Unix System Administration and Programming (USAP) Assignment Two (Puppet Enterprise) Report

Aaron Horler - s3481341

This report details the development of assignment two for USAP in semester two of 2017.

Proof of a successful run is provided via a puppet agent run log in the root of this repository named **run.log**. This log was generated by running **puppet agent -t -d >> run.log**.

Git

This project been been pushed to a bare Git repository on RMIT's coreteaching servers. It will also be publically released on GitHub from November 30, 2017.

The URL for the GitHub repository will be https://github.com/aghorler/usap-a2-puppet.

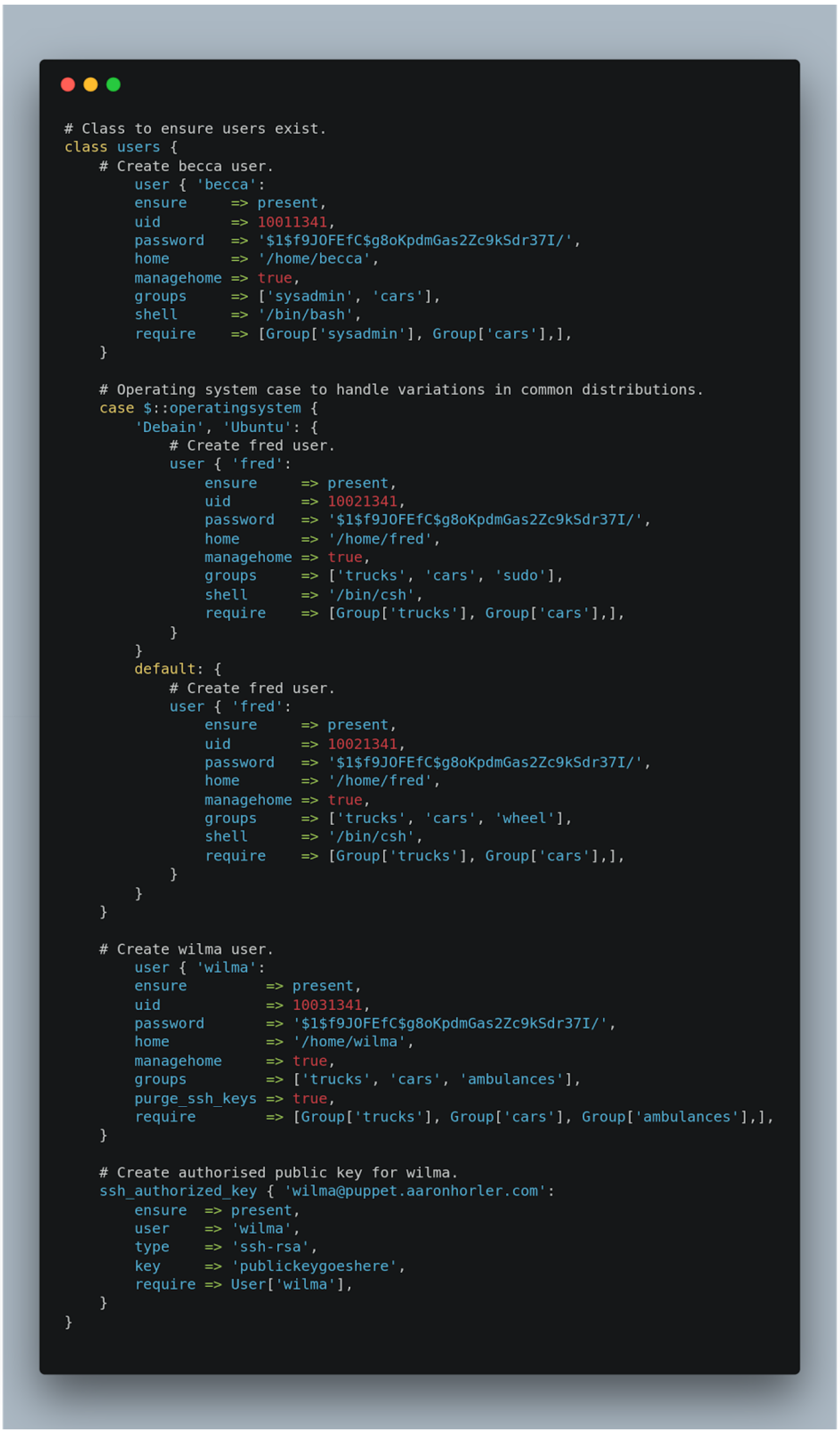
Notes

I did not use Amazon Web Services (AWS). Instead, I used Vultr.

I developed this on CentOS 7, although I have provided for basic compatibility with Ubuntu and Debian.

I have commented out the exec resource responsible for RMIT titan mounting, because it requires either my current RMIT password, or my private key, to run successfully. I am obviously not willing to publish either of these.

