COMP-4364 Course Syllabus

1 Course Information

Course Name	COMP 4364: Contemporary Systems Architecture
Prerequisites	COMP-4006 (graduate)
Required Textbook	No textbook required

2 Course Description

This course provides a practical, modern understanding of computer systems and the technologies that comprise contemporary infrastructure. The curriculum is structured as a progression, beginning with a foundation in core operating system concepts with a focus on practical mental models. It then builds on this foundation to explore how OS principles are applied in virtualization technologies, such as virtual machines and containers. Finally, the course examines how these technologies are orchestrated and scaled to create the large-scale, distributed systems that power modern cloud computing platforms. The objective is to equip students with a knowledge of the systems they will encounter in a modern professional software development career.

3 Textbook and Materials

There is no required textbook for participation in this class. All materials will be available for students enrolled in the course at `https://canvas.du.edu` and online at various internal and external web sites. You will need a good internet connection and a laptop that meets DU specifications. (See `https://www.du.edu/it/support/how-to/students/laptops`). For technical support in using Canvas, please go to `http://otl.du.edu/knowledgebase/canvas`

4 Course Learning Objectives (Student Learning Outcomes)

The overall learning objectives for this course are understanding and applying **operating system fundamentals**, **virtualization** and **containerization** in real-world scenarios to *solve problems*. Additionally, familiarity and knowledge of **cloud computing fundamentals**, and an ability to identify and analyze use cases for cloud computing architectures.

These objectives are provided in further detail in the following Subsections.

Learning Objectives for Unit 1: Operating System Fundamentals

1. **Understand the Role and Structure of Operating Systems**
2. **Analyze Hardware Components Managed by the OS**
3. **Apply Core OS Abstractions**
4. **Use System Calls and Shell Interfaces**
5. **Understand Processes and Threads**
6. **Implement CPU Scheduling Concepts**
7. **Address Concurrency and Synchronization Issues**
8. **Explain Deadlocks and Their Conditions**
9. **Understand Memory Management and Virtual Memory**
10. **Analyze File System Design and Implementation**

Learning Objectives for Unit 2: Virtualization and Containerization

1. **Understand Principles of Virtualization**
2. **Analyze Hypervisor Types and Techniques**
3. **Explore Linux Container Foundations**
4. **Construct Containers Manually**
5. **Apply Docker for Container Management**
6. **Implement Advanced Docker Features and Security Practices**

Learning Objectives for Unit 3: Cloud Computing

1. **Understand Cloud Computing Fundamentals**
2. **Apply Cloud Architecture and Design Patterns**
3. **Manage Containerized Applications**
4. **Evaluate Serverless Computing Models**
5. **Implement Infrastructure as Code (IaC)**
6. **Explore Major Cloud Platforms**
7. **Perform Hands-On Cloud Deployment**

5 Program Level Goals

Courses in the cyber security MS program, including this one, should contribute to overall program level outcomes for students. This course contributes to the following program level goals:

1. Understand how modern operating systems work (with a leaning towards Unix style OSes). Be familiar with their security components (identification, authentication, access controls, auditing) and how to configure them.
2. Employ common tools and techniques used in the cyber security field.
3. Comprehend and utilize cloud, virtualization and container technologies. Recognize the security implications of cloud and virtualization as well as the potential hazards of using these technologies.

6 Knowledge Units

This course covers the following NSA cybersecurity Knowledge Units:

1. Operating Systems Concepts (OSC, Core Technical CDE)
2. Low-level Programming (LSP, Optional KU)
3. Systems Programming (SPG, Optional KU)

7 Grading Policy

Example only. Grading policy may be different based on instructor.

Assignment	Weight
Homework	30%
Midterm Exam	30%
Final Exam	40%

8 Attendance Policy

Regular attendance is expected. Students are responsible for all material covered in class, including announcements and handouts.

9 Academic Integrity

All work submitted for this course must be the student's own original work. Any instance of plagiarism or cheating will be dealt with according to the university's academic integrity policy. Refer to the Student Rights and Responsiblities, as well as the University of Denver Student Honor code, here: `https://studentaffairs.du.edu/student-rights-responsibilities`

10 Disability Services

If you have a disability that may affect your ability to complete the work for this course, please contact the Disability Services office at: `https://studentaffairs.du.edu/disability-services-progr`

11 Concept Outline/Example Schedule

A detailed outline of the topics covered in this course are provided in this section. Note that assignments/labs are updated based on changes in underlying technology. Additionally your instructor will create different assignments/labs. As such, this is an example of one iteration of the course. While the underlying topics will not change, the specifics will.

