NYU DevOps Project

Welcome to the semester DevOps homework project for CSCI-GA.2820-001 DevOps and Agile Methodologies Fall 2020. Since one of the fundamental tenets of DevOps is *Collaboration*, all homework will be in the form of a continuing **group project**. This project will have milestones that are due in 2 weeks sprints. The first sprint starts on Thursday September 24th and ends two weeks later on Wednesday October 7th. These will continue in two week iterations for 5 sprints until the end of the semester.

At the end of each sprint you should have something to demo. This first sprint will just be developing the plan so the showing the plan is the demo. We will conduct a short Sprint Review in class at the end of each sprint.

Semester Project Scenario

In the spirit of how Spotify is organized into autonomous squads that have end-to-end responsibility for a piece of the entire Spotify service, you are being asked to develop the back end for an eCommerce web site as a collection RESTful services for a client. The class will be divided into 9 squads, each squad will develop and run one the following services end-to-end:

Squads	Description
Customers	Basic customer into (name, address, etc.)
Inventory	The count of how many of each product we have
Orders	A collection of order items created from products and quantity
Products	Products for a catalog (id, name, description, price. etc.)
Promotions	Deals on products (e.g., sale, buy 1 get 1 free, 20% off)
Recommendations	A relationship between two products (prod a, prod b)
Shopcarts	A collection of products to buy
Suppliers	Vendors that we get products from
Wishlists	A collection of products I wish I had

The primary requirement for creating these microservices is that they must be RESTful according to the guidelines that we will learn in class. Each microservices must support the complete Create, Read, Update, & Delete (CRUD) lifecycle calls plus List, Query, and at least one Action (7 API calls in all). This is the list of expected functions:

- List Resources

- Read a Resource
- Create a Resource
- Update a Resource
- Delete a Resource
- Query Resources by some attribute of the Resource
- Perform some stateful Action on the Resource

In addition your microservice root URL ('/') should return some useful information about the service like what your resource URL is. Later in the semester we will add full Swagger documentation but for now, just hint at how to use the service.

The semester project is worth 40% of your over all grade. It will be graded each sprint with the first two sprint covering 20% of your grade worth **10 points** each. The first two sprint assignments will bring together everything you have learned in the first half of the semester up to the mid-term exam before deploying to the cloud in the second half.

Sprint 0: Agile Development and Planning

The first homework assignment is called Sprint 0 because it will just be about planning with no real code deliverables. You are actually creating the plan for Sprint 1 at this point. We want to simulate what a real Agile development team would do to plan out their sprints. To this end, you need to create a Backlog of Stories and a Milestone called Sprint 0 to plan your first sprint. In later assignments you will create a Milestone called Sprint 1, Sprint 2, etc. Resist the urge to create future milestones until you understand what you team can accomplish each sprint.

You will need to:

Initial Project Setup

- Look at the `DevOps-Squads-Fall-2020.pdf` on **NYU Classes Resources** to identify what squad/team you are in.
- Nominate one person on your team to create a GitHub **Organization** for your squad. Invite others members of the team to your GitHub Organization.
- Create a GitHub **Repository** for your project under the squad's Organization. The name of the repository *must* match the plural name of the Resource that you are developing (e.g., customers, orders, etc.) Make sure that the repository is **Public** and the name is all lowercase.
- I will create a Slack channel of the same name (e.g., customers, orders, etc.) and invite all of the members of the squad to your Slack channel.
- Please post the URL of your repo to your Slack channel after you create it so that I can see how you are working on your Stories and Kanban board.

Execute

- Nominate someone to be the Product Owner and Scrum Master for the first Sprint. You might want to rotate this assignment each sprint so that everyone gets a chance to play each role.
- Write Stories (as Github Issues) to plan the work needed to create the service. You will need at least 8/10 Stories (depending on the number of people on your team) but you can write as many more as you'd like and are encouraged to do so. Remember you will need a story to set up your Vagrant development environment, to Create your resource, Read your resource, Update your resource, Delete your resource, List all resources, Query your resource, and perform an Action on your resource.
- Set up an `ISSUE_TEMPLATE.md` file in a `.github` folder to ensure that all of your Stories follow the desired format that we studied in class (i.e. **As a**, **I need**, **So that** with **Assumptions** and **Acceptance Criteria**).
- Be certain to think MVP (Minimal Viable Product). Make your stories small. Get basic functionality working first, then add more features later. Use the `nyu-devops/lab-agile-zenhub` stories as a guide
- Conduct a Backlog Grooming session to Order your Backlog of Stories from most important to least important. You can do this via nyu.zoom.us or any video meeting service you'd like.
- Use the Comments section in GitHub to discuss particular Stories.
- Create a "technical debt" label that is yellow and add appropriate labels to all of your stories that don't deliver customer value but are required to develop your service (hint: like creating a Vagrant environment).
- Assign Story Points to all of the Stories in the first Sprint (stick with 3,5,8 = S,M,L for now)
- Create a **Milestone** for your first Sprint and name it something useful like *Sprint 1: Local a local Servie* for the first Milestone to create your Sprint Plan. Don't forget to include a brief Sprint Goal as the description of the Milestone. Remember, you are creating the plan for Spring 1 which will begin in two weeks.
- Conduct a Sprint Planning session to plan out your Stories adding to the sprint Milestone.
- Make sure that the Stories in your Backlog are "Sprint Ready" by making sure that each of them has a label, an estimate, and are assigned to your milestone.

