ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ ΤΗΛΕΜΑΤΙΚΗΣ

ΜΑΖΙΚΗ ΔΙΑΧΕΙΡΙΣΗ ΛΕΙΤΟΥΡΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ LINUX ΜΕΣΩ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΩΝ ΔΙΑΔΙΚΑΣΙΩΝ

Διπλωματική Εργασία

Νάκος - Κοτσιώνης Κωνσταντίνος

Αθήνα, 2017

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ ΚΑΙ ΥΠΗΡΕΣΙΕΣ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

ΔΑΛΑΚΑΣ ΒΑΣΙΛΕΙΟΣ (Επιβλέπων)
Ε.ΔΙ.Π., Τμήμα Πληροφορικής & Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

ΚΑΜΑΛΑΚΗΣ ΘΩΜΑΣ
Επίκουρος Καθηγητής, Τμήμα Πληροφορικής & Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

ΤΣΑΔΗΜΑΣ ΑΝΑΡΓΥΡΟΣ
Ε.Τ.Ε.Π., Τμήμα Πληροφορικής & Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Ο Νάκος - Κοτσιώνης Κωνσταντίνος

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Στους παππούδες μου Σταύρο και Κώστα,

στις γιαγιάδες μου Μαίρη και Αγαθή,

με αγάπη και αιώνια ευγνωμοσύνη.

Εις ευ φρονών μυρίων μη φρονούντων κρείττων εστί.

Πλάτων, 427-347 π.Χ., Φιλόσοφος

Μετάφραση: Ένας που σκέφτεται σωστά είναι καλύτερος από μύριους που δεν σκέφτονται

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω την οικογένεια μου για την αγάπη τους, την υποστήριξη τους και την πίστη τους σε εμένα.

Κατά κύριο λόγο όμως οφείλω να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Δαλάκα Βασίλειο και τον κ. Τσαδήμα Ανάργυρο που με υποστήριξαν καθ' όλη τη διάρκεια της διπλωματικής εργασίας. Αισθάνομαι πολύ τυχερός που στο διάστημα αυτό είχα πάντα τη σιγουριά της υλικής αλλά και της ηθικής βοήθειας που χρειαζόμουν για να προχωρήσω.

Αθήνα, Φεβρουάριος 2017

Νάκος - Κοτσιώνης Κωνσταντίνος

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά

Η επιστήμη των υπολογιστών περιέχει πολλούς τομείς, σχεδόν όλοι όμως έχουν να κάνουν με την μερική ή ολική αυτοματοποίηση διαφόρων εργασιών. Η παραμετροποίηση και διαχείριση των υπολογιστών αποτελεί συχνά όμως μεγάλη ανάγκη και πρόκληση, ιδιαίτερα στις περιπτώσεις όπου το πλήθος των συσκευών είναι μεγάλο. Με αυτά τα δεδομένα αποφάσισα να ασχοληθώ με τις υπηρεσίες DevOps και πιο συγκεκριμένα με την εφαρμογή του Puppet. Απώτερος σκοπός η εφαρμογή του Puppet σε ένα από τα εργαστήρια του Χαροκοπείου πανεπιστημίου για την μαζική παραμετροποίηση των λειτουργικών συστημάτων των τερματικών.

Στο πρώτο κεφάλαιο γίνεται μια εισαγωγή στην έννοια DevOps και στα εργαλεία που μπορούν να χρησιμοποιηθούν για σκοπούς και ανάγκες αυτοματοποίησης και παραμετροποίησης λειτουργικών συστημάτων σε φάρμες υπολογιστών και σε Cloud συστήματα. Ακολουθεί η σύγκριση των εργαλείων αυτών και ο τρόπος εφαρμογής και προσέγγισης στο παρόν θέμα.

Στο δεύτερο κεφάλαιο λαμβάνει χώρα η ανάλυση των υπηρεσιών, των λειτουργικών συστημάτων που επρόκειτο να παραμετροποιηθούν καθώς και η κατανόηση αυτών. Επιπροσθέτως ακολουθεί μια εισαγωγή στη διαχείριση λειτουργικών συστημάτων Linux μέσω SSH, VI και προγραμματισμένων εργασιών. Ακολουθεί η ανάλυση του Puppet και των εργαλείων αυτού. Πιο συγκεκριμένα της γλώσσας προγραμματισμού Ruby που εφαρμόζεται στο Puppet αλλά και των εννοιών manifests, modules, forge, hiera. Στο τελευταίο κομμάτι αυτού του κεφαλαίου περιγράφεται ο τρόπος δοκιμής του εργαλείου αυτού σε ένα περιβάλλον με εικονικές μηχανές.

Στο τρίτο κεφάλαιο βρίσκεται η πρακτική εφαρμογή για ένα συγκεκριμένο σενάριο όπου προσπαθούμε να αυτοματοποιήσουμε κάποιες από τις διεργασίες σε ένα εργαστήριο υπολογιστών του Τμήματος Πληροφορικής και Τηλεματικής. Παρουσιάζονται όλα της τα στάδια από την εγκατάσταση μέχρι και τα αποτελέσματα. Στο πρώτο στάδιο εξηγούνται όλα τα βήματα για την εγκατάσταση του εργαλείου αυτού τόσο στον διακομιστή όσο και στο τερματικό, με εικόνες και ανάλυση των εντολών. Η εγκατάσταση ολοκληρώνεται με την επιτυχή σύνδεση του διακομιστή με τα τερματικά. Ακολουθεί το κύριο κομμάτι της εργασίας όπου γίνεται εκτενής ανάλυση των λειτουργιών του Puppet και της διαχείρισης αυτού ούτως ώστε να έχουμε το επιθυμητό αποτέλεσμα. Σε αυτό το σημείο αναλύεται ο τρόπος σκέψης και εξηγείται ο τρόπος προγραμματισμού των απαιτούμενων αρχείων για την παραμετροποίηση των υπηρεσιών LDAP, NSLCD, την εγκατάσταση της εφαρμογής VIM και φυσικά την αυτόματη ενημέρωση όλου του λειτουργικού συστήματος. Αυτό το κεφάλαιο κλείνει με την παρουσίαση των τρόπων επίβλεψης των αποτελεσμάτων μέσα από τα logs του συστήματος.

Στο τέταρτο και τελευταίο κεφάλαιο παρουσιάζονται συνολικά τα αποτελέσματα της εφαρμογής και ορισμένα συμπεράσματα που προέκυψαν μέσα από την εφαρμογή. Εντέλει γίνεται μια αναφορά για μελλοντική έρευνα πάνω στο ίδιο τομέα με επιπρόσθετες υπηρεσίες, εργαλεία και εφαρμογή.

Λέξεις κλειδιά: αυτοματοποίηση, διαχείριση, προγραμματισμός, λειτουργικά συστήματα, έλεγχος

Abstract

Computer science has many aspects, but most of them have to do with total or partial automation. Modification and management of computers is a constant need, and a challenge on the situation where the terminal devices are many. Given the above i have decided to focus on DevOps services and particularly on the implementation of Puppet software. The final goal is to apply Puppet on one of the Universities laboratories in needs of mass modification on the operating system of the terminals.

On first chapter, you will find an introduction for DevOps and the tools that can be used for means and needs of automating modification on operating systems on large scale, including cloud systems. Followed by a comparison of these tools, approach method and application on the given subject.

On second chapter, takes place an analysis on the services that will be used, the operating systems that will be modified and their understanding. Additionally, an introduction takes place on managing Linux operating systems via SSH, VI and scheduled tasks. Followed by an analysis on Puppet and its tools. More precisely on the programming language, Ruby that is used on Puppet and the meaning of manifests, modules, forge and hiera. On the last part of this chapter you will find a description on the testing method of these tools on a virtual machine environment.

On third chapter lays the practical application for a specific scenario, where we try to automate some tasks on a computer laboratory of the Department of Informatics and Telematics. All stages from installation through the results are presented. On the first stage the installation of this tool on the server and the client are explained with images and command analysis. Installation is completed with the successful connection of the server with the terminals. What comes next is the main subject where the analysis of Puppet operations takes place along with the management, so the desired result can occur. On this part the way of thinking is analyzed as well as the programming thinking for modifying the files needed for the services LDAP and NSLCD, the installation of VIM application and an automated operating system update. This chapter closes with the presentation of the ways to oversight the results through system logs.

On the fourth and last chapter, there is a presentation of the total results of the application and the conclusion through them. On the final part a future reference takes place for research on the same sector with additional services, tools and application.

Keywords: puppet, devops, linux, automation, management

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

Ops	Operations
Dev	Developers
DevOps	Developers and Operations
Xfce	XForms Common Environment
Conf	Configuration
SSH	Secure Shell
DNS	Domain Name System
VPN	Virtual Private Network
LDAP	Lightweight Directory Access Protocol

ΕΙΣΑΓΩΓΗ

Η πληροφορική στις μέρες μας εφαρμόζεται σε όλο και περισσότερους τομείς, με σκοπό την αυτοματοποίηση πολύπλοκων εργασιών και τον εκμηδενισμό του ενδεχομένου σφάλματος. Ο κάθε άνθρωπος σε αντίθεση με το παρελθόν πλέον είναι κάτοχος ενός ή και περισσότερων υπολογιστών, κάτι που μας κάνει να εξαρτώμαστε από αυτούς ολοένα και περισσότερο. Αποτελούν αναγκαίο εργαλείο διδασκαλίας και εκμάθησης και συχνά τους συναντούμε σε μεγάλος πλήθος μέσα σε εργαστήρια. Έτσι και στο Χαροκόπειο Πανεπιστήμιο οι ανάλογοι χώροι δέχονται κάθε μέρα πολλούς χρήστες. Σε ένα περιβάλλον με τόσους πολλούς χρήστες, οι απαιτήσεις για υπηρεσίες, ασφάλεια και παραμετροποίηση είναι αυξημένες. Ποιος είναι όμως ο καλύτερος τρόπος για να καλυφθούν αυτές οι ανάγκες; Η ομάδα των διαχειριστών όσο επαρκώς στελεχωμένη και να είναι πολλές φορές ίσως να μην είναι εφικτό να καλύψει άμεσα αυτές τις ανάγκες. Έτσι κρίνεται απαραίτητη η δημιουργία μίας υποδομής που θα αυτοματοποιεί συγκεκριμένες εργασίες για την άμεση παραμετροποίηση των υπολογιστών ενός εργαστηρίου. Επιπροσθέτως αυτοματοποιώντας αυτές τις εργασίες μπορεί να επιτευχθεί αύξηση της ασφάλειας των λειτουργικών συστημάτων και διασφάλιση της ορθής λειτουργίας τους.

ΚΕΦ.1: DevOps

Σε αυτό το κεφάλαιο αρχικά γίνεται μια εισαγωγή στις έννοιες και στα εργαλεία του κλάδου DevOps. Ακολουθεί μια σύντομη ανάλυση για τα εργαλεία αυτά και κλείνοντας η σύγκριση και επιλογή ενός από αυτά.

