UNIT–2 : Cloud Enabling Technologies & Service Oriented Architecture (PYQ Solution)

AKTU · Cloud Computing (BCAM051) · Solutions by **Chirag Jain**, Ycotes

SECTION A Answers (≈70 Words Each)

Q1. Explain virtualization and mention one real-life benefit of virtual machines. (Jan 2026, Jan 2025)

Virtualization is the technique of creating a virtual version of a physical resource (server, storage, network, OS) using a software layer called a hypervisor, so that many isolated virtual machines run on one physical machine. Benefit: server consolidation. A company can run ten applications on one server instead of ten, cutting hardware, power and space costs while keeping each application isolated.

Q2. Compare the implementation levels of virtualization. (Jan 2025)

Level	Description	Example
Instruction Set Architecture (ISA)	Emulates one ISA on another	QEMU, Bochs
Hardware abstraction level (HAL)	Hypervisor virtualizes CPU, memory, I/O directly on hardware	Xen, VMware ESXi, KVM
Operating system level	Containers share the host kernel	Docker, LXC
Library (user-level API) level	Intercepts API calls	WINE
Application level	Language VM runs bytecode	JVM, .NET CLR

Going from ISA to application level, performance improves and isolation decreases; HAL gives the best balance.

Q3. Define disaster recovery. (2024)

Disaster recovery (DR) is the set of policies, tools and procedures used to restore IT systems, applications and data after a disruption such as hardware failure, cyber-attack, fire or natural disaster. It is measured by RTO (recovery time objective: how fast service returns) and RPO (recovery point objective: how much data loss is acceptable). In the cloud, DR uses backups, replicas and failover to a secondary site.

Q4. List the types of virtualization. (2024)

- **Hardware (server) virtualization: full, para and hardware-assisted.**
- **Operating system virtualization: containers.**
- **Storage virtualization: pooling of physical storage (SAN, NAS).**
- **Network virtualization: virtual networks, VLAN, SDN.**
- **Desktop virtualization (VDI) and application virtualization.**
- **CPU, memory and I/O virtualization (components of hardware virtualization).**
- **Data and database virtualization.**

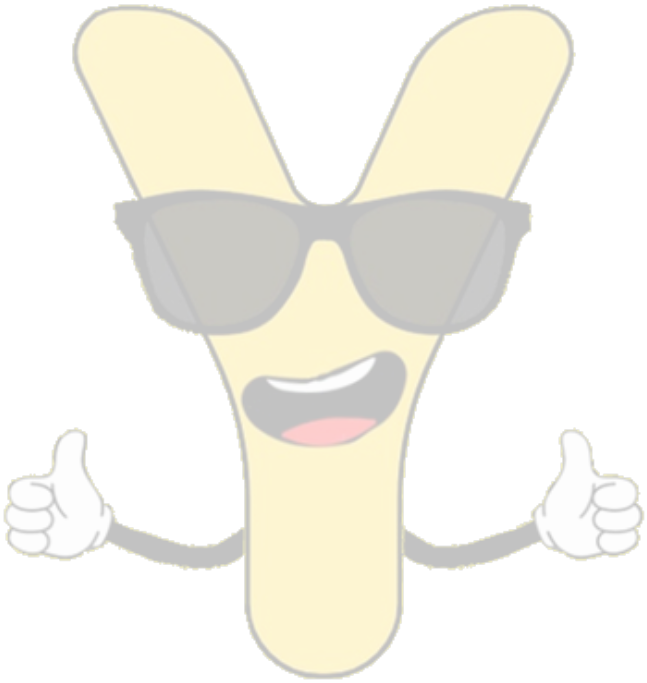
Q5. Explain the significance of Service-Oriented Architecture in cloud computing. (Feb 2024)

SOA structures software as loosely coupled, reusable services with well-defined interfaces that communicate through standard protocols (SOAP, REST). In the cloud it is significant because every cloud offering (IaaS, PaaS, SaaS) is exposed as a service through an API. SOA gives reusability, interoperability across platforms, scalability, easy integration and faster development, and supports service composition and orchestration.

Q6. Describe REST in the context of web services. (Feb 2024)

REST (Representational State Transfer) is an architectural style for web services where resources are identified by URIs and manipulated using standard HTTP methods: GET, POST, PUT, DELETE. It is stateless (each request carries all needed information), supports JSON or XML

representations, is cacheable and lightweight, and is the dominant style for cloud APIs, e.g. GET /students/10.



