Homework #1: Host a dynamic website using cloud virtualization

Due: Feb 1, 2018
Please read thoroughly through these instructions at least once before starting. There are 2 pages.

Your objectives with this first homework assignment are as follows:

- Create an account with a Cloud VM provider - Recommended Amazon EC2
  (some alternative suggestions have been provided in the Course Documents section)
- Use Linux (recommended Amazon Linux AMI or Ubuntu Linux)
- Use your UC GitHub account to store your code using a private repository (grant access to me, “tatavag”,”dhanasak”,”rawashoy”). Create a dedicated repository for this and do the same on future homework/projects (don’t create one project with sub-folders for the whole class).
- Host a webserver on your Linux instance that provides an HTTP application to operate on user-entered data, and provide a result. The content you write must be stored in the UC GitHub repository you create for this project.
- The web-based application you host in the web server must have the following elements:
  - Must contain a README or README.md file that describes how to use it
  - Must implement a simple algorithm that takes a user input and returns a result that is derived from operating on that input
  - Must provide a form to accept user input using either the GET or POST method. **Please do not implement by just Client side javascript.**
  - Must utilize at least one included file that is separate from the main application (such as a CSS stylesheet, server-side include, or javascript include)
- The web-based application must be hosted publicly during the grading time. Once you receive a grade in blackboard you may shutdown your VM.

The goal of this exercise is to familiarize you with the following concepts:

1. Use of SaaS for project management (GitHub)
2. Use of IaaS for infrastructure provisioning (Amazon EC2, etc.)
3. Leveraging cloud compute
4. Leveraging cloud networking (public access, cloud-based delivery)
5. User documentation
6. Web technologies to be used further (CSS, Javascript, includes, modular development)
7. Linux web app deployment

Submit the following urls to blackboard, one per each line without any headers or comments:

1. The public URL of the website you’ve hosted
2. The GitHub Project URL
3. The GitHub README Page LINK
4. The GitHub commit hash for the revision of the project you wish to be graded on.

Sample Submission would include the 3 urls lines and a hash line like below and nothing else:
http://myawesomecloudapp.com
https://github.uc.edu/tatavag/myawesomecloudapp
https://github.uc.edu/tatavag/myawesomecloudapp/blob/master/README.md
936797ae653863406b08d420c2be93b7c235cd92

Extra Credits 10 POINTS:

If you setup Python with a wsgi server and host behind Apache/Ngnix on port 80 (sorry No tomcat). Please document the steps for installation at a high level.

Extra Extra Credits 5 POINTS:

Enable HTTPS (Self signed certificate is okay for this Home Work). Please document the steps for enabling https at a high level.

If you start work early, and run into roadblocks, I (and your peers!) can be available for questions. If you wait until the days right before the assignment is due, I won't guarantee that I will be able to answer your question before the deadline!!

Tips, answers:

Your web application should implement a simple algorithm or operation, taking user input. You may use any language you wish. Remember that you are responsible for the level of difficulty incurred by the choice of language, frameworks, and/or libraries you decide upon.

Some great examples of web applications that have been submitted in the past:

- Fibonacci number calculator
- Multiplication table creator
- Simple calculator
- Prime number validator
- Large prime generator
- Meme generator

Using github, and committing revisions early & often will go a long way in helping you to diagnose bugs and roll-back from changes introducing them.

I recommend also detecting for and enforcing an upper limit on user input for tasks that can grow exponentially. This will help mitigate potential compute-heavy operations that can eat into your Amazon EC2 credits, etc.

Here are some application framework tutorials:

https://docs.djangoproject.com/en/1.10/intro/tutorial01/ (tutorial building an application in Django)

http://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html (setting up Django to deploy with uwsgi)

https://www.railstutorial.org/book (Ruby / Rails)

http://www.vogella.com/tutorials/ApacheTomcat/article.html (Apache Tomcat)