1.1 Τι σημαίνει DevOps;

Φανταστείτε τον Βασίλη, έναν εργαζόμενο σε μια εταιρία πληροφορικής με τομέα δράσης τον προγραμματισμό. Ο Βασίλης γράφει κώδικα για αυτή την επιχείρηση και συγκεκριμένα αναπτύσσει νέα προϊόντα , νέες επιλογές, ενημερώσεις ασφάλειας και επιδιορθώσεις στα ήδη υπάρχοντα. Ο Βασίλης πρέπει να περιμένει βδομάδες μέχρι το προϊόν της εργασίας του να βγει στην παραγωγή. Αυτή η καθυστέρηση αυξάνει την πίεση του ανταγωνισμού αφού οι ανταγωνιστές πολλές φορές το κάνουν πιο γρήγορα από την επιχείρηση που εργάζεται. Έτσι ο Βασίλης πρέπει να ακροβατεί μεταξύ παρελθοντικών, τωρινών και μελλοντικών εργασιών του. Όταν η εργασία του Βασίλη φτάνει στην παραγωγή πολλές φορές εμφανίζονται πολλά και διάφορα αναπάντεχα λάθη. Αυτό συνήθως συμβαίνει γιατί ο Βασίλης γράφει κώδικα σε ένα περιβάλλον προγραμματιστικό, κάτι που διαφέρει πολύ από ένα περιβάλλον παραγωγής.

Τώρα φανταστείτε τον Κώστα, έναν συνάδελφο του Βασίλη στην ίδια εταιρία στο τμήμα διαχείρισης λειτουργικών συστημάτων. Ο Κώστας είναι υπεύθυνος για την διαρκή λειτουργία των συστημάτων της επιχείρησης για την γραμμή παραγωγής. Ο αριθμός των διακομιστών που διαχειρίζεται ο Κώστας αυξάνεται συνεχώς αφού η επιχείρηση συνεχίζει να παράγει νέα προϊόντα και οι καταναλωτές χρησιμοποιούν όλο και περισσότερο τι υπηρεσίες

της επιχείρησης. Αυτή η αύξηση του αριθμού των διακομιστών δημιουργεί πολλές προκλήσεις και για τον Κώστα, αφού τα εργαλεία που χρησιμοποιεί για να διαχειρίζεται τους διακομιστές μερικές φορές μπορούν να αποδειχτούν ανεπαρκή για ένα μεγαλύτερο πλήθος διακομιστών. Αυτό επηρεάζει πώς ένα νέο κομμάτι κώδικα μπορεί να εφαρμοστεί πάνω στο παραγωγικό περιβάλλον. Συνήθως έναν νέο κομμάτι κώδικα πρέπει να δεχτεί τροποποιήσεις για να μπορέσει να ενσωματωθεί στην παραγωγή. Έτσι για να υπάρχει μια ισορροπία απαιτείται η παρουσία ενός χρονοδιαγράμματος εφαρμογής νέου κώδικα στην παραγωγή και για παράδειγμα, ότι αυτό μπορεί να γίνεται μόνο μια φορά τον μήνα. Είναι ευθύνη του Κώστα να ταιριάξει το νέο κομμάτι κώδικα στην παραγωγή καθώς και η διάγνωση σφαλμάτων και προβλημάτων σχετικά με αυτό και έτσι πολλές φορές ο Κώστας αισθάνεται σαν του έχουν πετάξει ένα κομμάτι κώδικα από το τμήμα του προγραμματισμού και πρέπει αυτός μόνος του να βγάλει την άκρη.

Οπότε τι θα μπορούσε να γίνει έτσι ώστε να βοηθηθεί η κατάσταση του Κώστα και του Βασίλη και να μπορέσουν να εργαστούν με έναν πιο αποδοτικό τρόπο; Στην ουσία και οι δύο θέλουν το ίδιο πράγμα που θέλει και η επιχείρηση, δηλαδή ευτυχισμένους τελικούς χρήστες. Για να το επιτύχουν αυτό θα μπορούσαν να εργάζονται μαζί ταυτόχρονα, θα μπορούσαν να προσπαθήσουν να σκέφτονται και να ενεργούν συλλογικά, να καταργήσουν τα τμήματα που τους διαχωρίζουν και να μοιραστούν την ευθύνη. Η αγγλική λέξη για τον Διαχειριστή λειτουργιών είναι Operation ή Ops και για τον Προγραμματιστή είναι Developer ή Dev, αναζητείται δηλαδή η ένωση αυτών των δύο σε ένα. Κάπως έτσι δημιουργήθηκε ο όρος DevOps.

Άρα DevOps είναι η ένωση διαχειριστών και προγραμματιστών, έτσι ώστε να υπάρξει βελτίωση συνεργασίας και παραγωγικότητας μέσω αυτοματοποιημένων υποδομών, χρονοδιαγραμμάτων και συνεχούς καταμέτρησης της απόδοσης των εφαρμογών [1].

Σε μια ομάδα DevOps σκοπός είναι η αυτοματοποίηση όλων των εργασιών, η αυτοματοποίηση του κώδικα ελέγχου, των χρονοδιαγραμμάτων και των εγκαταστάσεων. Το πλεονέκτημα βρίσκεται στο κοινό πεδίο δράσης και εφαρμογής των εργασιών. Με μικρότερα

κομμάτια κώδικα και ταχύτερη εφαρμογή μπορεί να επιτευχθεί πολύ μεγαλύτερη παραγωγικότητα, αποδοτικότητα και έλεγχος, χωρίς προβλήματα.

1.2 Τι θέλουμε να κάνουμε

Συχνά συναντούμε χώρους ειδικά διαμορφωμένους με μεγάλο πλήθος ηλεκτρονικών υπολογιστών, που φιλοξενούν πολλούς χρήστες. Οι χρήστες αυτοί μπορεί να έχουν διαφορετικές απαιτήσεις όσον αφορά την χρήση από τα συστήματα αυτά. Ένας τέτοιος χώρος είναι και τα εργαστήρια υπολογιστών στα πανεπιστήμια.

Το πεδίο εφαρμογής της διπλωματικής αυτής είναι το εργαστήριο κοινής χρήσης υπολογιστών στον δεύτερο όροφο του Χαροκοπείου πανεπιστημίου. Το εργαστήριο αποτελείται από περίπου 35 υπολογιστές ίδιου τύπου. Καθημερινά μεγάλος πλήθος φοιτητών επισκέπτονται τον χώρο αυτό για να παρακολουθήσουν μαθήματα, να εκτελέσουν έρευνα καθώς και να ικανοποιήσουν τις προσωπικές τους ανάγκες για κοινή χρήση. Όπως γίνεται αντιληπτό η χρήση και οι χρήστες ποικίλουν και έτσι οι απαιτήσεις των δυνατοτήτων αυτών των υπολογιστών είναι αυξημένες. Οι χρήστες θα πρέπει να έχουν πρόσβαση, μόνο στα απαραίτητα και διαθέσιμα στοιχεία σύμφωνα με τα πρότυπα. Για μία τέτοιου είδους ανάγκη η χρήση υπηρεσιών ενεργού καταλόγου είναι απαραίτητη [31]. Τι γίνεται όμως στην περίπτωση που το λειτουργικό σύστημα δεν είναι Windows αλλά Linux; Ακόμα και σε αυτή την περίπτωση υπάρχουν τα ανάλογα προγράμματα εφαρμογής των υπηρεσιών αυτών. Το λειτουργικό σύστημα των υπολογιστών αυτών είναι της διανομής Xubuntu και για την εφαρμογή υπηρεσιών ενεργού καταλόγου, γίνεται χρήση του πρωτοκόλλου LDAP μέσω proxy cache [30]. Τι γίνεται όμως όταν η μοναδική IP διεύθυνση του OpenLDAP διακομιστή που χρησιμοποιεί αλλάξει; Τι γίνεται όταν θέλουμε να εγκαταστήσουμε ένα νέο λογισμικό; Τι γίνεται όταν θέλουμε να περάσουμε ενημερώσεις; Πριν από λίγα χρόνια η απάντηση θα ήταν η εκτέλεση των απαραίτητων ενεργειών για την παραμετροποίηση και ενημέρωση των απαιτούμενων

προγραμμάτων και αρχείων σε κάθε ένα από τα μηχανήματα ξεχωριστά. Αυτό όμως δεν είναι απαραίτητο αφού όλη αυτή η διαδικασία μπορεί να αυτοματοποιηθεί με την χρήση των κατάλληλων εφαρμογών.

Σκοπός αυτής της διπλωματικής είναι η παραμετροποίηση των απαιτούμενων αρχείων που στην περίπτωσή μας είναι τα ldap.conf, nslcd.conf, επανεκκίνηση των διεργασιών LDAP και NSLCD, η εγκατάσταση της εφαρμογής VIM και η εγκατάσταση των τελευταίων ενημερώσεων λογισμικού σε λειτουργικό σύστημα Linux-Xubuntu με την χρήση κάποιου λογισμικού αυτόματης εφαρμογής κώδικα.

1.3 Σχετικά Εργαλεία

Τα εργαλεία ποικίλουν, όμως ποιες είναι οι διαφορές τους; Ποιο εργαλείο θα ήταν καλύτερο να χρησιμοποιήσουμε για τον σκοπό μας; Ας αναλύσουμε το κάθε εργαλείο ξεχωριστά.

- Chef

Το Chef είναι ένα εργαλείο αυτοματοποίησης, προϊόν της εταιρίας Chef παλαιότερα γνωστή και ως Opscode. Διαθέτει και ανοιχτή, χωρίς χρέωση έκδοση αλλά και επί πληρωμή που προσφέρει επιπλέον υπηρεσίες. Έχει πολλά βοηθητικά προγράμματα και μεγάλη κοινότητα υποστήριξης. Οι εφαρμογές του δεν χωρίζονται σύμφωνα με τις εκδόσεις λειτουργικού συστήματος που θα εφαρμοστούν αλλά σε διακομιστή, πελάτη και βοηθητικών υποπρογραμμάτων [2].

Εικόνα 1 : Λογότυπο Chef

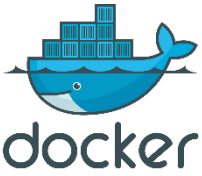
- Puppet

Το Puppet αποτελεί μαζί με το Chef ένα από τα πρωτοπόρα προγράμματα στον χώρο του. Διαθέτει και αυτό ανοιχτή και χωρίς χρέωση έκδοση καθώς και επί πληρωμή. Εδώ αξίζει να αναφέρουμε ότι το Puppet προσφέρει την εμπορική του έκδοση δωρεάν για διαχείριση μέχρι και δέκα μηχανημάτων. Η κοινότητα του είναι πολύ μεγάλη και διαθέτει τεράστιες βιβλιοθήκες με έτοιμα για χρήση modules. Διαθέτει μεγάλη γκάμα προγραμμάτων ανάλογα και με τον τύπο και την έκδοση του λειτουργικού συστήματος που πρόκειται να εφαρμοστεί. Βασίζεται και αυτό στο μοντέλο Πελάτη- Διακομιστή και είναι αρκετά ευέλικτο [3].