Users



```
# Class to ensure users exist.
class users {
    # Create becca user.
        user { 'becca':
        ensure     => present,
        uid        => 10011341,
        password   => '$1$f9JOFEfC$g8oKpdmGas2Zc9kSdr37I/',
        home       => '/home/becca',
        managehome => true,
        groups     => ['sysadmin', 'cars'],
        shell      => '/bin/bash',
        require    => [Group['sysadmin'], Group['cars'],],
    }

    # Operating system case to handle variations in common distributions.
    case $::operatingsystem {
        'Debain', 'Ubuntu': {
            # Create fred user.
            user { 'fred':
                ensure     => present,
                uid        => 10021341,
                password   => '$1$f9JOFEfC$g8oKpdmGas2Zc9kSdr37I/',
                home       => '/home/fred',
                managehome => true,
                groups     => ['trucks', 'cars', 'sudo'],
                shell      => '/bin/csh',
                require    => [Group['trucks'], Group['cars'],],
            }
        }
        default: {
            # Create fred user.
            user { 'fred':
                ensure     => present,
                uid        => 10021341,
                password   => '$1$f9JOFEfC$g8oKpdmGas2Zc9kSdr37I/',
                home       => '/home/fred',
                managehome => true,
                groups     => ['trucks', 'cars', 'wheel'],
                shell      => '/bin/csh',
                require    => [Group['trucks'], Group['cars'],],
            }
        }
    }

    # Create wilma user.
        user { 'wilma':
        ensure         => present,
        uid            => 10031341,
        password       => '$1$f9JOFEfC$g8oKpdmGas2Zc9kSdr37I/',
        home           => '/home/wilma',
        managehome     => true,
        groups         => ['trucks', 'cars', 'ambulances'],
        purge_ssh_keys => true,
        require        => [Group['trucks'], Group['cars'], Group['ambulances'],],
    }

    # Create authorised public key for wilma.
    ssh_authorized_key { 'wilma@puppet.aaronhorler.com':
        ensure  => present,
        user    => 'wilma',
        type    => 'ssh-rsa',
        key     => 'publickeygoeshere',
        require => User['wilma'],
    }
}
```

The users class ensures the presence of users becca, fred, and wilma.

All users have been configured per specification, with groups, a UID, a managed home directory, a default shell, and an encrypted password.

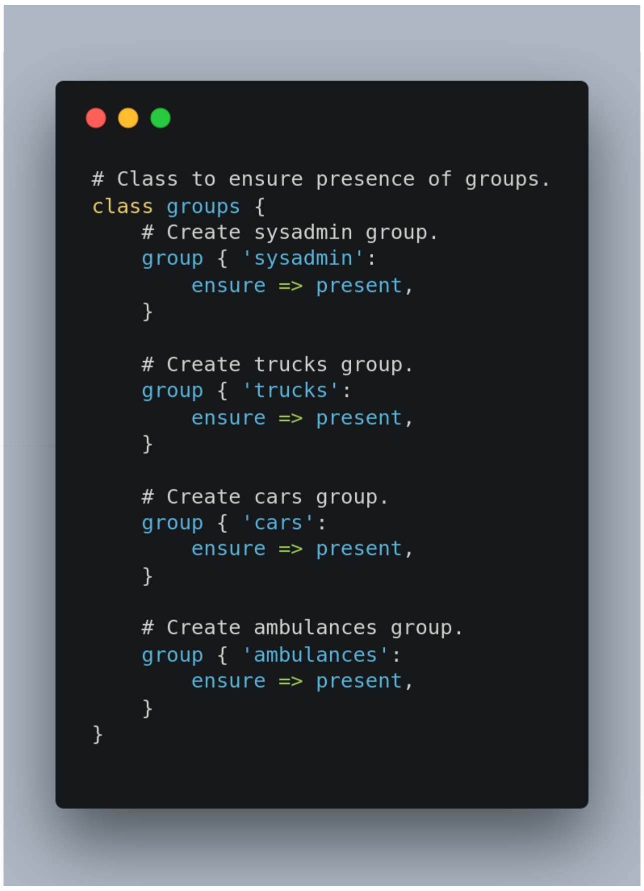
I generated passwords by running **openssl passwd -1**. All users passwords are *secret*.

In Debian and Ubuntu, fred is added to the group *sudo*, and in CentOS *wheel*. This is to allow usage of the *sudo* command - without modifying the sudoers file.

All users require that their groups are present.

User wilma’s public key is added to the *authorized_keys* file, requiring that the user is present.

