Initial Plan - Cloud Engineering University Teaching Module

Dom Routley

February 4, 2019

Description

In this project (provisionally named Cloud Engineering University Teaching Module) I intend to create a proof-of-concept university teaching module focusing on cloud engineering and cloud operations.

The Computer Science BSc degree at Cardiff University does not include a teaching module (either optional or core) that explores cloud computing in detail. As cloud engineering and operations has quickly become a large part of the information technology industry this project will attempt to build a proof-of-concept module that could be used as a basis for creating a fully fleshed out module in the future. There is currently a MSc teaching module entitled 'Distributed and Cloud Computing' for which this teaching module could serve as its BSc vanguard to prepare students for the MSc course. Distributed systems is already taught to BSc students in the third year as part of the 'High Performance Computing' and 'Emerging Technologies' teaching modules.

As this is a proof-of-concept design, not all elements are intended to be fully complete, I simply do not have the time or initial knowledge available to be able to create a fully ready module in three months. I therefore intend to create a module layout/syllabus/overview as well as building/writing as much of the actual teaching module as possible in the time given.

I intend to create a mix of lecture style resources, coursework, and exam material. I will focus on creating the coursework and its marking criteria as I believe that this is the single most valuable part of the project. During the creation of the coursework lecture style information will be created such that you should be able to study the lecture notes and use them to complete the coursework to at least a 70% grade level. From this lecture style information it should be possible to build out some exam questions. The lecture style information that I intend to create is to take the format of lecture slides with attached speaker notes, I intend to be able to pull together this content and create some fully formed deliverable lectures ready for use. To prove the viability of these lectures I will record myself giving each lecture (that I fully create) and upload the footage/audio to the internet such that it can be viewed as part of the viva.

For the coursework I intent to cover cloud server management, primarily through automated deployment/provisioning tooling such as Ansible, Puppet, Chef, and Terraform. Out of these four, I have prior experience mainly with Terraform and Ansible and believe that Ansible is the most versatile for what I would like to achieve. Puppet and Chef both are designed for client-server models of server configuration, Ansible is deployable by the user on their own machine and would therefore be more useful for a coursework style task. Terraform can be used as a server configuration management tool, but is primarily used for orchestration (creating virtual servers and their environments) rather than configuration.

Whilst I do not yet know exactly what the coursework will be about, the very basic plan that I think would work is that of a server and environment configuration task. Where the student must first build a cloud environment (or design it) and then write an Ansible configuration script that would configure the server/s in a particular manner. (Set up a web server, DHCP server ect). Terraform could be used in the back-end of the coursework to enable some automated environment testing (in OpenStack) of the coursework to achieve a grade for the student.

Other relevant information will be included as well, but as I am unlikely to be able to build a full exam along with a full semesters worth of lecture content alongside the