Εικόνα 2 : Λογότυπο Puppet

- Docker

Το Docker εστιάζει περισσότερο σε Cloud υπηρεσίες και πλατφόρμες εφαρμογής. Είναι αρκετά ποιο ελαφρύ και σε σχέση με τα υπόλοιπα όσον αφορά τις απαιτήσεις του. Περιλαμβάνει και ελεύθερη χωρίς χρεώσεις έκδοση αλλά και επί πληρωμή [4].



Εικόνα 3 : Λογότυπο Docker

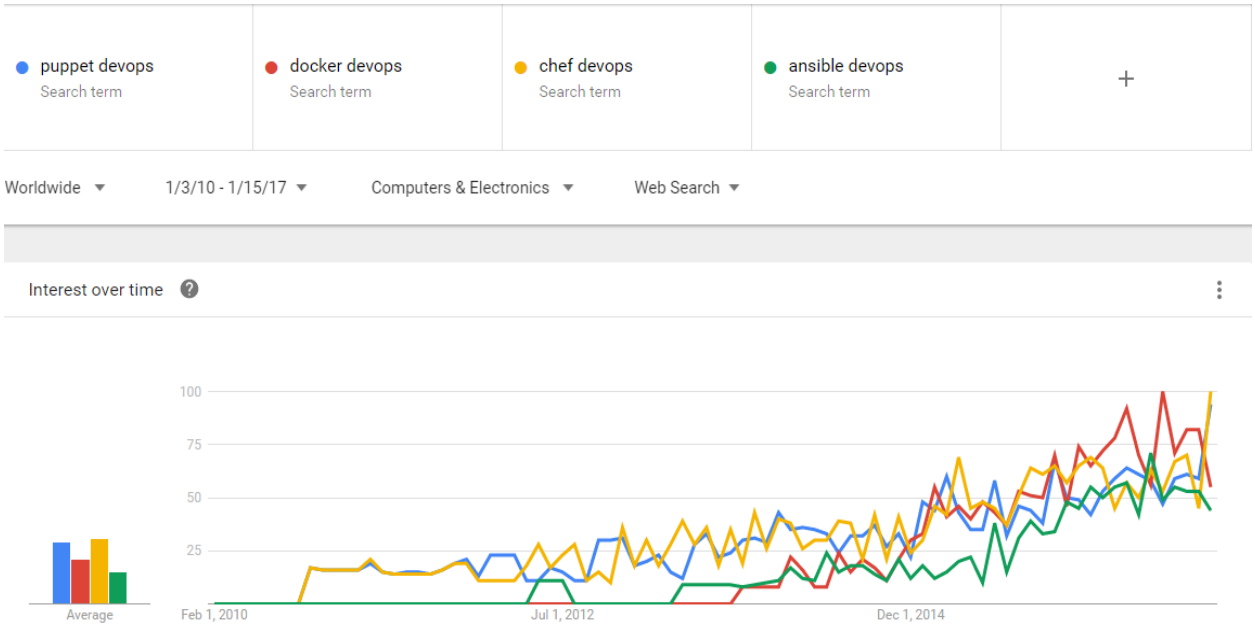
- Ansible

Όπως και τα προηγούμενα έτσι και το Ansible περιλαμβάνει και ελεύθερη αλλά και εμπορική έκδοση. Βασίζεται σε μία κεντρική πλατφόρμα διαχείρισης και εστιάζει περισσότερο σε κοινές και παραγωγικές διαδικασίες [5].

Εικόνα 4 : Λογότυπο Ansible

1.4 Σύγκριση και επιλογή

Όπως γίνεται αντιληπτό ο κλάδος περιέχει πολλά εργαλεία με διάφορες εφαρμογές. Τα πιο ευρέος διαδεδομένα εργαλεία αυτοματοποίησης εφαρμογής κώδικα παραμετροποίησης όμως σύμφωνα με τις αναζητήσεις είναι κατά σειρά το Chef, Puppet, Docker και Ansible [6].



Εικόνα 5 : Σύγκριση Υπηρεσιών

Λαμβάνοντας υπόψιν τα ιδιαίτερα χαρακτηριστικά του κάθε λογισμικού, το Puppet και το Chef θα μπορούσαν και τα δύο να αποτελούν μια πολύ καλή λύση για τον σκοπό αυτής της διπλωματικής εργασίας. Ανάμεσα σε αυτά τα δύο όμως η ελεύθερη προς χρήση έκδοση του Puppet είναι αρκετά ποιο πλούσια και ευέλικτη στην χρήση της. Το Puppet έχει τεράστια κοινότητα υποστήριξης και υποστηρίζεται από πάρα πολλές πλατφόρμες. Μπορεί να αποτελέσει κατάλληλο εργαλείο τόσο για μια απλή αυτοματοποίηση εφαρμογής παραμετροποιήσεων και ενημερώσεων όσο και για μια ποιο εξειδικευμένη εφαρμογή κώδικά με ιδιαίτερα χαρακτηριστικά. Για μια εφαρμογή σε λειτουργικά συστήματα όπου η διανομή βασίζεται σε Debian Linux είναι καλύτερο από όλα τα υπόλοιπα, διότι απευθύνεται περισσότερο προς αυτά και ταυτόχρονα υποστηρίζεται περισσότερο από αυτά [3].

ΚΕΦ.2: Ανάλυση υπηρεσιών και εφαρμογών

Σε αυτό το κεφάλαιο αναλύονται όλες οι υπηρεσίες οι οποίες θα τροποποιηθούν στα τερματικά, καθώς και αυτές που θα χρησιμοποιηθούν για την επεξεργασία, εγκατάσταση και διαχείριση αυτών. Στο τελευταίο κομμάτι του κεφαλαίου παρουσιάζεται η μέθοδος λειτουργίας του Puppet.

2.1 Διαχείριση Linux

Συχνά η τροποποίηση των λειτουργικών συστημάτων για τα οποία δεν υπάρχει επαρκής γνώση μπορεί να προκαλέσει προβλήματα. Έτσι σε αυτό το σημείο γίνεται μια αναφορά σε αυτά που θα χρησιμοποιηθούν και στον τρόπο διαχείρισης τους.

Όπως έχει προαναφερθεί και στο πρώτο κεφάλαιο τα τερματικά λειτουργούν με λειτουργικό σύστημα Xubuntu [7], το οποίο είναι βασισμένο σε Debian διανομή Linux και απλά χρησιμοποιεί έναν διαφορετικό τρόπο παρουσίασης και διεπαφής με τον χρήστη, το Xfce [8].

Εικόνα 6 : Λογότυπο Xubuntu

Για τις ανάγκες του διακομιστή θα χρησιμοποιηθεί μια εικονική μηχανή από τις ήδη υπάρχουσες στην υποδομή του ιδρύματος, με λειτουργικό σύστημα Debian [9].

Εικόνα 7 : Λογότυπο Debian

Πρόκειται για αρίστης ποιότητας και αξιοπιστίας λειτουργικά συστήματα διαρκώς ενημερωμένα και ανανεωμένα στο χώρο. Ένα λειτουργικό σύστημα όμως απαρτίζεται από πολλά προγράμματα και λειτουργίες, όμως στην παρούσα εργασία χρειάζονται μόνο μερικά από αυτά. Χωρίζοντας τους τομείς δράσης, προκύπτουν δύο σημεία εφαρμογής. Το ένα βρίσκεται στα τερματικά μηχανήματα μέσα στο εργαστήριο και το δεύτερο στον διακομιστή. Για τις ανάγκες διαχείρισης των συστημάτων είναι απαραίτητη μία εφαρμογή που να επιτρέπει την ενεργή εκτέλεση εντολών πάνω σε αυτό. Όλες οι υπηρεσίες διαθέτουν αρχεία ρυθμίσεων για την παραμετροποίηση τους και συνήθως στην μορφή conf. Για την παραμετροποίηση αυτών θα πρέπει να χρησιμοποιηθεί ένας κειμενογράφος. Ένα κομμάτι της αυτοματοποίησης των εφαρμογών απαιτεί την χρήση μιας εφαρμογής προγραμματισμένων ενεργειών. Η φυσική παρουσία όμως για τις ανάγκες διαχείρισης των συστημάτων αυτών δεν είναι πάντοτε εφικτή. Έτσι απαιτείται και η χρήση μιας εφαρμογής για απομακρυσμένη διαχείριση.

2.1.1 Διαχείριση και τηλεμετρία

Τα λειτουργικά συστήματα που χρησιμοποιούνται περιλαμβάνουν σχεδόν την ίδια γκάμα προγραμμάτων. Έτσι για τις ανάγκες διαχείρισης θα μπορούσε σε όλες τις περιπτώσεις να χρησιμοποιηθεί η εφαρμογή Terminal. Ταυτόχρονα όμως πρέπει να χρησιμοποιηθεί και μια εφαρμογή για διαχείριση εξ αποστάσεως. Θα μπορούσε να χρησιμοποιηθεί μία εφαρμογής τηλεμετρίας με την δυνατότητα υποστήριξης γραφικού περιβάλλοντος, όμως αυτό θα απορροφούσε αρκετούς πόρους από το σύστημα κάτι που δεν είναι πάντοτε επιθυμητό.

Όλες όμως αυτές οι ανάγκες μπορούν να καλυφθούν με την χρήση ενός και μόνο πρωτοκόλλου με την ονομασία SSH. Πρόκειται για ένα πρωτόκολλο επικοινωνίας βασισμένο σε αρχιτεκτονική Πελάτη - Εξυπηρετητή που επιτρέπει την διαχείριση του λειτουργικού συστήματος από απόσταση με την χρήση γραμμής εντολών. Ιδιαίτερα διάσημο για την ασφάλεια του και την αξιοπιστία του μπορεί να προσφέρει διαχείριση μέσω δικτύου για λειτουργικά συστήματα κυρίως Linux [10].

2.1.2 Κειμενογράφοι

Κάθε λογισμικό διαθέτει αρχεία απαραίτητα για την λειτουργία του και ένα από αυτά πολλές φορές έχει την ονομασία του λογισμικού και ακολουθείται από την κατάληξη .conf. Πρόκειται για αρχεία με ιδιαίτερη σημασία καθώς περιέχουν συγκεντρωμένη την πληροφορία ρυθμίσεων λειτουργίας για το εκάστοτε λογισμικό. Αυτά τα αρχεία περιέχουν κείμενο που ορίζει την κάθε διαθέσιμη για παραμετροποίηση παράμετρο. Έτσι η χρήση ενός λογισμικού διαχείρισης αρχείων κειμένου είναι απαραίτητη.

Οι επιλογές είναι αρκετές αλλά μια από τις παλαιότερες και πολύ αξιόπιστες είναι το VI. Πολύ γνωστή και συνήθως πάντα διαθέσιμη σε κάθε περιβάλλον [11]. Το συγκεκριμένο εργαλείο έχει κάποιες ιδιαιτερότητες στην χρήση του, που σε κάποιο ανάλογο του να μην υπάρχουν. Όμως αποτελεί συχνά την καλύτερη επιλογή λόγο της άμεσης διαθεσιμότητας του και αξιοπιστίας του.

