**Course Name:** Cloud Computing for Data Analysis

**Term:** Spring 2024

**Date:** 01/10/2024 - 04/17/2024

**Time:** We3:20-5:50PM

**Office Hours:** Before and after class and as needed

**Structure:**

* 3:20-4:00 Class Discussion on Reading
* 4:00-4:45 - Teach
* 4:45-5:00 - Break
* 5:00-6:00 - Lab

**Course Description:**

* This course is designed to give you a comprehensive view of cloud computing including Big Data, Machine Learning and Large Language Models (LLMS). A variety of learning resources will be used including interactive labs on AWS. This is a project-based course with extensive hands-on assignments and the language will be exclusively used.

**Learning Objectives:**

Upon successful completion of this course, you will be able to:

1. Summarize the fundamentals of cloud computing
2. Achieve AWS Solutions Architect Certification
3. Master the Rust language for general programming
4. Effectively utilize AI Pair Programming
5. Evaluate the economics of cloud computing
6. Accurately evaluate distributed computing challenges and opportunities and apply this knowledge to real-world projects.
7. Develop non-linear life-long learning skills
8. Build, share and present compelling portfolios using: GitLab, Hugging Face, YouTube, Linkedin and a personal portfolio website.
9. Develop [Metacognition skills](https://www.amazon.com/Know-Thyself-Self-Awareness-Stephen-Fleming-ebook/dp/B08F4ZMZQW/) (By teaching we learn)

**Core Tech Stack**

* Rust
* AWS (AWS Learner Labs, Optional Free Tier Account)
* GitLab
* Slack
* AWS Lightsail for Research (GPU)
* [CodeWhisperer](https://aws.amazon.com/codewhisperer/)

**Communication:**

* Slack
* GitLab

**Media and Labs:**

* [Duke AIML Group GitLab](https://gitlab.com/dukeaiml)
* [52 Weeks of AWS-The Complete Series](https://learning.oreilly.com/videos/52-weeks-of/080232022VIDEOPAIML/)
* AWS Academy Labs and Sandboxes
* [AWS Lightsail for Research (TBD)](https://aws.amazon.com/lightsail/research/)
* [Coursera Courses](https://www.coursera.org/instructor/noahgift)

**Reading Material:**

* [Developing on AWS with C#](https://learning.oreilly.com/library/view/developing-on-aws/9781492095866/)
* [Implementing MLOps in the Enterprise](https://learning.oreilly.com/library/view/implementing-mlops-in/9781098136574/)
* [Practical MLOps](https://learning.oreilly.com/library/view/practical-mlops/9781098103002/)
* [Python for DevOps](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/)
* [The Rust Programming Language](https://rust-book.cs.brown.edu/title-page.html) online book.
* [Know Thyself: The Science of Self-Awareness](https://www.amazon.com/Know-Thyself-Self-Awareness-Stephen-Fleming-ebook/dp/B08F4ZMZQW/)
* [AWS Whitepapers](https://aws.amazon.com/whitepapers) and Certification PPTs

**Projects**

**Project One: Continuous Delivery of Personal Website**

Requirements:

* Website built with [Zola](https://www.getzola.org), Hugo, Gatsby, Next.js or equivalent
* GitLab workflow to build and deploy site on push
* Hosted on Vercel, Netlify, AWS Amplify, AWS S3, or others.

Grading Rubric:

* Website Functionality (30 points)
* GitLab Workflow (30 points)
* Hosting and Deployment (20 points)
* Documentation (10 points)
* Demo Video (10 points)

**Project Two: Continuous Delivery of Rust Microservice**

Requirements:

* Simple REST API/web service in Rust
* Dockerfile to containerize service
* CI/CD pipeline files

Grading Rubric:

* Rust Microservice Functionality (30 points)
* Docker Configuration (20 points)
* CI/CD Pipeline (30 points)
* Documentation (10 points)
* Demo Video (10 points)

**Project Three: Rust Vector Database or Polar DataFrame API**

Requirements:

* High-performance vector database or Polars Dataframe invoked in Rust
* Data ingestion and query functionality
* Benchmarks API Performance

Grading Rubric:

