DA 410/510 Midterm Solutions

Spring 2026

Question 1

(10 Points) Analyze how moving from "buy capacity" to "rent capability" changes system design decisions for teams. In your answer, discuss planning horizon, elasticity, automation, and at least one cost-management implication. Also provide one deployment scenario where buying capacity is more cost-effective than renting cloud resources, and justify that claim using rough cost estimates.

Sample Answer

Moving from buying capacity to renting capability changes system design because the team no longer has to treat hardware procurement as the main constraint. The planning horizon becomes shorter: instead of forecasting peak demand years ahead and buying servers for that peak, the team can provision resources when needed and release them when demand falls. That shifts design toward modular systems that can be deployed and resized quickly.

Elasticity matters because systems can be designed to scale with demand. A team may choose stateless application servers, load balancing, and externalized state because those patterns work well when machines are created or removed dynamically. In a buy-capacity model, teams often overprovision for expected growth and failure margins. In a rent-capability model, the design can assume capacity is available through an API.

Automation becomes more important because the operational model is based on repeatable provisioning. If infrastructure is rented on demand, manual setup becomes a bottleneck. Teams are pushed toward scripts, templates, and infrastructure-as-code so that instances, storage, and networks can be created consistently.

One cost implication is that cloud reduces the cost of idle hardware for bursty workloads, but it can increase costs for steady-state workloads if rented resources run continuously at large scale. For example, suppose a research lab needs 10 servers running all day for three years. If comparable on-prem servers cost $4,000 each, then buying them is roughly $40,000 upfront, plus perhaps $10,000 to $15,000 for support, power, and replacement parts over time. A comparable cloud setup at roughly $0.20 per hour per server would cost about 10 x 0.20 x 24 x 365 x 3 = $52,560 just for compute, before storage and network charges. In that stable, predictable workload, buying capacity could be more cost-effective.

Question 2

(10 Points) Compare password-based versus public-key authentication for administrative access over SSH. Include at least one security advantage and one operational advantage of public-key authentication, and identify one administrative cost incurred when using cryptographic keys for remote login.

Sample Answer

Password-based authentication requires the administrator to prove identity by supplying a secret directly during login. Public-key authentication uses a key pair: the server stores the public key and the administrator proves possession of the private key.

A security advantage of public-key authentication is that the private key is not transmitted as a reusable login secret. This reduces exposure to password guessing, credential reuse, and many brute-force attacks. It also allows stronger authentication material than typical human-chosen passwords.

An operational advantage is that keys support automation and repeatable administration. Scripts, deployment tools, and remote commands can authenticate consistently without storing plaintext passwords in the same way. Key-based login is also convenient across many systems once the public key is installed.

The administrative cost is key lifecycle management. Someone has to generate keys, distribute public keys to the correct accounts, protect private keys, rotate or revoke keys when staff change roles, and clean up stale authorized keys on servers. At small scale this is manageable, but across many users and machines it becomes real administrative overhead.

Question 3

(10 Points) Compare containers and virtual machines in terms of isolation model, startup profile, and resource overhead. Conclude with one scenario where a container is the better fit and one where a VM is the better fit.

Sample Answer

Containers provide process-level isolation while sharing the host kernel. Virtual machines provide hardware-level virtualization and include a full guest operating system. Because of that, VMs usually provide stronger isolation boundaries, while containers trade some isolation strength for efficiency and portability.

Startup profile also differs. Containers usually start very quickly because they launch isolated processes in an already-running OS environment. VMs are slower because they must boot a guest operating system.

Resource overhead is lower for containers. They do not each carry a full guest OS, so many containers can run on a host with less memory and storage overhead than the equivalent number of VMs. VMs consume more resources because every instance includes its own OS image and virtualized hardware context.

A container is the better fit for a stateless web API that needs fast deployment, frequent rebuilds, and dense packing on shared infrastructure. A VM is the better fit for a workload that needs stronger isolation, a custom kernel environment, or separation between tenants that should behave like independent machines.

