CSC 648/848-04 Software Engineering – Spring 2020
Milestone 0

1.Introduction

The high level purpose of this milestone is for student teams to choose, install and prepare IT infrastructure for the development of team final project in CSC 648-848. Specifically, in M0 must achieve the following objectives working together:

- Configuring GitHub to enable team SW development
- Choosing a SW stack and delivery platform and service provider from given list
- Getting your choices from above approved by Class Instructor
- Creating a joint WWW page with info about team members using above infrastructure (this page will be graded but will also serve as ABOUT page for your team application)

The deployment server and ***all*** team applications for final team project must be running on a remote host. ***Localhost hosting will not be accepted.***

In this class the instructor will play the roles of **class CTO, class CEO and Customer.** Normally, those roles are assigned to different individuals in companies.

The work on M0 shall commence only after the teams have been formed and basic accounts set up – you will get e-mail on this from class instructor

Note that your team will be choosing most of the IT infrastructure for developing the team project (team choices will have to be approved by class instructor. After choosing the infrastructure and after it is approved all team members have to follow M0 and learn how to use it. M0 and class IT involves infrastructure designed to give teams more freedom but more responsibility in maintaining your infrastructure. Hence, there may be problems. Please contact class Instructor jortizco@sfsu.edu if you encounter any problems that you cannot solve yourself or with your team.

Your responsibility:

Since CSC 648-848 is a capstone senior course students are expected to learn necessary tools themselves and in working with their team members. Use of on-line resources and

asking around is encouraged. Class instructors will only help in case there are problems, and only at high level - coding details will be the duty of students to figure out.

The IT environment and setup we envision for the class is close to what you will be required to know to be successful SW engineer, so M0 is a great experience to improve your marketable skills. While some of the work is to be done *individually*, you must work together with your team to complete the task.   Note that your learning will comprise: a) individual usage of tools; and b) most important: how to use the tools in team setting (e.g. code merge etc.).

Your choices of IT stack and hosting platform will be from a limited list: this is how it is in most companies and it also simplifies our supervision and support to you.

<u>M0 carries 10 grade points.</u> Information about grading can be found at the end of the document. Again, each team member must complete his/her M0 tasks but to accomplish this each team member must also work with the rest of the team. **<u>This means that you can help each other, ask for help, work in pairs and are in fact encouraged to do this to not only help with M0 but also build the teamwork.</u>**

Your first task is to read through the entire document. Then, follow the steps starting with Section 2 “Setting up GitHub”. You will also give access to instructors, so they can monitor your work.

In section 3 we then describe actual M0 task where you are first to select deployment server and SW stack, then install them, and finally create a joint team web page. This WWW page can be used in your final project too (e.g. ABOUT page on your final team project WWW site). Section 4 describes M0 grading.

Again, M0 is a great vehicle to help you learn a very important part of SW engineering work which is setup, use and maintenance of professional development and deployment infrastructure.

<u>Note that we use *group* or *team* interchangeably.</u>

<u>Please read this document carefully and follow all the required steps.</u>

You must deliver M0 on schedule. Any delays must be approved by class instructor and asked for via e-mail at latest 24 H before the M0 deadline.

2. Setting up GitHub for team SW development

2.1 GitHub Team (Private) Repo

The purpose of this part of the exercise is for your team to set up the team private GitHub repository that is going to be used for storing your team's project (SW, documentation, formal milestone deliverables etc.) and be accessible only to the team members and instructors. Only one member needs to set up the private repository.

As a first step, all team members need to have their own GitHub account. This is mandatory, NO EXCEPTIONS - see step 1 below (you need to have your own GitHub these days anyway).

Creating GitHub Account (if you do not have one).

1. If needed, create your own GitHub account. If you already have an account, you can skip this step.
    a. When creating the GitHub account, select that you will have public repos. DO NOT SELECT private repos, or you will be asked to enter credit card information.

Creating GitHub Team private repo.

1. Select one team member to create the private repo.
2. Selected team member from step 1 uses the link that the instructor will post on iLearn to create the class team repository
3. A chosen team member (team leads must ensure this happens) will then ADD ALL MEMBERS of the team to the private repo. Each invited team member has to accept the invite. (**For each non-invited team member, 1 point will be deducted from M0 Grade for the whole team to promote teamwork.)**
    a. Simply inviting the team member is not enough. They need to accept the invite. Non-confirmed invites will get the same penalty as no invite for those who do not follow up.

