EMBL Position 000717 Assignment, Ramón Fallon

December 16, 2020

(These notes and code can be viewed at https://github.com/rafalcode/rf717)

Part A

The code defines a function for building a random permutation of unique integers starting from 1 up to a certain value which is passed in as an argument.

Yes, random permutations are often required, often to shuffle an array, though in this case we get a permutation starting from 1, so it is not exactly an index suffling code (subtractin one would do this)

Fisher-Yates is a neat method for it. range(length) is called for the usual 012... and a random index from other than the current index is chosen. The elements in the current and random index are then exchanged. One can progress from the beginning of and array to its one-but-last index and in this manner create a random permutation on the elements in the array.

But all this code is "from first principles" there are plenty libraries for this, numpy has random.shuffle() for example.

Part B

Test file is lines.txt.

- rlind.py uses a dictionary to print out (i.e. find) lines which are not unique.
- rlin.py is a navie method of finding duplicates, without dictionaries or hashes.

Part C

Difference between a VM and a container?

The key difference is autonomy, a VM is less dependent on the host system and controls many of its own system-level so that it appears more like a separate machine within the host system. It is however more resource hungry, and so only a relatively reduced number can be envisaged for a certain host system.

Collaboration between two virtual machines on the same host system is probably an unproductive task.

Containers on the other hand can be treated more like self contained services, and are designed to only lightly use host resources so one can contemplate having a higher number of them running at the same time. However, they are more tightly bound to the host system's resources and their processes are visible from the

host system.

Getting two docker containers to interact would certainly be a standard task to undertake, for which there are widely available tools and libraries.

In summary, VMs are better suited to a ”separated machines” context, while containers are more suited to ”separated services”.

VirtualBox is a widely used solution for VM (Virtual Machines), while Docker is is the most common container technology.

Gocd, cfengine, ansible and puppet

I'll deal with Ansible first because I have used it, for example, in setting up a Galaxy instance. Ansible sets up remote servers in automated fashion so that several servers can be set up together in a controlled and highly uniform manner.

I haven't used GoCD but it appears to be CD Continuous Delivery tool, so would be dedicated to automation of the build, test, deploy cycle. Usually this takes the form of a yaml file that makes / buidls the project, then applies a quick test and progresses onto deploy if the other stages have passed. My experience has been with gitlab's CI/CD method.

I have not used cfengine and puppet, but the also handle configuration management.

I would expect that there is an overlap between all four.

AWS charging for compute and storage

By the hour and depending on capacity and capability, the hourly rate goes up. Capability is used for compute instances (nodes) and the CPU power is tied to RAM available. There are some 30 or 40 types, though these fall under broader categories such as Memory Optimized, Compute-optimised etc. Despite this flexibility, you cannot choose a weak processor with vast memory, which, say, would be possible with a physical machine.

Some AWS images are not free to use, and the hourly charge may have an added licence cost.

Part C

I seldom have to go into the detail of XML, and have not used the libraries too much. I had a go at decoding the file you specified and though I printed out the entities, I did not count the occurrences.
This is similar to a pairwise analysis which I have done before.

Part D

Question 1

My attempt at this can be found https://github.com/rafalcode/adna and at site http://a.dna.ovh:8000/date/
It imports Django REST Framework (DRF) although, this is not necessary, as the application is so simple.

This will run with django's runserver and also with pure uwsgi, however it gives bad gateway when running on nginx suggesting a mismatch between uwsgi and nginx.

Question 2

There is also a Dockerfile, which would use gunicorn as webserver, but it has not been configured properly, so it not is not "dockerised". Some work would be required with supervisord, gunicorn and uwsgi to setup the docjeker image to serve the site correctly. A typical CD script I would use is

```
---
image: docker

stages:
  - build
  - test
  - deploy

build:
  stage: build
  script: [./scripts/build]

test:
  stage: test
  script: [./scripts/test]

deploy:
  stage: deploy
  script: [./scripts/deploy]
  only: [master39]
```

Question 3

For scalability I would lean on uwsgi, it's a powerful program and in the "Emperor" setting can spawn many child processes to cope with demand.

Part F

Unfortunately I was unable to make substantial progress on this part. Initially I was happy to see an API come up because I have recently successfully developed one using Django's REST Framework. But this requires interaction with an outside website, and there was some added complexities that I did not have time to solve.

Question 2

I have mentioned the ability to spawn child processes as key to scalability, and I must also mention the testing of scability which can well be called a separate category of testing: stress testing. This would requires a test harness of gradually increasing load.

Question 3

By coding Build, Test, Deploy scripts in the now common CI/CD fashion, as per .gitlab.yml example given above.

Candidate Notes

Myself I have to be involved in a Covid-19 testing startup project and a LIMS implementation right now, and the workload is very heavy. I was interested in doing this assignment because it was quite close to my core activities, but the truth is I could ill-afford the time. I was also interest in the heavy DevOps aspect which is a new tendency in Bioinformatics, which I've also found myself drifting towards. However, though the technologies (python) and tools I use are similar to the tasks you describe, my projects are different enough to have brought up a series of unknown aspects that I didn't have time to tackle properly.