* Vector Database or Polars DataFrame Implementation (40 points)
* Data Ingestion and Queries (30 points)
* Benchmarking (10 points)
* Documentation (10 points)
* Demo Video (10 points)

**Project Four: Rust AWS Lambda and Step Functions**

Requirements:

* Rust AWS Lambda function
* Step Functions workflow coordinating Lambdas
* Orchestrate data processing pipeline

Grading Rubric:

* Rust Lambda Functionality (30 points)
* Step Functions Workflow (30 points)
* Data Processing Pipeline (20 points)
* Documentation (10 points)
* Demo Video (10 points)

**Final Project: LLMOps - Model Serving with Rust**

Requirements:

* Obtain open source ML model
* Create Rust web service for model inferences
* Containerize service and deploy to Kubernetes
* Implement CI/CD pipeline

Grading Rubric:

* Rust Model Serving (25 points)
* Dockerization and Kubernetes (25 points)
* CI/CD Pipeline (25 points)
* Monitoring and Metrics (10 points)
* Documentation (5 points)
* Demo Video (10 points)

**Section Zero: Getting Started with Rust *(Optional, but recommended before you start class)***

* **Section Topics:** 
  + Software Testing
  + System Programming
  + Rust Programming
  + AU pair programming
* **Section Learning Objectives:**
  + Apply Rust's core concepts like ownership, borrowing, and lifetimes to write efficient, reliable, and safe code.
  + Develop, document, test and debug Rust projects using Cargo, Rust Analyzer, and other tools in the Rust ecosystem.
  + Use Rust's advanced features like enums, structs, traits, and generics to build robust applications.
* **Section Description:** This comprehensive Rust programming course welcomes learners of all levels, including beginners and those with some programming experience.
* **Media and Readings:**
  + [Rust Fundamentals Coursera Course](https://www.coursera.org/learn/rust-fundamentals)
  + [The Rust Programming Language](https://rust-book.cs.brown.edu/title-page.html) online book.

**Weekly Schedule:**

**Section One: Cloud Computing Foundations**

* **Section Topics:** 
  + Overview of Cloud Computing
  + Cloud Adoption Framework(s)
  + Economics of Cloud Computing
  + Types of Cloud Services: SaaS, PaaS, IaaS, MaaS, Serverless
  + IaC (Infrastructure as Code) w/ Terraform
  + Continuous Delivery

**Week One: Cloud Computing Foundations Part-Part One**

* **Week Learning Objectives**
  + Understand the core concepts and definitions of cloud computing
  + Compare and evaluate IaaS, PaaS, and SaaS cloud service models
  + Explain the business drivers and economic impacts of cloud adoption
  + Gain hands-on experience with cloud onboarding and account setup
  + Explore DevOps principles and practices for faster software delivery
  + Get introduced to continuous integration and continuous delivery
  + Create a portfolio site to showcase course projects
* **Week Description:** This week provides an introduction to core cloud computing concepts including definitions, service models, adoption considerations, and economic impacts. Students will get hands-on experience with cloud onboarding.
* **In Class Lab-Cloud onboarding lab exercise:** Spin up a new cloud-based development environment using AWS Learner Lab
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Exploring Cloud Onboarding](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/3): Learn the basics of cloud onboarding including account setup, core services, tools, and best practices.
    - [Coursera-Evaluating the Cloud Service Model](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/4): Understand IaaS, PaaS, and SaaS service models and how to evaluate the best fit.
    - [Coursera-Applying DevOps Principles](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/5): Explore core DevOps principles and practices for delivering software faster.
    - [O’Reilly-Python for DevOps-Chapter 6-Continuous Integration and Continuous Deployment](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/ch06.html): Introduction to continuous integration and continuous delivery pipelines.
    - Module 1 - Welcome to AWS Academy Cloud Architecting
  + **Weekly Mini-Project Topic:** Create a static site with [Zola](https://www.getzola.org), a Rust static site generator, that will hold all of the portofolio work in this class. Store source code in a GitLab repo in our Duke GitLab organization.
  + Requirements
    - Static site generated with Zola
    - Home page and portfolio project page templates
    - Styled with CSS
    - GitLab repo with source code
  + Grading Criteria
    - Correct Zola usage: 25%
    - Page templates: 35%
    - Styling: 20%
    - GitLab repo: 20%
  + Deliverables
    - Link to live static site or screenshot showing it running
    - Link to GitLab repo