What I am looking for is your understanding of how a self-directed Agile team works. You won't be telling me who did what. I should be able to look at the Kanban board from ZenHub and determine exactly what was done, and who wrote which story. This should also help you understand where you are in the assignment and if you will reach your sprint goal of submitting your homework on time. I want to see Milestones with Story Points, Labels, etc.

Please have someone from the team post the URL of your group's GitHub repository into your teams Slack channel to complete Part 1 of this homework.

This part is worth **10 points** and is due on Wednesday October 7th, and I will continue to accept submissions until Friday October 9th but they will incur a 10% grade penalty (i.e., your squad will start with 9 points instead of 10). No submissions will be accepted after the second deadline.

Submission

Each student must submit the URL of your group's GitHub repository as the response to this homework assignment to complete Sprint 0.

Sprint 1: Develop the RESTful Service locally

In Sprint 1 we will use the **Test Driven Development** techniques that you learned in class to develop the service based on the Stories you wrote. Start by creating tests for the Model that represents your resource and develop the code for it so that the tests pass. Then create tests for the RESTful Flask service that uses the model and develop the code for that. Don't forget to write test cases to test if the proper RESTful return codes are headers are being returned.

You will need to:

- Create a `Vagrantfile` so that members of the team can quickly get a development environment running
- The `Vagrantfile` should install all of the needed software to run the service so that the all a team member needs to do is `vagrant up && vagrant ssh` and be able to start developing.
- Create a README.md file and make sure that you give instructions on what calls are available and what inputs they expect and how to run and test your code so that users will know hot it works.
- Make sure that you set up a `.gitignore` file to keep unwanted files out of your repo. I should not see `.vagrant/` folders in your repos.
- Use **PyLint** to make sure that your code conforms to the PEP8 Python standard as shown in class. Your pylint score must be at least 9.0/10.0.
- Create a `setup.cfg` file with all of your `nosetests` options so that you only have to type `nosetests` to run the test suite.
- Place your test cases in a `./tests` folder.
- If needed, use Docker inside of Vagrant for any 3rd part services you want to use. e.g., if you use a Redis, MongoDB, PostgreSQL, MySQL database, etc. it should be installed using Docker. Note: You do not need to use persistence for this first assignment.
- If you copied the `Vagrantfile` from my repo, don't forget to remove anything from your `Vagrantfile` that is not needed.
- Start developing by creating the TDD `test_xxx.py` files that contains your Unit Tests to test your service and model.
- Use the standard `PyUnit` syntax that we learned in class.

- Create a `models.py` file to hold your model definitions of your resource
- Write test cases to test your model
- Create a `service.py` file to contain your Flask service
- Write test cases to test your service
- There should be tests for each of Create, Read, Update, Delete, List, Query, Action, traversing both happy paths and sad paths
- Run these tests using `nosetest` as you develop the code making them all green (*passing*)
- Use the coverage tool to measure your test coverage. Your goal is get it above 95% for full credit.
- Following good REST API practices, your API should only return `json`. This includes any error messages from the web server which means you must override the default error handlers and ensure that only json messages are returned.
- Logging is critical for debugging so make sure to log your actions (you will thank me when you get to the cloud deploy)
- Move these Stories through the ZenHub pipeline as you work on them.
- As you execute this plan using ZenHub, each member of your squad should assign a Story to themselves, move it to In Process, and begin working on it. When completed, move the Story to Done (not Closed) until you have a Sprint Review and the Product Owner declares it Closed.
- All work should be done in branches and Pull Requests should be used to merge those working branches into master. (i.e., do not commit to master)
- When you create your Pull Request, associate the Story that the request is for using the ZenHub button "Connect with an Issue".
- Create a **Burndown** chart to track your progress.
- After we conduct the Sprint Review in class at the end of the sprint, move the completed stories from Done to Close and mark the Sprint Milestone as Closed.

Think about what you need to do in this part and make sure that there is a Story to cover the work. (*hint*: there should be a Story for someone to create the `Vagrantfile` for everyone else to use and it should be pretty high up in the Backlog.)

Grading Criteria

Working as an Agile Team

Unlike other classes where only your final submission is graded, I will will be watching how you work during the sprint. I expect to see stories get moved across the various pipelines of your Kanban board. I expect to see your Burndown chart actually "burn down". I expect to see Pull Requests being used to submit your work to the master branch. The importance here is to demonstrate that you know how to collaborate as a self-directed team.

95% Test Coverage

You will be judged by my ability to clone your project, issue `vagrant up`, and run `nosetests` and see all of the tests pass, as well as run your service in a VM and go to `http://localhost:5000/` so that I can see it running. Make sure all of that works.

What I am also looking for is a demonstration of your understanding of how write good test cases. I am also looking for good code testing coverage. You must achieve over **95%** code coverage for full credit.

RESTful Service Implementation
I will also be grading on how RESTful and complete your API is. Make sure that the URI's follow RESTful conventions and that the proper HTTP returns codes are used as covered in the lectures.

This part is worth **10 points**.

Submission

Each student must submit the URL of your group's GitHub repository as the response to this homework assignment to complete the Sprint 1 homework.
————————————