**Virtualization & Cloud — Detailed Step-by-Step Notes**

**1) सारांश — काय आहे आणि का महत्वाचं?**

* **Virtualization** = एका physical मशीन वर multiple isolated virtual machines (VMs) चालवणे (each with own OS).
* **Cloud** = इंटरनेटवर/on-demand रूपात compute, storage, network आणि managed services मिळवण्याची पद्धत (IaaS, PaaS, SaaS).
* फायदा: resource consolidation, isolation, elasticity, snapshots, live migration, faster provisioning.

**PART A — Virtualization (step-by-step)**

**A1. Hypervisors — प्रकार**

* **Type-1 (bare-metal)**: VMware ESXi, KVM (on bare metal via libvirt), Xen — direct hardware वर चालतात (production datacenters).
* **Type-2 (hosted)**: VirtualBox, VMware Workstation — host OS वर रन होतात (dev/desktops).

**Interview short:** Type-1 म्हणजे direct hardware, Type-2 host OS वर.

**A2. आवश्यक गोष्टी (Hosts)**

* CPU virtualization extensions: **Intel VT-x** किंवा **AMD-V** (BIOS/UEFI मध्ये enable करा).
* Sufficient RAM, CPUs, disk, and networking.
* For passthrough: IOMMU (enable via kernel boot intel\_iommu=on / amd\_iommu=on).

**A3. KVM + libvirt — Quick install (RHEL/CentOS/Ubuntu)**

# RHEL/CentOS

sudo yum install -y qemu-kvm libvirt virt-install virt-manager

# Ubuntu/Debian

sudo apt update

sudo apt install -y qemu-kvm libvirt-daemon-system libvirt-clients virtinst virt-manager

Enable/start libvirt:

sudo systemctl enable --now libvirtd

# check

sudo systemctl status libvirtd

**A4. Create storage pool & network (libvirt)**

# List default networks

virsh net-list --all

# Start default network if down

virsh net-start default

virsh net-autostart default

# Create storage pool (dir)

virsh pool-define-as --name vmstore --type dir --target /var/lib/libvirt/images

virsh pool-build vmstore

virsh pool-start vmstore

virsh pool-autostart vmstore

**A5. Create VM with virt-install (example)**

virt-install \

--name testvm \

--ram 4096 \

--vcpus 2 \

--os-variant ubuntu20.04 \

--cdrom=/path/to/ubuntu-20.04.iso \

--network network=default \

--disk path=/var/lib/libvirt/images/testvm.qcow2,size=20 \

--graphics none \

--console pty,target\_type=serial

* GUI/console: virt-manager वापरून GUI मध्ये VM बनवता येते.

**A6. virsh — महत्वाचे commands**

virsh list --all # all VMs

virsh start testvm

virsh shutdown testvm

virsh destroy testvm # force off

virsh console testvm # serial console

virsh dominfo testvm # VM info

virsh snapshot-create-as testvm snap1 "pre-change" --disk-only=false

virsh snapshot-list testvm

virsh snapshot-revert testvm snap1

virsh migrate --live testvm qemu+ssh://target-host/system # live migration

**A7. Snapshots & backups**

* **Libvirt snapshot**: fast, but watch space. Example above.
* **LVM snapshot**: create snapshot LV, mount and rsync off to backup server.
* For full image backup: virsh dumpxml testvm > testvm.xml and copy qcow2 image.

**A8. Live migration (prereqs & steps)**

**Prereq:** shared storage (NFS/CEPH) or pre-copy migration, identical CPU features or --copy-locks/cpu model tuning.

# Example: live migrate

virsh migrate --live --persistent --undefinesource testvm qemu+ssh://target/system

* Verify networks, storage accessibility, and CPU compatibility. Always test in lab.

**A9. PCI passthrough / SR-IOV (overview steps)**

1. Enable IOMMU in kernel boot: intel\_iommu=on / amd\_iommu=on.
2. Identify device: lspci -nn.
3. Bind to vfio driver: add to modprobe config or virsh nodedev-detach.
4. Add <hostdev> entry in VM xml to pass device.

**Warning:** passthrough changes hardware availability; test before prod.

**A10. Containers vs VMs (short)**

* **VM:** full OS per VM, strong isolation, heavier.
* **Container:** OS-level virtualization (Docker, Podman), lightweight, fast startup, shares host kernel.
* Use containers for microservices; VMs when OS isolation or different kernels needed.

