**Cloud basics**

**Provision and manage cloud infrastructure as code (IaC).**

**What Are They?**

| **Tool** | **Description** |
| --- | --- |
| **ARM Templates** | JSON-based templates native to Azure for defining infrastructure declaratively. |
| **Azure Bicep** | A newer, cleaner abstraction over ARM templates. It's a domain-specific language (DSL) for Azure. |
| **Terraform** | An open-source, cloud-agnostic IaC tool by HashiCorp that supports multiple cloud providers (not just Azure). |

**Key Differences**

| **Feature/Aspect** | **ARM Templates** | **Bicep** | **Terraform** |
| --- | --- | --- | --- |
| **Syntax** | JSON (verbose) | Bicep DSL (simpler) | HCL (HashiCorp Configuration Language) |
| **Provider Support** | Only Azure | Only Azure | Multi-cloud (Azure, AWS, GCP, etc.) |
| **Ease of Use** | Verbose & hard to manage | Cleaner & more readable | User-friendly, powerful |
| **Modularity** | Limited | Good | Excellent |
| **State Management** | Managed by Azure | Managed by Azure | Local or remote state management |
| **Tooling** | Azure Portal, CLI, VS Code | Same as ARM, better with VS Code | Terraform CLI, VS Code, IDE plugins |
| **Maturity** | Oldest, battle-tested | Newer but rapidly growing | Very mature |
| **Community Support** | Microsoft Docs, smaller | Growing fast | Huge community, lots of modules |

**Cloud Service Models: IaaS, PaaS, SaaS**

**1. SaaS (Software as a Service)**

* **Definition**: Delivers fully functional software applications over the internet.
* **User Responsibility**: Only application usage.
* **Provider Responsibility**: Manages everything else (infrastructure, platforms, updates).
* **Access**: Via web browsers or apps; typically subscription-based.
* **Ideal For**: End-users needing ready-to-use software without technical involvement.
* **Examples**: Salesforce, Microsoft 365, Zoom, Slack.

**2. PaaS (Platform as a Service)**

* **Definition**: Offers a platform with tools to develop, run, and manage applications.
* **User Responsibility**: Application development and data management.
* **Provider Responsibility**: Handles infrastructure, operating systems, and runtime environments.
* **Ideal For**: Developers focusing on app development without managing underlying infrastructure.
* **Examples**: Google App Engine, AWS Elastic Beanstalk.

**3. IaaS (Infrastructure as a Service)**

* **Definition**: Provides virtualized computing resources over the internet.
* **User Responsibility**: Managing applications, data, runtime, middleware, and operating systems.
* **Provider Responsibility**: Manages virtualization, servers, storage, and networking.
* **Ideal For**: Network architects and IT administrators needing control over infrastructure.
* **Examples**: Amazon Web Services (AWS), Microsoft Azure.

**What is NFR (Non-Functional Requirement)?**

Non-Functional Requirements (NFRs) define how a system should behave, not what the system should do.

If functional requirements are the **features** (“What it should do”),

then NFRs are the **qualities** (“How it should perform while doing it”).

**Example:**

Let’s say you're building a **food delivery app** like Swiggy or Zomato.

* **Functional requirement**:

→ “User can place an order.”

* **Non-functional requirement**:

→ “The app should load the menu in under 2 seconds.”

→ “It should handle 10,000 users simultaneously.”

→ “App should be available 99.99% of the time.”

**Common Types of NFRs**

| **Category** | **Description** | **Example** |
| --- | --- | --- |
| **Performance** | Speed, responsiveness | "Page should load in 1s" |
| **Scalability** | Handle growth in users/data | "Support 1M users concurrently" |
| **Availability** | Uptime expectations | "99.99% uptime guaranteed" |
| **Reliability** | System should work consistently | "No crashes during usage" |
| **Security** | Data protection & access control | "Encrypt passwords at rest" |
| **Maintainability** | Easy to update/fix code | "Modular code structure" |
| **Usability** | User-friendly interface | "Users can learn app in 10 mins" |
| **Portability** | Runs on multiple platforms | "Supports Android & iOS" |
| **Compliance** | Follow legal/industry standards | "HIPAA/GDPR compliant" |

**Cloud Deployment Models**

**1. Public Cloud**

* **Definition**: Cloud services offered over the public internet and available to anyone who wants to purchase them.
* **Ownership**: Owned and operated by third-party cloud service providers.
* **Examples**: Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP).
* **Advantages**:
  + Cost-effective due to shared resources.
  + Scalable and flexible to meet varying demands.
  + No need for physical hardware management.
* **Disadvantages**:
  + Less control over security and compliance.
  + Potential for performance variability.

**2. Private Cloud**

* **Definition**: Cloud infrastructure dedicated to a single organization, offering greater control and security.
* **Ownership**: Owned and managed internally or by a third-party exclusively for the organization.
* **Examples**: On-premises data centers, private cloud solutions like OpenStack.
* **Advantages**:
  + Enhanced security and privacy.
  + Greater control over resources and data.
  + Customization to meet specific business needs.
* **Disadvantages**:
  + Higher costs due to dedicated resources.
  + Requires in-house expertise for management.

**3. Hybrid Cloud**

* **Definition**: A combination of public and private clouds, allowing data and applications to be shared between them.
* **Ownership**: A mix of on-premises infrastructure and third-party cloud services.
* **Examples**: Using private cloud for sensitive data and public cloud for less-critical resources.
* **Advantages**:
  + Flexibility to move workloads between cloud solutions as needs change.
  + Optimized existing infrastructure, security, and compliance.
  + Cost-effective scalability.
* **Disadvantages**:
  + Complex to manage and integrate.
  + Potential security challenges in data movement

**Quick Comparison**

| **Feature** | **Public Cloud** | **Private Cloud** | **Hybrid Cloud** |
| --- | --- | --- | --- |
| **Ownership** | **Third-party providers** | **Single organization** | **Combination of both** |
| **Cost** | **Lower, pay-as-you-go** | **Higher, capital investment** | **Variable, depends on usage** |
| **Security** | **Standard security measures** | **Enhanced security controls** | **Balanced security approach** |
| **Scalability** | **High** | **Limited by internal resources** | **High, with strategic resource allocation** |
| **Management** | **Managed by provider** | **Managed internally** | **Shared management responsibilities** |