Team members are strongly **encouraged to** practice creation of branches and code merge, which turned out to be occurring problems with previous teams. Please consult on-line resources for **GitHub** best practices on this topic and the slides we will post.

The team repository created with the above link will be used to store your team's source code and milestone documents. This repo is already private and configured for you to begin

work right away. *The instructor is* already attached to the repo. Only your team members need to be added. There is *NO* need for a paid subscription for the private repo. This have been taken care of for you. Info is in readme file, may include some necessary modifications to make your repo is team specific..

Appoint GitHub master

Teams are also requested to appoint a GitHub master who will help organize and maintain GitHub according to best practices – M0 is a good time to do this. This can also be back end or team lead.

3. M0 tasks

This section describes how to develop actual M0 tasks, after you have completed all the previous steps.

Task 1: Selecting Server/Platform Provider, Software stack and Deploying Your Team Web Application

The purpose of this task is for you as a team to do the following:

- Select a *Server/Platform provider*
  Some possible choices are below:
  - Amazon AWS
  - Google Compute Engine
  - Please refrain from using ***Heroku***. While simple, it has caused issues for many teams in the past that have delayed application development that has affected teams final grade.
- Select *a Software Stack*. An example would be a LAMP Stack. This includes: a) the OS choice; b) web server choice; c) database choice; d) and server-side language choice, and e) any other important technologies needed your Software Stack.
- The server-side language for your web application must be one of the following:
  a. Python
  b. JavaScript
  c. PHP

  You may ***NOT*** use another server-side language that is not listed above. If you strongly disagree you may contact class instructor with your suggestion.
- For databases we only allow SQL types of databases and recommend MySQL

When using AWS or Google Compute Engine please refrain from using their app features. These auto-deployment tools cause some issues for students and in some cases caused students to be charged (Most of it not all were refunded). If you do use them, use at your own risk we are not responsible for accrued costs for running the server.

When deciding on the technology to be used in your Software Stack, besides functionality (a common factor to start from), you should keep a few additional factors in mind when comparing technologies side by side. Some key factors may be:

- How often is the technology updated?
- How well is the technology supported?
- How mature is the technology?
- How well is the documentation?
- How good is the community using this tool?
- What features do is offer compare to other technologies.
- And very important: How easy is it to use/learn for all your team members given specific class schedules?
- Please avoid choosing the technologies where there is no expertise in the team. Remember: your focus is to deliver the application and NOT necessarily learn/use newest technologies

Note: your choices must be made such as not to incur any costs, which is possible today given many free offerings on the market.

Task 2: Getting Server/Platform and Software Stack from Task 1 Approved:

The task of selection has to be done very soon after formal start of M0 (TBD e.g. within a week as per schedule posted).

When your team has decided on a software stack, **team lead** needs to email the class Instructor (cc'ing the class CEO Prof. Petkovic as well) detailing your software stack. The email **must** have the following format:

- Receiver: Class Instructor (jortizco@sfsu.edu)
- Subject: CSC 648-848 Spring 2020 Section N Team M
    - **(N is section; M is team number)**
- List the Technologies used in your software stack.
- This includes the following:
    - Server Host, Instance size (CPU and RAM)
    - Operating System and Version Number
    - Database and Version Number

        - ***Only SQL based Databases are allowed***
    - Webserver and Version Number
    - Server-Side language and Version Number
    - Also list any technologies or packages you will need that you think is important. (you can exclude things like git and ssh). There is no wrong answer here, just list what you can. The more the better, it'll help me determine how sound your software stack is.
- Then list each member's familiarity with the backend language. A sample is given in the Appendix of this document.

MAKE SURE TO FOLLOW EMAIL FORMAT. MOST IMPORTANTLY USE THE CORRECT SUBJECT HEADING. ANY EMAILS SENT WITH WRONG SUBJECT HEADING WILL BE EITHER RETURNED OR MISSED(IGNORED).

A sample email has been given in the Appendix of this document. Please do your best to follow this format. This will ensure a speedy response. If the class instructor has any questions about your software stack, make sure that your team replies promptly. Delayed responses can delay Software Stack approval.

Class Instructor will send formal checkpoint request to solicit your e-mail on completion of task 2 in order to ensure this task is completed on time.

Once your software stack has been approved via e-mail from class instructor you must begin installing and configuring your server and completion of M0 immediately.

Task 3: Installing and Configuring Approved Software Stack.

After your server/platform provider and software stack have been approved by e-mail from Instructor it is now time to begin installing everything.