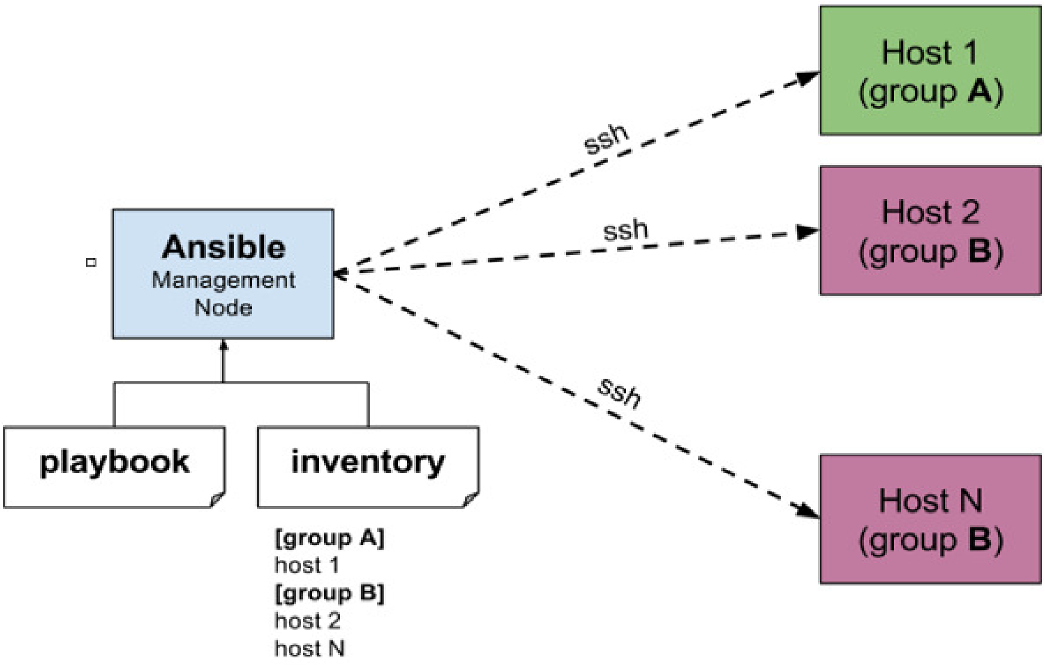


Figure 1: Ansible server management

https://www.tutorialspoint.com/ansible/ansible_introduction.htm

coursework, I expect this to be brief and simple an outline to be covered in the "What I would do if I had more time" section of the final report.

The coursework is going to be the most significant portion of the project. I want to not only design the coursework questions, but also construct at least one model answer. I am also very interested in automated testing and linting (syntax/grammar), so I would like to build at least a plan of how these could be implemented into the marking system to aid the course lecturer. If there is time, then I would like to make these working and available to be shown at the viva.

Objectives

The objectives that I want to achieve can all be split into a set of deliverables that are then given weekly target dates for their completion. The discussion of the target dates is in the work plan section below. None of this is set in stone as I may decide that some or all of these are bad ideas and I will then pivot to different ideas/objectives.
Each of the deliverables has been allocated into a work block, these work blocks are:

- A mostly to fully complete coursework question sheet and model answers
  - Block 1a
    - A task asking for a logical design of a Virtual Private Cloud (VPC) that is appropriate for a certain task
    - A "correct" design of the VPC meeting the specifications given in the question - 100% grade example
    - A "correct" written answer for why the VPC meets the criteria - 100% grade example

