RISK AND RESILIENCE BOOTCAMP

TEKsystems Global Services

WORKFORCE DEVELOPMENT

CLOUD COMPUTING

This section is an introduction to cloud computing systems

- What they are
- The IT challenge of cloud computing
- The risk and resilience issues associated with cloud computing

CLOUD COMPUTING

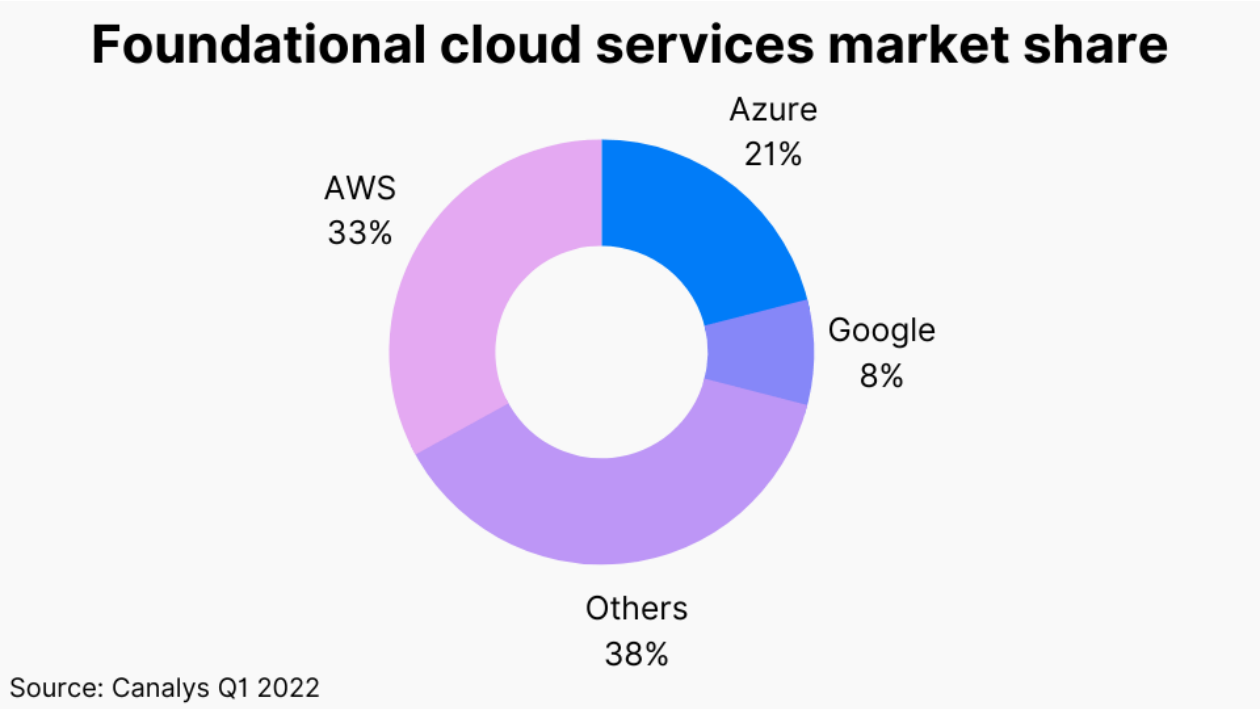
- History and evolution
  - Early roots: virtualization and grid computing in the 2000s
  - Amazon launched EC2 in 2006: commonly cited as birth of modern public IaaS
    - Allowed the use of virtualized infrastructure like disks, servers (recall this from a previous module)
  - Growth of PaaS, serverless, containers, Kubernetes, microservices
    - Allowed for the deployment of operational environments without a virtual infrastructure
  - Increasing focus on AI/ML services, data lakes, managed services
- Adoption
  - The initial wave of migration to the cloud was motivated by cost
  - Instead of capital investment in hardware, the infrastructure was leased in the cloud
  - Modern trend is to re-host cloud applications on site
    - The declining cost of hardware and changed the ROI equation

SERVICE MODELS

- IaaS (Infrastructure as a Service)
  - Virtual machines, storage, networking
  - Customer controls OS, middleware, applications
- PaaS (Platform as a Service)
  - Managed runtime, middleware, development frameworks
  - Infrastructure is opaque to the users
- SaaS (Software as a Service):
  - Fully managed applications like CRM, email, and ERP
- FaaS / Serverless
  - Event-driven, function-execution abstraction
- Other models:
  - DBaaS, CaaS: Database as a service, containers as a service, etc.

VENDORS

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)
- Others / regional players (Alibaba, IBM Cloud, Oracle Cloud, etc.)