There will be no detailed instructions given for this as there are too many possible configurations. But you may follow these simple steps:

1. Start server Instance with your server host
2. SSH/log into your server
3. Update your server
4. Install Webserver
5. Install DB
6. Install Server-Side language
7. Install remaining needed packages
8. Make any needed configurations.

Some notable Stack installation instructions:

- AWS w/ Node

- LAMP Stack AWS EC2
- LAMP Google Cloud Platform

This may not be the most detailed set of instructions be it gives you an order of items to work through. **Please use on-line resources and follow the documentation!** If you run into any issues there is help available. You may ask anyone in your team, your class, and class Instructor.

THE CLASS CTO MUST BE GIVEN ROOT ACCESS TO THE TEAM SERVER (SSH ACCOUNT) AND DATABASE (DB USER WITH FULL PRIVILEGES). THERE IS NO EXCEPTION TO THIS.

Task 4: Create a Team Website and ABOUT page

The purpose of this part of the exercise is to get you to work individually to create your own webpage within your team's framework context, and then to work with your team to join these pages together using your teams GitHub account and chosen framework into a single site. We recommend that you in fact create ABOUT WWW page which introduces the team members. This can then be part of your final application and is great for your portfolio.

Everyone in your team should clone the team's private GitHub repository into his/her individual shell account or onto your local computer, and within the chosen team framework create their own WWW page that at a minimum displays their name and their or some other image (if you are comfortable it is good idea to use your own photo which is useful for your ABOUT page and portfolio. **Photo is optional**). Please make sure that you have the right to use any posted photo. This work must be completed by individual student and then pushed to the team's private GitHub repository. The file(s) created/added should then be merged into the team's private GitHub repository. **The website must be deployed onto the team's remote server setup as in Task 3. ABOUT page and student individual pages must also be deployed on the server.**

Make sure your individual test environment is identical to the one on the server, see end of this section.

One of your teammates will need to modify your framework's home page to point to all the different team member pages (all residing at the server). The finished site must look something reasonably close to the example in Figure 1. The user 1...user 6 buttons should be replaced with buttons pointing to the individual pages labeled with the username of the person who created that page. **The website shall be deployed onto the team's group shell account**.

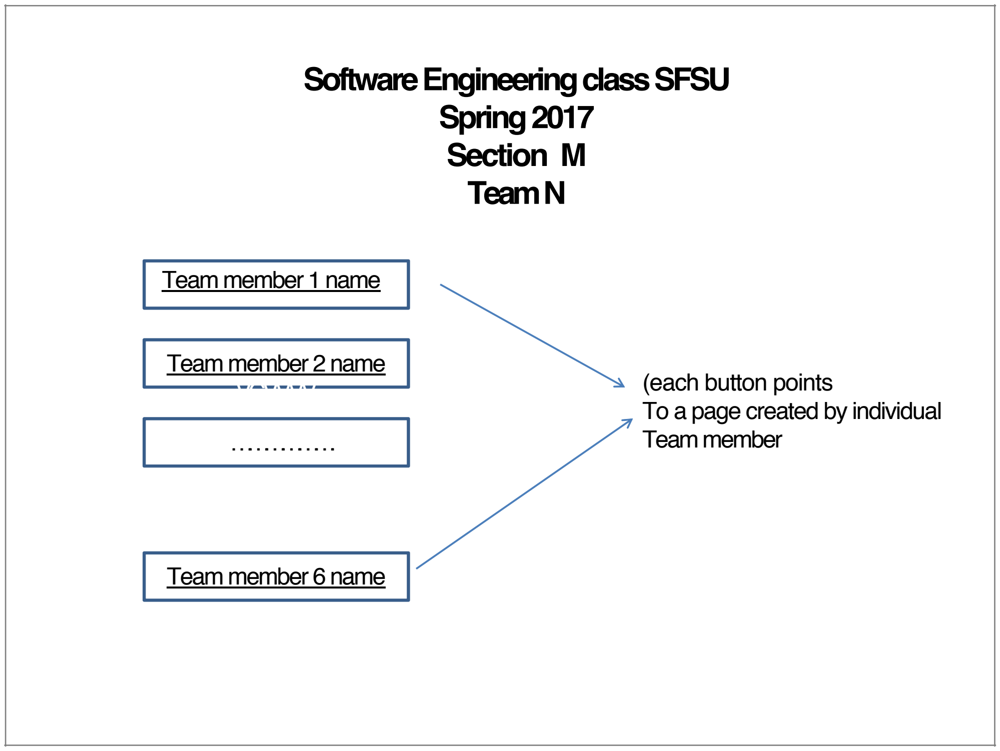


Figure 1. Example mockup of Milestone 0 team home page.