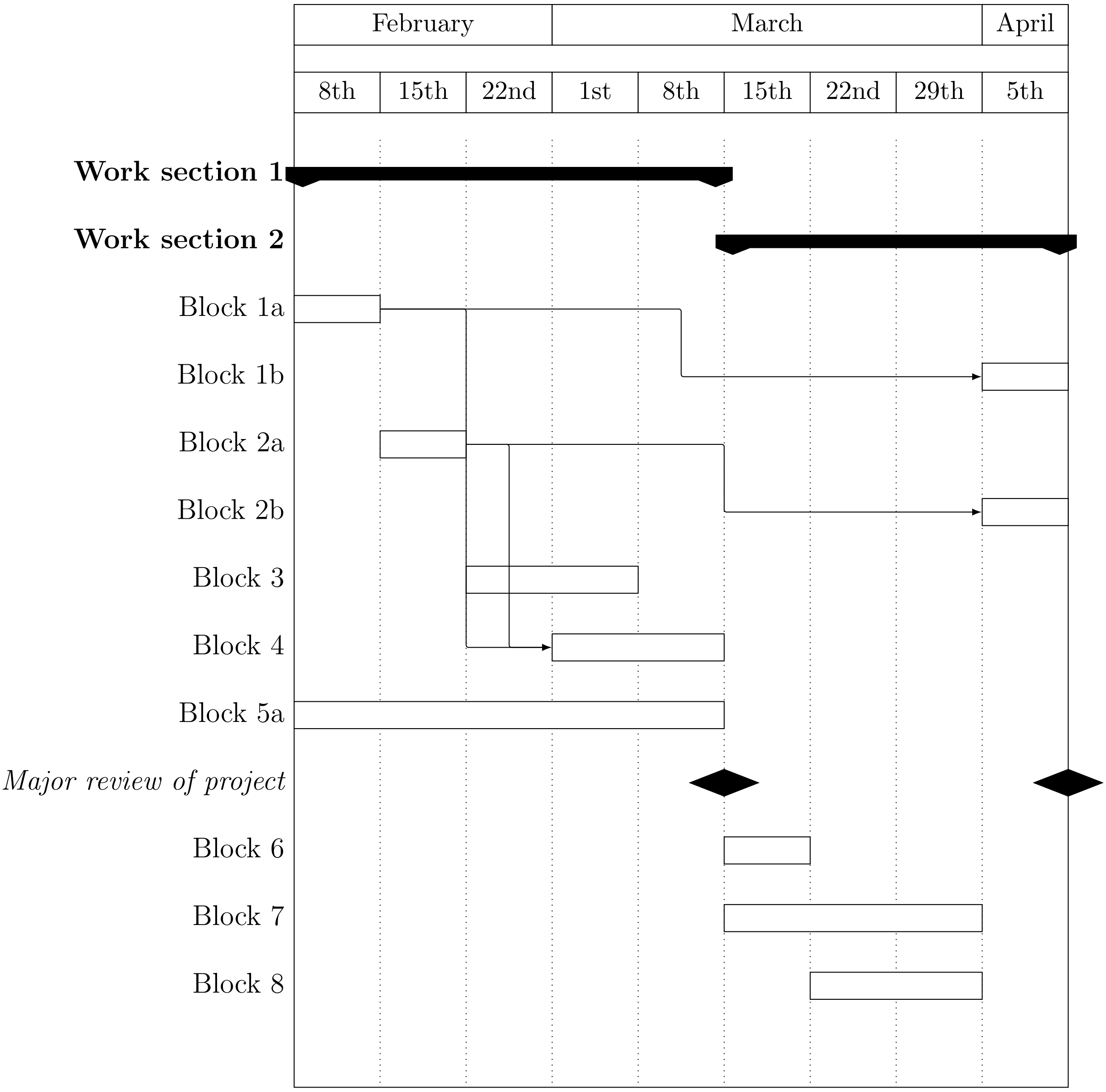
    - Lecture content, (slides + written content) explaining all the information needed to design the VPC correctly
    - Written explanations for why the VPC design question is useful for the module and pointing to the lecture content that it examines
  - Block 1b - This is a continuation of Block 1a, but is classed as surplus to Minimum Viable Product (MVP) so will be done after the MVP has been met and has a lower priority than other work blocks
    - A “correct” design of the VPC meeting the specifications given in the question - 70% grade example
    - A “correct” design of the VPC meeting the specifications given in the question - 60% grade example
    - A “correct” design of the VPC meeting the specifications given in the question - 50% grade example
    - A “correct” design of the VPC meeting the specifications given in the question - 40% grade example
    - A “correct” written answer for why the VPC meets the criteria - 70% grade example
    - A “correct” written answer for why the VPC meets the criteria - 60% grade example
    - A “correct” written answer for why the VPC meets the criteria - 50% grade example
    - A “correct” written answer for why the VPC meets the criteria - 40% grade example
  - Block 2a
    - A task asking for a Ansible Playbook to configure a server (or group of servers) for specific tasks
    - A “correct” Ansible Playbook meeting the specifications given in the question - 70% grade example
    - Lecture content, (slides + written content) explaining all the information needed to write the Ansible Playbook (up to a 70% grade)
    - Written explanation for why the Ansible Playbook question is useful for the module and pointing to lecture content that it examines
  - Block 2b - This is a continuation of Block 2a, but is classed as surplus to MVP so will be done after the MVP has been met and has a lower priority than other work blocks
    - A “correct” Ansible Playbook that goes further than what is explained in lectures and achieves more than 70%
    - Written explanation for what would constitute valid “further than in lectures” content and how you would grade this
    - A “correct” Ansible Playbook meeting the specifications given in the question - 60% grade example
    - A “correct” Ansible Playbook meeting the specifications given in the question - 50% grade example