CLOUD DEPLOYMENT MODELS

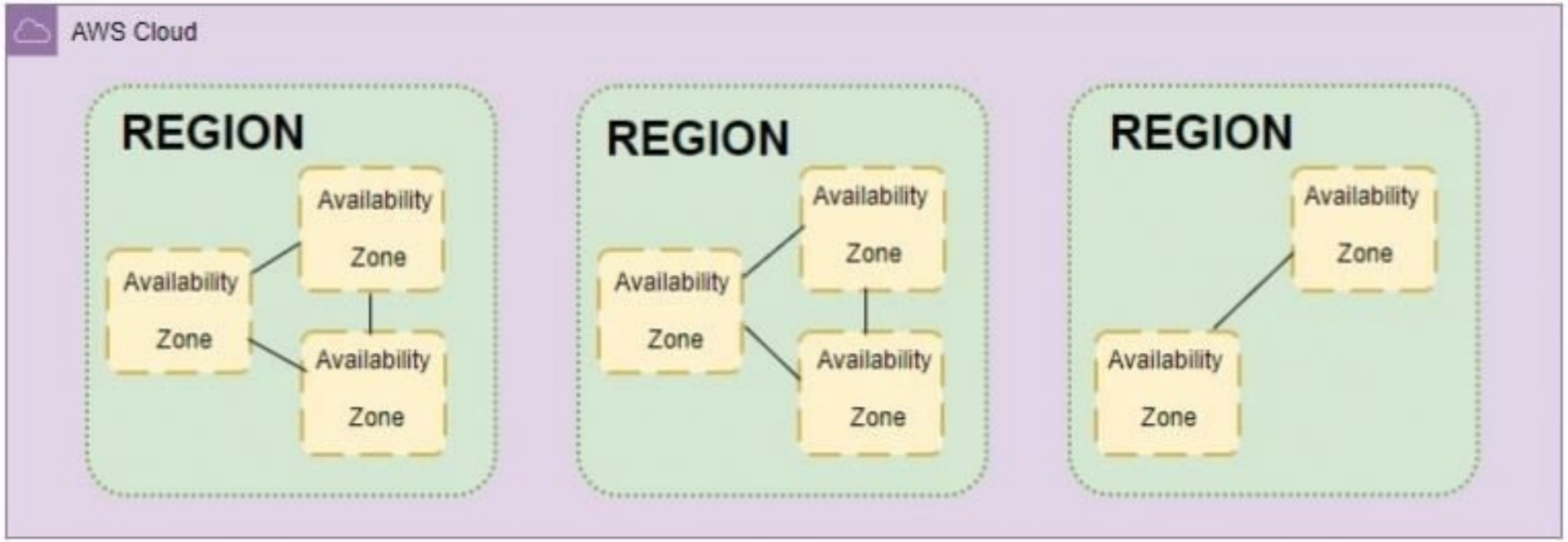
- Public cloud
  - Operated by a third-party provider that offers computing resources (servers, storage, databases, networking, software) over the internet to multiple customers on a pay-as-you-go basis
  - Characteristics
    - Shared infrastructure (multi-tenant environment)
    - On-demand self-service and elasticity
    - Highly scalable and globally distributed
    - Provider responsible for physical and infrastructure security
    - Customer responsible for workloads, configurations, and access (per shared responsibility model)

CLOUD DEPLOYMENT MODELS

- Public cloud
  - Common use cases
    - Web hosting and APIs
    - Data analytics and machine learning
    - Development and testing environments
    - Backup and disaster recovery
    - Fintech and digital banking front-end services
  - Risk considerations
    - Data security and privacy (multi-tenancy and jurisdiction issues)
    - Compliance and data residency constraints in financial and government sectors
    - Vendor lock-in risk due to proprietary APIs and services
  - Resilience impact
    - High resilience due to redundant data centers, availability zones, and automated recovery
    - However, customers depend on the provider's uptime and region architecture
    - Overreliance on one cloud provider can create concentration risk