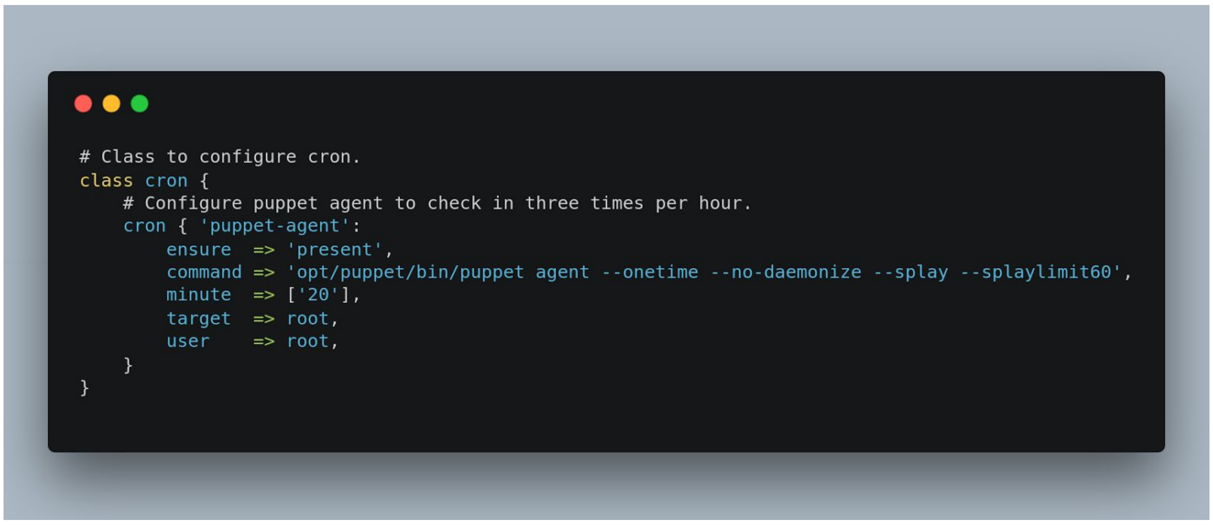
Groups



```
# Class to ensure presence of groups.
class groups {
    # Create sysadmin group.
    group { 'sysadmin':
        ensure => present,
    }

    # Create trucks group.
    group { 'trucks':
        ensure => present,
    }

    # Create cars group.
    group { 'cars':
        ensure => present,
    }

    # Create ambulances group.
    group { 'ambulances':
        ensure => present,
    }
}
```

The groups class ensures the presence of groups sysadmin, trucks, cars, ambulances.

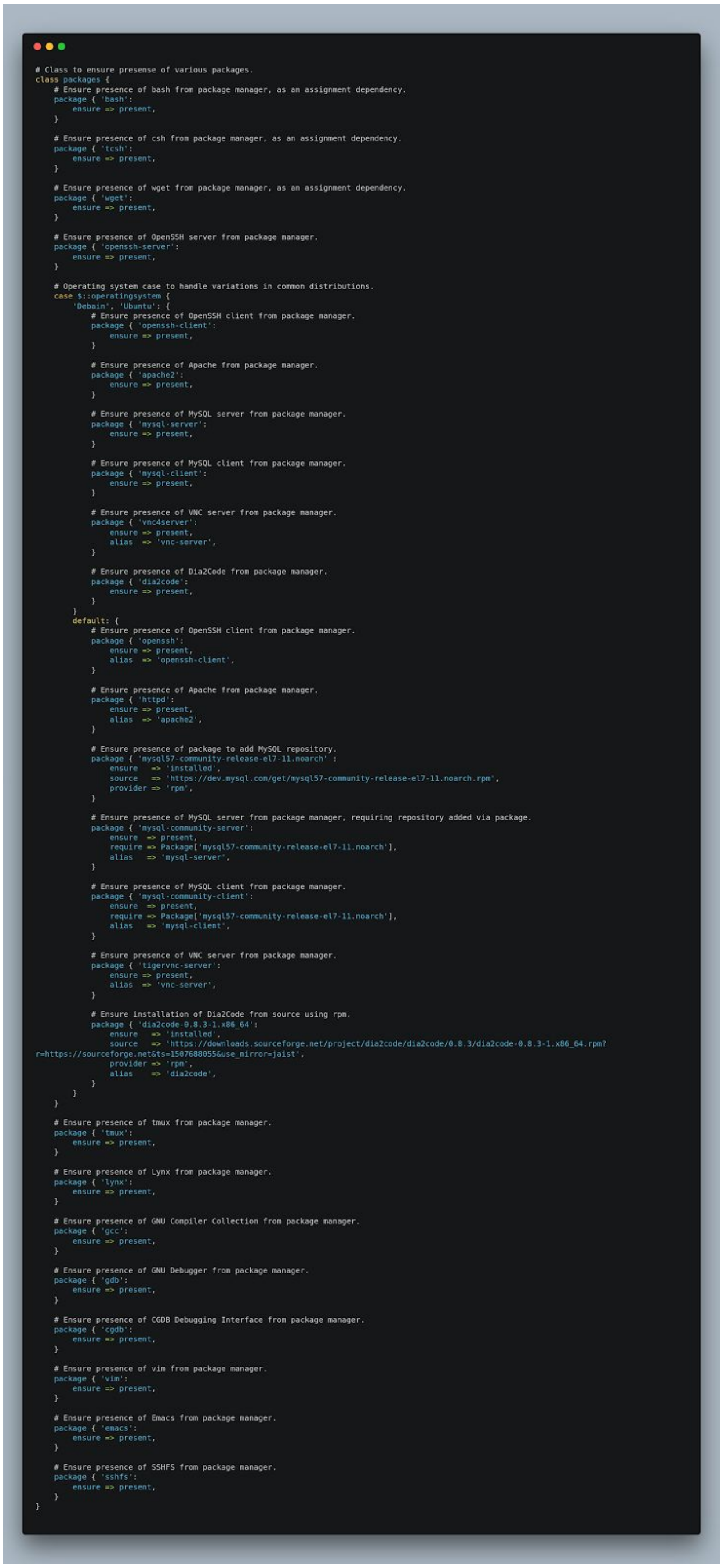
Cron



```
# Class to configure cron.
class cron {
    # Configure puppet agent to check in three times per hour.
    cron { 'puppet-agent':
        ensure  => 'present',
        command => 'opt/puppet/bin/puppet agent --onetime --no-daemonize --splay --splaylimit60',
        minute  => ['20'],
        target  => root,
        user    => root,
    }
}
```

The cron class ensures the puppet agent checks in three times per hour - using cron.