**A11. Troubleshooting Virtualization**

* Check libvirt service: systemctl status libvirtd
* VM logs: /var/log/libvirt/qemu/<vm>.log
* qemu error: journalctl -u libvirtd -f / /var/log/messages
* Network issues: virsh net-dumpxml default and check bridge brctl show or ip a.

**PART B — Cloud (Basic Awareness, step-by-step)**

खाली AWS CLI आधारित उदाहरणे देत आहे कारण CLI examples सर्वात practical असतात. संकल्पना सर्व public cloud providers (AWS/Azure/GCP) मध्ये जवळजवळ सारख्या असतात.

**B1. Cloud models (concepts)**

* **IaaS**: VM, storage, network (EC2, Compute Engine)
* **PaaS**: Managed platform (Elastic Beanstalk, App Engine)
* **SaaS**: Software as a Service (Gmail, Office365)

**B2. Cloud deployment models**

* **Public** (AWS/Azure/GCP), **Private** (on-prem cloud like OpenStack), **Hybrid** (mix of both), **Multi-Cloud**.

**B3. Key Cloud Concepts**

* **Region** (geographic) → **Availability Zone (AZ)** (isolated datacenter).
* **VPC / Virtual Network** → logical network, subnets, routing.
* **Security Group / Network ACL** → virtual firewall rules.
* **IAM** → identity & access management (roles, policies).
* **Object storage** (S3/GCS) vs **Block storage** (EBS) vs **File storage** (EFS, Filestore).
* **Autoscaling**, **Load Balancer**, **Managed DB** (RDS / Cloud SQL), **Serverless** (Lambda / Cloud Functions).

**B4. AWS — basic step-by-step (CLI examples)**

**Prereq:** configure AWS CLI (aws configure) with access key, secret, region.

1. **Create keypair**

aws ec2 create-key-pair --key-name mykey --query 'KeyMaterial' --output text > mykey.pem

chmod 400 mykey.pem

1. **Create security group (allow SSH, HTTP)**

aws ec2 create-security-group --group-name web-sg --description "web SG" --vpc-id vpc-xxxxx

aws ec2 authorize-security-group-ingress --group-id sg-xxxxx --protocol tcp --port 22 --cidr 203.0.113.0/24

aws ec2 authorize-security-group-ingress --group-id sg-xxxxx --protocol tcp --port 80 --cidr 0.0.0.0/0

1. **Launch EC2 instance**

aws ec2 run-instances --image-id ami-0abcdef1234567890 --count 1 --instance-type t3.micro \

--key-name mykey --security-group-ids sg-xxxxx --subnet-id subnet-xxxx --associate-public-ip-address

1. **SSH to instance**

ssh -i mykey.pem ec2-user@<PUBLIC\_IP>

1. **Attach EBS volume & snapshot**

# create volume

aws ec2 create-volume --size 20 --availability-zone ap-south-1a --volume-type gp3

# attach

aws ec2 attach-volume --volume-id vol-xxxxx --instance-id i-xxxxx --device /dev/xvdf

# create snapshot

aws ec2 create-snapshot --volume-id vol-xxxxx --description "backup"

1. **Create AMI from instance**

aws ec2 create-image --instance-id i-xxxxx --name "myserver-ami-$(date +%F)"

**B5. Azure / GCP equivalents (conceptual)**

* **Azure**: VM (VM Scale Sets), VNet, NSG (security groups), Managed Disks, Azure Blob, Azure Functions.
* **GCP**: Compute Engine, VPC, Firewall rules, Persistent Disk, Cloud Storage, Cloud Functions.  
  (Each has CLI: az, gcloud similar steps.)

**B6. Cloud networking basics (VPC)**

1. Create VPC.
2. Create subnets (public/private).
3. Create Internet Gateway and attach to VPC.
4. Create route table and route public subnet default route via IGW.
5. Security groups (stateful) and Network ACLs (stateless).
6. Load Balancer (ALB/NLB) in public subnet forwarding to instances.

**B7. IAM & Security best practices**

* Use roles for EC2 (no embedded keys).
* Least privilege policies.
* MFA for root/console.
* Rotate keys & monitor via CloudTrail.
* Enable encryption at rest (EBS, S3 default encryption).