CLOUD DEPLOYMENT MODELS

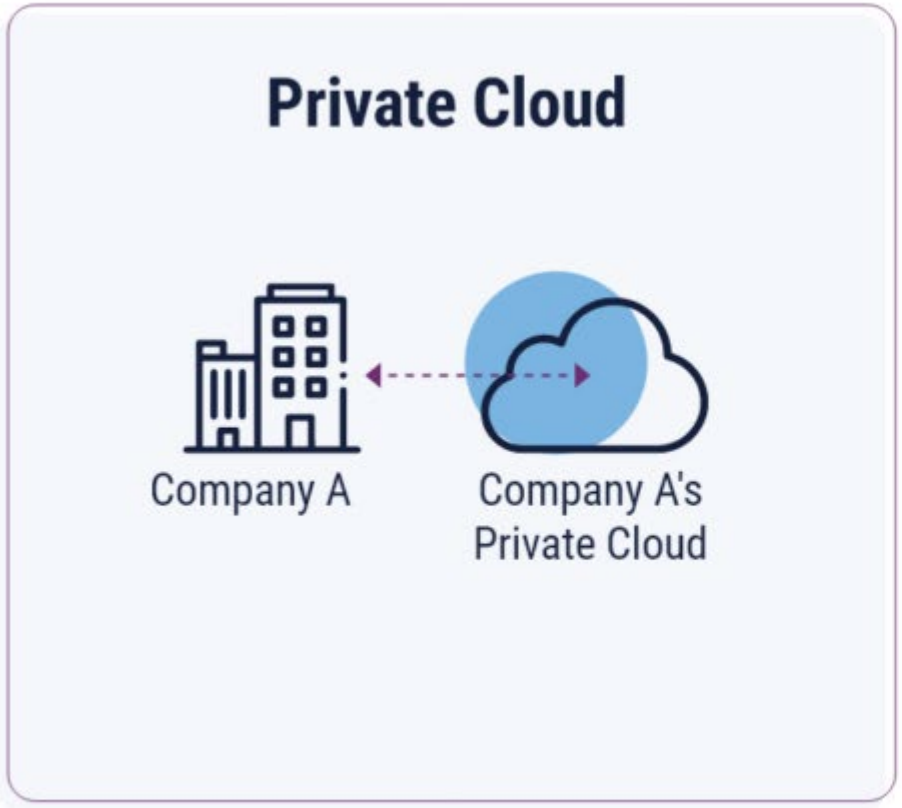
CLOUD DEPLOYMENT MODELS

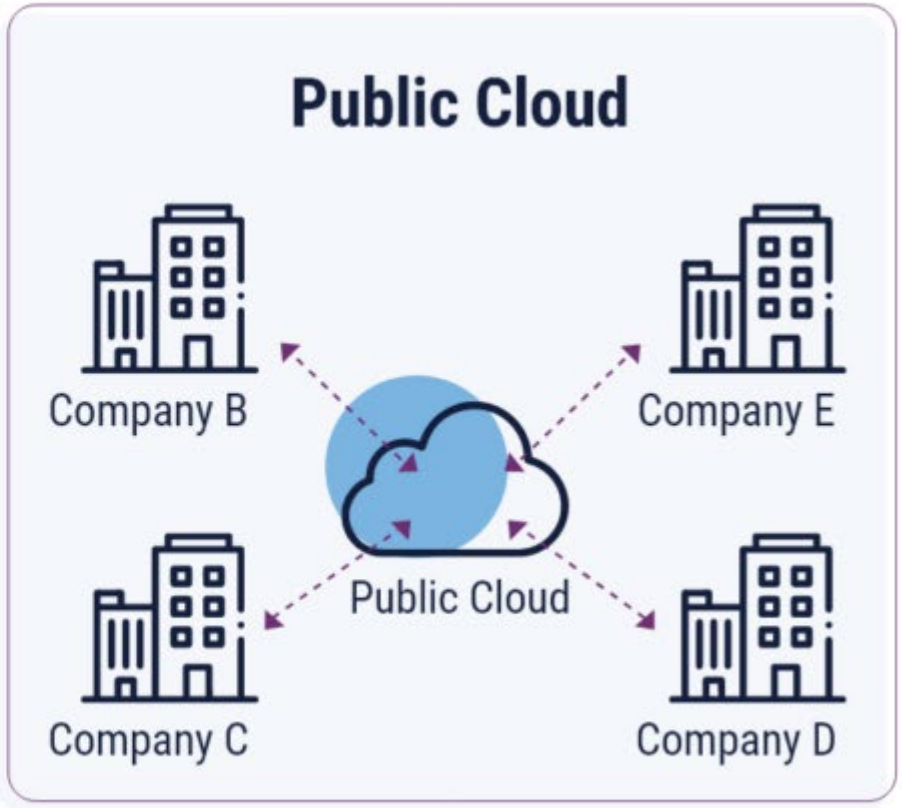


CLOUD DEPLOYMENT MODELS

- Private cloud
  - Infrastructure operated solely for one organization, hosted either on-premises or by a third-party provider.
    - It offers many of the benefits of cloud computing (virtualization, elasticity, self-service) but within a controlled, dedicated environment
  - Examples
    - VMware vCloud Suite
    - OpenStack private clouds
    - Red Hat OpenShift (for container-based private clouds)
    - IBM Cloud Private
  - Characteristics
    - Single-tenant (exclusive resources)
    - Customizable for specific security, governance, or compliance requirements
    - Often managed internally or through a managed service provider (MSP)
    - Can integrate with existing legacy systems

CLOUD DEPLOYMENT MODELS





CLOUD DEPLOYMENT MODELS

- Private cloud
  - Common use cases
    - Core banking systems
    - Payment processing and transaction settlement systems
    - Sensitive healthcare or government data handling
    - Critical workloads that must meet strict regulatory controls
  - Risk considerations
    - High capital and operational costs (infrastructure ownership)
    - Limited elasticity compared to public cloud
    - Internal management risk: resilience depends on in-house expertise and redundancy of the design
  - Resilience impact
    - Full control over security, recovery, and redundancy planning
    - Strong isolation reduces exposure to shared infrastructure threats
    - Resilience depends entirely on internal design: if redundancy is underfunded, the system remains fragile

CLOUD DEPLOYMENT MODELS

- Hybrid cloud
  - A computing environment that combines public and private clouds, allowing data and applications to move between them
    - It integrates on-premises or private resources with public cloud services for flexibility, scalability, and compliance control
  - Examples
    - Azure Arc, AWS Outposts, Google Anthos (hybrid orchestration tools)
  - Characteristics
    - Enables “best of both worlds”: control and security for sensitive workloads, scalability for others
    - Connectivity between environments via VPNs, private links, or dedicated interconnects
    - Supports bursting (overflowing workloads to public cloud during demand spikes)
  - Common use cases
    - Banks running transaction systems on private infrastructure while using public cloud for analytics or mobile apps
    - Governments keeping citizen identity databases on-prem while hosting e-services in public clouds
    - Disaster recovery and backup in public cloud while operations continue on private infrastructure

CLOUD DEPLOYMENT MODELS

- Hybrid Cloud
  - Risk considerations
    - Integration and interoperability risks between environments
    - Complex security and identity management
    - Data movement risks and potential misconfiguration between cloud boundaries
  - Resilience impact
    - Increased resilience through multiple failover options and flexible recovery paths
    - Complexity of integration can introduce operational fragility if not managed carefully
    - Requires strong orchestration, monitoring, and consistent policies across environments