Packages



```
# Class to ensure presense of various packages.
class packages {
    # Ensure presence of bash from package manager, as an assignment dependency.
    package { 'bash':
        ensure => present,
    }

    # Ensure presence of csh from package manager, as an assignment dependency.
    package { 'tcsh':
        ensure => present,
    }

    # Ensure presence of wget from package manager, as an assignment dependency.
    package { 'wget':
        ensure => present,
    }

    # Ensure presence of OpenSSH server from package manager.
    package { 'openssh-server':
        ensure => present,
    }

    # Operating system case to handle variations in common distributions.
    case $::operatingsystem {
        'Debain', 'Ubuntu': {
            # Ensure presence of OpenSSH client from package manager.
            package { 'openssh-client':
                ensure => present,
            }

            # Ensure presence of Apache from package manager.
            package { 'apache2':
                ensure => present,
            }

            # Ensure presence of MySQL server from package manager.
            package { 'mysql-server':
                ensure => present,
            }

            # Ensure presence of MySQL client from package manager.
            package { 'mysql-client':
                ensure => present,
            }

            # Ensure presence of VNC server from package manager.
            package { 'vnc4server':
                ensure => present,
                alias  => 'vnc-server',
            }

            # Ensure presence of Dia2Code from package manager.
            package { 'dia2code':
                ensure => present,
            }
        }
        default: {
            # Ensure presence of OpenSSH client from package manager.
            package { 'openssh':
                ensure => present,
                alias  => 'openssh-client',
            }

            # Ensure presence of Apache from package manager.
            package { 'httpd':
                ensure => present,
                alias  => 'apache2',
            }

            # Ensure presence of package to add MySQL repository.
            package { 'mysql57-community-release-el7-11.noarch' :
                ensure   => 'installed',
                source   => 'https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm',
                provider => 'rpm',
            }

            # Ensure presence of MySQL server from package manager, requiring repository added via package.
            package { 'mysql-community-server':
                ensure  => present,
                require => Package['mysql57-community-release-el7-11.noarch'],
                alias   => 'mysql-server',
            }

            # Ensure presence of MySQL client from package manager.
            package { 'mysql-community-client':
                ensure  => present,
                require => Package['mysql57-community-release-el7-11.noarch'],
                alias   => 'mysql-client',
            }

            # Ensure presence of VNC server from package manager.
            package { 'tigervnc-server':
                ensure => present,
                alias  => 'vnc-server',
            }

            # Ensure installation of Dia2Code from source using rpm.
            package { 'dia2code-0.8.3-1.x86_64':
                ensure   => 'installed',
                source   => 'https://downloads.sourceforge.net/project/dia2code/dia2code/0.8.3/dia2code-0.8.3-1.x86_64.rpm?
r=https://sourceforge.net&ts=1507688055&use_mirror=jaist',
                provider => 'rpm',
                alias    => 'dia2code',
            }
        }
    }

    # Ensure presence of tmux from package manager.
    package { 'tmux':
        ensure => present,
    }

    # Ensure presence of Lynx from package manager.
    package { 'lynx':
        ensure => present,
    }

    # Ensure presence of GNU Compiler Collection from package manager.
    package { 'gcc':
        ensure => present,
    }

    # Ensure presence of GNU Debugger from package manager.
    package { 'gdb':
        ensure => present,
    }

    # Ensure presence of CGDB Debugging Interface from package manager.
    package { 'cgdb':
        ensure => present,
    }

    # Ensure presence of vim from package manager.
    package { 'vim':
        ensure => present,
    }

    # Ensure presence of Emacs from package manager.
    package { 'emacs':
        ensure => present,
    }

    # Ensure presence of SSHFS from package manager.
    package { 'sshfs':
        ensure => present,
    }
}
```

The packages class ensures the presence of various packages.

For the ***openssh*** requirement, I have installed both *openssh-client* and *openssh-server*. On CentOS, *openssh-client* is named *openssh*. On Ubuntu and Debian, *openssh-client* is named *openssh-client*. The operating system case handles this. *openssh-server* is named identically on each supported distribution.

For the ***Apache*** requirement, I installed *Apache*. On CentOS, *Apache* is named *httpd*. On Ubuntu and Debian, *Apache* is named *apache2*. The operating system case handles this.

For the ***MySQL Server*** requirement, I installed *mysql-server*. On CentOS, *mysql-server* is not included in the default repositories, and requires the installation of a package from the MySQL website to add the MySQL repository - *mysql-server* is then named *mysql-community-server*. The presence of the repository package is required before *mysql-community-server* is installed. On Ubuntu and Debian, *mysql-server* is included in the default repositories.

For the ***MySQL Client*** requirement, I installed *mysql-client*. On CentOs, *mysql-client* is not included in the default repositories, and requires the installation of a package from the MySQL website to add the MySQL repository - *mysql-client* is then named *mysql-community-client*. The presence of the repository package is required before *mysql-community-client* is installed. On Ubuntu and Debian, *mysql-client* is included in the default repositories.

