# Projects

## Corso di Sistemi Distribuiti e Cloud Computing

A.A. 2024/25

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

## Choice and deadline

- Project choice is mandatory
- Deadline: February 14, 2025
- How to choose: fill the Google form made available on January 28 (link on Teams)
  - Team members
  - Chosen project (and description of service, if applicable)
  - Alternative if chosen project is no longer available
- Maximum number of available slots for each project
  - Assigned with FIFO discipline
  - Project change is subject to availability
- Communicate promptly and motivate any change to your team
- Project is valid only for A.Y. 2024/25

# Delivery: what

- What to deliver
  - Link to shared cloud folder or code repository containing:
    - Project code
    - Instructions to configure and run your code
    - Report
    - (if applicable) Dataset of experimental results
  - Write your report with the structure of a scientific article
    - E.g.,: Title, Author(s), Abstract, Introduction, Background, Solution Design, Solution Details, Results, Discussion, Conclusions, References
    - Maximum 8 pages using ACM or IEEE double-column format
      - ACM proceedings templates https://www.acm.org/publications/proceedings-template
      - IEEE proceedings templates https://www.ieee.org/conferences/publishing/templates.html



# Delivery: when

- When to deliver project code and report
  - By September 19, 2025
  - About one week before project presentation
  - No prefixed dates for presentation, we will agree on the date *after (and not before)* you deliver the project
  - Team members > 1: all team members present their project on the same day

# Presentation

- What to present
  - Prepare slides
  - Prepare live demo of your project
  - Team members > 1: Each team member discusses a part of the project (it is your choice how to divide the presentation)
  - Maximum 10 minutes per team member
    - I will stop the presentation when the allotted time is up
    - Live demo is not included!
  - Q&A during and at the end of presentation



# Common requirements for all projects

- Programming language: depends on project
- You are allowed to use supporting libraries and tools to develop your project (not overlapping with project goals!)
  - Be careful: their use should be properly mentioned in the project report
- Your system/service should have configurable parameters (no hard-coded!)
  - Through configuration file/service
- You should test all the functionalities of your developed system/service and present and discuss (if any) testing results in the project report

# Common requirements for all projects (2)

- System/service should be made by distributed components/nodes
  - Allowed centralized services are limited to: service discovery, users logging, and other housekeeping tasks
- System/service should support multiple entities (e.g., concurrent clients) which may contend for shared resources
- System/service should support update to some form of shared state (if any)
- Reqs that depend on project:
  - System/service scalability and elasticity
  - System/service fault tolerance
    - In particular, system/service continues operation even if one node crashes (optional: crashed node recovers after crash and resume operation)



# Grant for cloud services

- Projects require to use Amazon Web Services (AWS) through Learner Lab provided by AWS Academy
  - You received invitation email to your institutional address *(@students.uniroma2.eu* or *@alumni.uniroma2.eu*) on December 15
  - Available budget: 50$
  - Current grant expires on June 12, 2025
    - A new grant will be activated in proximity to expiration date and invitation will be sent to interested students
  - Team members > 1: team or group interaction is not supported; by design, cross-account interaction is not allowed
- Check current list of services and service restrictions
  - README file located within Learner Lab interface
- Plus AWS Free Tier for 12 months (and some always free offering) but requires credit card
  https://aws.amazon.com/free

# Project types

- See first lesson of the course
- Type A
  - Exam score:
    - 50% written exam (plus elective oral exam)
    - 50% project (2-4 students per team)
- Type B
  - Exam score:
    - 75% written exam (plus elective oral exam)
    - 25% individual project



# Projects A: summary

- A1: Microservice application for green smart cities
  - 2-4 students per team, max 4 teams
- A2: Distributed system for computation offloading in the compute continuum
  - 2-4 students per team, max 4 teams
- A3: Distributed application in the compute continuum
  - 2 students per team, max 3 teams

- For all A projects: programming language of your choice
- Goal: favour participation to CINI Smart Cities University Challenge 2025

# CINI Smart Cities University Challenge

- One team for projects A1 and A2 can be selected as local winner to participate to CINI Smart Cities University Challenge, co-located with I-Cities 2025
- Requirement: well-defined and documented API
- Deadlines:
  - By July 11: project delivery
  - By July 31: online meeting of local winner with challenge organizers and teams from other universities to present your project