CLOUD DEPLOYMENT MODELS

- Community cloud
  - A collaborative cloud infrastructure shared by several organizations with common goals, compliance requirements, or interests
    - Ownership may be collective or managed by a third-party provider.
  - Examples
    - Government agencies sharing a cloud platform for national services
    - A consortium of banks using a jointly managed secure cloud (e.g., IBM Cloud for Financial Services)
    - Healthcare networks pooling cloud infrastructure compliant with HIPAA
  - Characteristics
    - Shared governance model
    - Customized compliance and security controls relevant to the community
    - Cost shared among participants
    - May be hosted on dedicated infrastructure or managed by a trusted third party

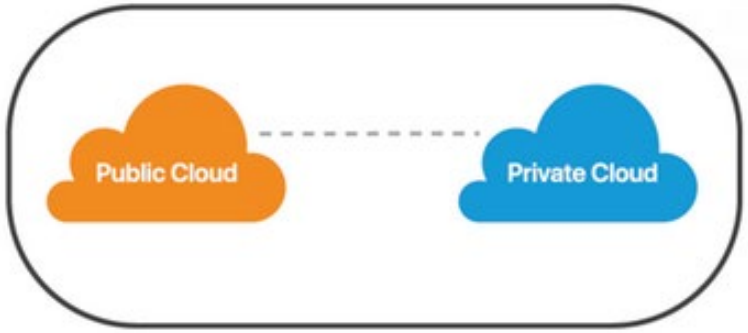
CLOUD DEPLOYMENT MODELS

- Community cloud
  - Common use cases
    - Government ministries sharing infrastructure for e-services
    - Financial consortiums for payment clearing or data exchange
    - Research collaborations between universities
  - Risk considerations
    - Governance complexity because of shared responsibility and decision-making
    - Data segregation and confidentiality among tenants
    - Dependency on central management or service providers
  - Resilience impact
    - Pooled resources can increase redundancy and load distribution
    - Governance delays can hinder incident response
    - Shared operational standards can strengthen collective resilience if coordinated well

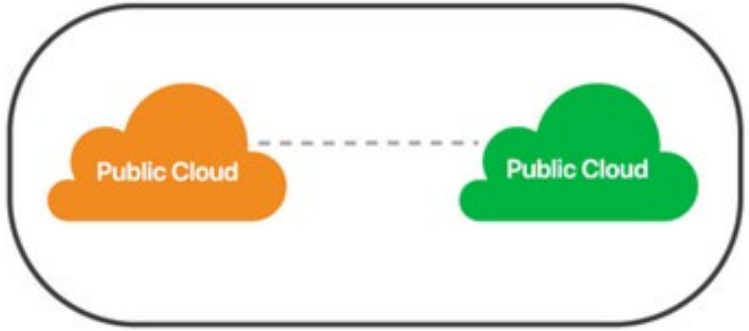
CLOUD DEPLOYMENT MODELS

- Multi-cloud
  - Use of two or more public cloud providers (e.g., AWS + Azure + GCP) for different workloads or redundancy
    - A multi-cloud strategy avoids dependence on a single provider and can leverage unique capabilities from each vendor
  - Characteristics:
    - Distribution of workloads across multiple clouds
    - Common tools or orchestration layers manage deployment and monitoring
    - For example: Terraform, Kubernetes, HashiCorp Consul
    - Data replication or workload failover across clouds
    - Policy-driven workload placement for performance or compliance

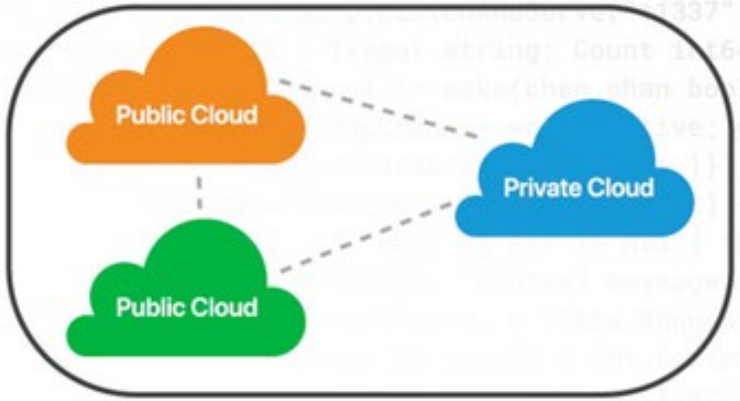
CLOUD DEPLOYMENT MODELS



Hybrid cloud



Multicloud



Multi/hybrid cloud

CLOUD DEPLOYMENT MODELS

- Multi-cloud
  - Common use cases
    - Banks or governments ensuring critical workloads can continue even if one provider experiences an outage
    - Enterprises adopting best-of-breed services (e.g., AI from Google, databases from AWS)
    - Businesses managing global operations with provider-specific regional advantages
  - Risk considerations
    - Operational complexity: managing APIs, security, and monitoring across vendors
    - Cost overhead due to redundant services and data transfer
    - Data consistency and latency issues between providers
  - Resilience impact
    - Strong protection against single-provider failure (reduces concentration risk)
    - Enables flexible disaster recovery and vendor diversification
    - Complexity can reduce manageability: resilience only improves if properly designed and tested