**Week Two: Cloud Computing Foundations Part-Part One**

* **Week Learning Objectives**
  + Evaluate different cloud service and deployment models
  + Analyze the economics of cloud computing
  + Explore serverless computing concepts
* **Week Description:** This week dives deeper into key aspects of cloud computing including service models, economic considerations, and an introduction to serverless architectures.
* **In Class Lab-Cloud onboarding lab exercise:** Module 2 - Introducing Cloud Architecting
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Exploring Cloud Onboarding](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/3)
    - [Coursera-Evaluating the Cloud Service Model](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/4)
    - [Coursera-Applying DevOps Principles](https://www.coursera.org/learn/cloud-computing-foundations-duke/home/week/5)
    - [O’Reilly-Python for DevOps-Chapter 6-Continuous Integration and Continuous Deployment](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/ch06.html)
    - AWS Academy Cloud Architecting-Module 2 - Introducing Cloud Architecting
  + **Weekly Mini-Project:**  Create a simple AWS Lambda function that processes data.
    - **Requirements**
      * Rust Lambda Function using [Cargo Lambda](https://www.cargo-lambda.info)
      * Process and transform sample data
    - **Grading Criteria**
      * Lambda functionality: 30%
      * API Gateway integration: 30%
      * Data processing: 30%
      * Documentation: 10%

**Week Three: Cloud Computing Foundations Part-Part Three**

* **Week Learning Objectives**
  + Apply IaC best practices
  + Cloud Storage
  + Explain infrastructure as code (IaC) concepts
* **Week Description:** This week introduces infrastructure as code with for reproducibly managing infrastructure**.**
* **In Class Lab-Cloud onboarding lab exercise:** AWS Academy Cloud Architecting-Module 3 - Adding a Storage Layer
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Data Engineering with Rust-Week3-Rust Data Engineering Libraries and Tools](https://www.coursera.org/learn/data-engineering-rust)
    - [O’Reilly-Python for DevOps-Chapter 10-Infrastructure as Code](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/ch10.html)
    - [O’Reilly-Python Devops Two Hours](https://learning.oreilly.com/videos/python-devops-in/61272021VIDEOPAIML/61272021VIDEOPAIML-c1_s0/)
    - [Hello World IAC with AWS CDK](https://learning.oreilly.com/videos/hello-world-iac/10262021VIDEOPAIML/)
    - AWS Academy Cloud Architecting-Module 3 - Adding a Storage Layer
  + **Weekly Mini-Project:** Create an S3 Bucket using CDK with AWS CodeWhisperer.
    - Requirements
      * Create S3 bucket using AWS CDK
      * Use CodeWhisperer to generate CDK code
      * Add bucket properties like versioning and encryption
    - Grading Criteria
      * Correct CDK bucket creation: 30%
      * Use of CodeWhisperer: 30%
      * Bucket properties: 30%
      * Documentation: 10%
    - Deliverables
      * CDK app code
      * Generated S3 bucket screenshot
      * Writeup explaining CodeWhisperer usage

**Week Four: Cloud Virtualization, Containers and API: Virtualization and Containers**

* **Week Learning Objectives**
  + Evaluate different virtualization abstractions.
  + Build solutions with containers.
  + Build solutions with virtual machines.
* **Week Description:** This week explores concepts of virtualization and containers in the cloud. Students will get hands-on experience with deploying containers.
* **In Class Lab-Cloud onboarding lab exercise:** AWS Academy Cloud Architecting-Module 4 - Adding a Compute Layer
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Data Engineering with Rust-Week3-Rust Data Engineering Libraries and Tools](https://www.coursera.org/learn/data-engineering-rust)
    - [Virtualization, Docker, and Kubernetes for Data Engineering](https://www.coursera.org/learn/virtualization-docker-kubernetes-data-engineering)
    - [Developing on AWS with C#-Chapter 5. Containerization of .NET](https://learning.oreilly.com/library/view/developing-on-aws/9781492095866/ch05.html)
    - AWS Academy Cloud Architecting-Module 4 - Adding a Compute Layer
  + **Weekly Mini-Project:** Containerize a Rust Actix Web Service
    - Requirements
      * Containerize simple Rust Actix web app
      * Build Docker image
      * Run container locally
    - Grading Criteria
      * Container functionality: 40%
      * Dockerfile and build: 40%
      * Documentation: 20%
    - Deliverables
      * Dockerfile
      * Screenshots or demo video showing container running
      * Writeup of process