For the ***vncserver*** requirement, I installed *tigervnc-server* on CentOS, and *vnc4server* on Ubuntu and Debian.

For the ***tmux*** requirement, I installed *tmux*.

For the ***dia2code*** requirement, I installed *dia2code*. On CentOS, *dia2code* is not included in the default repositories, and requires installation from source. On Ubuntu and Debian, *dia2code* is included in the default repositories.

For the ***lynx*** requirement, I installed *lynx*.

For the ***gcc*** requirement, I installed *gcc*.

For the ***gdb*** requirement, I installed *gdb*.

For the ***cgdb*** requirement, I installed *cgdb*.

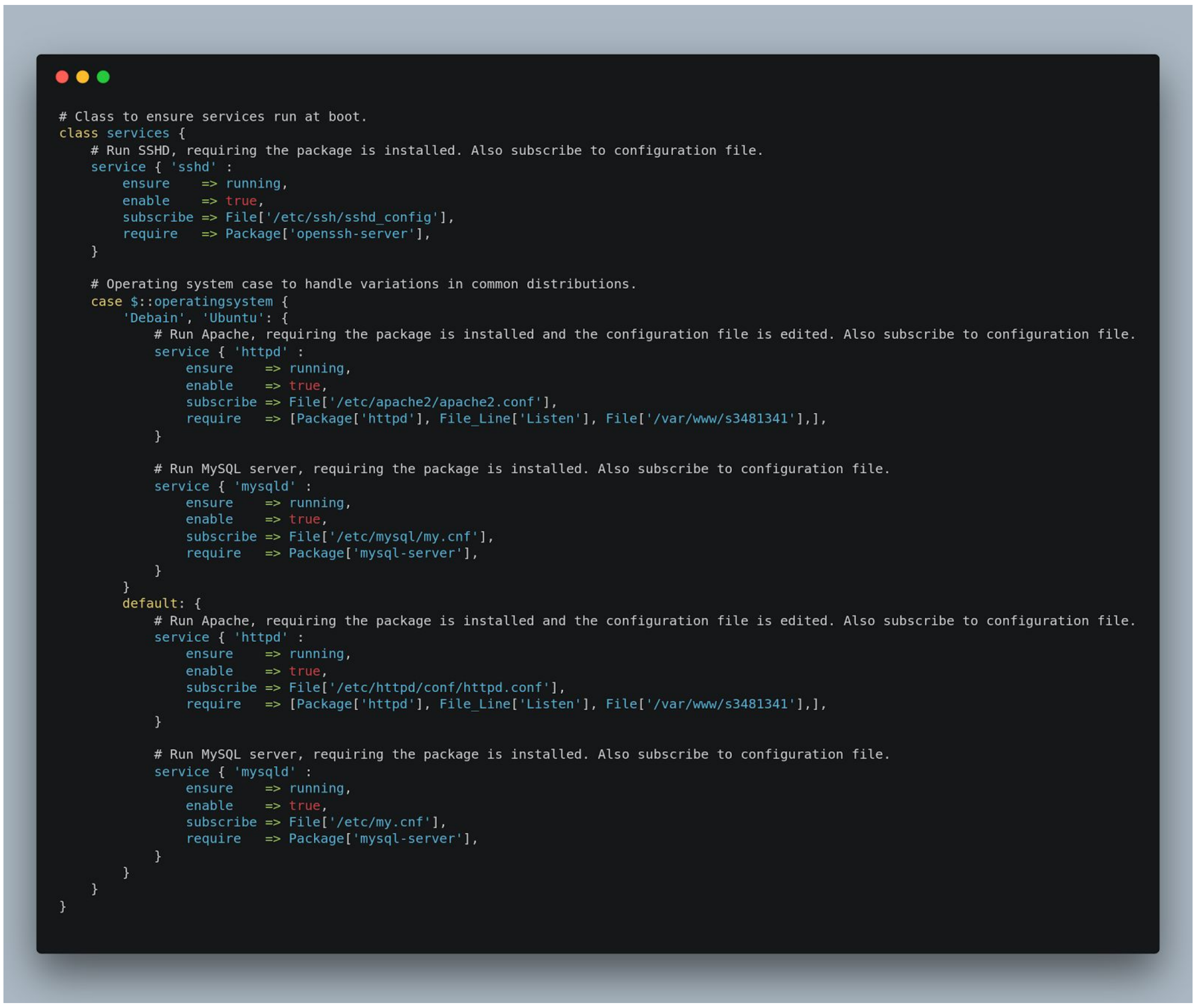
For the ***vim*** requirement, I installed *vim*.

For the ***emacs*** requirement, I installed *emacs*.

For the ***sshfs*** requirement, I installed *sshfs*.

I also installed *bash*, *tcsh*, and *wget* to support other assignment requirements and basic functionality.