CLOUD DEPLOYMENT MODELS

Model	Ownership	Key Advantage	Primary Risk	Resilience Strength
Public Cloud	Third-party	Low cost, high scalability	Data security, vendor lock-in	High (provider-level)
Private Cloud	Single organization	Control, compliance	Cost, internal management	Depends on internal design
Hybrid Cloud	Combined (public + private)	Flexibility, compliance + scalability	Integration complexity	Moderate to high (if well-orchestrated)
Community Cloud	Shared among organizations	Shared compliance, cost efficiency	Governance complexity	Moderate (depends on coordination)
Multi-Cloud	Multiple providers	Avoid lock-in, redundancy	Operational complexity	Very high (if managed effectively)

CORE TECHNOLOGIES & CONCEPTS

- Virtualization and hypervisors
  - Virtualization allows multiple virtual machines (VMs) to run on a single physical server by abstracting hardware resources (CPU, memory, storage, and network)
    - A hypervisor is the software layer that manages this abstraction
  - Common hypervisors
    - Xen: used by AWS and others
    - KVM (Kernel-based Virtual Machine): Open source, used by Google Cloud
    - Hyper-V: Microsoft's hypervisor for Azure and Windows Server environments
    - VMware ESXi: Widely used in enterprise private clouds
  - How it works
    - The hypervisor partitions physical resources into multiple isolated "guest" systems
    - Each virtual machine acts like an independent computer, running its own operating system and applications

CORE TECHNOLOGIES & CONCEPTS

- Virtualization and hypervisors
  - Resilience benefits
    - Enables resource pooling and fault isolation: one VM crash doesn’t affect others
    - Supports live migration: VMs can be moved between hosts during maintenance or failure
    - Simplifies disaster recovery and backup: VMs can be snapshotted and restored quickly
  - Risk considerations
    - Hypervisor vulnerabilities can compromise all VMs on a host
    - Noisy neighbor effects: one tenant consuming excessive resources can affect others.
    - Complex management: VM sprawl and patching issues if governance is weak.

CORE TECHNOLOGIES & CONCEPTS

- Containers and orchestration
  - Containers are lightweight, portable units that package applications with their dependencies, sharing the host OS kernel
    - They start faster and use fewer resources than VMs
  - Common tools
    - Docker: containerization platform
    - Kubernetes (K8s): orchestration system for managing containers at scale
    - Podman, OpenShift, Amazon ECS/EKS, Google GKE, Azure AKS: cloud-native container orchestration services.
  - How it works
    - Containers isolate application processes in lightweight independent processes
    - Orchestration tools manage deployment, scaling, failover, and networking of containers across multiple hosts

CORE TECHNOLOGIES & CONCEPTS

- Containers and orchestration
  - Resilience benefits:
    - Rapid recovery: failed containers can be automatically restarted or rescheduled
    - Scalability: workloads can scale horizontally based on demand
    - Microservice isolation: faults in one service don’t crash the whole application
    - Portability: easy migration across environments (dev, test, production)
  - Risk considerations:
    - Configuration drift and weak isolation if mismanaged
    - Security of container images: vulnerable base images or misconfigured registries
    - Complexity: orchestrators like Kubernetes introduce new attack surfaces and require careful monitoring

CORE TECHNOLOGIES & CONCEPTS

- Virtual networking
  - Separates the control plane (decision-making about where traffic goes) from the data plane (actual traffic forwarding)
    - This allows network configuration to be automated through software rather than manual hardware changes
  - Uses
    - Enables virtual private clouds (VPCs) and subnets
    - Provides programmable firewalls, load balancers, and routing rules
    - Allows dynamic traffic rerouting and segmentation across tenants
  - Resilience benefits
    - Enables rapid recovery from network failures via automated rerouting
    - Supports isolation between workloads, improving security and fault containment
    - Allows for multi-region connectivity and seamless scaling
  - Risk considerations
    - Misconfigurations can expose entire networks
    - Centralized controllers can become single points of failure
    - Complex visibility: attacks or faults can propagate quickly in virtualized networks if not monitored

CORE TECHNOLOGIES & CONCEPTS

- Object and block storage services, distributed file systems
  - Cloud storage systems abstract physical storage into logical pools that can be accessed over networks
  - Types
    - Block Storage: low-level data storage used by VMs: e.g., Amazon EBS, Azure Disks
    - Object Storage: stores unstructured data as objects in buckets: e.g., Amazon S3, Azure Blob Storage, Google Cloud Storage
    - Distributed File Systems: file-based access across multiple nodes: e.g., Google File System, Ceph, Lustre
  - Resilience benefits
    - High durability: data replicated across multiple availability zones or regions
    - Versioning & snapshot features: rapid recovery from corruption or accidental deletion
    - Elastic capacity: scales automatically as data grows
  - Risk considerations
    - Access misconfigurations (e.g., public S3 buckets) are a major data breach source
    - Eventual consistency: updates may not immediately propagate globally
    - Latency or regional outages can affect dependent services