**Week Five: Cloud Virtualization, Containers and API: Microservices**

* **Week Learning Objectives**
  + Evaluate Microservice architectures.
  + Build Microservices with Rust
  + Apply DevOps best practices for Serverless Microservices.
* **Week Description:** This week focuses on microservice architectures, building microservices in Rust, and DevOps considerations.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 5 - Adding a Database Layer
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Data Engineering with Rust-Week4-Designing Data Processing Systems in Rust](https://www.coursera.org/learn/data-engineering-rust)
    - AWS Academy Cloud Architecting-Module 5 - Adding a Database Layer
  + **Weekly Mini-Project:** Serverless Rust Microservice
    - Requirements
      * Create a Rust AWS Lambda function
      * Implement a simple service
      * Connect to a database
    - Grading Criteria
      * Rust Lambda functionality: 30%
      * Database integration: 30%
      * Service implementation: 30%
      * Documentation: 10%
    - Deliverables:
      * Rust code
      * Screenshots or demo video showing successful invocation
      * Writeup explaining service

**Week Six: Cloud Computing Foundations-DevOps**

* **Week Learning Objectives**
  + Build effective and actionable monitoring and alerting.
  + Evaluate different infrastructure configurations that optimize Cloud computing performance.
  + Evaluate best practices for Operations including alerts, load testing and Kaizen methodology.
* **Week Description:** This week focuses on DevOps concepts like monitoring, infrastructure optimization, and operational best practices.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 6 - Creating a Networking Environment
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Python for DevOps-Chapter-7-Monitoring and Logging](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/ch07.html#idm46114710287784)
    - [Coursera-Rust for DevOps-Week-2-Implementing logging and monitoring strategies](https://www.coursera.org/learn/rust-for-devops)
    - AWS Academy Cloud Architecting-Module 6 - Creating a Networking Environment
  + **Weekly Mini-Project:** Instrument a Rust Lambda Function with Logging and Tracing
    - **Requirements**
      * Add logging to a Rust Lambda function
      * Integrate AWS X-Ray tracing
      * Connect logs/traces to CloudWatch
    - **Grading Criteria**
      * Logging implementation: 30%
      * X-Ray tracing: 30%
      * CloudWatch centralization: 30%
      * Documentation: 10%
    - Deliverables
      * Updated Rust code
      * Screenshots or demo video showing logs and traces

**Week Seven: Cloud Data Engineering: Getting Started with Cloud Data Engineering**

* **Week Learning Objectives**
  + Evaluate Microservice architectures.
  + Build Microservices with Rust
  + Apply DevOps best practices for Serverless Microservices.
* **Week Description:** This week provides an introduction to cloud data engineering, including scaling data pipelines and various data stores.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 7 - Connecting Networks
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Applying Key Data Engineering Tasks](https://www.coursera.org/learn/cloud-data-engineering-duke/home/week/1)
    - [Achieving Scalability with Vector, Graph, and Key/Value Databases](https://learning.oreilly.com/videos/achieving-scalability-with/10192023VIDEOPAIML/)
    - AWS Academy Cloud Architecting-Module 7 - Connecting Networks
  + **Weekly Mini-Project:** Data Processing with Vector Database
    - Requirements
      * Ingest data into Vector database
      * Perform queries and aggregations
      * Visualize output
    - Grading Criteria
      * Data ingestion: 30%
      * Query functionality: 30%
      * Visualizations: 30%
      * Documentation: 10%
    - Deliverables
      * Rust code
      * Screenshots or demo video showing successful invocation
      * Writeup explaining service

