CS 6343: CLOUD COMPUTING
Mid Term Project Milestones

Midterm requirements

- Study the PaaS platforms
  - Discuss how they are different in important features
  - Discuss similar functionalities in different PaaSs and how their APIs are different
- RoboCode should have basic functionalities, fully running
  - Allow user to read, create, edit, save, compile, and play the robot programs
  - Improve the GUI and coding of the current web-based RoboCode program
- Develop and deploy the basic RoboCode on Cloud Foundry (CF)
  - Run the CF on a cluster of servers (can be one simple CF deployment on each machine)
  - Deploy services on CF instances
    - Pick a few Java programs you have written (can be one duplicated into multiple instances)
    - Better to have some Java programs that do perform some significant computation
    - Deploy each on the CF cluster
  - Build RoboCode as a few microservices and deploy them on each CF
  - Include one authentication mechanism that CF supports
  - Include one shared DB for all CF instances (distributed DB preferred, one that CF supports)
  - Include a router package to route the http requests securely to the RoboCode instance in each CF
- Multiple users and security
  - Show that you can create users on CF and assign different privileges to them
    - Some users may not be able to access some services
      - If you only deploy your RoboCode as one service, then you can deploy other services to show your service level access control
  - Store the data at the local database or in a distributed DB that CF supports
    - Each user has a list of robots
    - Each robot has its source code and its executable (will need to add other properties later, such as playing results, ranking, score, etc.)
    - Each user should be able to access her/his own data (robots, etc.) and other users should not be allowed to access them
- Overall system
  - Your RoboCode on CF should be a complete solution, without some detailed components
  - From the user login and management service to RoboCode functionalities to the DB service with some basic security features
- Preliminary design of the overall system you are going to implement as the final project
    - Does not have to be a complete design, but should have been well thought and planned
    - Will be discussed after midterm demo
  - RoboCode
    - Additional functionalities in RoboCode, such as scoring, etc
      - Cloud be a separate gaming microservice that is responsible for general scoring and ranking, etc.
      - Cloud consider other potential functionalities
    - Improved GUI
    - Improved design
  - Design of the multi-tenant access control system
    - The overall admin should be able to create new tenants or delete tenants
    - Each tenant admin should be able to create and delete users

- The admin of a tenant should be able to
  - Define a role hierarchy and the access rights for each role
  - Assign users in the domain to roles
  - Define access rights
    - For example, the access rights to the robots should include read/update/play
- Each user may access the services and data according to their rights
  - All the data created by the users in the domain belong to the domain
  - E.g., the robots created by the users in the domain belongs to the domain
- The system should support cross tenant accesses
  - Each tenant should also have role mapping tables for other tenants
  - Roles from other tenants (not necessarily all tenants) are mapped to local roles for accesses to local data
  - It is possible that a domain has only a single user but multiple roles
- To make the access right assignment on a large number of resources easier, your system should also support the grouping of resources (data and services, especially data)
  - Better to consider resource hierarchy
  - Access rights can be assigned to resource groups
- How to integrate with CF
  - It is above CF
  - Consider the APIs needed by the upper level applications (like your RoboCode)
  - Consider the CF capabilities to be used to achieve the goal
  - Consider the features that are not offered in the CF and need your implementation

- Design of the router package
  - It is above CF
  - The routing should be secure
  - You may consider load balancing, but optional
- Experimentation procedure
  - Goal of your experiments (what you want to measure or observe)
  - Additional code needed for the experimentation
  - How to collect the experimental results
- Planned work distribution
  - Very high level, based on components in the system

Final requirements

- Please refer to the original project description and midterm design description
- More information will be given if additional requirements are needed

Submissions

- Submission guideline
  - Each team only needs one submission through e-learning
  - Report submission
    - Attach the doc file during e-learning submission
    - The file name has to be <team-label>-report.doc or <team-label>-report.docx
    - Do not submit pdf
    - We will let you know your team label, e.g., A1, A2, …
  - Code submission
    - Zip or tar your source code and the deployable microservices
    - Zip file name has to be <team-label>-code.zip
    - Attach the zip file during e-learning submission
  - The workload distribution report

- Just collect the logs you prepared week by week and combine them into one pdf file
  - Should be ordered by week and dated for each week
- Principles in preparing the weekly detailed workload log
  - The items about the workload for each member should be distinct
  - When multiple members are together performing a task, the role of each member should be clearly stated (e.g., if multiple members are together performing a certain installation, then state who actually typed in the commands, who simply made observation, and who provided the guidance on what to do)
  - When multiple members are attempting to resolve the same problem encountered when performing a task, state clearly the problem, the resolution approach(es) each member came up with (even if they were not successful), and the references used for the resolution approach
  - When multiple members are helping implement the same component, state the role of each member (coding, testing, debugging, giving guidance, etc.)
  - Whenever multiple members are performing the same task (in the same or different roles), clearly state the percentage of contribution to the task
    - If the percentage cannot be agreed upon by the team members, state the believed percentage of each and let the TA and the instructor know about the issue
  - Provide any other information that can help us understand your contribution and efforts toward the project
- Pdf file name has to be <team-label>-work.pdf
- Attach the zip file during e-learning submission

Required information in the report (can include more than listed)

- In the cover page, provide team-label, title, and team members
- Introduction
  - Goal of the project
    - What you would offer in your system and why what you offered is important
    - What you should show in the report and/or in a demo of your project (at an upper level, based on the goal)
- Study of related work
  - Summary of related works (in paper or similar products available)
  - How your project is different from or is similar to some existing works
- Approach
  - System architecture
    - Activity diagram (workflow)
    - Architecture diagrams (from high level to low level decomposition of the system)
    - Description of the nodes in the architecture
  - Detailed design
    - For each component in the architecture, provide the list of code files for the component
    - Discuss the APIs of the important components and the algorithms used in some components, etc.
    - Indicate which components have been implemented and which have not been
      - You can simply do color coding or something alike and explain the indicator
  - Implementation details
    - Description of each code file (what it does) and the relations between the code files
    - Description of the system environment
      - Major components in the system environment
      - Your steps for installing these components (No need to give details, just refer to the online resources you used for your installation)

      - Problems encountered during installation of the system environment and how they are resolved
  - Experimental setup
    - Architecture of the experimentation system
      - The system you plan to build and explore in the experimental study can be a black box (or a few black boxes)
    - List of experiments and the goal of each experiment
      - What you plan to learn from each experiment
      - The data to be collected
      - The metrics you plan to use
      - The control parameters you plan to use
  - ...
- Experimental results
  - Clearly state the control parameters and the metrics for measurement in the results
  - Experimentation needs to be thorough and results need to be easy to read (e.g., use graphs and table)
- Installation manual
  - Focus on how to set up your program
  - The url and version number of the open source components needed to set up your system
    - No need to give installation guide for open sources
- User manual, discussing how to use your system
- Team member contribution
  - A high-level summary
  - Can be done based on the detailed system architecture, experimentation architecture and system environment
    - Indicate who have contributed to each component, what type of contribution (installation, design, coding, testing, debugging, experimental data collection, reporting), and at what percentage
- …