Question 4

Read the following Dockerfile and answer the three subquestions below about the state of the resulting container.

```
FROM python:3.12-slim
```

```
ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1

WORKDIR /app

COPY requirements.txt /app/requirements.txt
RUN pip install --no-cache-dir -r /app/requirements.txt

COPY main.py /app/main.py
RUN useradd --create-home --shell /usr/sbin/nologin appuser \
    && chown -R appuser:appuser /app

USER appuser
EXPOSE 8000

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

(a) **(3 Points)** What files and application content will be present in the image under `/app` after the build completes?

(b) **(3 Points)** When a container starts from this image, what command will run, and which network port is the application intended to listen on?

(c) **(3 Points)** Which user will the application run as inside the container, and what earlier Dockerfile steps make that possible?

Sample Answer

(a) Under `/app`, the image will contain at least `requirements.txt` and `main.py`, because both are copied there explicitly. The working directory is also set to `/app`. The Python dependencies from `requirements.txt` are installed into the image, but those packages are typically installed into the Python environment rather than stored under `/app`.

(b) The container will run the command `uvicorn main:app -host 0.0.0.0 -port 8000`, because that full command is specified in `CMD`. The intended listening port is 8000, which is also documented by `EXPOSE 8000`.

(c) The application runs as user `appuser`. That is possible because the Dockerfile first creates the account with `useradd`, then changes ownership of `/app` to that user with `chown -R appuser:appuser /app`, and finally switches the default runtime identity with `USER appuser`.

Question 5

(10 Points) Analyze the risk of storing persistent application state only on an EC2 root volume. Then explain why using a separate EBS volume is a better design when you need data to survive instance replacement, rebuilds, or recovery operations.

Sample Answer

If persistent application state lives only on the EC2 root volume, that state is tightly coupled to the life of one VM. If the instance is terminated, replaced, or rebuilt, the root volume may be deleted or at least remain bound to that specific machine. That makes recovery harder and increases the

chance of data loss. It also makes operations like replacing a failed instance more disruptive because the application data is not independently movable.

Using a separate EBS volume is better because the storage lifecycle is separated from the compute lifecycle. The instance can fail or be replaced while the data volume survives. That supports a core cloud design principle: compute should be replaceable, while persistent state should be portable and durable.

Operationally, a separate EBS volume can be detached and attached to another instance in the same availability zone, and it can also be snapshotted for backup and recovery. This makes rebuilds, migrations, and recovery procedures much more credible than keeping all state on the root filesystem.

Question 6

Answer the two subquestions below.

(a) **(5 Points)** Describe the three-tier model for application design.

(b) **(5 Points)** Explain why database-backed applications often run the database as a separate process or machine. In your answer, identify the role of the data tier and address resource contention, isolation, and operational flexibility.

Sample Answer

(a) The three-tier model separates an application into a presentation tier, an application tier, and a data tier. The presentation tier handles user interaction, the application tier contains business logic, and the data tier stores and retrieves persistent state.

(b) In that model, the database belongs to the data tier. Running it as a separate process or machine reflects the fact that data storage is a distinct architectural responsibility rather than just another part of the web server.

One reason for separation is resource contention. Databases compete heavily for memory, CPU, disk throughput, and IOPS. If the database and application server run in the same process space or on the same small machine, they can interfere with each other and create unpredictable performance.

Another reason is isolation. Separating the data tier reduces the blast radius of failures and makes it easier to secure, monitor, back up, and tune the database independently from the application logic.

It also improves operational flexibility. The application tier and data tier can be scaled, patched, restarted, or moved independently. That is useful because application servers often scale horizontally, while databases frequently require a different operational strategy.

Question 7

A web application running on a single VM is experiencing high latency during peak traffic. Answer the two subquestions below.

(a) **(5 Points)** Explain a vertical scaling solution for this problem. Describe how it would help, what tradeoff it introduces, and when you would choose it.

(b) **(5 Points)** Explain a horizontal scaling solution for this problem. Describe how it would help, what tradeoff it introduces, and when you would choose it.

Sample Answer

(a) A vertical scaling solution is to move the application from the current VM to a larger instance with more CPU, memory, and possibly better network or disk performance. That helps because a single machine can process more requests concurrently and hold more working data in memory. The tradeoff is that cost rises quickly and the system still depends heavily on one machine, so failure and redundancy concerns remain. I would choose this when the application is simple, stateful in a way that is hard to distribute, or needs a fast short-term capacity increase.

(b) A horizontal scaling solution is to run multiple application servers and place a load balancer in front of them. That helps because requests can be spread across several smaller machines, increasing aggregate capacity and improving redundancy. The tradeoff is additional architectural complexity: the system may need stateless application servers, shared storage, better deployment automation, health checks, and private networking. I would choose this when the workload is expected to keep growing or when availability and fault tolerance matter in addition to raw capacity.