**Week Eight: Cloud Data Engineering: Examining Principles of Data Engineering**

* **Week Learning Objectives**
  + Analyze best practices in Data Engineering
  + Build Rust command-line tools
  + Apply software testing to Rust tools
* **Week Description:** This week focuses on best practices in data engineering, building Python command-line tools, and testing considerations.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 8 - Securing User and Application Access
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Python and Rust with Linux Command Line Tools-Week1-Introduction to Command-line tools with Python and Rust](https://www.coursera.org/learn/python-rust-linux)
    - [Python for DevOps-Chapter 15-Data Engineering](https://learning.oreilly.com/library/view/python-for-devops/9781492057680/ch15.html#idm46114692728552)
    - AWS Academy Cloud Architecting-Module 8 - Securing User and Application Access
  + **Weekly Mini-Project:** Rust Command-Line Tool with Testing
    - Requirements
      * Rust command-line tool
      * Data ingestion/processing
      * Unit tests
    - Grading Criteria
      * Tool functionality: 30%
      * Data processing: 30%
      * Testing implementation: 30%
      * Documentation: 10%
    - Deliverables
      * Rust tool source code
      * Sample output
      * Testing report

**Week Nine: Cloud Data Engineering: Building Data Engineering Pipelines**

* **Week Learning Objectives**
  + Build a serverless Data Engineering system.
  + Evaluate effective Data Governance in Cloud solutions.
* **Week Description:** This week explores building serverless data pipelines and governance best practices.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 9 - Implementing Elasticity, High Availability, and Monitoring
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Building Data Engineering Pipelines](https://www.coursera.org/learn/cloud-data-engineering-duke/home/week/3)
    - [Coursera-Python and Rust with Linux Command Line-Week4-Rust-AWS-Lambda](https://www.coursera.org/learn/python-rust-linux)
    - AWS Academy Cloud Architecting-Module 9 - Implementing Elasticity, High Availability, and Monitoring
  + **Weekly Mini-Project:** Rust Lambda Data Processing Pipeline
    - Requirements
      * Series of Rust Lambda functions
      * Implement data transformations
      * Orchestrate pipeline with Step Functions
    - Grading Criteria
      * Lambda functionality: 30%
      * Data transformations: 30%
      * Pipeline orchestration: 30%
      * Documentation: 10%
    - Deliverables
      * Rust code
      * Screenshots and architecture diagrams
      * Explanation writeup

**Week Ten: Cloud Data Engineering: Applying Key Data Engineering Tasks**

* **Week Learning Objectives**
  + Develop Cloud ETL (Extract, Load, Transfer) pipelines.
  + Evaluate best practices for Cloud databases and storage.
* **Week Description:** This week explores key data engineering tasks like Cloud ETL, database best practices.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 10 - Automating Your Architecture
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera-Applying Key Data Engineering Tasks](https://www.coursera.org/learn/cloud-data-engineering-duke/home/week/4)
    - [Spark, Hadoop, and Snowflake for Data Engineering](https://www.coursera.org/learn/spark-hadoop-snowflake-data-engineering)
    - [Achieving Scalability with Vector, Graph, and Key/Value Databases](https://learning.oreilly.com/videos/achieving-scalability-with/10192023VIDEOPAIML/)
    - AWS Academy Cloud Architecting-Module 10 - Automating Your Architecture
  + **Weekly Mini-Project:** Serverless ETL Pipeline with Rust Polars
    - Requirements
      * Extract and transform data with Rust Polars
      * Orchestrate ETL steps using Step Functions
      * Load processed data into S3
    - Grading Criteria
      * Rust Polars data processing: 30%
      * Step Functions workflow: 30%
      * S3 data loading: 30%
      * Documentation: 10%
    - Deliverables
      * Rust Polars ETL code
      * Step Functions workflow screenshot
      * Sample transformed S3 data