    - A “correct” Ansible Playbook meeting the specifications given in the question - 40% grade example
  - Block 3
    - Two more tasks for coursework designed
    - Those two tasks written with lecture content also created and justifications for all
  - Block 4
    - Automated marking/linting system designed and made for the Ansible Playbook coursework task
  - Block 5
    - A test area in OpenStack created that allows the students to run tests for the coursework and also allows the automated testing to run
    - Instructions on use of test area created to a professional standard
- Multiple exam questions with model answers
  - Block 6
    - Design (minimum) four exam questions
    - Model answers for all exam questions
  - Block 7
    - Lecture content for all exam questions that covers everything needed to know
- Example Lecture/s
  - Block 8
    - More lecture content that is useful to know, but is not expressly assessed in exams or coursework so far generated
    - One or more fully compiled lecture presentations
    - All fully compiled lecture presentations are presented and the audio/presentation content uploaded to the internet (publicly available)

Work Plan

I have split the project into two parts, each intended to take approximately one third of the total time that is remaining from now (4th of February) to the submission date of the project (10th May). This gives my fourteen (14) weeks to divide between my two sections of work and pulling all of my notes together into the final report. I intend to split the time in a 5/4/5 split, five weeks for the first section of work, four for the second, and five for honing the report. The reason for the sections having uneven time is that I expect the first section to be harder as I expect there to be some false starts/discarded work in the first section. I will place the most valuable work in the first section, that way if my time planning is wrong I can drop second section work in order to focus on more valuable work.

Work section one will run from the 5th February to the 8th March. It will consist of internal objective dates: 8th/15th/22nd February, 1st/8th March. Work section one will run from the 9th March to the 5th April. It will consist of internal objective dates: 15th/22nd/29th March, 5th April. Internal objective dates are Fridays, they correspond with the meetings between myself and my project supervisor and will act as minor reviews of how the project is going. There will be major reviews of the project at the end of both of the work sections. (March 8th, April 5th)



Glossary

Ansible Ansible is an open-source software provisioning, configuration management, and application deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system configuration. Unlike competing products, Ansible is agentless - temporarily connecting remotely via SSH or remote PowerShell to do its tasks. 1

Ansible Playbook An Ansible playbook is an organized unit of scripts that defines work for a server configuration managed by the automation tool Ansible. 3, 4

Chef Chef is a configuration management tool. Chef is used to streamline the task of configuring and maintaining a company's servers, and can integrate with cloud-based platforms to automatically provision and configure new machines. Chef can run in client/server mode, or in a stand-alone configuration named "chef-solo". In client/server mode, the Chef client sends various attributes about the node to the Chef server. The server uses Elasticsearch to index these attributes and provides an API for clients to query this information. Chef then query's these attributes and uses the resulting data to help configure the node. Chef supports both Linux and Windows. 1

MVP Minimum Viable Product 3

OpenStack OpenStack is a free and open-source software platform for cloud computing, mostly deployed as infrastructure-as-a-service, whereby virtual servers and other resources are made available to customers. Cardiff University School of Computer Science maintains an OpenStack cluster. 1, 4

Puppet Puppet is an open-source software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows, and includes its own declarative language to describe system configuration. Puppet Server is installed on one or more servers, and Puppet Agent is installed on all the machines that the user wants to manage. Puppet Agents communicate with the server and fetch configuration instructions. The Agent then applies the configuration on the system and sends a status report to the server. 1

Terraform Terraform is an open-source infrastructure as code tool. It enables users to define and provision a datacenter infrastructure. 1

VPC Virtual Private Cloud 2, 3