Attendance System Service

American University of Beirut

EECE 351
Fall 2020-2021

Tentative Deadline: First Week of December, 2020

1. Project description

You are asked to develop a Web Service named “Attendance System Service”, which will be used to keep track of student attendance during the online learning period. Hence, the purpose of the app is to take attendance data using an automatic process. The instructors would use the web service to view or modify student attendance records during or after an online session. As a result, details like Student ID, date, in-time, out-time, and total attendance time should be stored in a database. The database should be private and only accessed through the service. The instructor is provided by a username and password to access the service. A predefined set of registered instructors are saved in a database, such that only allowed users can access. Different instructors can access the service, such that each instructor, after logging in, can choose one of his registered courses to access that course’s database.

For the purpose of this project, you are required to create 2 databases. One for instructor records (ID, username and password), where the service should search the database for each login (username and password) to match (authenticate) the entered username and password. The database should have al least two instructor records entries. A second database is for the 351 online course, which can be accessed by both instructors. The database columns should be as instructed above: S_ID, Data, In_Time, Out_Time, Total_Time.

It is assumed that the service can access the application (e.g Webex, Zoom, Teams) over which the online session is held, and extract attendance data and automatically store them in their corresponding course/database. So, it is assumed that the database contains the attendance records.

After successful login, the instructor has the option to view attendance records or modify them in case he discovers issues with the automatic retrieval or records. The instructor should be provided, through the service, with a clear and simple user interface to do so.

You will be supplying the functionality of attendance monitoring in two forms: as a server (milestone 1) and as a web service (milestone 2). The server shall be multithreaded and is similar in structure to the sample code that is posted on Moodle. It will be contacted by a client that communicates with it using sockets, as described in class. The web service shall be hosted

on a Cloud Server, and shall be communicated with by clients as also will be described in class. Moreover, the service will be accessed as a Software as a Service (SaaS), as you will also learn.

Hence, the client will behave differently for the two milestones:

- Milestone 1: the client application, will contact the server and provide the user (instructor) a graphical user interface to interact with the server (e.g. to query, update, and add attendance records).

- Milestone 2: the client application will use a URL to access the service using a web browser. It will also provide the user interface to supply the right input to the service (e.g., range of dates and student name) and get back from it the desired results.

General Cloud Deployment Tips:

As you will see in class, deployment of a service on the cloud requires containerizing your service then deploying it over one of the cloud service providers. Major cloud providers include google cloud, AWS, Azure and others. Different service providers offer free deployment trials that you can benefit from in this project.

Grading Criteria

1. **Design and Functionality**

    a. Client and server communication (20%)

    b. Hosting the web service on the cloud (10%)

    c. Client and web service communication (20%)

    d. Database of instructors and student attendance (10%)

e. User interface design (10%)

f. Available options for the instructor to manage the database (10%)

2. **Presentation and Reporting**

a. Source Code Quality (10%)

b. Presentation and documentation (10%)

Deliverables:

1. **Progress presentation and report (around November 15):**

a. Detailed documentation of what work have been accomplished.

b. Road map details (with timeline) of the remaining work.

c. Teamwork summary.

d. Presented and delivered over Teams.

2. **Final Presentation and report and project files (around the first week of December)**

a. Final Presentation.

b. Final report.

c. Complete project files with "How to use and test" document.

d. Teamwork collaboration and work details over the project timeline.

e. Presented over Teams and delivered over both Teams and Moodle.

Notes:

a. All communications and collaborations (message communications, online meeting, and file sharing) between team members should be done over the private channel of each team on Microsoft Teams.

b. Communication statistics will be collected to determine and acknowledge teamwork.

Finally, for any clarifications or questions contact me:

Ali Hussein (ah203@aub.edu.lb)

Appendix:

Cloud-based services are applications or software that are available remotely and hosted on the vendor's server (cloud provider) on behalf of the customer. One easy way to look at this is to still think of the software as being on-premise, but it just happens to also be available remotely. Cloud-based services integrate cloud computing were the service processing load is done in the cloud, while you can still benefit from whatever this service provides as if it was implemented locally. In Other words, Cloud computing is the on-demand availability of service resources, especially data storage and processing power, without direct active management by the user.

