compute | GCE, App Engine, Azure App Service | Manage provisioning, scaling, release coordination |
| Storage | GCS, Azure Blob | Govern access, cost controls, data lifecycle |
| Database | RDS, DynamoDB, Cloud SQL | Handle backup, compliance, downtime windows |

**🧠 TPM Knowledge Nuggets**

* **Avoid over-architecting**: Choose managed services where possible.
* **Work with FinOps**: Compute/storage costs scale fast → tagging and budgeting matter.
* **Track SLAs**: Document SLAs per service class and ensure they’re reflected in program planning.
* **Secure early**: TPMs must align with security/legal on encryption, data masking, and retention.
* **Prioritize observability**: Always ensure monitoring, logging, and alerting are in-scope.

**Module 3: Cloud-Native vs On-Prem vs Hybrid**

**🧭 TPM-Focused Study Guide + Hands-on Lab**

**🎯 Objective**

Enable TPMs to understand different cloud deployment models, when to use each, their trade-offs, cost considerations, and how to design a basic web application architecture in a public cloud.

**📍 Section 1: Cloud-Native vs On-Prem vs Hybrid – What’s the Difference?**

| **Model** | **Description** | **Core Idea** |
| --- | --- | --- |
| Cloud-Native | Built specifically for the cloud using cloud-first services | Scale fast, automate, pay-as-you-go |
| On-Premises | Deployed within your company’s data centers | Full control but high cost and effort |
| Hybrid Cloud | Combines on-prem and cloud | Balance legacy investments with cloud agility |

**📌 TPM-Focused Comparison Table**

| **Attribute** | **Cloud-Native** | **On-Prem** | **Hybrid** |
| --- | --- | --- | --- |
| **Cost Model** | OpEx (Pay-as-you-go) | CapEx (Upfront investment) | Mixed |
| **Speed** | Fast provisioning, CI/CD | Manual provisioning | Slower due to integration |
| **Scalability** | Auto-scaling, elastic | Hardware-bound | Cloud components can scale |
| **Compliance** | May need config | Total control | Can isolate sensitive workloads |
| **Maintenance** | Minimal (managed services) | Full responsibility | Dual burden |
| **TPM Role** | Manage services, CI/CD, uptime | Coordinate infra, hardware, vendors | Manage integration, data flows, security |

**📍 Section 2: When to Use What – TPM Decision Guide**

**🟢 Cloud-Native: When to Use**

* **Startups and greenfield projects**
* Microservices, mobile/web apps
* Scale-heavy apps (e.g. e-commerce, gaming)
* **TPM Focus**: Fast delivery, DevOps automation, lower infra overhead

**🔵 On-Premises: When to Use**

* Sensitive data workloads (e.g., government, financial institutions)
* Compliance-restricted workloads (e.g., HIPAA, RBI)
* When existing infra already exists and works
* **TPM Focus**: Hardware lifecycles, vendor management, disaster recovery drills

**🟣 Hybrid: When to Use**

* Migration in progress from on-prem to cloud
* Need for local data processing (e.g., IoT, edge use cases)
* Want cloud agility without discarding legacy systems
* **TPM Focus**: Integration planning, latency, network design, data residency

**📍 Section 3: Cloud Cost Considerations**

**🔹 Cost Categories**

| **Type** | **Description** |
| --- | --- |
| **Compute** | VM uptime, container runs, functions (billed per second/minute) |
| **Storage** | Object (GCS), block (EBS), database usage (RDS) |
| **Data Transfer** | Ingress (often free) vs Egress (usually charged) |
| **Licensing** | OS, DB licenses, 3rd-party apps (e.g., Windows, Oracle) |

**🔹 Pricing Models**

| **Model** | **Description** | **TPM Use Case** |
| --- | --- | --- |
| On-demand | Pay per second/minute | Dev/test environments, short-lived workloads |
| Reserved | Commit to 1–3 years | Production workloads with steady usage |
| Spot / Preemptible | Unused capacity at low price | Non-critical, fault-tolerant jobs |
| Serverless | Pay-per-execution | Event-driven or bursty workloads (e.g. Cloud Functions) |

**✅ TPM Best Practices for Cloud Cost Control**

| **Practice** | **Description** |
| --- | --- |
| **Tagging resources** | Enforce cost attribution per team/project |
| **Budgets + Alerts** | Configure cost thresholds to avoid surprise bills |
| **Use cost calculators** | Use GCP/Azure calculators during design stage |
| **Rightsizing** | Regularly adjust VM sizes to match actual usage |
| **Use serverless where possible** | No idle costs, pay only when used |

**📍 Section 4: Architecture Lab – Design a Basic Web App in Cloud**

**🧪 Goal**

TPMs should be able to support or lead discussions on designing a minimal viable web app architecture using public cloud services.

**🌐 Sample Requirements**

* Static front-end (HTML/CSS/JS)
* Backend (API or logic tier)
* Database (SQL/NoSQL)
* Logging & monitoring
* Public access via domain

**🟢 Example: GCP Architecture**

| **Layer** | **Service** | **Purpose** |
| --- | --- | --- |
| Frontend | **GCS + Cloud CDN** | Host static website content |
| Backend | **Cloud Run / App Engine** | Run Python/Node microservice |
| Database | **Cloud SQL** (PostgreSQL) | Store structured user data |
| Auth | **Firebase Auth / IAM** | User auth and identity |
| Observability | **Cloud Logging + Monitoring** | Health checks, alerts |
| Domain | **Cloud DNS** | Map domain to app IP |

**🔵 Example: Azure Architecture**

| **Layer** | **Service** | **Purpose** |
| --- | --- | --- |
| Frontend | **Azure Blob + Azure CDN** | Host static files |
| Backend | **Azure App Service** | App logic |
| Database | **Azure SQL DB** | Relational data |
| Identity | **Azure AD B2C** | Login/authentication |
| Monitoring | **Azure Monitor + App Insights** | Logs, alerts, dashboards |
| Domain | **Azure DNS** | Custom domain resolution |

**🧠 TPM Knowledge Nuggets**

* Always ask: **What is managed vs. self-managed?**
* Be proactive about **network egress charges** in multi-region setups.
* Maintain a **central cost dashboard** for visibility.
* Emphasize **early-stage trade-off analysis**: security, performance, portability, and cost.
* Partner with **FinOps and DevOps** teams from the start of program lifecycle.