DA 410/510 Midterm Review Guide

Spring 2026

How to Use This Guide

- For each section, focus on: definitions, tradeoffs, failure modes, and operational workflows.
- If a topic involves commands, practice typing the commands from memory and explaining why each command is used.
- For "compare" questions, always structure your answer as: model A, model B, key tradeoff, when to choose each.
- For "design/propose" questions, give a step-by-step plan and justify each step in terms of risk, performance, or maintainability.

Slide Topics

Foundations and Service Models

Ideas to understand:

- Cloud is an evolution of shared computing, abstraction, automation, and metering.
- Core shift: buying fixed capacity → renting elastic capability (via APIs).
- Historical arc: mainframes/time-sharing → client-server → virtualized cloud platforms.
- IaaS/PaaS/SaaS/FaaS progression: more abstraction means less infra ownership but less customization.

Virtual Machines

Ideas to understand:

- Hypervisor role and isolation goals.
- Type 1 vs Type 2 architectures.
- Memory/device virtualization and I/O overhead sources.
- GPU virtualization and tradeoffs.
- VM instance building blocks: image, instance type, storage, network config.
- Storage performance metrics: latency, throughput, IOPS.

Linux System Administration Basics

Ideas to understand:

- Kernel vs user space, Linux distributions, package ecosystems.
- Filesystem hierarchy and key directories (`/etc`, `/var`, `/tmp`, `/home`).
- User/group ownership and file/directory privileges.
- Shell command model, exit codes, redirection, pipes, manual pages.
- SSH key authentication, SCP usage, and `tmux` workflows.

Gitea Demo on EC2

Ideas to understand:

- End-to-end deployment sequence from launch to first repository.
- Why each setup command exists (user creation, directory ownership, service file fields).
- Post-install lock-down steps (`/etc/gitea`, `app.ini` permissions).

Containers, Images, Dockerfiles

Ideas to understand:

- Container vs VM isolation and resource profile.
- Image vs container mental model and lifecycle stages.
- Docker Engine/CLI and client-daemon separation.
- Layering, build caching, and reproducibility.
- `Dockerfile` purpose and instruction-level roles (`FROM`, `RUN`, `COPY`, `CMD`, `ENTRYPOINT`).
- Base-image trust/provenance considerations.

Containers and Reverse Proxies

Ideas to understand:

- Docker install chain on EC2 (keys, repo, engine install, verification).
- HTTP method semantics and API testing with `curl`.
- Reverse proxy fundamentals, request flow, and forward vs reverse distinction.

Storage (Linux + EBS + S3 + IAM)

Ideas to understand:

- Block vs character devices and Linux storage layering.
- Device naming conventions and practical command use (`lsblk`, `df -h`).
- Filesystem creation/mount workflow (`mkfs`, `mount`, `umount`, ownership fix-up).
- EC2 lifecycle vs EBS lifecycle and billing implications.

- EBS configuration dimensions (type, size, IOPS/throughput, AZ scope, snapshots).
- S3 object model and EBS vs S3 architectural contrast.
- IAM role-based access on EC2 for AWS API operations.

Cloud Application Design Patterns

Ideas to understand:

- Client-server baseline and database-backed service decomposition.
- Three-tier architecture responsibilities and boundaries.
- Capacity thinking: request latency, core count, memory, bandwidth limits.
- Vertical vs horizontal scaling tradeoffs.
- Load balancing (L4/L7), replication, caching, CDNs.
- Stateful vs stateless systems and sticky-session implications.
- Message queues, logs/metrics, and architecture-level observability.

Assignment-Derived Core Concepts

Assignment 1: Self-Hosted Gitea on EC2

Ideas to understand:

- Strategic tradeoff: self-hosting on IaaS vs managed SaaS for developer platforms.
- Benefits and costs across control, customization, operational burden, and risk ownership.
- Minimum bar for responsible operation of an internet-facing service (availability, access control, recovery, change discipline).

Assignment 2: FastAPI + Docker Compose + Nginx

Ideas to understand:

- Architectural boundaries: public edge vs private internal services.
- Why boundary design changes both attack surface and operational visibility.
- Conceptual distinction between stateless and stateful concerns in containerized systems.
- How state concerns drive storage choices and validation strategy.

Assignment 3: Persistent Gitea (Docker + EBS + S3)

Ideas to understand:

- Compute-state separation as a first-class cloud design principle.
- Relationship between decoupled state and service mobility, replaceability, and disaster recovery.
- What makes backup/restore "credible" in operations: tested recovery, objective evidence, and secure identity handling.

Rapid Self-Test Checklist

1. Can I explain the Week 1 core frame: cloud evolution + service-model responsibility shifts (IaaS/PaaS/SaaS/FaaS)?
2. Can I reason about VM design choices on AWS: instance family, storage profile (latency/throughput/IOPS), access controls, and lifecycle states?
3. Can I explain Linux admin fundamentals clearly (filesystem model, permissions, shell I/O, SSH/SCP, `tmux`)?
4. Can I justify edge/internal boundary decisions and stateless vs stateful choices in containerized web architectures?
5. Can I explain where persistent state lives (root volume vs EBS vs S3) and what evidence proves recoverability?
6. Can I design and defend scaling patterns (vertical/horizontal, load balancing, replication, caching/CDN, queues) for a given workload?
7. Can I answer conceptual questions using explicit tradeoffs, clear assumptions, and a concrete "when to choose what" conclusion?