Services



```
# Class to ensure services run at boot.
class services {
    # Run SSHD, requiring the package is installed. Also subscribe to configuration file.
    service { 'sshd' :
        ensure    => running,
        enable    => true,
        subscribe => File['/etc/ssh/sshd_config'],
        require   => Package['openssh-server'],
    }

    # Operating system case to handle variations in common distributions.
    case $::operatingsystem {
        'Debain', 'Ubuntu': {
            # Run Apache, requiring the package is installed and the configuration file is edited. Also subscribe to configuration file.
            service { 'httpd' :
                ensure    => running,
                enable    => true,
                subscribe => File['/etc/apache2/apache2.conf'],
                require   => [Package['httpd'], File_Line['Listen'], File['/var/www/s3481341'],],
            }

            # Run MySQL server, requiring the package is installed. Also subscribe to configuration file.
            service { 'mysqld' :
                ensure    => running,
                enable    => true,
                subscribe => File['/etc/mysql/my.cnf'],
                require   => Package['mysql-server'],
            }
        }
        default: {
            # Run Apache, requiring the package is installed and the configuration file is edited. Also subscribe to configuration file.
            service { 'httpd' :
                ensure    => running,
                enable    => true,
                subscribe => File['/etc/httpd/conf/httpd.conf'],
                require   => [Package['httpd'], File_Line['Listen'], File['/var/www/s3481341'],],
            }

            # Run MySQL server, requiring the package is installed. Also subscribe to configuration file.
            service { 'mysqld' :
                ensure    => running,
                enable    => true,
                subscribe => File['/etc/my.cnf'],
                require   => Package['mysql-server'],
            }
        }
    }
}
```

The *services* class ensures that various services run on boot.

The assignment specification requires that all required packages are run at boot. This is not possible, as many of the packages are not services.

Of the required packages, *openssh* (server), *Apache*, *MySQL Server*, and *vncserver* are services.

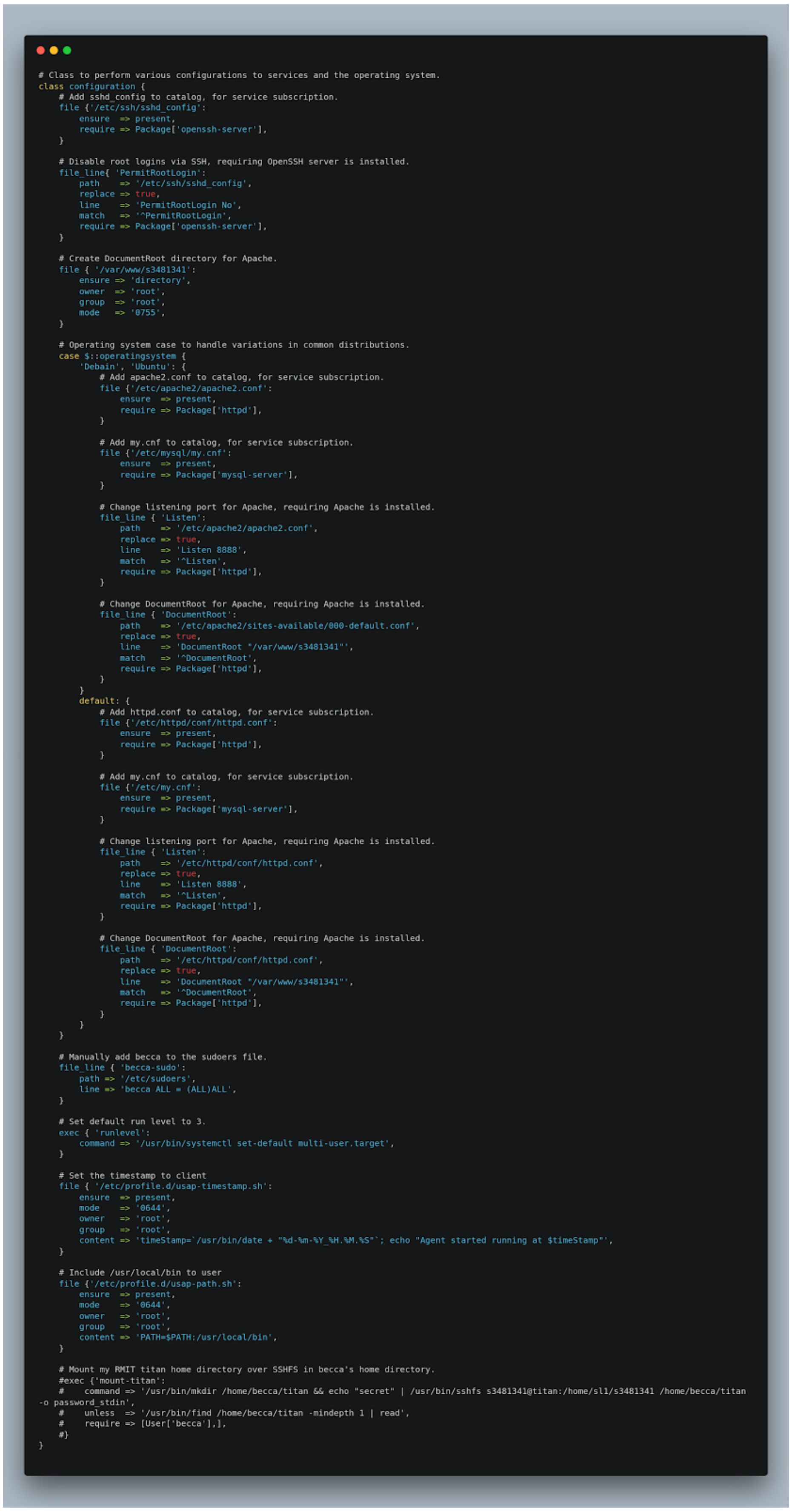
For the ***openssh*** requirement, the *openssh-server* service is ensured running and enabled. The */etc/sshd/sshd_config* file is subscribed to. The *openssh-server* package is required.