Cloud Provider Sample list:

AWS:
Amazon's premier cloud web service, AWS, is divided into three products: Elastic Compute Cloud (EC2), Elastic Beanstalk, and Lightsail. The last two are designed as easy-to-use platforms that utilize the cloud computing resources that EC2 provides.
EC2 has a free tier that offers complete access to a limited set of resources, allowing you to familiarize yourself with AWS' service and configure it to your exact needs. The AWS free tier includes 12 months of access, during which you'll get 750 computing hours a month.

Azure:

Azure offers a free product system very similar to AWS': Some are free for 12 months, others for life, and some are available for trial through the use of credit. Among the cloud resources that are free for 12 months, you'll find Linux virtual machines, storage space, databases, and bandwidth.
Azure's generous free resources allotment, specifically the bandwidth, make it a great place for projects aiming for production-readiness. The resources you'll get for free (1 vCPU, 1 GB RAM, 5 GB space) are actually more than what a lot of shared hosting providers offer on their intermediate to advanced plans!

Heroku: https://www.heroku.com/free:
A free cloud hosting for Developers. A combination of zero cost and ease of use that Heroku becomes such a great option for students and developers. Once you sign up and get your dyno going, you can immediately get a runtime environment up. Multiple languages and frameworks, such as Python, Node.js, and Ruby, are available for deployment without any extra configurations.
The free dyno will net you 512 MB of RAM and 2 process types (equal to 1 vCPU), making it a great platform for school projects and general experimentation.

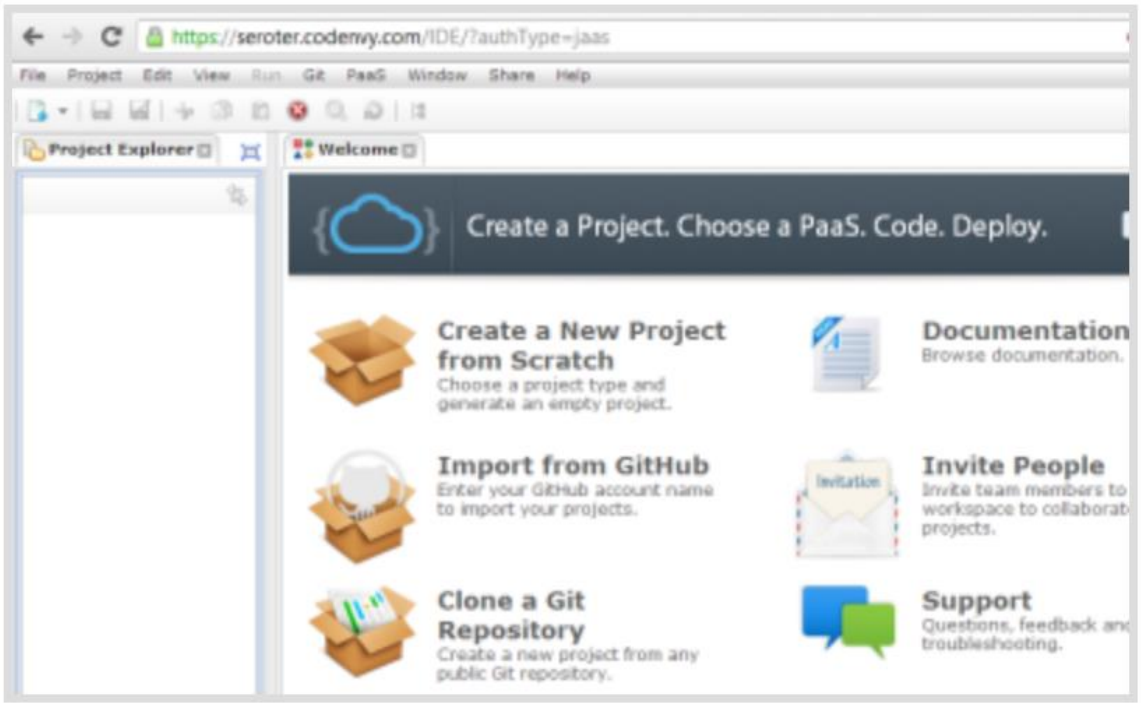
Another simple method to deploy your application on the cloud with the ability to access it is shown in the following:

https://www.ctl.io/blog/post/the-simplest-way-to-build-and-deploy-web-applications-to-the-cloud/

This provides Platform as a Service (PaaS) in order to deploy your SaaS. This platform provider a free online IDE to implement your code using a language of your choice and deploy it using one of multiple free cloud vendors such as cloud Foundry.