2.1.3 Προγραμματισμένες εργασίες

Πέραν του βασικού λογισμικού (Puppet) που θα χρησιμοποιηθεί για την αυτοματοποίηση, είναι απαραίτητη και η χρήση ενός ακόμα που να προσφέρει την δυνατότητα προγραμματισμένων διεργασιών με χρονοδιάγραμμα. Αυτό συμβαίνει διότι πρέπει με κάποιο τρόπο να οριστεί από την μεριά των υπολογιστών του εργαστηρίου, μια αυτόματη διαδικασία εκτέλεσης των απαραίτητων εντολών για την εκκίνηση των εργασιών ενημέρωσης αυτών από το Puppet.

Για τις ανάγκες αυτές επιλέχθηκε το Cron. Πρόκειται για μια υπηρεσία που εφαρμόζει αυτόματα συγκεκριμένες και προκαθορισμένες εργασίες με βάση ένα προκαθορισμένο χρονοδιάγραμμα. Η χρήση του είναι εφικτή μόνο σε λειτουργικά συστήματα τύπου Linux και συνήθως υπάρχει ήδη διαθέσιμο σε σχεδόν όλες τις διανομές [12].

2.1.4 Ασφαλές περιβάλλον δοκιμών με εικονικές μηχανές

Το Puppet αποτελεί ένα πολύ ισχυρό εργαλείο το οποίο μπορεί να “επισκευάσει” πολύ γρήγορα ένα μεγάλο πλήθος λειτουργικών συστημάτων αλλά μπορεί και πολύ γρήγορα να το καταστρέψει.

Για τον λόγο αυτό η εξοικείωση με το εργαλείο αυτό κρίνεται αναγκαία μέσω πειραματικών δοκιμών. Ένα πολύ ασφαλές, αξιόπιστο και εύχρηστο περιβάλλον χρήσης θα ήταν μια εικονική μηχανή. Έτσι με χρήση της εφαρμογής VMWare Fusion [21] σε περιβάλλον λειτουργικού συστήματος OSX [22], δημιουργήθηκαν δύο εικονικές μηχανές με λειτουργικό σύστημα Linux Debian στην πρώτη, για τις ανάγκες του εξυπηρετητή και Linux Xubuntu στην δεύτερη, για του πελάτη.

Εικόνα 8 : Λογότυπο WMware Fusion

Η επιλογή λειτουργικού συστήματος δεν ήταν καθόλου τυχαία αλλά ανάλογη του πραγματικού περιβάλλοντος εφαρμογής μέχρι και στην έκδοση του λειτουργικού συστήματος. Με την τεχνολογία αυτή υπάρχει το πλεονέκτημα του snapshot που προσφέρει μεγαλύτερη ευελιξία στις δοκιμές, αφού μπορεί πολύ εύκολα και γρήγορα να εκτελεστεί επαναφορά του λειτουργικού συστήματος σε μία παλαιότερη κατάσταση του.

2.2 Υπηρεσίες προς παραμετροποίηση

Όπως στα λειτουργικά συστήματα έτσι και στις υπηρεσίες τους η τροποποίηση τους όταν δεν υπάρχει επαρκής γνώση μπορεί να προκαλέσει σφάλματα και δυσκολίες πολλές. Έτσι σε αυτό το σημείο γίνεται αναφορά στις υπηρεσίες που θα τροποποιηθούν στα τερματικά του εργαστηρίου.

2.2.1 LDAP, NSLCD, GETENT

Ένας από τους πιο αποδοτικούς τρόπους μαζικής διαχείρισης τερματικών σε μεγάλο πλήθος είναι οι υπηρεσίες αυθεντικοποίησης. Πρόκειται για υπηρεσίες όπου ο κάθε χρήστης διαθέτει μια ταυτότητα ή αλλιώς λογαριασμό χρήστη στον διακομιστή την οποία μπορεί να

χρησιμοποιήσει για να συνδεθεί σε κάθε τερματικό. Η ταυτότητα αυτή περιέχει όλες τις πληροφορίες για το ιστορικό χρήσης και απαιτήσεις του χρήστη αυτής.

Το πρωτόκολλο που προσφέρει αυτή την υπηρεσία σε περιβάλλον Linux, είναι το LDAP. Το πρωτόκολλο βρίσκεται ήδη σε λειτουργία στα συγκεκριμένα τερματικά, υπάρχει όμως η ανάγκη τροποποίησης του λόγο αλλαγών στον εξοπλισμό διαχείρισης. Για την παραμετροποίηση της υπηρεσίας αυτής απαιτείται η εισαγωγή των κατάλληλων παραμέτρων στο αρχείο ldap.conf. Το αρχείο αυτό κατά την εγκατάσταση τοποθετείται στην τοποθεσία /etc/ldap/ [13].

Στο ρόλο εφαρμογής του LDAP υπάρχει και η διεργασία ή αλλιώς daemon με την ονομασία NSLCD. Η βασική του λειτουργία είναι να κάνει ερωτήματα στον διακομιστή που έχει οριστεί [14]. Το NSLCD όμως διαθέτει και αυτό αρχείο κατάλληλο για την διαμόρφωση του. Το αρχείο αυτό ονομάζεται ανάλογα nslcd.conf και βρίσκεται στον φάκελο /etc/ .

Η τρίτη λειτουργία που θα πρέπει να εκτελεστεί αφορά την χρήση του Getent. Πρόκειται για ένα πρόγραμμα ανάκτησης στοιχείων λογαριασμών χρηστών [15].

Συνολικά θα πρέπει να εισαχθούν οι απαραίτητες πληροφορίες για την παραμετροποίηση του LDAP, NSLCD στα αρχεία διαμόρφωσης τους και να εκτελεστεί μια εντολή για την ανάκτηση την λίστας που θα περιέχει τους λογαριασμούς που έχουν καταγραφεί σε κάθε σύστημα. Στην περίπτωση λειτουργίας όμως σε Linux μια παραμετροποίηση για να λειτουργήσει απαιτείται η επανεκκίνηση των συγκεκριμένων υπηρεσιών στο σύστημα. Έτσι μετά την αλλαγή των αρχείων /etc/ldap/ldap.conf και /etc/nslcd.conf θα πρέπει να εκτελεστούν οι απαραίτητες εντολές αυτόματα στο τερματικό για την επανεκκίνηση των υπηρεσιών και ύστερα να γίνει η καταγραφή της λίστας χρηστών.

2.2.2 VIM, Update

Μία ακόμα συχνή απαίτηση σε ένα τέτοιο περιβάλλον όπως το εργαστήριο του Τμήματος Πληροφορικής και Τηλεματικής, είναι η διαθεσιμότητα νέων υπηρεσιών. Μια υπηρεσία που δεν υπήρχε πριν διαθέσιμη από την εγκατάσταση του λειτουργικού συστήματος, είναι ο γνωστός κειμενογράφος VIM [16]. Ένα από τα πλεονεκτήματα των λειτουργικών συστημάτων που είναι βασισμένα σε Debian διανομή, είναι η ύπαρξη της υπηρεσίας Aptitude [17]. Μέσω αυτής ο διαχειριστής του συστήματος μπορεί να έχει άμεσα πρόσβαση σε μεγάλος πλήθος λογισμικού. Υπηρεσία πολύ γνωστή σε χρήστες εξοικειωμένους σε αυτού του είδους διανομές. Για παράδειγμα η χρήση της μέσω της γραμμής εντολών για την εγκατάσταση του VIM είναι :

```
apt-get install vim
```

Η υπηρεσία αναγνωρίζει την αίτηση και το περιεχόμενο αυτής ενώ προσφέρει αυτόματα και την εγκατάσταση των ενδεχομένως επιπλέον απαιτούμενων προγραμμάτων για την λειτουργία του. Μπορεί να διαχειριστεί όχι μόνο αιτήσεις εγκατάστασης αλλά και απεγκατάστασης και αυτόματης ενημέρωσης και αφαίρεσης πεπαλαιωμένου λογισμικού.

Σαν τελευταίος στόχος έχει οριστεί και η ενημέρωση όλου του λειτουργικού συστήματος. Ένα πλήρως ενημερωμένο σύστημα παρέχει και καλύτερη ασφάλεια αλλά και αξιοπιστία. Το Aptitude μπορεί να προσφέρει και αυτή την δυνατότητα, καθώς διαθέτει αυτόματη ενημέρωση στην τελευταία έκδοση κάθε λογισμικού που περιλαμβάνει το λειτουργικό σύστημα. Ένα παράδειγμα χρήσης του για την συγκεκριμένη λειτουργία θα ήταν :

```
1. apt-get update
2. apt-get upgrade
```

Κατά την πρώτη εντολή ενημερώνει τους τοπικούς του καταλόγους με όλες τις διαθέσιμες εκδόσεις για όλα τα προγράμματα που διαθέτει και κατά την δεύτερη εκτελεί την λήψη και εγκατάσταση αυτών.

2.3 Puppet

Ακολουθεί μια εισαγωγή στο Puppet, των εργαλείων και λειτουργιών του. Κάπου εδώ αξίζει να σημειωθούν μερικές φράσεις που χρησιμοποιεί η εταιρία στον επίσημο ιστότοπο της για να περιγράψει τις δυνατότητες του λογισμικού που προσφέρει [3].

- *"Το πιο σύντομο μονοπάτι για καλύτερο λογισμικό" (Ελεύθερη μετάφραση)*
- *"Αν έχει IP τότε μπορεί να διαχειριστεί" (Ελεύθερη μετάφραση)*

  Δύο φράσεις που υπόσχονται πολλά για τις δυνατότητες αυτού του λογισμικού.

2.3.1 Γενικά

Το Puppet όπως έχει προαναφερθεί και στο πρώτο κεφάλαιο αποτελεί ένα εργαλείο αυτοματοποίησης λειτουργιών παραμετροποίησης σε λειτουργικά συστήματα. Ένα λογισμικό με πολλές εφαρμογές και πολύ έρευνα στο ελεύθερο λογισμικό που τρέχει [18].

Στον ιστότοπο της εταιρίας θα βρούμε δύο εκδόσεις για το λογισμικό αυτό. Η πρώτη έκδοση είναι το Puppet Enterprise και πρόκειται για μία έκδοση λογισμικού επί πληρωμή. Διατίθεται δωρεάν για 10 κόμβους ή αλλιώς για δέκα μηχανήματα και έπειτα απαιτεί την ανάλογη χρέωση. Μια μικρή επιχείρηση σε περίπτωση εφαρμογής του Puppet ίσως να προτιμούσε αυτή την έκδοση αφού περιέχει πολλά εργαλεία έτοιμα προς εφαρμογή που προσφέρουν ευκολία στην χρήση, παρακολούθηση και συλλογή στατιστικών στοιχείων. Όλα αυτά σε αντίθεση με την δεύτερη έκδοση που αποτελεί και το "κορμό" του Puppet και ονομάζεται Open Puppet. Πρόκειται για το ανοιχτό, ελεύθερο, και δωρεάν προς χρήση λογισμικό. Προφανώς και όλα τα επιπλέον εργαλεία τα οποία περιλαμβάνει η Enterprise

έκδοση θα μπορούσανε να αναπτυχθούν παρομοίως και στην δωρεάν έκδοση μετά από ανάλογες ενέργειες. Αν δεν υπάρχουν ήδη στο αποθετήριο του Open Puppet σε μια προγενέστερη μορφή τους.