For the ***Apache*** requirement, in CentOS, the *httpd* service is ensured running and enabled, and the */etc/httpd/conf/httpd.conf* file is subscribed to. In Ubuntu and Debian, the *apache2* service is ensured running and enabled, and the */etc/apache2/apache2.conf* file is subscribed to. In both cases, the *httpd* package, and two configuration changes are required.

For the ***MySQL Server*** requirement, the *mysqld* service is ensured running and enabled. In CentOS, the */etc/my.cnf* file is subscribed to. In Ubuntu and Debian, the */etc/mysql/my.cnf* file is subscribed to.

I was unable to start the **vncserver** services for each respective operating system. In CentOS, following the documentation, I was unable to start the *tingervnc-server* service under any configuration due to a generic error and no available log.

Configuration



```
# Class to perform various configurations to services and the operating system.
class configuration {
    # Add sshd_config to catalog, for service subscription.
    file {'/etc/ssh/sshd_config':
        ensure  => present,
        require => Package['openssh-server'],
    }

    # Disable root logins via SSH, requiring OpenSSH server is installed.
    file_line{ 'PermitRootLogin':
        path    => '/etc/ssh/sshd_config',
        replace => true,
        line    => 'PermitRootLogin No',
        match   => '^PermitRootLogin',
        require => Package['openssh-server'],
    }

    # Create DocumentRoot directory for Apache.
    file { '/var/www/s3481341':
        ensure => 'directory',
        owner  => 'root',
        group  => 'root',
        mode   => '0755',
    }

    # Operating system case to handle variations in common distributions.
    case $::operatingsystem {
        'Debain', 'Ubuntu': {
            # Add apache2.conf to catalog, for service subscription.
            file {'/etc/apache2/apache2.conf':
                ensure  => present,
                require => Package['httpd'],
            }

            # Add my.cnf to catalog, for service subscription.
            file {'/etc/mysql/my.cnf':
                ensure  => present,
                require => Package['mysql-server'],
            }

            # Change listening port for Apache, requiring Apache is installed.
            file_line { 'Listen':
                path    => '/etc/apache2/apache2.conf',
                replace => true,
                line    => 'Listen 8888',
                match   => '^Listen',
                require => Package['httpd'],
            }

            # Change DocumentRoot for Apache, requiring Apache is installed.
            file_line { 'DocumentRoot':
                path    => '/etc/apache2/sites-available/000-default.conf',
                replace => true,
                line    => 'DocumentRoot "/var/www/s3481341"',
                match   => '^DocumentRoot',
                require => Package['httpd'],
            }
        }
        default: {
            # Add httpd.conf to catalog, for service subscription.
            file {'/etc/httpd/conf/httpd.conf':
                ensure  => present,
                require => Package['httpd'],
            }

            # Add my.cnf to catalog, for service subscription.
            file {'/etc/my.cnf':
                ensure  => present,
                require => Package['mysql-server'],
            }

            # Change listening port for Apache, requiring Apache is installed.
            file_line { 'Listen':
                path    => '/etc/httpd/conf/httpd.conf',
                replace => true,
                line    => 'Listen 8888',
                match   => '^Listen',
                require => Package['httpd'],
            }

            # Change DocumentRoot for Apache, requiring Apache is installed.
            file_line { 'DocumentRoot':
                path    => '/etc/httpd/conf/httpd.conf',
                replace => true,
                line    => 'DocumentRoot "/var/www/s3481341"',
                match   => '^DocumentRoot',
                require => Package['httpd'],
            }
        }
    }

    # Manually add becca to the sudoers file.
    file_line { 'becca-sudo':
        path => '/etc/sudoers',
        line => 'becca ALL = (ALL)ALL',
    }

    # Set default run level to 3.
    exec { 'runlevel':
        command => '/usr/bin/systemctl set-default multi-user.target',
    }

    # Set the timestamp to client
    file { '/etc/profile.d/usap-timestamp.sh':
        ensure  => present,
        mode    => '0644',
        owner   => 'root',
        group   => 'root',
        content => 'timeStamp=`/usr/bin/date + "%d-%m-%Y_%H.%M.%S"`; echo "Agent started running at $timeStamp"',
    }

    # Include /usr/local/bin to user
    file {'/etc/profile.d/usap-path.sh':
        ensure  => present,
        mode    => '0644',
        owner   => 'root',
        group   => 'root',
        content => 'PATH=$PATH:/usr/local/bin',
    }

    # Mount my RMIT titan home directory over SSHFS in becca's home directory.
    #exec {'mount-titan':
    #    command => '/usr/bin/mkdir /home/becca/titan && echo "secret" | /usr/bin/sshfs s3481341@titan:/home/sl1/s3481341 /home/becca/titan
-o password_stdin',
    #    unless  => '/usr/bin/find /home/becca/titan -mindepth 1 | read',
    #    require => [User['becca'],],
    #}
}
```

The *configuration* class ensures that various packages and the operating system are configured to requirement.

