COSC349 assignment 1 (2023): Use virtualisation to effect portable building and deployment of software applications

Submission information

Due date: Monday, 4th September 2023, at 11:59 PM.

Weight: This assignment is worth 20% of the mark for the paper.

Participation: You are permitted to work individually or in pairs.

How you will make submissions: The COSC349 Blackboard section will be used to collect the following four pieces of information:

1. a URL such that the command '`git clone` URL' would download your repository (you must have shared this repository—see below);
2. the Git commit ID within your repository that you want to be marked;
3. a URL to a screen recording showing you using your software;
4. a project report in PDF format.

All students must make their own personal submission within Blackboard. Members of pairs must both submit exactly the same material.

Requirements:

- You are expected to work on your assignment within a Git repository. The easiest way to acquire URLs that reach your repository is to push your work to a (free) cloud-based Git service, such as GitHub, Bitbucket, GitLab, or Altitude (a GitLab server run locally within the School of Computing that you can access). The aforementioned cloud services allow you to create private repositories.
- You must invite teaching staff to collaborate on your repository, so that they can access the URL that you submit. Inviting David is sufficient: he is user `dme26` on GitHub, Bitbucket, and GitLab in the cloud. He is user `dme` on Altitude. Otherwise you can send an email invitation to `dme@cs.otago.ac.nz`.
- Do not apply Git history-rewriting operations to your repository: it is important that the markers can see how your commits completed your assignment, and how you worked on the assignment over time.
- Your repository must include all of the files required to build your application, other than stable, publicly-available, external resources downloaded automatically during your build process. For example you do *not* need to include Vagrant box files, or data that can be downloaded from cloud-based repository servers (*e.g.*, GitHub, *etc.*), but be careful to test that such material is being successfully fetched by your build process just before you declare your project complete.

- Your screen recording must be no longer than two minutes. Do not add additional graphics to your video beyond what is seen on screen. You may narrate your screen recording, or leave the audio track blank, and instead briefly describe in your report the important time points in your recording. You may use any tool to create your screen recording, but note that Zoom facilitates this (join your own meeting; screen share into that meeting; and then record the meeting onto your computer). You can share your recording with David using your University of Otago OneDrive storage (or YouTube, *etc.*).
- Your repository should be set up so as to allow a new developer to join your project, and to rapidly get up to speed regarding how to use and extend your work. For example an informative '`README`' file of some sort is expected. Assume that such a developer will not receive your submitted report. Ideally your repository will not contain too many indicators that it is aiming to satisfy a university assignment.
- Your project report is where you can communicate with your markers, explaining aspects of your project that might not be reflected clearly within your repository, such as unsuccessful attempts that you might have made to use particular technologies. The marking guide, below, indicates a number of points that your report must cover.

Late submissions: The timestamp of the Git commit that you select for marking will be taken as when you submitted your assignment. COSC349 applies a late submissions policy typical of other COSC papers. Late submissions will incur a 10% penalty per working day, rolling over at 11:59 PM. Submissions that are more than five days overdue will not be accepted.

Learning objectives

The aims of the assignment include the following learning objectives:

- to demonstrate that you understand how virtualisation can usefully underpin application development; (This is also particularly true of application development for cloud computing.)
- to employ, in practice, fundamental concepts in cloud-ready software development, such as automated provisioning of virtual machines;
- to carry out research to discover open source projects and materials that you can use within your application development work; and
- to deliver software that is well tested and documented.

Problem description

For this assignment you should design and develop an application whose deployment relies on virtualisation. Your application should operate through coordination of at least three different virtual machines (VMs). One of your VMs should be responsible for data storage that is important to your application.

However, this assignment is not focused on the functionality of your application: the assignment is focused on how you *build* your application, and in particular **how you facilitate other developers potentially modifying, (re)building and running your application**. Your use of virtualisation should allow developers to build and run your application even if they check out your project's Git repository on a different host operating system from yours.

Vagrant can be used to complete the assignment, but use of other tools available on the CS Lab computers is acceptable too, *e.g.*, Docker Desktop, and (non-Vagrant) shell scripting to automate VirtualBox functionality.

The easiest way for a virtualised application to provide an interface to (non-developer) users is through the use of web technologies; and a common way to manage application data is through use of a relational database. Starter material for both web and database technology is included in the lab exercises. However, because COSC349 is not a web technology or database paper, marking will not focus on the details of your web implementation or database modelling. Any other form of user interface or storage is acceptable too, provided that your design does not require your users to have advanced computing qualifications.

You are welcome to use others' code within software components and as a starting point for your assignment, but you still need to build an application of your own design. Any use of others' resources must be attributed by you, both in your in-code documentation *and* in your project report. Further, the history of commits in your Git repository will be examined to determine what, and how much code and configuration you added into your project.

A developer must be able to modify your application by changing source files in their clone of your Git repository, and then be able to rebuild their instance of your application on their virtualisation host.

Example applications

If you cannot think of an application to build, some suggestions are listed below. You may want to consider developing a project that you can present to future potential employers. Also, given that the assignment does not focus on the application itself, you are welcome to rework and extend material that you have completed for other assignments within Computer Science or Information Science papers, for example.

- An online service for booking use of shared resources of some type. One VM could run a web interface for users, another VM could run a database that records allocation of the shared resource, and a third VM could provide administrative functionality.

- A rental share-house budget management helper: one VM could provide the interface renters use to enter bills, payments, and view the results of queries. Another VM could store the users' data in a database or other storage system. A third VM could provide functionality for the owner/manager of the property.

- A tool for recording collections of websites, *e.g.*, their URLs, and classifying information such as a short description, or tags, *etc.* Different VMs could run a web interface for the editor, a database server, and a tool for producing reports.

Rough marking guide

The assignment is worth 20% of your COSC349 mark but will be marked out of 100. A typical breakdown of marks is included below. This is not a strict marking scheme, as having one has been found to tend to disadvantage students. For example, if you are unable to achieve a section of the marking breakdown below, you may be able to gain compensatory marks by explaining in your project report what you intended to do, what problem arose, and what compromise you reached. Also, it may be considered that you have gone well beyond what was expected for part of the assignment, and should be awarded additional marks for a section, thus helping compensate for some minor problems elsewhere.

30% Application design and use of VMs

- Your application should be built using at least three VMs that interact. Your application needs to be of your design, even if you use others' open-source components to help build it.
- Your repository (not your report) should include an explanation to new developers about the design of your application, including the purpose of each of your VMs, and how they interact.
- Your report should justify the importance and benefits of using separate VMs, rather than aggregating functionality into a single server.
- Your screen recording—no longer than two minutes—makes it clear how a user should use your application. You may narrate your screen recording using audio, or instead include a textual description of your screen recording's timeline within your report.

30% Automated build process

- After starting your application's build process, your application should build and start, unattended.
- Your repository should default to preloading test data for demonstration purposes into your application, rather than requiring significant amounts of manual interaction with your application's user interface before useful results are seen.
- In your report, explain what software components your build process will download, and the approximate download volumes involved, both for a clean first-time build, and for subsequent redeployment.
- Your project report can explain why manual interaction is required, if you are unable to achieve an entirely unattended build process.

20% Suggested improvements

- In your report, detail two different types of change—along with a specific example of each—that a developer might make to the code of your repository, and how they can subsequently rebuild and rerun the application after they have made such changes.

20% The commit history in your Git repository should make it clear how you **incrementally developed and debugged** your application.