Οι βασικές διαφορές μεταξύ της επί πληρωμής έκδοσης και της ελεύθερης είναι ότι η εμπορική έκδοση υποστηρίζει επιπλέον τα παρακάτω [19].

	Ανοιχτή έκδοση Puppet	**Εμπορική έκδοση Puppet**
Πληροφόρηση		
Διαδραστική απεικόνιση		ΝΑΙ
Επιθεώρηση γεγονότων		ΝΑΙ
Πληροφόρηση Puppet Server		ΝΑΙ
Αυτοματισμός διαμόρφωσης		
Γλώσσα Puppet	ΝΑΙ	ΝΑΙ
Puppet Server	ΝΑΙ	ΝΑΙ
Βάση δεδομένων Puppet	ΝΑΙ	ΝΑΙ
Ενιαίος agent	ΝΑΙ	ΝΑΙ
Διεπαφή μέσω ιστοσελίδας		ΝΑΙ
Εταιρικές πλατφόρμες		ΝΑΙ
Εγκατάσταση με καθοδήγηση		ΝΑΙ
Υποστηριζόμενες ενότητες		ΝΑΙ
Ολοκληρωμένες εταιρικές λύσεις		ΝΑΙ
Ενορχήστρωση		
Επιπρόσθετα εργαλεία γλώσσας προγραμματισμού	ΝΑΙ	ΝΑΙ
Ενορχηστρωμένο περιβάλλον διεπαφής για προγραμματισμό		ΝΑΙ
Εργαλεία γραμμής εντολών		ΝΑΙ
Άμεσος και πλήρης έλεγχος		ΝΑΙ
Δυνατότητα βοήθειας και προβολής σε πραγματικό		ΝΑΙ

χρόνο		
Βελτιωμένη νοημοσύνη		ΝΑΙ
Αυτοματοποιημένη τροφοδότηση		
Υποστίρηξη Cloud	ΝΑΙ	ΝΑΙ
Αποθήκες	ΝΑΙ	ΝΑΙ
Εικονικές μηχανές		ΝΑΙ
Δυνατότητα υποστηριζόμενης αναβάθμισης		ΝΑΙ
Διαχείριση κώδικα		
Εργαλεία κοινότητας για την διαχείριση κόμβων	ΝΑΙ	ΝΑΙ
Ενοποίηση με το Git	ΝΑΙ	ΝΑΙ
Υποστήριξη ροής εργασίας		ΝΑΙ
Περιβάλλον διεπαφής με γραμμή εντολών		ΝΑΙ
Συγχρονισμός αρχείων μεταξύ πολλαπλών εξυπηρετητών		ΝΑΙ
Διαχείριση κόμβων		
Κατηγοριοποίηση κόμβων βάση ρόλου		ΝΑΙ
Κατηγοριοποίηση κόμβων μέσω διεπαφής ιστοσελίδας		ΝΑΙ
Φιλτράρισμα αποθεμάτων		ΝΑΙ
Έλεγχος πρόσβασης βάση ρόλου		
Έλεγχος πρόσβασης βάση ρόλου		ΝΑΙ
Έλεγχος ταυτότητας και ανάκλησης		ΝΑΙ
Ενοποίηση με υπηρεσίες ενεργού καταλόγου		ΝΑΙ
Υποστήριξη επιχειρήσεων		
Υποστηριξη κοινότητας	ΝΑΙ	ΝΑΙ
Εταιρική υποστηριξη		ΝΑΙ
Επαγγελματικές υπηρεσίες		ΝΑΙ
Εκπαίδευση εικονική και με φυσική παρουσία		ΝΑΙ

Πίνακας 1 : Σύγκριση εκδόσεων Puppet

Διαθέτει πολλές εκδόσεις για διάφορα λειτουργικά συστήματα όπως Windows, Apple OSX και Linux (Debian, Ubuntu, Fedora, Red Hat Enterprise and more) για το Open Puppet, ενώ η εμπορική έκδοση διαθέτει ακόμα περισσότερες. Το Puppet είναι βασισμένο στην αρχιτεκτονική πελάτη - διακομιστή και στις δύο εκδόσεις του. Αξιοσημείωτο ακόμα είναι ότι σε περίπτωση εφαρμογής του σε πολύ μεγάλη κλίμακα μπορούν να οριστούν και βοηθητικοί διακομιστές υπό του κεντρικού για αυξημένες απαιτήσεις. Η έκδοση λογισμικού για τον διακομιστή ονομαζόταν Puppet Master παλαιότερα και πλέον Puppet Server. Η έκδοση για την πλευρά του πελάτη ονομάζεται Puppet Agent. Στην περίπτωση εφαρμογής του με βοηθητικούς servers τότε ο κάθε server ενημερώνει τους agents που του έχουν οριστεί και ο βοηθητικός server δέχεται παραμετροποίηση και ενημέρωση από τον κεντρικό Puppet Server. Το Puppet αποτελεί εργαλείο προτίμησης για πολλούς παρόχους υπηρεσιών Cloud. Τα περισσότερα από τα προ εγκατεστημένα λειτουργικά συστήματα και εφαρμογές που προσφέρουν οι πάροχοι αυτοί χειρίζονται από εργαλεία σαν το Puppet.

2.3.2 Πηγές λογισμικού

Τα λειτουργικά συστήματα που έχουν επιλεχθεί παρέχουν την δυνατότητα εγκατάστασης του Puppet μέσω του Aptitude. Σε ποια έκδοση όμως και με τι υποστήριξη;

Στον επίσημο ιστότοπο του, το Puppet διατίθεται μέσω αποθετηρίου ανάλογου του τύπου του λειτουργικού συστήματος που θα εφαρμοστεί. Έτσι δημιουργείται ένα δίλημμα για το ποια οδός πρέπει να επιλεχθεί για την εγκατάσταση του Puppet. Στα επίσημα αποθετήρια λογισμικού του Ubuntu, το Puppet είναι διαθέσιμο σε παλαιότερες εκδόσεις και άρα λιγότερο ενημερωμένο. Αυτό μπορεί να το αντιληφθεί κανείς με μια πρώτη ματιά στην μορφή που υπάρχει διαθέσιμο, ως Puppet Master που όπως έχει προαναφέρει αποτελεί παλαιότερη έκδοση του Puppet. Στην λειτουργία ενός τέτοιου συστήματος αυτοματοποίησης όλη η διεργασία και η παραμετροποίηση πραγματοποιείται στην πλευρά του διακομιστή, σε

αντίθεση με την πλευρά του πελάτη, που δεν έχει πολλές απαιτήσεις και απλά δέχεται ενημερώσεις από τον διακομιστή. Αυτό όμως δεν σημαίνει ότι όταν υπάρχει μεγάλη διαφορά στις εκδόσεις δεν θα υπάρχουν επιπλοκές. Έτσι λοιπόν θεωρείται ορθή η χρήση του Puppet από τις τελευταίες και πιο ενημερωμένες βιβλιοθήκες του ίδιου του λογισμικού και όχι από του Ubuntu. Για να το επιτύχουμε αυτό θα πρέπει να ενημερωθούν οι ανάλογες βιβλιοθήκες στο λειτουργικό σύστημα έτσι ώστε να μπορούν να επικοινωνήσουν με αυτές του Puppet. Οι εν λόγω βιβλιοθήκες είναι προσβάσιμες μέσω διαδικτύου [20]. Παρατηρείται ότι τα αρχεία εγκατάστασης έχουν κωδική ονομασία ανάλογη της έκδοσης Debian Linux η οποία χρησιμοποιείται για παράδειγμα trusty, willy ή xenial. Άρα θα πρέπει να εγκατασταθεί και η ανάλογη έκδοση. Πρόκειται για αρχεία που περιέχουν πληροφορίες σχετικά με την τοποθεσία του ανάλογου αποθετηρίου και όχι για αρχεία που εγκαθιστούν το ίδιο το λογισμικό.

2.3.2 Ruby

Ο ρόλος και οι δυνατότητες του Puppet έχουν αναφερθεί ήδη, αλλά όλα αυτά είναι δυνατά μέσω του σωστού προγραμματισμού του. Για τον σκοπό αυτό το Puppet βασίζεται στην γλώσσα προγραμματισμού Ruby. Πρόκειται για μία δυναμική ανοιχτού κώδικα γλώσσας προγραμματισμού, που εστιάζει στην απλότητα και ευκολία χρήσης καθώς και στην παραγωγικότητα. Συγκριτικά με τις υπόλοιπες γλώσσες προγραμματισμού διαθέτει έναν πιο κομψό τρόπο σύνταξης κάτι που την κάνει περισσότερο ευανάγνωστη και εύκολη στην ανάπτυξη κώδικα [21].

Εικόνα 9 : Λογότυπο Ruby

Στην περίπτωση εφαρμογής του στο Puppet χρησιμοποιείται στην σύνταξη των αρχείων που προορίζονται για τις επιθυμητές εργασίες. Τα αρχεία αυτά είναι τύπου κειμένου και πρέπει έχουν κατάληξη .pp για να τα αναγνωρίσει το Puppet ως αρχεία προς εκτέλεση. Το καλύτερο μέρος για την κατανόηση του τρόπου σύνταξης των αρχείων αυτών είναι τα επίσημα αποδεικτικά έγγραφα από τον επίσημο ιστότοπο του Puppet [22]. Υπάρχουν βέβαια και πολλά άλλα σημεία στο διαδίκτυο και μάλιστα σε ορισμένα έτοιμες και παραγωγικές λύσεις. Το Puppet μεταφράζει τις λειτουργίες προς εκτέλεση ανάλογα με το σύστημα στο οποίο είναι εγκατεστημένο στην μεριά του πελάτη. Ο τρόπος αίτησης μιας συγκεκριμένη λειτουργίας διαφέρει κατά πολύ από ότι μιας κατευθείαν πάνω στο λειτουργικό σύστημα για παράδειγμα μέσω της γραμμής εντολών. Αυτό δεν σημαίνει όμως ότι δεν παρέχει και την δυνατότητα εκτέλεσης εντολών με τον ίδιο τρόπο μέσα από αυτό. Για την καλύτερη κατανόηση των λειτουργιών αυτών, ακολουθεί ένα παράδειγμα με περίπτωση χρήσης την εγκατάσταση του SSH.