SECTION B Answers (≈250 Words Each)

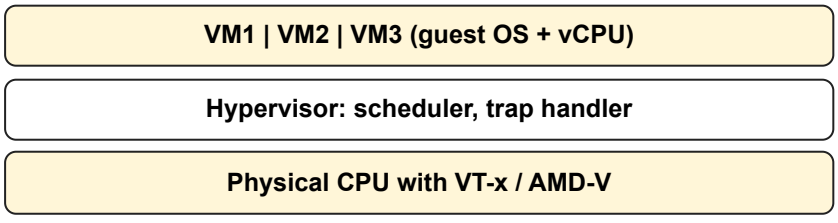
Q1. Describe the types of virtualization and explain how CPU virtualization is achieved and enables multiple VMs on a single processor. (Jan 2026, Feb 2024)

Types of virtualization:

- **Hardware/server virtualization:** hypervisor runs multiple VMs (Type 1 bare-metal like Xen, ESXi; Type 2 hosted like VirtualBox).
- **OS-level virtualization:** containers share one kernel (Docker).
- **Storage virtualization:** logical pooling of physical disks.
- **Network virtualization:** virtual switches, VLAN, SDN.
- **Desktop and application virtualization:** VDI, streamed applications.

CPU virtualization: the hypervisor lets several guest OSs believe they own the processor.

1. **Privilege levels:** the guest kernel runs in a lower-privilege ring while the hypervisor runs at the highest privilege.
2. **Trap and emulate:** privileged or sensitive instructions executed by the guest cause a trap; the hypervisor emulates the effect on a virtual CPU (vCPU) and resumes the guest.
3. **Binary translation:** for sensitive x86 instructions that do not trap, the hypervisor rewrites the code on the fly (full virtualization).
4. **Paravirtualization:** the guest kernel is modified to use **hypercalls** instead of privileged instructions.
5. **Hardware-assisted virtualization:** Intel VT-x and AMD-V add a root/guest mode, so sensitive instructions trap automatically.
6. **Scheduling:** the hypervisor **time-slices** the physical CPU among vCPUs (like an OS schedules processes), saving and restoring each vCPU state.



CPU virtualization

Because each VM has its own vCPU state, memory mapping and I/O, many VMs share one processor in isolation.

Q2. Discuss Service-Oriented Architecture and its relevance in cloud computing. Analyze its impact on application scalability. (Jan 2025)

SOA is a design style in which an application is built from **independent services** that expose functions through published interfaces (WSDL/REST) and are discovered and invoked over a network. Roles: **service provider, service consumer, service registry**.



SOA publish-find-bind

Relevance to cloud:

- Cloud models (IaaS/PaaS/SaaS) are delivered as services over APIs.
- Services are **loosely coupled and platform independent**, so they run anywhere in the cloud.
- **Reuse and composition** speed up development; **interoperability** links different vendors.

Impact on scalability:

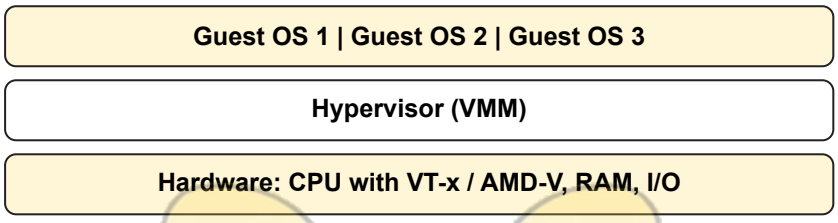
- Each service can be **scaled independently** (more instances of only the busy service), instead of scaling the whole application.
- Stateless services work behind **load balancers** and autoscaling.
- Failures are isolated, improving availability.
- Drawbacks: network overhead, service-management complexity and need for governance.

Conclusion: SOA is the architectural basis for scalable, composable cloud applications.

Q3. Describe the techniques used for implementing hardware virtualization, with diagram. (2024)

Hardware (server) virtualization lets a hypervisor create multiple VMs. Techniques:

1. **Full virtualization:** the guest OS runs **unmodified**; the hypervisor traps and translates privileged instructions (binary translation + direct execution). Example: VMware ESXi. Strong isolation, some overhead.
2. **Paravirtualization:** the guest OS is **modified** to make hypercalls to the hypervisor. Example: Xen PV. Lower overhead but needs guest changes.
3. **Hardware-assisted virtualization:** CPU extensions (Intel VT-x, AMD-V, EPT/NPT for memory) let the hypervisor run unmodified guests efficiently. Example: KVM, Hyper-V.
4. **OS-level (container) virtualization** is a lightweight alternative.