Developers can push their application to Platform as a Service in a number of ways. While most developers are familiar with command line interfaces and GUI tools that run on their desktop, a new crop of cloud-based integrated development environments (IDEs) can make PaaS deployments even simpler. Cloud IDEs offer excellent collaboration capabilities, easy accessibility, and "no-touch" setup.

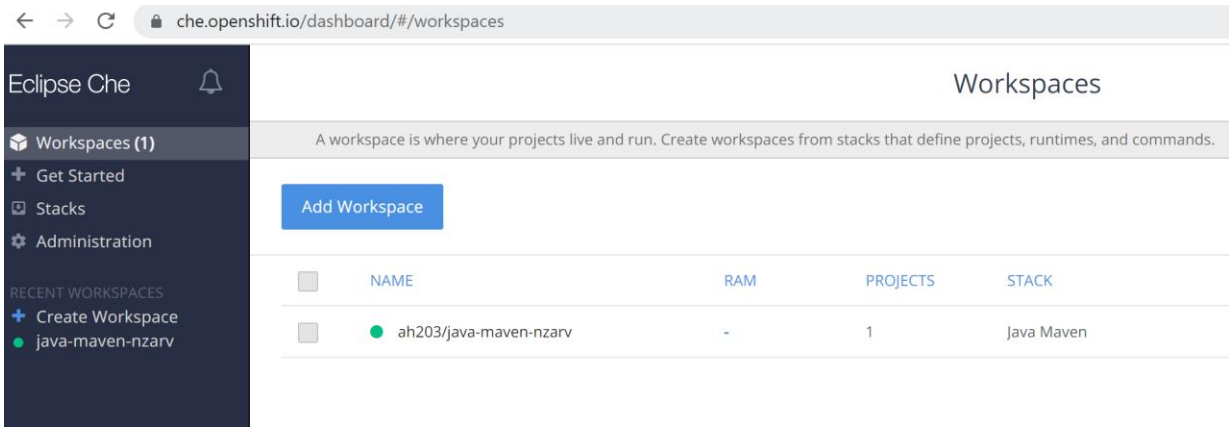
One such cloud IDE is Codenvy. This tool works natively with Cloud Foundry, making it easy to build Java/Ruby/Python/PHP applications and then push them to Platform as a Service.