**Week 11: DevOps Concepts for MLOps**

* **Week Learning Objectives**
  + Apply DevOps practices to MLOps with Rust
  + Deploy transformer models on cloud platforms
* **Week Description:** This week introduces DevOps concepts tailored for MLOps using Rust and transformer models.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 11 - Caching Content
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Implementing MLOps in the Enterprise-MLOps: What Is It And Why Do We Need It?](https://learning.oreilly.com/library/view/implementing-mlops-in/9781098136574/ch01.html)
    - [Coursera Rust LLMOPs-Week1-DevOps Concepts for MLOps](https://www.coursera.org/learn/rust-llmops)
    - AWS Academy Cloud Architecting-Module 11 - Caching Content
  + **Weekly Mini-Project:** Rust Serverless Transformer Endpoint
    - Requirements
      * Dockerize Hugging Face Rust transformer
      * Deploy container to AWS Lambda
      * Implement query endpoint
    - Grading Criteria
      * Transformer packaging: 30%
      * Serverless deployment: 30%
      * Endpoint functionality: 30%
      * Documentation: 10%
    - Deliverables
      * Dockerfile and Rust code
      * Screenshot of AWS Lambda
      * cURL request against endpoint

**Week 12 : Rust Hugging Face Candle**

* **Week Learning Objectives**
  + Combine Rust with Candle ML framework
  + Deploy Hugging Face transformers
  + Apply MLOps best practices
* **Week Description:** This week explores using Rust with Candle ML and Hugging Face transformers, applying DevOps to MLOps.
* **In Class Lab-Cloud lab exercise:**  Module 12 - Building Decoupled Architectures
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera Rust LLMOPs-Week2-Rust Hugging Face Candle](https://www.coursera.org/learn/rust-llmops)
    - [Implementing MLOps in the Enterprise-Chapter 10. Implementing MLOps Using Rust](https://learning.oreilly.com/library/view/implementing-mlops-in/9781098136574/ch10.html)
    - Module 12 - Building Decoupled Architectures
  + **Weekly Mini-Project:** Candle Transformer Model Serving
    - Requirements
      * Candle + Hugging Face Rust model
      * Containerize prediction endpoint (advanced students can create single file binary)
      * Query model endpoint
    - Grading Criteria
      * Model building: 30%
      * Containerization: 30%
      * Prediction queries: 30%
      * Documentation: 10%

**Week 13 : Key LLMOps Technologies**

* **Week Learning Objectives**
  + Customize and fine-tune pre-trained Bert models with Rust
* **Week Description:** This week explores BERT models in Rust for optimized NLP performance.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 13 - Building Microservices and Serverless Architectures
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera Rust LLMOPs-Week3-Key LLMOps Technologies](https://www.coursera.org/learn/rust-llmops)
    - AWS Academy Cloud Architecting-Module 13 - Building Microservices and Serverless Architectures
  + **Weekly Mini-Project:** Fine-Tuned BERT Classifier
    - Requirements
      * Take base BERT model
      * Add task-specific fine-tuning
      * Evaluate model metrics
    - Grading Criteria
      * Fine-tuning implementation: 30%
      * Performance benchmarking: 30%
      * Model analysis: 30%
      * Documentation: 10%

**Week 14 :Key Generative AI Technologies**

* **Week Learning Objectives**
  + Apply generative AI to software development
* **Week Description:** This week, you will learn to utilize GenAI Systems to enhance your ability to write production software and solve problems.
* **In Class Lab-Cloud lab exercise:**  AWS Academy Cloud Architecting-Module 14 - Planning for Disaster
* **Assignments**
  + **Weekly Readings (Will be in class discussion)**
    - [Coursera Rust LLMOPs-Week4-Key Generative AI Technologies](https://www.coursera.org/learn/rust-llmops)
    - AWS Academy Cloud Architecting-Module 14 - Planning for Disaster
  + **Weekly Mini-Project:**  Evaluating Code Generation Tools
    - Requirements
      * Generate calculator code with 3 different tools (AWS CodeWhisperer, An Open Source LLMs like Star Coder, Gemni)
      * Implement generated code
      * Unit test each calculator function
    - Grading Criteria
      * Code generation integration: 30%
      * Unit testing implementations: 40%
      * Comparison analysis: 20%
      * Documentation: 10%
    - Deliverables
      * Generated code samples
      * Unit testing code
      * Report comparing tools

**Course Grade**

* Individual Projects:
  + Project 1: (6.25%)
  + Project 2: (6.25%)
  + Project 3: (6.25%)
  + Project 4: (6.25%)
* Team Project (25%)
* Class Discussion Grade (25%)
* Mini-projects (25%)