Τυπικά ένας διαχειριστής θα μπορούσε να συνδεθεί με τον λογαριασμό του στο τερματικό το οποίο θέλει να εφαρμόσει τις αλλαγές και να εκτελέσει την εντολή :

```
sudo apt-get install ssh
```

Με την εντολή αυτή χρησιμοποιεί τα δικαιώματα λογαριασμού διαχειριστή (sudo) στο τερματικό για να τρέξει το πρόγραμμα Aptitude (apt-get) και να εγκαταστήσει (install) το λογισμικό στην λίστα (ssh).

Εναλλακτικά και χρησιμοποιώντας το Puppet θα μπορούσε να συντάξει ένα αρχείο εκτέλεσης κώδικα στο τερματικό με δύο διαφορετικούς τρόπους.

1. Θα μπορούσε να χρησιμοποιήσει την δυνατότητα του για αναγνώριση της λειτουργίας γράφοντας :

```
package { 'ssh':
                ensure => 'installed',
}
```

Ορίζοντας τον τύπο λειτουργίας που έχει να κάνει με ένα πακέτο λογισμικού (package) και ύστερα εισάγοντας την ονομασία του (ssh), ζητά την εξασφάλιση της εγκατάστασης του (ensure => 'installed').

2. Θα μπορούσε να χρησιμοποιήσει την δυνατότητα κλήσης μία εντολής από το ίδιο το τερματικό γράφοντας :

```
Exec { path => [ "/bin/", "/sbin/" , "/usr/bin/", "/usr/sbin/" ] }
exec { "Install SSH":
        command => "/usr/bin/apt-get -y install ssh",
}
```

Ορίζοντας τον τύπο λειτουργίας που έχει να κάνει με την εκτέλεση μίας εντολής (Exec) και τα μονοπάτια φακέλων στα οποία μπορεί να βρίσκεται αυτή (path => ["/bin/", "/sbin/" , "/usr/bin/", "/usr/sbin/"]), ορίζεται η εντολή ακριβώς και εκτελείται (command => "/usr/bin/apt-get -y install ssh").

2.3.3 Manifests and Modules

Τα αρχεία αυτά στα οποία εμπεριέχεται ο κώδικας γραμμένος σε Ruby που θα χρησιμοποιήσει το Puppet για να εφαρμόσει τις παραμετροποιήσεις ονομάζονται Manifests. Είναι τα ίδια αρχεία που αναφέρθηκαν στην προηγούμενη ενότητα με κατάληξη .pp . Σε μία καινούργια εγκατάσταση του, το Puppet ψάχνει σε ένα συγκεκριμένο φάκελο με το ακόλουθο μονοπάτι :

/etc/puppetlabs/code/enviroments/production/manifests

για αρχεία τύπου .pp και εκτελεί όλα τα αρχεία με αυτή την κατάληξη ακόμα και αν αυτά είναι περισσότερα από ένα. Στην περίπτωση ύπαρξης πολλών αρχείων δημιουργεί μια λίστα και ταξινομεί τα αρχεία με αλφαβητική σειρά ανάλογα με την ονομασία τους και τα εκτελεί κατά αυτήν την σειρά. Για παράδειγμα αν στην τοποθεσία αυτή υπήρχαν τα αρχεία με ονομασίες site.pp, hello.pp, update.pp και reboot.pp το Puppet θα τα εκτελούσε με την εξής σειρά :

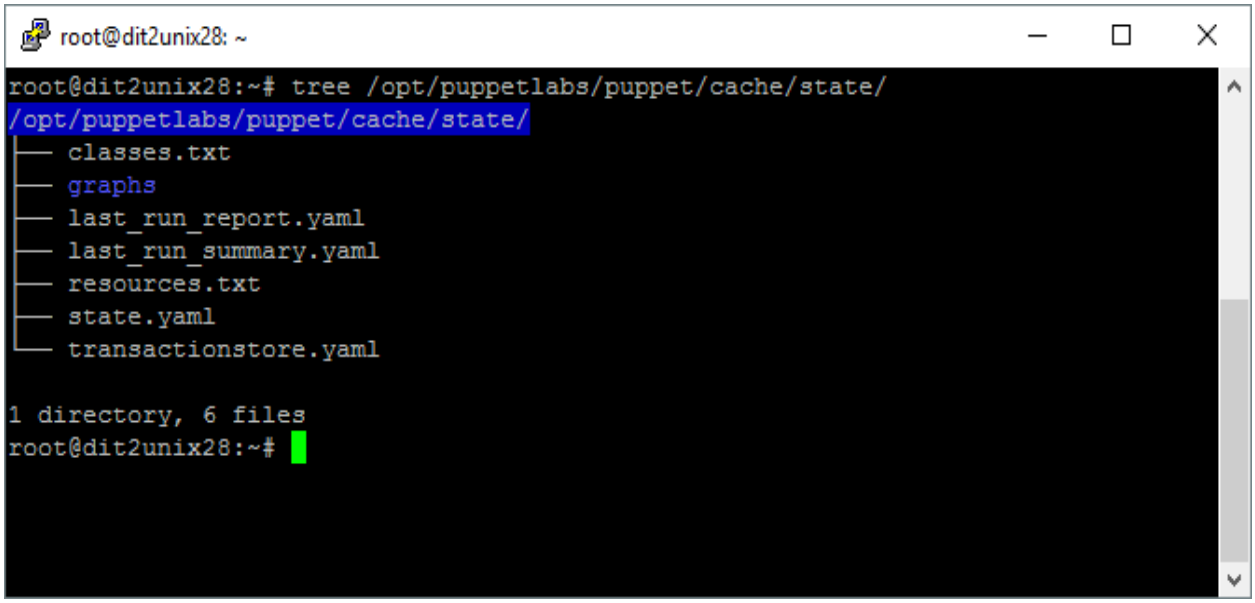
1. hello.pp
2. reboot.pp
3. site.pp
4. update.pp

Το Puppet δεν περιορίζεται όμως σε αυτήν την απλή λειτουργία. Διαθέτει modules ή αλλιώς και ενότητες. Μία ενότητα μπορεί να διαθέτει τα δικά της μανιφέστα προς εκτέλεση, ενώ μπορεί και να περιέχει και αρχεία που ενδεχομένως να μπορεί να μεταφέρει μεταξύ του πελάτη και του διακομιστή. Μπορεί ακόμα να περιέχει και διάφορα πρότυπα που να χρησιμοποιούνται αναλόγως. Ένα πρότυπο έχει την δυνατότητα να περιέχει και τις δικές του βιβλιοθήκες, τα δικά του δοκιμαστικά εργαλεία και εάν είναι αναγκαίο και τις δικές του προδιαγραφές. Μέσα από το Puppet παρέχεται και η δυνατότητα δημιουργίας προσωπικών ενοτήτων και η ανάρτηση τους στην δημόσια βιβλιοθήκη του λογισμικού που φέρει το όνομα Forge. Φυσικά δεν θα μπορούσε να μην υπάρχει και η δυνατότητα να μην δημοσιευτεί μια ενότητα και να διασφαλιστεί το απόρρητο σε αυτή. Αναλόγως το Forge παρέχει την δυνατότητα άντλησης έτοιμων προς χρήση ενοτήτων που έχουν αναπτυχθεί από άλλους χρήστες.

2.3.4 Εφαρμογή και έλεγχος

Οι τρόποι εφαρμογής του ποικίλουν ανάλογα με τις ανάγκες τόσο για τον εξυπηρετητή όσο και για τον πελάτη. Στην περίπτωση του εξυπηρετητή δίνεται η επιλογή εφαρμογής των τροποποιήσεων και σε όλους του συνδεδεμένους κόμβους αλλά και ξεχωριστά στον κάθε έναν. Υπάρχει ακόμα και η δυνατότητα εφαρμογής διαφορετικών ενοτήτων στον κάθε κόμβο ταυτόχρονα. Στην περίπτωση του πελάτη η εφαρμογή των ρυθμίσεων μπορεί να γίνει είτε αυτόματα μέσω μια προγραμματισμένης διεργασίας με χρονοδιάγραμμα και ημιαυτόματα με την εκτέλεση μίας εφαρμογής. Κάθε μέθοδος εφαρμογής έχει τα θετικά και τα αρνητικά της ανάλογα με την περίπτωση. Η ανοιχτή προς χρήση έκδοση του Puppet όπως έχει προαναφερθεί έχει αρκετές διαφορές με την επί πληρωμή και μία από αυτές τις διαφορές είναι και ο έλεγχος λειτουργίας. Αυτό όμως δεν σημαίνει ότι οι πληροφορίες σχετικά με την εφαρμογή δεν είναι διαθέσιμες. Στην αρχική του μορφή το Puppet περιλαμβάνει ιστορικό για κάθε εφαρμογή καταλόγου από αυτό και στην πλευρά του πελάτη αλλά και του εξυπηρετητή. Τα λεγόμενα αρχεία logs περιέχουν πολλές πληροφορίες για τις εφαρμογές καταλόγων τροποποιήσεων. Για τον Puppet Agent βρίσκονται στον φάκελο :

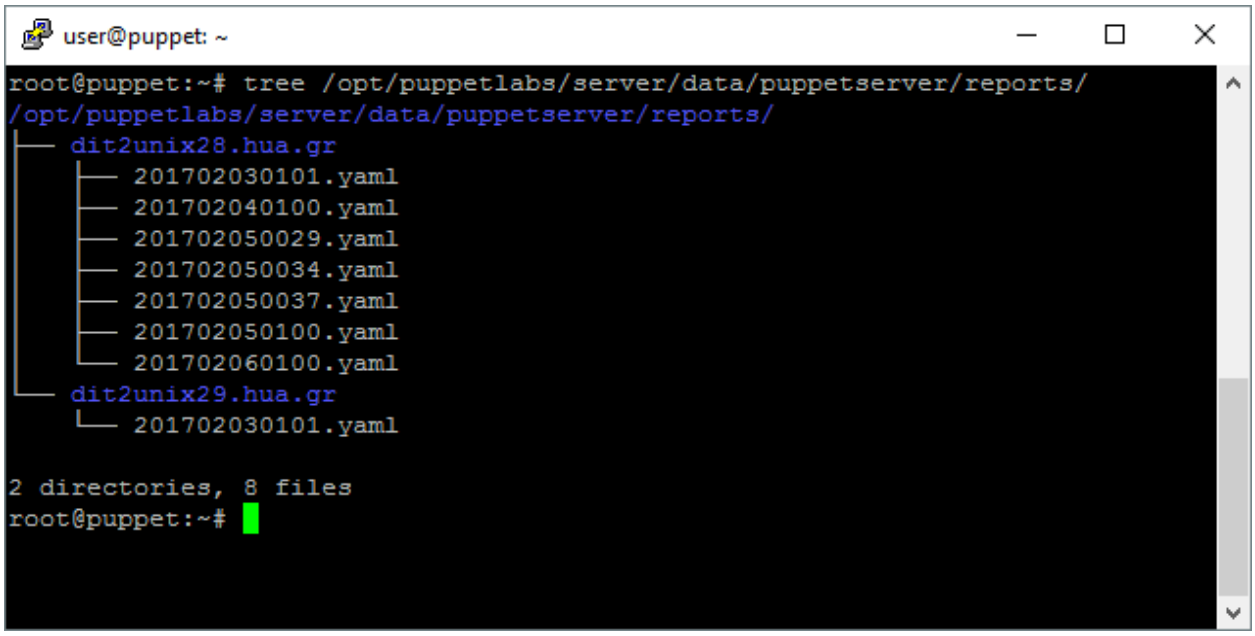
/opt/puppetlabs/puppet/cache/state/



Εικόνα 10 : Log αρχεία του Puppet Agent

και περιέχουν όλες τις απαραίτητες πληροφορίες σχετικά με την τελευταία φορά ενημέρωσης. Στην πλευρά του Puppet Server όμως τα πράγματα είναι εντελώς διαφορετικά αφού περιέχονται πληροφορίες για την εφαρμογή κάθε καταλόγου και κάθε κόμβου. Αυτές οι πληροφορίες βρίσκονται στο μονοπάτι φακέλου :