This single ABOUT page and individual team member pages) site must **<u>be served from the remote server you setup in Task 3.</u>** This is the same way you will deploy your final app, hence this is useful to make sure you learn and test it, especially how to deal with branching and merging code. Every team must understand how your team's application is deployed and managed.

NOTE on student individual test environment:
Every student should have a correct test environment on their working PC e.g. **make SURE the same steps you do to configure your deployment sever, you also do for your test environment.** This has caused a lot of issues in the past. For example, students would setup test environments on their personal laptops, but the environment would not match what the deployment server had. If your deployment and test environment differ too much, then deployment becomes a hassle especially running the app on the deployment server.

For example, last semester the deployment server was running Node Js with NGINX webserver in front of the Node apps. Many students would have test environments on their PCs mimicking their node app on their PC but forgot to add NGINX as well. Forgetting NGINX caused the following problems:

- Apps would not even start after being pulled from GitHub
- Apps paths did not work

- Apps routes did not work.
- Apps were missing dependencies obtained from test environment.
- Apps would perform differently compared to test environment.
- And the list goes on.

A team from a previous semester had set up a test environment on a virtual machine that I given one of their teammates for another class. That teammate configured the VM to match their production server and then gave it to their team so they can use. It was not a requirement but it helps ensure your test environment is as close to your production as possible. The tricky part is managing architecture changes and pushing these changes to each individual VMs.

NOTE about using deployment server: most choices will be on some form of a cloud service. To use cloud service properly one must understand how they charge. For example, it may happen that unintentionally students create many instances of SW or applications and have them running silently (unless they are stopped), thus incurring extra costs. It is imperative to avoid this and we strongly suggest that somebody in the team be in charge of monitoring cloud usage. Also, cloud storage is very expensive so make sure you use it minimally and understand how it is charged (class project requires only minimal storage). Keep in mind that if your team uses app engine for google or the AWS equivalent stopping an instance is NOT enough. Killing an instance will only cause it to respawn since the app is active. To stop this the APP itself needs to be stopped or deleted which then will terminate all instances associated with that APP. Failure to do this may incur large costs for the service

4. Submission and Grading of M0

4.1 Submission of M0 for grading

All projects will be inspected and graded on the due date TBD. The key for grading will be your team's correct installation and configuration of software stack, correct setup and usage of GitHub, and the deployment of your About Me Web Application from ***YOUR CHOICE OF REMOTE SERVER (section 2).*** Emphasis for M0 grading is on correctness (not so much on UI design of the final team page) and proper usage of development tools and deployment method.

Once you are done and ready for grading you must send e-mail to the instructor with the following subject line, e-mail body and the attachment (**you MUST follow this protocol**)

E-mail subject line:
"CSC 648-848 Spring 2020 M0 Section N Team M"
(N is section; M is team number)

E-mail body:
In the e-mail say that M0 is ready for grading. Provide a link to your application. Please do not send emails with empty body messages.

E-mail Attachment:
Cut and paste the form below and send it as PDF e-mail attachment with file name as **"CSC 648-848 Spring 2020 M0 section N Team M DOC".** The below form is used for submission of needed information for the current milestone. The below table is used to help access certain parts of your web application. Please make sure the information submitted is accurate and up to date to ensure grading of the milestone is completed in an efficient manner.

San Francisco State University

CSC 648 - 848

Milestone 0 Submission Form

Section M Team N

Item	Credentials
Website URL	
SSH URL	
SSH Username	
SSH Password/ Key	
Database URL	
Database Username	
Database Password	

Addition information, anything to help with executing or grading of the current milestone

It is important that the information in the above tables is accurate. If the form is filled out incorrectly or not at all your team risks losing points. Class instructor cannot grade what cannot be accessed. If your team is using SSH keys they can be attached in the email as well.

You must deliver M0 on schedule. Any delays must be approved by class instructor and asked for via e-mail at latest 24 H before the M0 deadline.