CORE TECHNOLOGIES & CONCEPTS

- Load balancers, auto-scaling, and elasticity
  - Load balancers distribute incoming traffic across multiple servers or instances to ensure availability and responsiveness
    - Auto-scaling adjusts computing capacity automatically based on demand.
  - Examples
    - Elastic Load Balancer (AWS), Azure Load Balancer, Google Cloud Load Balancer
    - Auto-scaling Groups (ASGs), Horizontal Pod Autoscaler (Kubernetes)
  - Resilience benefits
    - Prevents overload of individual nodes: avoids single points of failure
    - Auto-scaling ensures service continuity during traffic spikes
    - Improves fault tolerance by rerouting traffic to healthy instances
  - Risk considerations
    - Misconfigured scaling policies can cause cost overruns or instability loops
    - Load balancer failure can disrupt entire service access
    - Monitoring and policy testing are critical for predictable behavior

CORE TECHNOLOGIES & CONCEPTS

- Regions, availability zones, and edge locations
  - Cloud providers organize their infrastructure geographically for performance, compliance, and resilience
    - Region: a physical area (e.g., “US-East-1”) containing multiple data centers
    - Availability Zone (AZ): independent data centers within a region, isolated by power, cooling, and network
    - Edge Location: smaller, distributed sites closer to end-users for caching and low-latency access (used by CDNs and edge computing)
  - Resilience benefits
    - Redundancy: replication across AZs mitigates hardware or power failures
    - Disaster recovery: regional replication supports continuity after catastrophic events
    - Latency optimization: edge locations improve responsiveness and user experience
  - Risk considerations
    - Region-level outages still occur (e.g., AWS US-East-1)
    - Data residency issues arise from cross-region replication
    - Overconfidence in “region redundancy” can lead to hidden concentration risk

CORE TECHNOLOGIES & CONCEPTS

- Serverless and event-driven architectures
  - Serverless computing allows developers to run code without managing servers
    - The cloud provider handles scaling, provisioning, and availability
    - Event-driven systems respond automatically to triggers like API calls, file uploads, or messages
  - Examples
    - AWS Lambda, Azure Functions, Google Cloud Functions
    - Event buses: Amazon EventBridge, Kafka, Pub/Sub
  - Resilience benefits
    - Automatic scaling and failover: no idle capacity or manual provisioning
    - Fine-grained recovery: only affected functions restart on error
    - Operational simplicity: less infrastructure to patch or monitor
  - Risk considerations
    - Cold start latency: delay when functions scale up
    - Limited observability in complex workflows
    - Vendor lock-in: heavy dependence on proprietary runtime services
    - Hidden dependencies: chained serverless functions can fail in unpredictable ways

CORE TECHNOLOGIES & CONCEPTS

- Managed services (databases, queues, AI, analytics)
  - Managed services are pre-built components operated by the provider
    - Customers consume functionality (e.g., database, analytics, AI) without managing the underlying infrastructure
  - Examples
    - Databases: Amazon RDS, Azure SQL Database, Google Cloud Spanner
    - Message Queues: Amazon SQS, Azure Service Bus, Kafka on Confluent Cloud
    - AI/ML: AWS SageMaker, Azure Cognitive Services, Google Vertex AI
    - Analytics: BigQuery, Redshift, Snowflake, Databricks
  - Resilience benefits
    - Providers handle replication, backups, patching, and scaling
    - Built-in multi-AZ redundancy and automated failover
    - Allows focus on business logic instead of infrastructure
  - Risk considerations
    - Vendor dependency and interoperability risks (e.g., proprietary databases)
    - Opaque control: customers rely on provider's DR and maintenance policies
    - Complex failure modes: cascading effects when managed dependencies fail

CORE TECHNOLOGIES & CONCEPTS

- Service meshes, API gateways, and microservices architectures
  - Modern cloud-native systems decompose applications into microservices that communicate via APIs
    - A service mesh manages service-to-service communication, while API gateways expose services securely to clients
  - Key technologies
    - Service Mesh: Istio, Linkerd, Consul Connect
    - API Gateways: Kong, Apigee, AWS API Gateway, Azure API Management
  - Resilience benefits
    - Fault isolation: failure in one microservice doesn't crash the entire system
    - Traffic control: routing, retries, and circuit breakers built into the service mesh
    - Observability: tracing and metrics provide deep insight into performance and issues
    - Security: unified policies for authentication, encryption, and rate limiting
  - Risk considerations
    - Complexity explosion: too many moving parts and dependencies
    - Configuration errors can create cascading failures
    - Latency overhead: additional network hops and monitoring layers

CORE TECHNOLOGIES & CONCEPTS

Technology	Resilience Strength	Key Risks
Virtualization	Fault isolation, fast recovery	Hypervisor exploits, shared host exposure
Containers & Orchestration	Auto-healing, scalability	Misconfigurations, image vulnerabilities
SDN & Virtual Networking	Automated failover, isolation	Controller failures, network misconfig
Storage Services	High durability, replication	Access misconfig, latency, consistency
Load Balancers & Auto-scaling	Elasticity, traffic resilience	Cost overrun, unstable scaling
Regions & AZs	Geographic redundancy	Concentration risk, data sovereignty
Serverless	Built-in scaling and DR	Lock-in, cold starts, observability gaps
Managed Services	Operational resilience	Provider dependency, opaque control
Service Meshes & APIs	Fault isolation, visibility	Complexity, cascading misconfig errors