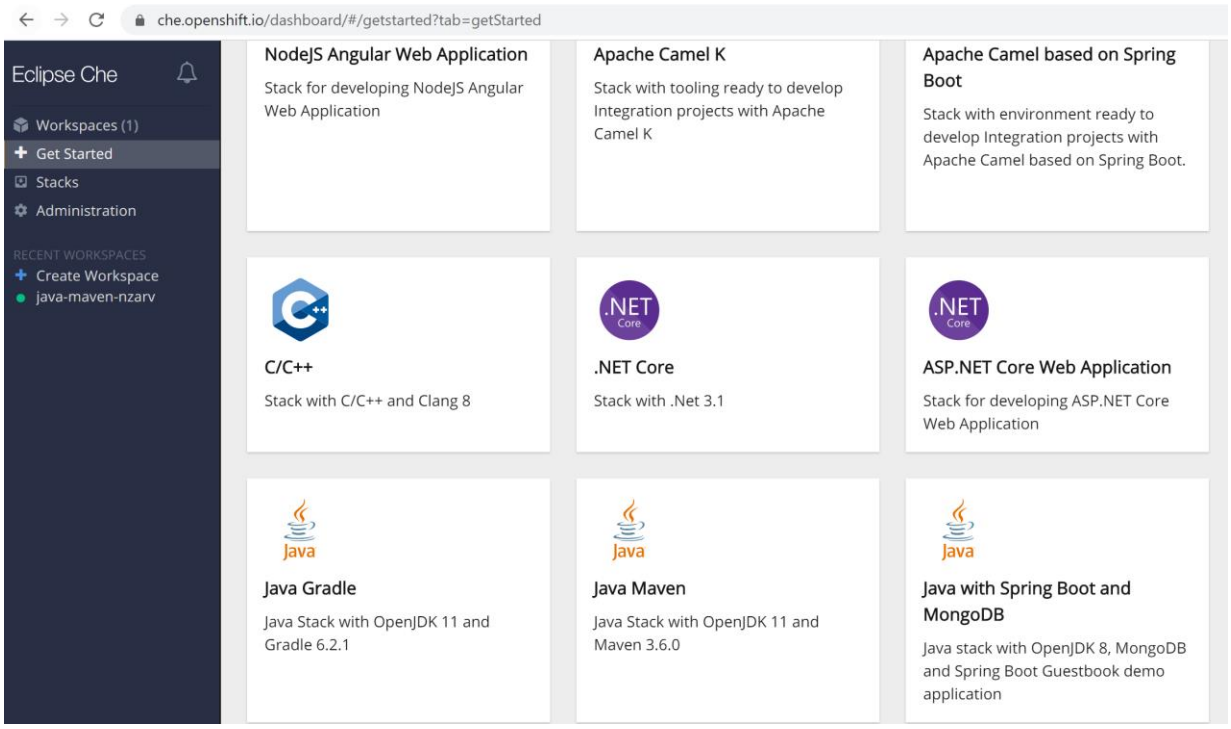
Codenvy uses a handy “new project” wizard experience to help the developer choose which programming language to use, and then which (supported) PaaS to push to. In the short animation below, observe how I created a new Java Spring project, chose Cloud Foundry (Platform as a Service) as a destination, finish the wizard and publish the application to Platform as a Service.

Upon signing up for a free account, you will be given a workspace in a continarized medium in the cloud. You can the create a new project and paste your service and build it in the cloud. You also have the option to upload your application.

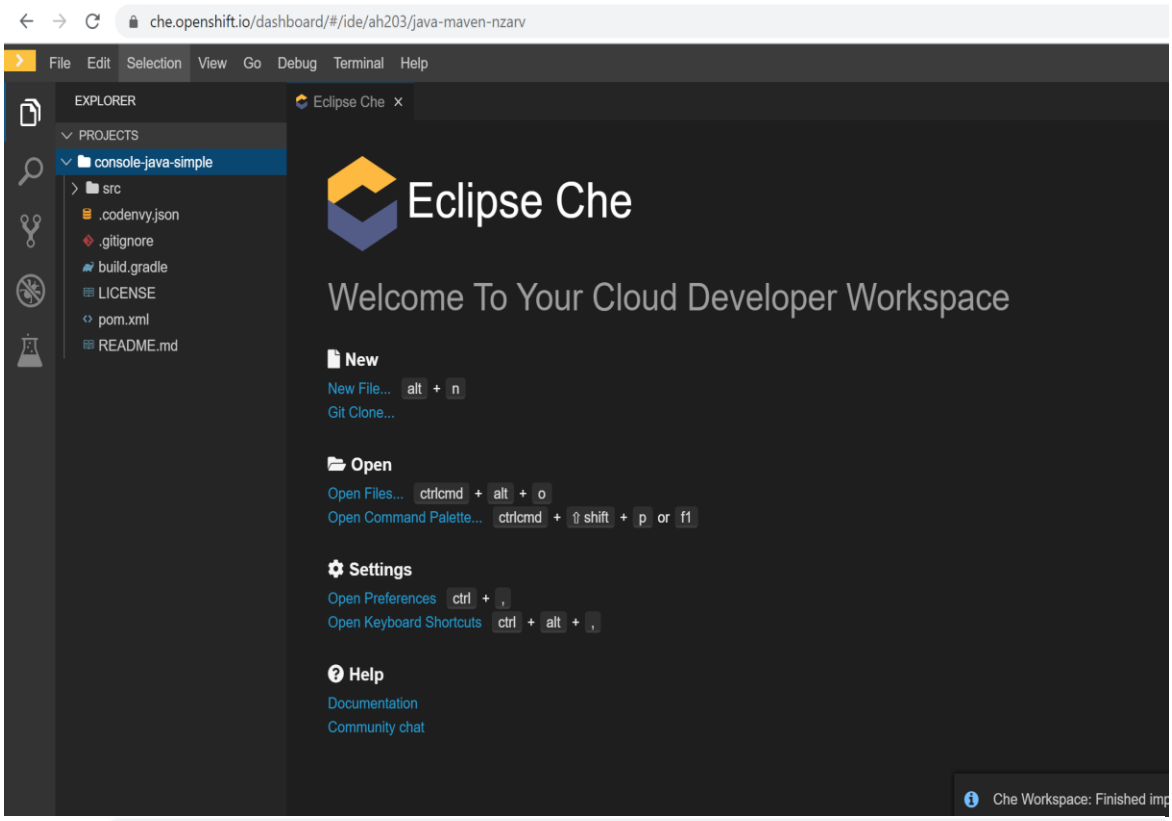
New Workspace:

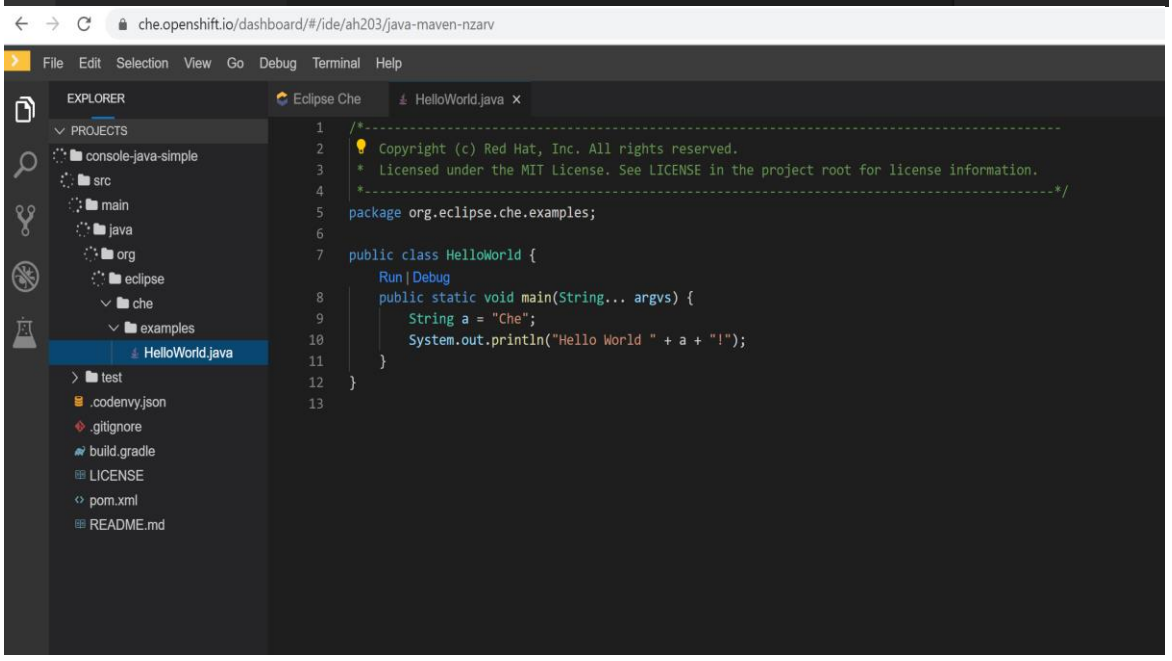


Getting Started:

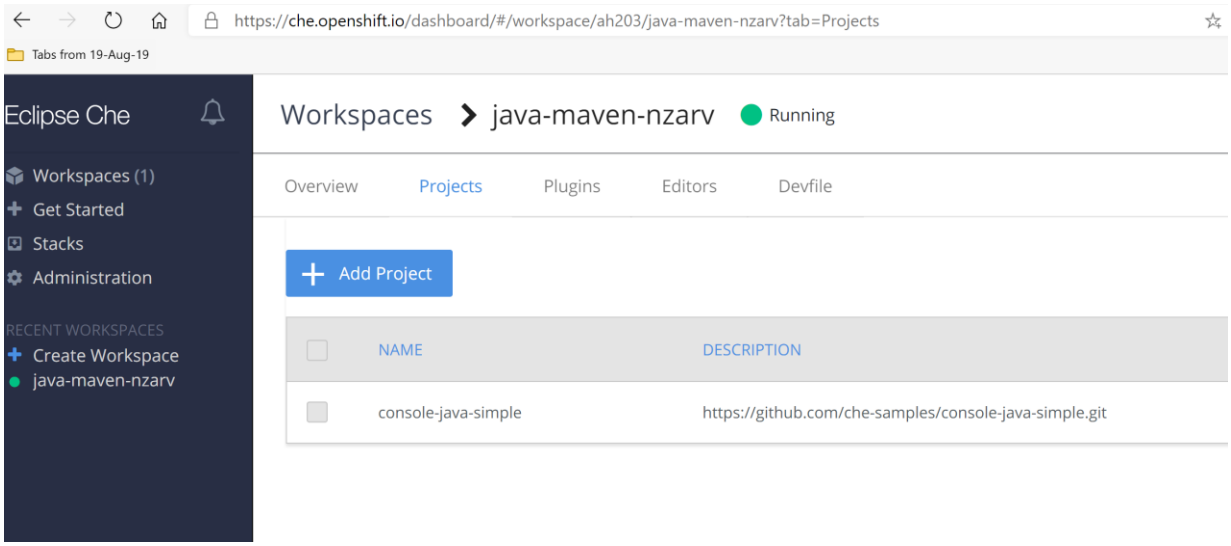


New Project (Hello World):

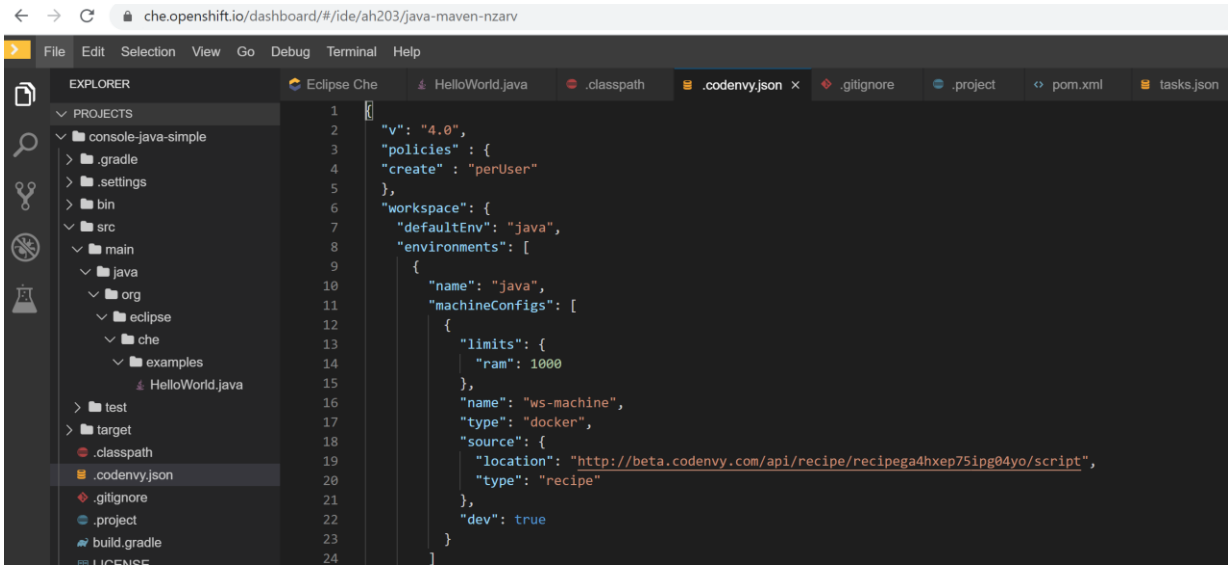




Add New Project:



In the .codenvy.json you can fide the location of your container in the cloud:



After creating and building your application you will find the public url of your service in the workspace Overview of the project.

Resources:

https://docs.openshift.com/dedicated/3/getting_started/access_your_services.html