# Project A1

- Microservice application for green smart cities
  - 2-4 students per team, max 4 teams
  - Application (proposed by team) related to Green Revolution and Ecological Transition PNRR themes, including:
    - Energy consumption
    - Health and telemedicine (e.g., remote patient monitoring)
    - Sustainable agriculture and circular economy
    - Waste management
  - Programming language of your choice
  - At least 2 microservices patterns and 2 microservices using persistent communication
  - Application deployment using preferably Kubernetes (at least Docker Compose)
  - Number of students >=3: support application elasticity and fault tolerance

# Project A2

- Distributed system for computation offloading in the compute continuum
  - 2-4 students per team, max 4 teams
  - System nodes vary from resource-constrained devices at edge to powerful resources in cloud, differences in energy-efficiency and carbon-awareness
  - Applications composed by pipeline of tasks and containerized
  - Manage application deployment in a smart way by selecting *at runtime* system node(s) for task processing (energy, load, network proximity, …)
  - Basic idea:
    - Process low-demanding and latency-sensitive tasks at edge
    - Dynamically offload high-demanding computing tasks to another edge node or to cloud
  - Optional: migrate running tasks to another node
  - If green application, project can be selected for CINI Smart Cities University Challenge 2025



# Project A3

- Distributed application in the compute continuum
- 2 students per team, max 3 teams
  - Design, develop, and deploy a distributed application using microservices or serverless and related technologies
  - Exploit the idea of Compute Continuum in your application
  - Application of your choice but agreed with professor!
    - Some examples

    https://www.agendadigitale.eu/infrastrutture/compute-continuum-nuove-opportunita-di-calcolo-efficiente-e-pervasivo-sfide-e-vantaggi/
  - Go as programming language
  - Select at design time where app components are executed, depending on task
  - Automate application deployment
  - Test application performance

# Projects B: summary

- B1: Fault-tolerant MapReduce
- B2: SAGA: orchestration vs. choreography
- B3: Microservice application of your choice
- B4: Small-scale DHT system
- B5: Gossip-based service discovery and failure detection

- For all B projects:
  - Go as programming language
  - 1 student per team, max 5 students per project



# Project B1

- Fault-tolerant MapReduce using Go and RPCs
  - 1 student per team, max 5 students
  - Goal: extend your Go exercise
  - Programming language: Go
  - Consider distributed sorting as MapReduce application
  - Manage service discovery (do not use a SQL database!)
  - Manage crashes on master, mappers and reducers
  - Test your solution (crashes included!)
  - Deployment using Docker Compose and EC2 instance

# Project B2

- SAGA: orchestration vs. choreography
  - 1 student per team, max 5 students
  - Goal: implement SAGA pattern (on your own, no libraries) in a simple microservice app of your choice
  - Programming language: Go
  - Both orchestration- and choreography-based SAGA https://microservices.io/patterns/data/saga.html
  - Test your solution
  - Deployment using Docker Compose and EC2 instance
  - Kubernetes as optional



# Project B3

- Microservice application of your choice
  - 1 student per team, max 5 students
  - Goal: design, develop and deploy a small microservice application (at least 3 services, not including API gateway)
  - Application of your choice but agreed with professor!
    - Avoiding microservices that only perform CRUD operations on SQL database
  - Programming language: Go
  - RPC communication style
  - Use at least 2 microservice patterns
  - Test your solution
  - Deployment using Docker Compose and EC2 instance
  - Kubernetes as optional

# Project B4

- Small-scale DHT system
  - 1 student per team, max 5 students
  - Goal: design, develop and test a small-scale DHT system different from Chord
    - Suggestions on project description
  - Programming language: Go
  - Test your solution
  - Deployment using Docker Compose and EC2 instance



# Project B5

- Gossip-based service discovery and failure detection
  - 1 student per team, max 5 students
  - Goal: design, develop and test a gossip-based service for service discovery and failure detection
    - Suggestions on project description
  - Idea: use gossip to disseminate changes in a decentralized way
  - Programming language: Go
  - Test your solution (including services added/removed, crashes)
  - Deployment using Docker Compose and EC2 instance