Hardware virtualization

Technique	Guest modified?	Performance	Example
Full	No	Good	VMware
Para	Yes	Very good	Xen
Hardware-assisted	No	Best	KVM

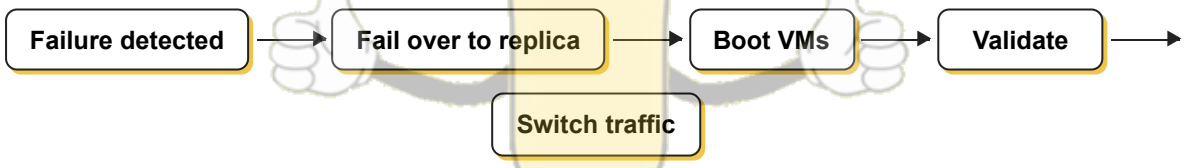
SECTION C Answers (≈500 Words Each)

Q1. Explain how virtualization supports disaster recovery and business continuity. Simulate a recovery scenario for a financial system and calculate the restoration time. (Jan 2025, Feb 2024)

Virtualization and DR: a VM is a set of files (disk image, configuration), so it is easy to copy, move and restart on different hardware.

- **Snapshots and backups** of whole VMs capture OS, applications and data.
- **Replication** of VMs to a secondary site or cloud keeps a standby copy (low RPO).
- **Hardware independence:** a VM restarts on any compatible host, no need for identical hardware.
- **Live migration and high availability (HA):** VMs move or restart automatically when a host fails (VMware HA, vMotion).
- **Fast, automated failover and testing** without disturbing production.
- **Lower cost:** the standby site needs fewer resources, and uses cloud pay-per-use.

Business continuity: these mechanisms keep critical services running with short downtime, protect against data loss, meet compliance and preserve customer trust.



DR failover flow

Simulation (financial system, 2 TB data):
Case 1: restore from backup. Steps:

- Detection and declaration: 10 min.
- Data restore at 200 MB/s: 2 TB = 2,097,152 MB ÷ 200 = 10,486 s ≈ 175 min.
- Boot VMs and services: 10 min.
- Data validation and reconciliation: 15 min.
- DNS/load-balancer cutover: 5 min.
  Total = 10 + 175 + 10 + 15 + 5 = 215 min ≈ 3 h 35 min.

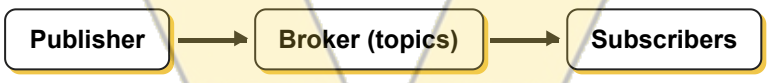
Case 2: continuous replication to a standby site (data already present).

- Detection 5 min + failover and VM boot 10 min + validation 10 min + cutover 5 min = **30 min**.

Replication reduces the recovery time from 215 to 30 minutes (about 86% less), meeting an RTO of 1 hour, with an RPO of seconds.

Q2. Describe the Publish-Subscribe model and its implementation in real-time systems. Simulate its use in a stock trading application and compute latency for 1 million events. (Jan 2025)

Publish-Subscribe (pub-sub) is a messaging model where **publishers** send messages to **topics** (channels) without knowing the receivers, and **subscribers** register interest in topics; a **broker** routes each message to all subscribers of its topic. It gives **decoupling in space, time and synchronisation**, so the system scales easily and works for real-time event delivery.



Publish-subscribe model

Implementation: brokers such as Apache Kafka, RabbitMQ, MQTT or cloud services (AWS SNS, Google Pub/Sub); filtering by topic or content; persistence and acknowledgements for reliability; partitions for parallelism.

Stock trading simulation: publishers = exchange feeds publishing price ticks to topics like `RELIANCE` and `TCS`; subscribers = trading engines, dashboards and risk systems.
Assume: 1,000,000 events; broker has 4 partitions each processing 25,000 events/s = 100,000 events/s; network publish delay 1 ms; delivery delay 1 ms.

- **Per-event latency (steady flow)** = publish 1 ms + broker service (1/25,000 s = 0.04 ms) + delivery 1 ms ≈ **2.04 ms**.
- **Time to process all events** = 1,000,000 ÷ 100,000 = **10 s**.
- If all events arrive as one burst, the first has ≈ 2 ms latency and the last waits about 10 s; the **average queueing delay ≈ 5 s**. Adding partitions (8 partitions → 200,000 events/s) halves total time to 5 s.