**B8. Auto-scaling & High Availability (step-by-step concept)**

1. Create Launch Template/Configuration (instance type, AMI, user-data).
2. Define Auto Scaling Group (ASG) with min/max/desired.
3. Attach to Load Balancer target group.
4. Health checks -> replace unhealthy instances automatically.

**B9. Cost & Monitoring**

* Use CloudWatch / Stackdriver / Azure Monitor for metrics & logs.
* Tag resources for cost center.
* Use reserved instances/savings plans where long-term.

**PART C — Storage & Networking in Virtual/Cloud environments**

**Storage choices**

* **Local**: fastest, not shareable.
* **Block (EBS)**: attachable block device to single VM.
* **File (NFS, EFS)**: shared across machines.
* **Object (S3)**: scalable, cheap, REST access.

**Network choices**

* **Bridged** vs **NAT** vs **Host-only** (VM networks).
* Cloud: private subnets (no internet) vs public subnets.

**PART D — Troubleshooting Virtualization & Cloud**

**Virtualization troubleshooting quick commands**

* virsh list --all → VM power states.
* virsh dominfo <vm> → memory/vCPU state.
* Logs: /var/log/libvirt/qemu/<vm>.log, journalctl -u libvirtd.
* KVM host status: virsh nodeinfo, free -m, top.
* VMware: use vSphere client or esxcli / vim-cmd on ESXi.

**Cloud troubleshooting**

* Check instance system logs (console logs in provider).
* Check security groups / NACL blocking ports.
* Check cloud provider status & console for events.
* Review IAM permission errors via CloudTrail logs.

**PART E — Best Practices & Hardening (short list)**

* Use templates/AMI/Golden images for consistent provisioning.
* Tag everything (owner, env, cost center).
* Backup: snapshots + automated schedules.
* Monitor: metrics, logs, alerting, health checks.
* Security: IAM least privilege, security groups, patching.
* Limit SSH access (bastion hosts, SSM Session Manager).
* Use Infrastructure as Code (Terraform/CloudFormation) for reproducibility.

**PART F — Hands-on mini-lab (suggested steps you can run)**

**Lab 1 — KVM VM**

1. Install qemu/libvirt/virt-manager.
2. Start libvirtd, ensure default network up.
3. Create a storage pool and create VM with virt-install.
4. Use virsh to snapshot and revert.

**Lab 2 — AWS EC2**

1. aws configure with credentials.
2. Create keypair + security group.
3. Launch EC2 with aws ec2 run-instances.
4. SSH, attach EBS, create snapshot, create AMI.

(मग मी step-by-step commands + sanity checks तयार करून देईन.)

**Interview Q&A (common & concise)**

1. **Q:** Hypervisor types?  
   **A:** Type-1 (bare-metal) vs Type-2 (hosted).
2. **Q:** VM vs Container — difference?  
   **A:** VM = full OS + kernel; Container = shares host kernel, lightweight.
3. **Q:** What is live migration? prerequisites?  
   **A:** Move running VM to another host with minimal downtime. Needs shared storage or pre-copy, matching CPU features, network config.
4. **Q:** How to snapshot a VM?  
   **A:** virsh snapshot-create-as <vm> name or via hypervisor UI; or take storage (EBS) snapshot in cloud.
5. **Q:** What is a VPC?  
   **A:** Virtual network in cloud that isolates resources—has subnets, route tables, gateways.
6. **Q:** How to secure cloud access?  
   **A:** IAM roles, MFA, restrict IPs in security groups, use bastion/SSM, rotate keys.
7. **Q:** When use bare-metal virtualization vs cloud?  
   **A:** Bare-metal for predictable perf/legacy needs; cloud for elasticity, managed infra, OPEX model.

**📘 Virtualization & Cloud (Basic Awareness)**

**1. What is Virtualization?**

🔹 **Definition**: Virtualization is the process of creating a virtual (rather than physical) version of computing resources like servers, storage, or network.  
🔹 **Purpose**: Efficient use of hardware, cost reduction, flexibility, isolation of workloads.

**2. Types of Virtualization**

1. **Server Virtualization** – Multiple servers running on one physical machine (via hypervisor).
2. **Desktop Virtualization** – Running virtual desktops (VDI).
3. **Storage Virtualization** – Pooling physical storage into a single logical unit.
4. **Network Virtualization** – Virtual networks using VLANs, VXLAN, SDN.
5. **Application Virtualization** – Running apps isolated from OS.

**3. Hypervisors**

Hypervisor is the software that enables virtualization.