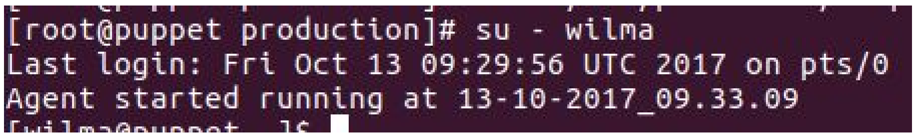
Root SSH logins are disabled by replacing any line that starts with *"PermitRootLogin"* with *"PermitRootLogin No"*. This modification requires the presence of the *openssh-server* package.

The document root for Apache is changed by ensuring the presence of the */var/www/s3481341* directory, and replacing any line in the Apache configuration (for each distribution) that starts with *"DocumentRoot"* with *"DocumentRoot "/var/www/s3481341""*.

User becca is granted sudo access via direct manual configuration of the */etc/sudoers* file. This is achieved by ensuring *"becca ALL = (ALL)ALL"* is appended to the end of the file.

/usr/local/bin is added is every user's path on the system by creating a script in */etc/profile.d* that executes **PATH=$PATH:/usr/local/bin**.

My home directory on RMIT's titan server is mounted over ssh in becca's home directory using an exec resource that executes **/usr/bin/mkdir /home/becca/titan && echo "secret" | /usr/bin/sshfs s3481341@titan:/home/sl1/s3481341 /home/becca/titan -o password_stdin**. *secret* is to be replaced with the user's password. This requires the presence of the becca user.


```
[root@puppet production]# su - wilma
Last login: Fri Oct 13 09:29:56 UTC 2017 on pts/0
Agent started running at 13-10-2017_09.33.09
```

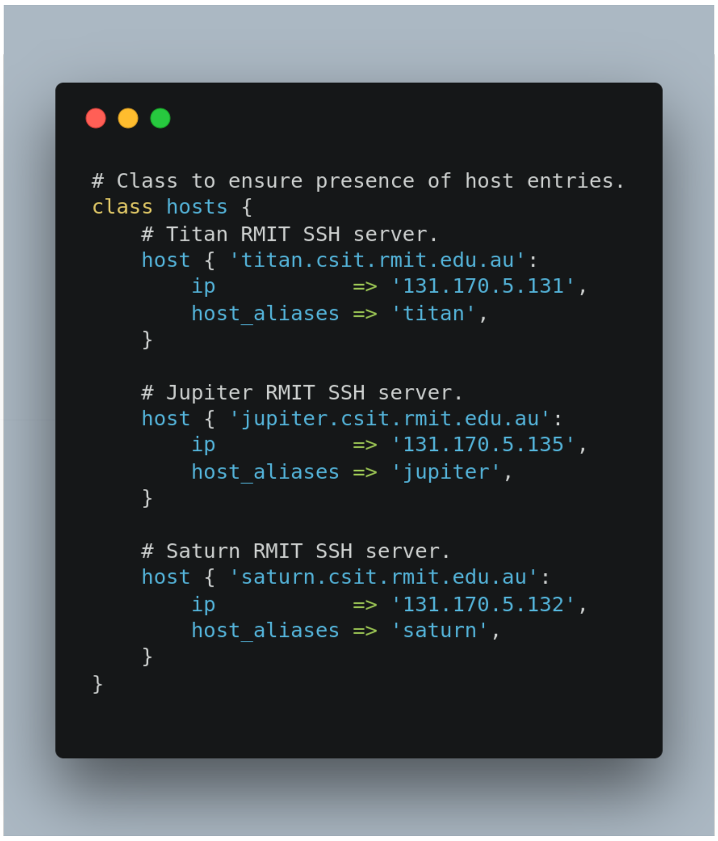
The timestamp requirement was unclear. The requirement was achieved by creating a script in */etc/profile.d* that executes an echo command on login/boot.

Services are started at run level three by using an *exec* resource to run the command **/usr/bin/systemctl set-default multi-user.target**.

I also had to change the Apache listening port. This is because *puppet-server* was bound to port 80. I changed the Apache listening port to 8888 via a *file_line* resource that replaces any line that starts with *"Listen"* with *"Listen 8888"*. This requires the presence of the *httpd* package.

Variations in supported distributions are handled by a case in this class.

Hosts



```
# Class to ensure presence of host entries.
class hosts {
    # Titan RMIT SSH server.
    host { 'titan.csit.rmit.edu.au':
        ip           => '131.170.5.131',
        host_aliases => 'titan',
    }

    # Jupiter RMIT SSH server.
    host { 'jupiter.csit.rmit.edu.au':
        ip           => '131.170.5.135',
        host_aliases => 'jupiter',
    }

    # Saturn RMIT SSH server.
    host { 'saturn.csit.rmit.edu.au':
        ip           => '131.170.5.132',
        host_aliases => 'saturn',
    }
}
```

The *hosts* class adds host entries for RMIT's titan, jupiter, and saturn servers to */etc/hosts* via a *host* resource.