4.2 Grading of M0

This assignment is worth 10% of your class grade, as indicated in the syllabus. The grade for this assignment will be determined according to the following criteria:

Category	**Points**
Correctly Installed and Configured Software Stack	2
Given Class CTO root Server and Database access	2
Correct use of Git and GitHub	2
Correctly Filled out M0 Submission Document.	2
Correct team WWW page functionality, deployment and proper usage of team's Software Stack for creating web page	2
Total:	10

Appendix

i. Sample Email for Software Stack Approval

```
Hello Class CTO

     Below is a list of the technologies used in TeamNN's
software stack:
     Sever Host: Google Compute Engine 1vCPU 2 GB RAM
     Operating System: Ubuntu 16.04 Server
     Database: PostgreSQL 10.1
     Web Server: NGINX 1.12.2
     Server-Side Language: Python
     Additional Technologies: Web Framework: Flask
                              IDE: PyCharm InteliJ,
                              Web Analytics: Google Analytics
                              SSL Cert: Lets Encrypt (Cert Bot)
                              SASS: 3.5.5



          Member's Familiarity with Server-Side Language on a
scale of 1 to 5,
          with 5 being very familiar and 1 being never used it.

          Jane: 5
          Jack: 2
          John: 3
          Joe: 1
          Jill: 4
          Jake: 5


          Then any extra information you think is necessary.
     Best,
          TeamNN
```

ii. Connecting to MySQL Database via cmd line or Workbench

Connecting to MySQL Server via MySQL Workbench

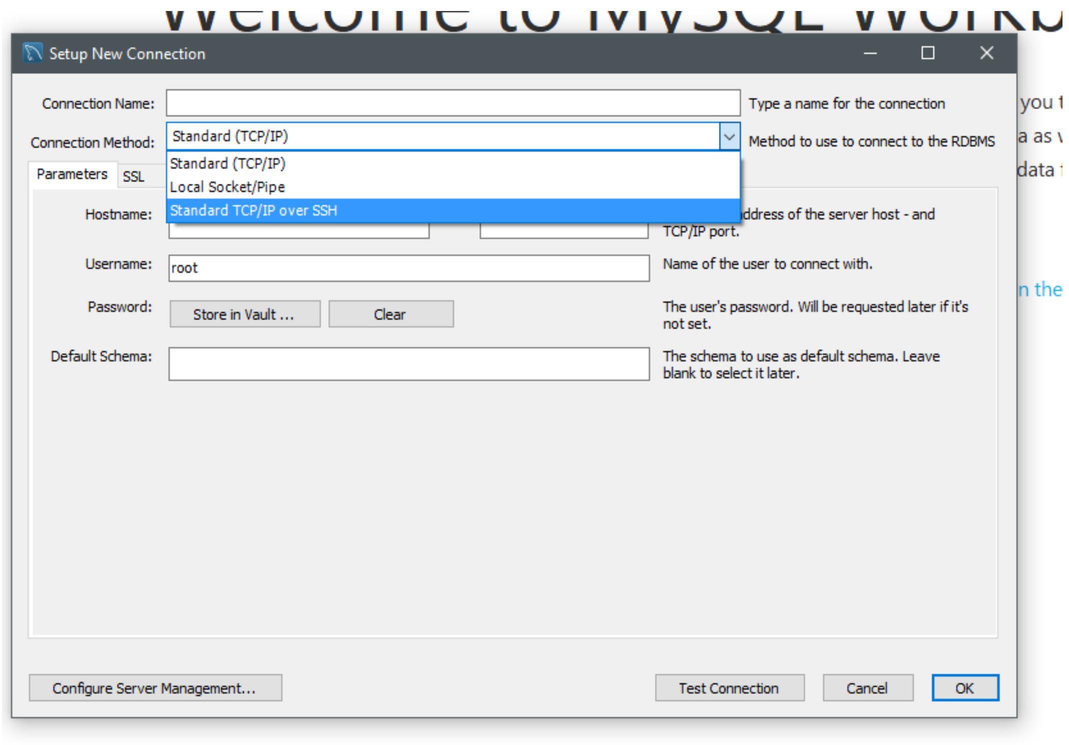
There are two ways you may access your database. One way is logging into the remote Google Compute Engine server and accessing your database via command line with the following commands:

```
mysql -u username -p
(where username is your database username and password is your db password)
connect db_name
(where db name is the database name you wish to connect to)
```