Unit 1: Operating System Fundamentals

1. **Introduction to Operating Systems**
   Role of the OS as an abstraction layer and resource manager; kernel vs. user mode.

2. **Computer Hardware and OS Structure**
   Overview of CPU, memory hierarchy, and I/O devices; monolithic OS structure and boot process.
   Example Lab: Compile and modify xv6 in QEMU.

3. **Core OS Abstractions: Processes and Files**
   Processes as programs in execution; file abstraction and directory hierarchy.

4. **System Calls and Shell**
   System call interface, POSIX standard, and shell as a command interpreter.
   Example Lab: Implement a user-level sleep program in xv6.

5. **Processes and Threads**
   Process lifecycle, states, hierarchies; threads as lightweight execution units.

6. **CPU Scheduling**
   Scheduling algorithms: FCFS, SJF, Round-Robin, Priority; modern schedulers like Linux CFS.
   Example Assignment: Implement priority scheduling in xv6.

7. **Concurrency and Synchronization**
   Race conditions, mutual exclusion, semaphores, mutexes; Producer-Consumer problem.
   Example Lab: Compare threading vs. multiprocessing in Python.

8. **Deadlocks**
   Conditions for deadlock; Dining Philosophers example.

9. **Memory Management and Virtual Memory**
   Base/limit registers, paging, page tables, MMU, TLB; page replacement algorithms.
   Example Lab: Modify xv6 page tables for shared memory.

10. **File Systems**
    File attributes, allocation methods (contiguous, linked, indexed), links, journaling, VFS.
    Example Assignment: Recover deleted files from FAT32, NTFS, EXT4 images.

Unit 2: Virtualization and Containerization

1. **Principles of Virtualization**
   Virtualization basics; VMs on a single host; hypervisor requirements (safety, fidelity, efficiency); types: full system, OS-level (containers), process-level.

2. **Hypervisors and Techniques**
   Type 1 vs. Type 2 hypervisors; handling privileged instructions: hardware-assisted, binary translation, paravirtualization.
   Example Assignment: Inspect VMs with QEMU; manage Ubuntu VMs with Multipass.

3. **Linux Containers: Namespaces and Cgroups**
   Kernel features for isolation and resource control; namespaces (PID, network, mount); cgroups for CPU/memory limits.
   Example Lab: Configure and compare cgroups v1 vs. v2.

4. **Containerization in Practice**
   Combine kernel features (unshare, chroot, namespaces, cgroups) to build containers; network namespaces; rootless containers.
   Example Assignment: Manually construct a container environment.

5. **Introduction to Docker**
   Docker concepts: image, container, Dockerfile, Docker Hub; build and run images; multi-container apps and networks.
   Example Lab: Build and inspect Docker image layers; observe build cache behavior.

6. **Advanced Docker and Security**
   Docker Compose for multi-container apps; image security risks; best practices: minimal base images, multi-stage builds, non-root users.
   Example Assignment: Forensic analysis of Docker image layers.

Unit 3: Cloud Computing and Distributed Systems

1. **Introduction to Cloud Computing**
   Cloud model and economic shift (CapEx to OpEx); essential characteristics; service models (IaaS, PaaS, SaaS); deployment models (public, private, hybrid, multi-cloud).

2. **Cloud Architecture and Design Patterns**
   Patterns for resiliency (clustering, failover) and scalability (load balancing, autoscaling); distributed system design patterns (retry, circuit breaker, queue-based load leveling). *Example Assignment: Complete cloud provider "Intro course" (e.g., AWS Academy Cloud Foundations*

3. **Container Orchestration**
   Principles of orchestrating containerized applications; cluster architecture; core objects for deployment and networking.

4. **Serverless Architectures**
   Event-driven computing; models (functions vs. containers); pay-per-use billing; automatic scaling; benefits and limitations.

5. **Infrastructure as Code (IaC)**
   Automating resource provisioning; declarative configuration; workflow for planning and applying infrastructure changes.

6. **Cloud Platform Case Studies**
   Overview of major cloud providers; global infrastructure concepts; compute, storage, database, networking, and identity services; practical deployment exercises.