**Letter Grades**

* A+: 97-100% A: 93-96% A-: 90-92%
* B+: 87-89% B: 83-86% B-: 80-82%
* C+: 77-79% C: 73-76% C-: 70-72%
* D+: 67-69% D: 63-66% D-: 60-62%
* F: Below 60%

**Notes**

* The final grade will be calculated using the weighted average of the scores for each project.
* The team project will be graded on the following criteria:
* Teamwork: The team worked effectively together and communicated well.
* Communication: The team communicated effectively with the instructor and other students.
* Technical skills: The team used the appropriate technical skills to complete the project.
* Creativity: The team demonstrated creativity in their approach to the project.
* The class discussion grade will be based on the following criteria:
* Participation: The student participated actively in class discussions.
* Thoughtfulness: The student's contributions to class discussions were thoughtful and insightful.
* Respect: The student was respectful of other students' opinions and ideas.
* The course instructor reserves the right to adjust the grading scale as needed. Please note that the total weight of all assignments and class activities equals 100%.

**Graded Items:**

* ***Class Discussion***
  + In-Person Attendance Mandatory
  + 25% of Grade
  + Participation is self-reported in a Google sheet:
    - Each class must either note a good point and the name of the student or mention the point made in class
    - The final self-reported Google sheet must have approximately a 50/50 mix of both. See [here for example](https://docs.google.com/spreadsheets/d/1JEE0Jv_81Js5bNRBLa3WG_jl3E-ZPwF9Q-k2KtdQC-Y/edit#gid=0):
    - The final grade will follow [Nick Eubanks Rubric and the TAs and Instructor will validate the self-grading](https://github.com/nickeubank/unifyingdatascience/blob/master/syllabus/Syllabus_UnifyingDataScience.pdf)
    - Since all students cannot be called upon in class, if you were not able to speak in class, please put your comments in Slack if not called upon in class. Do this AFTER CLASS, since your comments will reflect what actually happened in class vs only what you read.
* ***Weekly Mini-Project***
  + 25% of Grade
  + 100%-Rust (Core Language)
    - Rubric:
      * Must pass lint/test/format in **GitLab CI/CD**
      * Small Tool or service: CLI/Serverless/Microservice
      * Average time 1-2 hours max

**Grading Rubric for Weekly Mini-Project**

See [this template](https://github.com/carrieli15/706-Data-Engineering-Template) for Python) which MUST be mimicked for ALL mini-projects, but with Rust ([ideas her](https://github.com/nogibjj/rust-new-project-template)e). See each week for a breakout of the sub-component of the mini-project, as it contains directions on achieving full credit. For example, the Project Development and Language Use are subdivided into individual mini-project requirements, which is approximately 50% of the grade of the project. ALL projects must have linting, testing, formatting, and a **GitLab** README; this is the other 50% of the grade.

**Project Development (25 points):** The project must be a small tool or service: CLI/Serverless/Microservice.

* Appropriate project scope and complexity: 10 points
* The functionality of the tool or service: 15 points

**Language Use (25 points):** Over 15 weeks, 100% of the mini-projects must be developed

using only Rust.

* Correct and efficient use of Rust: 25 points

**Linting, Testing, and Formatting (25 points):** The project must pass lint, test, and format in Continuous Integration in GitLab.

* Passing lint check: 8 points
* Passing tests: 8 points
* Proper code formatting: 9 points

**GitLab README and Submission (25 points):** The project must include a **GitLab** README with badges proving the Makefile actions work, and the README should be submitted into Slack.

* Clear and accurate README with badges: 15 points
* Proper submission on Slack (including pasted version of README): 10 points

T**otal: 100 points**

* ***Four Individual Project***
  + 25% of Grade
* ***Group Project***
  + 25% of Grade
    - Rubric:
      * Group Feedback Session Required as Part of Final Writeup
        + Each Team Member is required at the end of class to add three positives and three areas to improve on during a Zoom or in-person meeting.
        + Each student then writes a reflection statement about what they learned they could improve during the final project submission.
      * The final project must include the following criteria to pass:
        + Demonstrated load-test
        + README explains project, and includes architectural diagram
        + Must pass test/lint/format in **GitLab** and have badge providing it
        + Must continuously deploy