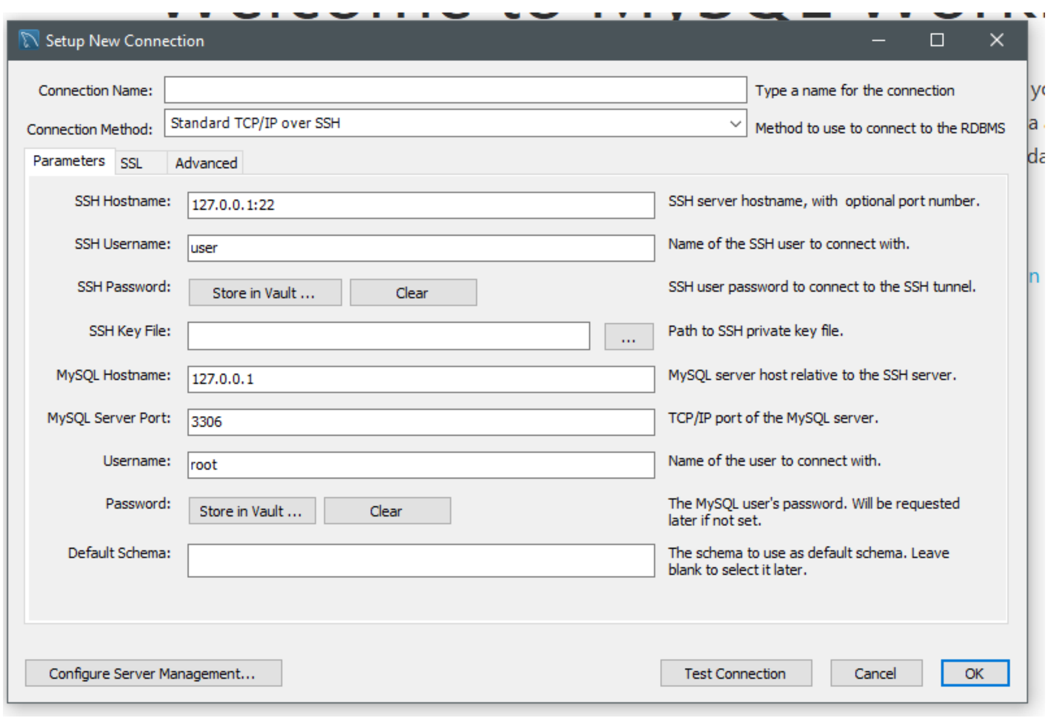
At this point you may start executing SQL commands against your connected database.

The other way is via MySQL Workbench. This is GUI application used to manage MySQL databases.

To connect to your database, you first need to create a new connection. This can be done by clicking the plus sign (+) near the text MySQL Connections. This will open a new tab. You will then select a connection type. The connection to select is "Standard TCP/IP over SSH". Figure shown below.


Setup New Connection
Connection Name:
Type a name for the connection
Connection Method:
Standard (TCP/IP)
Method to use to connect to the RDBMS
Standard (TCP/IP)
Local Socket/Pipe
Standard TCP/IP over SSH
Parameters
SSL
Hostname:
ddress of the server host - and TCP/IP port.
Username:
root
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's not set.
Default Schema:
The schema to use as default schema. Leave blank to select it later.
Configure Server Management...
Test Connection
Cancel
OK

Next, you will need to fill in the required information for creating a new connection over SSH.



Setup New Connection

Connection Name: [blank] — Type a name for the connection

Connection Method: Standard TCP/IP over SSH — Method to use to connect to the RDBMS

SSH Hostname:	127.0.0.1:22	SSH server hostname, with optional port number.
SSH Username:	user	Name of the SSH user to connect with.
SSH Password:	Store in Vault ... / Clear	SSH user password to connect to the SSH tunnel.
SSH Key File:	...	Path to SSH private key file.
MySQL Hostname:	127.0.0.1	MySQL server host relative to the SSH server.
MySQL Server Port:	3306	TCP/IP port of the MySQL server.
Username:	root	Name of the user to connect with.
Password:	Store in Vault ... / Clear	The MySQL user's password. Will be requested later if not set.
Default Schema:		The schema to use as default schema. Leave blank to select it later.

Using the above figure as an example, fill in the following information:

- Connection Name: teams_db
- SSH Hostname: your host name (could be an IP)
- SSH username: your_ssh_username
- SSH Key File: path you your key file
- MySQL Hostname: 127.0.0.1
- MySQL Server Port: 3306
- Username: database username, same as shell account name
- Password: database password, should be student id, unless changed.

Default Schema: student_username, this your database name, where username is your database account name.

Other Option is to Connect with username and passed.

This is a simple process and is the default way to connect. Therefore, the steps will not be shown. But you should note the following:

- To connect to database remotely not via SSH a port needs to be opened on the server pointed to the database. Note that this is against good security practices.
- A user account in your database needs to be made for the connection. DO NOT use root to do this, while easy and simply its lazy and goes against good security practices.