RISK IN CLOUD COMPUTING

- Security and data breach risk
  - Unauthorized access, data leaks, and breaches due to misconfigurations, weak credentials, or provider vulnerabilities
    - Shared infrastructure and API-based access increase exposure
  - Common causes
    - Misconfigured storage (e.g., public S3 buckets)
    - Weak identity or access management
    - Insecure APIs or credentials in code repositories
    - Poor key management
    - Multi-tenancy and hypervisor vulnerabilities

RISK IN CLOUD COMPUTING

- Security and data breach risk
  - Mitigation strategies
    - Encryption everywhere: encrypt data at rest, in transit, and in use
    - Identity and Access Management (IAM): enforce least privilege, role-based access, and multi-factor authentication
    - Continuous monitoring: use SIEM and CSPM (Cloud Security Posture Management) tools
    - Configuration baselines and automation: enforce security-as-code using tools like Terraform and AWS Config
    - Regular penetration testing and red teaming
    - Zero Trust Architecture: verify every connection, even internal ones
  - Reference example
    - Capital One's 2019 AWS breach caused by misconfigured firewall and credential exposure remains a classic case for misconfiguration risk

RISK IN CLOUD COMPUTING

- Compliance and legal risk
  - Cloud use must align with industry regulations (e.g., GDPR, PCI DSS, HIPAA, GLBA, FedRAMP)
    - Non-compliance can lead to fines, lawsuits, and reputational harm
  - Common causes
    - Data stored outside permitted jurisdictions
    - Lack of auditable access logs or chain-of-custody
    - Unverified provider certifications
    - Cloud provider subcontracting outside visibility
  - Mitigation strategies
    - Due diligence on providers: verify certifications (ISO 27001, SOC 2, FedRAMP)
    - Data sovereignty management: choose region-specific deployments
    - Shared Responsibility Model: ensure the organization meets its side of compliance
    - Audit and logging: maintain immutable audit trails
    - Legal clauses in contracts: ensure rights to data access, audit, and exit
  - Example
    - European banks often require “in-region” cloud hosting under the EU’s Digital Operational Resilience Act (DORA)

RISK IN CLOUD COMPUTING

- Operational risk
  - Failures in day-to-day management, misconfigurations, or outages that affect service delivery
    - This includes both provider-level incidents and internal mismanagement
  - Common causes
    - Provider outages (e.g., AWS region downtime)
    - Automation scripts gone wrong (IaC mis-deployments)
    - Insufficient monitoring or alerting
    - Change management failure
  - Mitigation strategies
    - Redundancy: multi-AZ or multi-region deployments
    - Monitoring and observability: integrate logs, traces, metrics with alerting
    - Infrastructure as Code (IaC): controlled, versioned deployments
    - Incident Response Plans: define and test recovery procedures
    - Chaos testing: proactively simulate failures
  - Example
    - AWS outage (Dec 2021) impacted Netflix, Disney+, and Amazon logistics — illustrating concentration risk in provider dependence

RISK IN CLOUD COMPUTING

- Vendor lock-in and dependency risk
  - Difficulty migrating to another provider due to proprietary services, APIs, or data formats, increasing dependency risk and reducing flexibility
  - Common causes
    - Use of provider-specific databases, messaging, or analytics services
    - Lack of multi-cloud strategy
    - Proprietary APIs or SDKs
  - Mitigation strategies
    - Use open standards and portable architectures (e.g., Kubernetes, containers)
    - Abstraction layers: deploy middleware that can run across clouds
    - Data portability: maintain off-cloud backups and export formats
    - Exit strategies in contracts: define exit timelines, data export rights, and penalties
  - Example
    - A financial institution using AWS-native services (Lambda, DynamoDB) made it prohibitively complex to migrate to Azure during vendor review, a real-world lock-in scenario

RISK IN CLOUD COMPUTING

- Concentration and systemic risk
  - Dependence on a few hyperscalers (AWS, Azure, GCP) introduces systemic fragility.
    - A single provider outage or policy change can affect multiple industries simultaneously
  - Common causes
    - Market dominance by three major CSPs
    - Shared third-party dependencies (e.g., DNS, CDN)
    - Geographic or geopolitical dependencies
  - Mitigation strategies
    - Multi-cloud deployment: diversify across providers
    - Hybrid cloud models: keep critical systems on-prem or in private clouds
    - Resilience exercises: simulate provider outages
    - Regulatory oversight: especially in critical sectors (financial services, utilities)

RISK IN CLOUD COMPUTING

- Data loss or corruption risk
  - Loss or corruption of data due to accidental deletion, ransomware, replication errors, or failed recovery
  - Common causes
    - Misconfigured lifecycle policies
    - Application bugs overwriting data
    - Provider replication errors
    - Inadequate backup verification
  - Mitigation strategies
    - Regular backups and replication testing
    - Versioning and immutability in object storage (e.g., S3 Object Lock)
    - Cross-region replication for critical data
    - Automated backup validation

RISK IN CLOUD COMPUTING

- Identity and access management (IAM) risk
  - Mismanaged user privileges, API keys, or service roles can lead to privilege escalation or unauthorized access
  - Common causes
    - Overly permissive IAM policies (wildcard patterns)
    - Shared credentials or hardcoded keys
    - Insufficient monitoring of user behavior
  - Mitigation strategies
    - Principle of least privilege and role-based access control
    - Federated identity and SSO
    - Key rotation and secret vaulting (e.g., AWS KMS, HashiCorp Vault)
    - IAM policy audits and anomaly detection