Conclusion: pub-sub with partitioned brokers delivers low-latency, scalable event streaming for trading.

Q3. Discuss Service-Oriented Architecture (SOA). Also explain the building blocks of SOAP. (2024)

SOA: an architectural style in which software is organised as **loosely coupled, reusable, interoperable services** that provide business functions through standard interfaces. Principles: standard contracts, loose coupling, abstraction, reusability, composability, autonomy, statelessness and discoverability. Components: **service provider, consumer, registry (UDDI)**, with the **enterprise service bus** for routing. Benefits: reuse, integration of heterogeneous systems, flexibility and scalability. Challenges: performance overhead, security and governance.

SOAP (Simple Object Access Protocol): an XML-based protocol for exchanging structured information between web services, usually over HTTP or SMTP. **Building blocks of a SOAP message:**

1. **Envelope:** mandatory root element defining the start and end of the message and namespaces.
2. **Header:** optional; carries metadata such as authentication, routing, transaction information.
3. **Body:** mandatory; contains the actual request or response data.
4. **Fault:** optional element inside the body for error reporting (code, string, actor, detail).



```
<soap:Envelope>
  <soap:Header> ... auth token ... </soap:Header>
  <soap:Body>
    <GetPrice><item>Laptop</item></GetPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP is described by **WSDL** and discovered through **UDDI**. It is platform-neutral and supports WS-Security, but is heavier than REST.

Q4. Explain the architectural constraints of web services. (2024)

Web services follow **REST architectural constraints** (Fielding):

1. **Client-server:** separation of user interface from data storage improves portability and scalability.
2. **Stateless:** each request contains all information needed; the server stores no client session, enabling scaling.

3. **Cacheable:** responses are marked cacheable or not to reduce load and latency.
4. **Uniform interface:** resource identification by URI, manipulation through representations (JSON/XML), self-descriptive messages, and HATEOAS (hypermedia links).
5. **Layered system:** clients cannot tell whether connected directly to the server or through proxies, gateways and load balancers.
6. **Code on demand (optional):** server can send executable code (JavaScript).

For SOAP-style services, additional constraints are standard contracts (WSDL), XML messaging and a defined protocol stack (SOAP, WSDL, UDDI).



Layered, cacheable, stateless web service

Conclusion: these constraints give scalable, simple, loosely coupled web services.

Q5. Discuss the key virtualization structures, full virtualization and paravirtualization, and their impact on performance and efficiency. (Feb 2024)

Full virtualization: the hypervisor presents a complete virtual copy of the hardware; the **guest OS runs unmodified**. Sensitive instructions are trapped or rewritten by **binary translation**, with other instructions executed directly. Examples: VMware ESXi, VirtualBox.

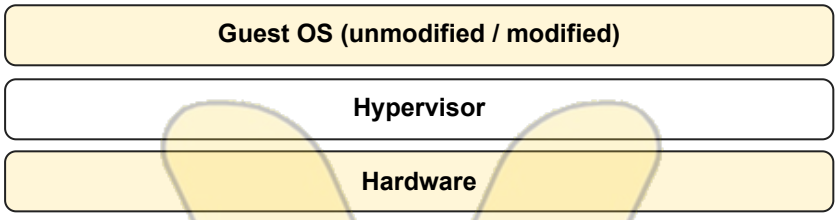
- Advantages: no guest changes, strong isolation, many OSs supported.
- Disadvantages: translation and trap overhead, a complex hypervisor.

Paravirtualization: the hypervisor exposes an interface similar but not identical to hardware; the **guest OS kernel is modified** to call **hypercalls** instead of privileged instructions, and uses split drivers for I/O. Example: Xen.

- Advantages: lower overhead, higher performance, efficient I/O.
- Disadvantages: needs a modified guest kernel; not available for closed-source OSs; portability limited.

Feature	Full	Para
Guest OS	Unmodified	Modified
Mechanism	Binary translation/traps	Hypercalls
Performance	Moderate (5-15% overhead)	Better (about 2-5% overhead)
Compatibility	High	Limited
Examples	VMware, VirtualBox	Xen

Hardware-assisted virtualization (VT-x, AMD-V) now gives full virtualization near paravirtual speed.



Virtualization structure

Impact: paravirtualization yields better efficiency where guest kernels are controllable; full virtualization gives flexibility. Both improve utilisation by consolidating servers.