/opt/puppetlabs/server/data/puppetserver/reports/



Εικόνα 11 : Log αρχεία του Puppet Server

και μέσα σε αυτό υπάρχουν φάκελοι με το όνομα υπολογιστή του κάθε Puppet Agent. Μέσα στο κάθε φάκελο του κάθε κόμβου υπάρχουν ξεχωριστά αρχεία πληροφοριών για κάθε προσπάθεια εφαρμογής καταλόγου σε αυτόν. Αυτό σημαίνει ότι κρατείται πλήρες ιστορικό και για επιτυχείς και ανεπιτυχείς προσπάθειες εφαρμογής. Το όνομα των αρχείων δεν είναι τυχαίο αλλά περιλαμβάνει κατά σειρά μια σειρά αριθμών που υποδηλώνει το έτος, μήνα, ημέρα και ώρα προσπάθειας εφαρμογής.

2.4 Απαιτήσεις συστημάτων

Για την περάτωση της πρακτικής εφαρμογής είναι αναγκαίες και οι ανάλογες προϋποθέσεις που πρέπει να τηρούν τα συστήματα. Τα τερματικά συστήματα του εργαστηρίου του ιδρύματος διαθέτουν όπως έχει προαναφερθεί λειτουργικό σύστημα Xubuntu στην έκδοση xenial. Για τις ανάγκες του διακομιστή παραχωρήθηκε από το ίδρυμα ένα εικονικό μηχάνημα με λειτουργικό σύστημα Debian σε έκδοση Jessie. Πέραν της διανομής και έκδοσης του λειτουργικού συστήματος όμως ο διακομιστής πρέπει να διαθέτει και τους ανάλογους πόρους. Ύστερα από έρευνα πάνω στο documentation του Puppet, παρατηρήθηκε ότι ο διακομιστής (Debian VM Machine) λειτουργεί ομαλά με το ελάχιστο 2 πυρήνες επεξεργαστική ισχύ και 1 Giga Byte μνήμη [23]. Για τις ανάγκες της εργασίας όμως ο διακομιστής δημιουργήθηκε με δυνατότητες επεξεργαστικής ισχύος 2 πυρήνων και 4 Giga Byte μνήμη.

ΚΕΦ.3: Πρακτική Εφαρμογή

Το κεφάλαιο περιέχει την πρακτική εφαρμογή του Puppet στο περιβάλλον του εργαστηρίου. Ξεκινά με την διαδικασία απομακρυσμένης διαχείρισης και συνεχίζει με την εγκατάσταση των απαραίτητων υπηρεσιών σε πελάτη και εξυπηρετητή. Ακολουθεί η ανάλυση του τρόπου προγραμματισμού του Puppet Server καθώς και η εφαρμογή των παραμετροποιήσεων στους κόμβους. Κλείνοντας αναφέρονται οι τρόποι ελέγχου των αποτελεσμάτων.

3.1 Προετοιμασία υπηρεσιών

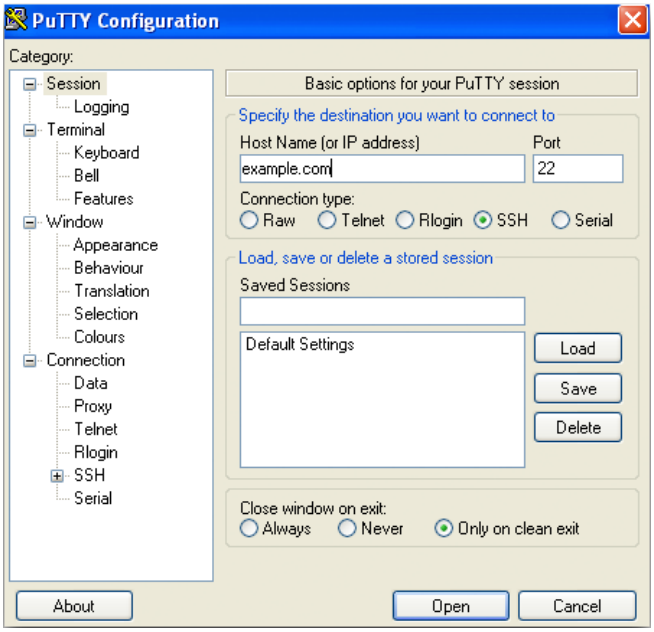
Για την έναρξη διεξαγωγής των εργασιών παραμετροποίησης υπάρχουν κάποιες βασικές προϋποθέσεις. Μια από αυτές είναι η ικανότητα απομακρυσμένης διαχείρισης, καθώς η φυσική παρουσία μέσα στο δίκτυο του πανεπιστημίου δεν είναι πάντοτε εφικτή. Ένα ακόμα σημαντικό στοιχείο είναι η διασφάλιση της συνεχούς διαθεσιμότητας των μηχανημάτων και φυσικά η εγκατάσταση του απαραίτητου λογισμικού σε αυτά.

3.1.1 VPN και SSH

Χρησιμοποιώντας τον οδηγό για την υπηρεσία του VPN [24] που διαθέτει το πανεπιστήμιο για όλους του σπουδαστές του και ήδη κατέχοντας έναν λογαριασμό χρήσης η σύνδεση είναι απλή. Με την χρήση αυτών των υπηρεσιών, ο υπολογιστής για την διαχείρισή

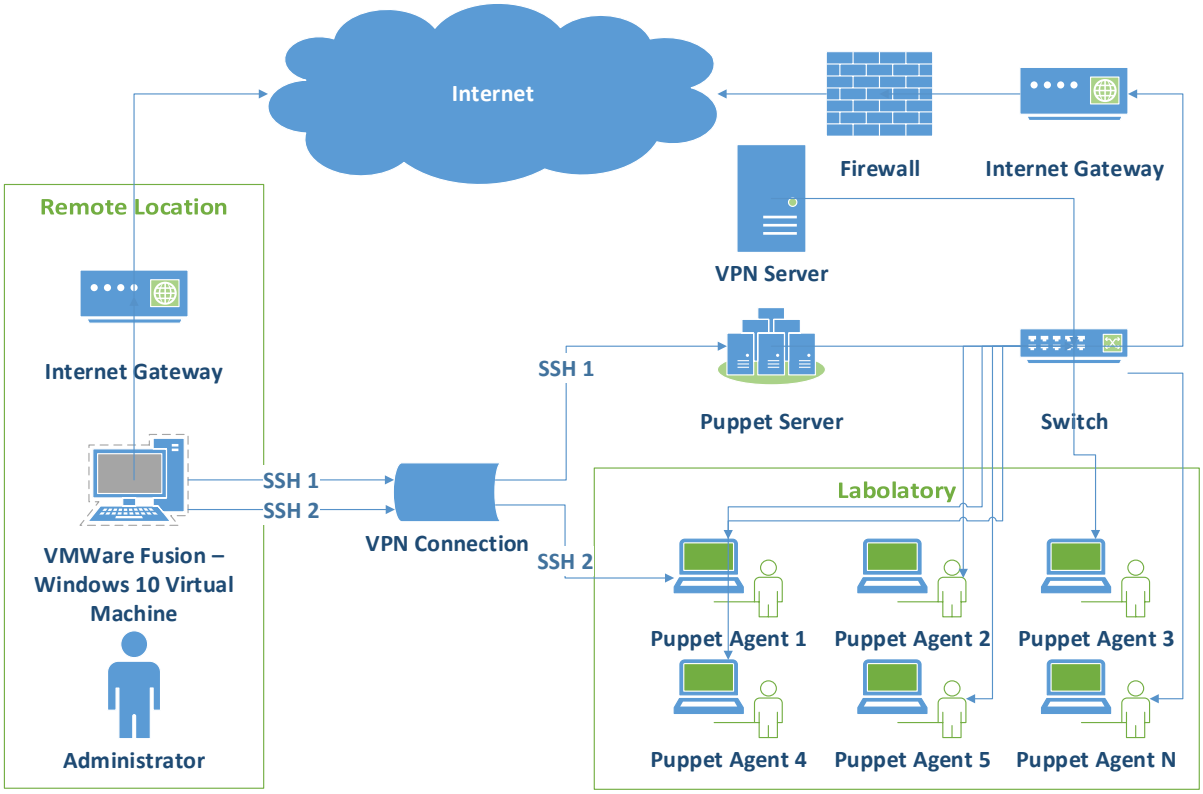
και την παραμετροποίηση μπορεί εύκολα να αποτελέσει ένα κομμάτι του δικτύου του πανεπιστημίου [25].

Απαραίτητη προϋπόθεση είναι όμως και οι υπολογιστές που έχουν τον ρόλο του πελάτη και του εξυπηρετητή να έχουν εγκατεστημένη και εν λειτουργία την δυνατότητα απομακρυσμένης διαχείρισης. Άρα οι υπολογιστές αυτοί πρέπει να διαθέτουν την υπηρεσία SSH. Για τις ανάγκες αυτές είναι υπεύθυνη η ομάδα διαχειριστών των υπολογιστών και υπηρεσιών του πανεπιστημίου, η οποία και τις παρείχε. Το SSH αποτελεί ένα πρωτόκολλο που υποστηρίζεται από όλα τα λειτουργικά συστήματα. Για τις ανάγκες σύνδεσης και διαχείρισης χρησιμοποιήθηκε μια εικονική μηχανή με λειτουργικό σύστημα Windows 10. Ένα πολύ δημοφιλές λογισμικό για σύνδεση σε υπηρεσίες SSH αποτελεί το Putty . Έτσι χρησιμοποιώντας την υπηρεσία VPN η σύνδεση μπορεί να πραγματοποιηθεί εύκολα απλά εισάγοντας στο Putty την τοπική και προσωπική IP διεύθυνση των μηχανημάτων μέσα στο δίκτυο. Για την σύνδεση σε αυτό το πρωτόκολλο χρησιμοποιήθηκε η βασική πόρτα λειτουργίας του (22).