* **Type 1 (Bare Metal)** – Installed directly on hardware (e.g., VMware ESXi, Microsoft Hyper-V, KVM).
* **Type 2 (Hosted)** – Runs on top of OS (e.g., VirtualBox, VMware Workstation).

**4. Virtualization in Linux (KVM / QEMU)**

**🔹 Install KVM**

# Check if CPU supports virtualization

egrep -c '(vmx|svm)' /proc/cpuinfo

# Install KVM and tools

sudo yum install -y qemu-kvm libvirt virt-install bridge-utils virt-manager

**🔹 Start & Enable libvirtd service**

sudo systemctl start libvirtd

sudo systemctl enable libvirtd

**🔹 Create a Virtual Machine**

virt-install \

--name testvm \

--ram 2048 \

--vcpus 2 \

--disk path=/var/lib/libvirt/images/testvm.img,size=20 \

--os-type linux \

--os-variant rhel7 \

--network bridge=virbr0 \

--graphics vnc \

--cdrom /iso/rhel7.iso

**5. Storage Management in Virtualization**

* Virtual disks are usually **qcow2** or **raw** format.
* Commands:

# List VMs

virsh list --all

# Show VM details

virsh dominfo testvm

# Attach disk

virsh attach-disk testvm /var/lib/libvirt/images/data.img vdb --persistent

**6. Networking in Virtualization**

* **NAT Networking (Default)** – VMs can reach outside, but external cannot reach VM directly.
* **Bridge Networking** – VM gets IP from physical network, acts like another machine.
* **Host-only Networking** – Communication only between host and VM.

**7. Snapshots & Cloning**

# Take snapshot

virsh snapshot-create-as testvm snap1 "Before update"

# Revert snapshot

virsh snapshot-revert testvm snap1

**☁️ Cloud Computing Basics**

**1. What is Cloud Computing?**

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

**2. Cloud Service Models**

1. **IaaS (Infrastructure as a Service)**
   * Provides VM, storage, networking.
   * Example: AWS EC2, Azure VMs, GCP Compute Engine.
2. **PaaS (Platform as a Service)**
   * Provides platform (runtime, DB, app hosting).
   * Example: AWS Elastic Beanstalk, Google App Engine.
3. **SaaS (Software as a Service)**
   * Ready-to-use applications delivered over internet.
   * Example: Gmail, Salesforce, Office 365.

**3. Cloud Deployment Models**

* **Public Cloud** – AWS, Azure, GCP.
* **Private Cloud** – OpenStack, VMware vCloud.
* **Hybrid Cloud** – Mix of both.
* **Community Cloud** – Shared infrastructure for specific group.

**4. Basic AWS Examples**

**🔹 Create an EC2 Instance**

# Using AWS CLI

aws ec2 run-instances \

--image-id ami-0abcdef1234567890 \

--count 1 \

--instance-type t2.micro \

--key-name MyKeyPair \

--security-group-ids sg-12345 \

--subnet-id subnet-6789

**🔹 S3 Storage**

# Create bucket

aws s3 mb s3://mybucket123

# Upload file

aws s3 cp file.txt s3://mybucket123/

**5. Cloud Best Practices**

* Always follow **least privilege** (IAM).
* Enable **monitoring** (CloudWatch, Azure Monitor, GCP Stackdriver).
* Use **auto-scaling** for cost efficiency.
* Enable **backups & snapshots** for disaster recovery.

**🎤 Interview Questions (with Answers)**

1. **Q: What is the difference between virtualization and cloud?**
   * Virtualization allows multiple OS to run on single hardware.
   * Cloud uses virtualization to provide services on demand over the internet.
2. **Q: What is the difference between Type 1 and Type 2 hypervisors?**
   * Type 1 runs directly on hardware → better performance (KVM, ESXi).
   * Type 2 runs on top of OS → easier for testing (VirtualBox).
3. **Q: Explain IaaS, PaaS, SaaS with examples.**
   * IaaS → AWS EC2
   * PaaS → Google App Engine
   * SaaS → Gmail, Office 365
4. **Q: What is the difference between public, private, and hybrid cloud?**
   * Public: Shared infrastructure (AWS, Azure).
   * Private: Dedicated infra (OpenStack).
   * Hybrid: Combination.
5. **Q: How do you secure your cloud environment?**
   * IAM policies, encryption, network security groups, monitoring, patching.