RISK IN CLOUD COMPUTING

- Financial and cost management risk
  - Uncontrolled cloud consumption can lead to cost overruns, billing surprises, or unprofitable architectures
  - Common causes
    - Overprovisioning or idle resources
    - Poor visibility of cost per application or department
    - Unused data egress or cross-region traffic costs
  - Mitigation strategies
    - FinOps practices: budgeting, monitoring, tagging, and cost allocation
    - Auto-scaling and serverless usage
    - Cost alerts and budgets in cloud consoles
    - Architecture reviews for cost optimization

RESILIENCE IN CLOUD COMPUTING

- Dependency on cloud provider availability
  - Issue: If a region, service, or control plane fails, all dependent workloads go offline
  - Examples
    - AWS US-East-1 outages repeatedly affected global services
    - Microsoft Azure regional authentication failures blocked user logins
  - Resilience strategies
    - Multi-AZ and multi-region architectures
    - Active-active failover across regions
    - Cross-provider redundancy (multi-cloud)
    - Automated failback testing using simulated outages

RESILIENCE IN CLOUD COMPUTING

- Latency, network, and connectivity risk
  - Issue: Connectivity between regions or from users to the cloud may degrade due to network congestion or ISP failures
  - Resilience strategies
    - Edge caching and CDNs for end-user content
    - Hybrid deployments for latency-sensitive systems
    - Redundant connectivity: VPN and Direct Connect links
    - Traffic routing policies (Anycast, Route53 health checks)

RESILIENCE IN CLOUD COMPUTING

- Cascading failures across cloud services
  - Issue: Cloud applications depend on multiple managed services
    - If one fails (e.g., storage or identity), dependent services may also fail causing a cascading failure
  - Resilience strategies
    - Dependency mapping: identify service interdependencies
    - Graceful degradation: build apps to continue with limited functionality
    - Circuit breakers and retry logic in code
    - Resilience testing (“game days”) to uncover hidden dependencies

RESILIENCE IN CLOUD COMPUTING

- Complexity and misconfiguration risk
  - Issue: Cloud environments have thousands of configuration options
    - Misconfigurations are one of the top causes of outages and breaches
  - Resilience strategies
    - Automation with guardrails: use IaC templates, enforce policy-as-code
    - Continuous compliance tools (e.g., AWS Config, Azure Policy)
    - Change management with approvals and version control
    - Monitoring and alerting for configuration drift

RESILIENCE IN CLOUD COMPUTING

- Shared responsibility and misaligned expectations
  - Issue: Organizations may assume providers handle resilience end-to-end
    - In reality, providers ensure infrastructure uptime, while customers must ensure application-level continuity
  - Resilience strategies
    - Understand shared responsibility model
    - Design for application-level redundancy
    - Perform joint DR planning with providers
    - Define SLAs/SLOs for both provider and internal operations

RESILIENCE IN CLOUD COMPUTING

- Recovery and data replication challenges
  - Issue: While providers replicate data, logical errors (deletions, corruption) can propagate across replicas
  - Resilience strategies
    - Immutable backups and air-gapped copies
    - Cross-region backup separate from production data
    - Regular restore testing (not just backups)
    - Version control for datasets and storage buckets

RESILIENCE IN CLOUD COMPUTING

- Multi-cloud and hybrid complexity
  - Issue: Using multiple clouds improves resilience but increases operational complexity, tooling inconsistency, and synchronization challenges
  - Resilience strategies
    - Centralized observability and orchestration
    - Cloud-agnostic platforms (Kubernetes, Terraform, HashiCorp Consul)
    - Standardized deployment pipelines across environments
    - Regular failover testing between clouds

RESILIENCE IN CLOUD COMPUTING

- Human factors and skill gaps
  - Issue: Resilience failures often result from misoperation, lack of cloud expertise, or poor incident response readiness
  - Resilience strategies
    - Training and certification programs
    - Incident response drills and game days
    - DevSecOps culture: shared ownership of resilience
    - Automated runbooks and playbooks

RESILIENCE IN CLOUD COMPUTING

Risk / Issue Type	Examples / Causes	Mitigation / Resilience Measures
Security & Breach Risk	Misconfigured storage, weak IAM	Encryption, Zero Trust, CSPM
Compliance Risk	Cross-border data, missing logs	Data residency, audit, certifications
Operational Risk	Outages, IaC errors	Redundancy, monitoring, automation
Vendor Lock-in	Proprietary APIs	Open standards, hybrid design
Concentration Risk	Overreliance on hyperscalers	Multi-cloud, hybrid, exit strategies
Data Loss	Replication failure, deletion	Immutable backups, restore testing
IAM Risk	Excessive privileges	Least privilege, MFA, rotation
Financial Risk	Cost overruns	FinOps, auto-scaling, budgets
Provider Availability	Region outage	Multi-region, cross-cloud failover
Cascading Failures	Service dependency	Circuit breakers, graceful degradation
Connectivity Issues	Latency, ISP outage	Edge caching, redundant links
Complexity	Misconfiguration	Automation, guardrails, IaC
Recovery Gaps	Logical errors	Immutable backups, cross-region DR

Q&A AND OPEN DISCUSSION