Εικόνα 12 : Διαμόρφωση σύνδεσης στο Putty

Οι συνδέσεις για την διεξαγωγή των εργασιών είναι δύο συνολικά. Η πρώτη στον εξυπηρετητή με λειτουργικό σύστημα Debian Linux και η δεύτερη στον πελάτη με Xubuntu Linux.



Σχήμα 1 : Διάγραμμα δικτύου για σύνδεση και διαχείριση

Για την εφαρμογή όμως των απαραίτητων τροποποιήσεων, όπως εγκατάσταση και παραμετροποίηση βασικών λειτουργιών του λειτουργικού συστήματος απαιτείται ένας λογαριασμός διαχειριστή για το εκάστοτε σύστημα. Για την περίπτωση όμως χρήσης αυτού του λογαριασμού θα ήταν απαραίτητη η συνεχής χρήση της παραμέτρου *sudo* σε κάθε εντολή που χρησιμοποιείται Με την παράμετρο αυτή μπορεί κάποιος εξουσιοδοτημένος από το σύστημα χρήστης να εκτελέσει μια εντολή με δικαιώματα διαχειριστή. Για ένα λειτουργικό σύστημα Linux δεν υπάρχει κάτι πιο κατάλληλο από τον κεντρικό λογαριασμό διαχειριστή που διαθέτει σχεδόν τα ίδια δικαιώματα με αυτά του συστήματος, τον λεγόμενο root user. Για την σύνδεση και χρήση του συγκεκριμένου λογαριασμού είναι συνήθως απαραίτητη φυσικά η γνώση του κωδικού και η εντολή *su* . Στις Debian διανομές όμως αυτό δεν είναι απαραίτητο καθώς με την χρήση της εντολής *sudo -s* μπορεί ένας λογαριασμός με τα κατάλληλα δικαιώματα να συνδεθεί ως root. Για τις ανάγκες περάτωσης αυτής της εργασίας όλες οι εντολές εκτελέστηκαν από τον λογαριασμό και το περιβάλλον του χρήστη root. Η συγκεκριμένη ενέργεια είναι αναγκαία και για λόγους διατήρησης κοινού σημείου αναφοράς

στην υπηρεσία. Επιπλέον με αυτό τον τρόπο αποφεύγονται και άλλα ενδεχομένως δυσάρεστα μελλοντικά προβλήματα που μπορούν να προκύψουν όπως από την διαγραφή του συγκεκριμένου χρήστη ή παρεμβολή στις ρυθμίσεις από κάποια άλλη υπηρεσία.

3.1.2 Repositories

Για κάθε εγκατάσταση ενός νέου λογισμικού σε ένα λειτουργικό σύστημα, είναι απαραίτητα και τα ανάλογα αρχεία. Στην εργασία αυτή όμως, όπως έχει προαναφερθεί η πηγή αυτών των αρχείων θα είναι από τον επίσημο ιστότοπο της ίδιας της εταιρίας Puppet, για την διασφάλιση της τελευταίας έκδοσης [20].

Εικόνα 13 : Αποθετήριο λογισμικού Puppet

Για να ενημερωθεί όμως το λειτουργικό σύστημα για τις διευθύνσεις αυτές απαιτούνται και οι κατάλληλες ενέργειες για την εισαγωγή τους σε αυτό. Η ενημέρωση των repositories για την διάθεση νέου λογισμικού σε Debian διανομές, μπορεί να γίνει είτε

χειροκίνητα, είτε με εγκατάσταση του ανάλογου και συμβατού πακέτου ενημέρωσης. Στον αναφερόμενο ιστότοπο τα διαθέσιμα αρχεία έχουν μορφή ονομασίας “puppetlabs-release-pc1-xenial.deb”. Η παράμετρος pc1 είναι συντομογραφία του Puppet Collection 1 και μεταφράζεται σε Συλλογή 1 για το Puppet. Η τελευταία παράμετρος σε κάθε όνομα αυτών των αρχείων αφορά την διανομή για την οποία προορίζεται και η χρήση του. Για την προβολή της διανομής του λειτουργικού συστήματος στα διαθέσιμα μηχανήματα είναι αρκετή η εκτέλεση της εντολής :

```
lsb_release -a
```

Τα αποτελέσματα αυτής της εντολής ορίζουν ότι η διανομή των τερματικών του εργαστηρίου είναι xenial και του εξυπηρετητή είναι Jessie. Άρα και τα αρχεία που πρέπει να χρησιμοποιηθούν είναι “puppetlabs-release-pc1-xenial.deb” και “puppetlabs-release-pc1-jessie.deb” αντίστοιχα. Το επόμενο βήμα είναι η επιτυχής λήψη του ανάλογου αρχείου στα μηχανήματα. Για την ανάγκη αυτή χρησιμοποιήθηκε η εντολή wget προσθέτοντας την απαραίτητη πληροφορία σε αυτή, που δεν είναι καμία άλλη από την πλήρη διεύθυνση του αρχείου στο διαδίκτυο.

Για τα τερματικά του εργαστηρίου :

```
wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb
```

Για τον εξυπηρετητή :

```
wget https://apt.puppetlabs.com/puppetlabs-release-pc1-jessie.deb
```

Η λειτουργία ενημερώνει για την πρόοδο λήψης του δηλωθέντος αρχείου και αν δεν οριστεί καμία ακόμα παράμετρος, αποθηκεύει το αρχείο στον φάκελο από του οποίου την

θέση εκτελέστηκε η εντολή. Στην παρούσα περίπτωση και εφόσον έχει πραγματοποιηθεί μόνο σύνδεση με τον χρήστη root, το πλήρες μονοπάτι του φακέλου αυτού είναι /root/ .

Μετά την ολοκλήρωση της λήψης, ακολουθεί η εγκατάσταση του πακέτου. Για τον σκοπό αυτό χρησιμοποιήθηκε η εντολή dpkg, προσθέτοντας τις κατάλληλες παραμέτρους για εγκατάσταση και δήλωση του συγκεκριμένου αρχείου.

Για τα τερματικά του εργαστηρίου :

```
dpkg -i puppetlabs-release-pc1-xenial.deb
```

Για τον εξυπηρετητή :

```
dpkg -i puppetlabs-release-pc1-jessie.deb
```

Η δήλωση του πλήρους μονοπατιού τοποθεσίας του αρχείου δεν είναι αναγκαία αφού το αρχείο βρίσκεται στον ίδιο φάκελο από όπου έχει εκτελεστεί η εντολή. Σε αυτό το σημείο έχουν εισαχθεί οι κατάλληλες εγγραφές έτσι ώστε να είναι διαθέσιμο το εν λόγο αποθετήριο. Για να τεθεί όμως σε ισχύ είναι απαραίτητη και η ενημέρωση στο λογισμικό Aptitude. Αυτό επιτυγχάνεται εκτελώντας την ακόλουθη εντολή :

```
apt-get update
```

Σε αυτή την κατάσταση τα σύστημα είναι έτοιμα για την εγκατάσταση του λογισμικού Puppet.

3.1.3 Εγκατάσταση Puppet

Για την εγκατάσταση του Puppet στα τερματικά μέσα στο εργαστήριο θα πρέπει να χρησιμοποιηθεί το λογισμικό Puppet-Agent. Πρόκειται για το κατάλληλο λογισμικό για λειτουργία σε ρόλο πελάτη. Το λογισμικό υπάρχει πλέον διαθέσιμο μέσα από τα νέα αποθετήρια, στην μεγαλύτερη δυνατή έκδοσή του σύμφωνα με την τρέχουσα διανομή και την υποστήριξη σε αυτή. Για την εκτέλεση μιας ολοκληρωμένης αυτόματης εγκατάστασης όλα τα απαραίτητα επιπλέον λογισμικά, πρέπει να χρησιμοποιηθεί το λογισμικό Aptitude. Η κατάλληλη εντολή για την εγκατάσταση του Puppet-Agent είναι :

```
apt-get install puppet-agent
```

Η εντολή αναλύεται στην κλήση της υπηρεσίας Aptitude με την παράμετρο install που υποδηλώνει την λειτουργία εγκατάστασης και φυσικά την δήλωση του συγκεκριμένου λογισμικού με την ακριβή ονομασία.

Στην πλευρά του εξυπηρετητή όμως η διαδικασία εγκατάστασης έχει κάποιες μικρές διαφορές. Η εντολή εγκατάστασης του λογισμικού διαφέρει μόνο στο όνομα. Η ακριβής ονομασία του πακέτου λογισμικού για τον εξυπηρετητή είναι Puppet Server. Η εντολή εγκατάστασης είναι :

```
apt-get install puppetserver
```

Αυτό όμως δεν είναι αρκετό αφού πρέπει να διασφαλιστεί η συνεχής λειτουργία και διαθεσιμότητα των υπηρεσιών του εξυπηρετητή. Θα πρέπει λοιπόν η ανάλογη διεργασία να είναι ενεργοποιημένη και να τρέχει. Η διεργασία έχει το ίδιο όνομα με την υπηρεσία και για την διαχείριση της είναι απαραίτητες οι ακόλουθες εντολές :

1. *puppet resource service puppetserver enable=true*
2. *puppet resource service puppetserver ensure=running*

Στη ουσία με τις εντολές αυτές ενημερώνεται το λειτουργικό σύστημα για την προτεινόμενη κατάσταση μιας υπηρεσίας (resource service puppetserver), με την χρήση των κατάλληλων παραμέτρων. Πρώτον την ενεργοποίηση με την παράμετρο enable=true και δεύτερον την εξασφάλιση της λειτουργίας με την παράμετρο ensure=running. Η εγκατάσταση του λογισμικού με τον συγκεκριμένο τρόπο δεν περιλαμβάνει την προσθήκη της τοποθεσίας αυτού, για την αναγνώριση και της εντολής του. Αυτό στην προκειμένη περίπτωση είναι αρκετά βολικό και κυρίως για λόγους επιπρόσθετης ασφαλείας και ελέγχου. Για τον λόγο αυτό η εκτέλεση οποιασδήποτε εντολής από τον Puppet Server ή Puppet Agent θα πρέπει να γίνεται με την δήλωση της τοποθεσίας του λογισμικού. Έτσι για την περίπτωση του agent η εντολή εκτελείτε από /opt/puppetlabs/bin/puppet και του server από /opt/puppetlabs/bin/puppetserver . Για λόγους αισθητικής όμως είναι προτιμώμενη η παρουσίαση των εντολών χωρίς το πλήρες μονοπάτι. Σε αυτό το σημείο θα πρέπει να επισημανθεί ότι παρότι και τα δύο λογισμικά έχουν εγκατασταθεί, αυτό δεν σημαίνει ότι επικοινωνούν κιόλας.

3.2 Σύνδεση υπηρεσιών

Η διαθεσιμότητα του λογισμικού από μόνη της δεν είναι αρκετή, καθώς για να λειτουργήσει πρέπει να υπάρχει μια επιτυχής σύνδεση μεταξύ του εξυπηρετητή και των κόμβων προς διαχείριση. Για την σύνδεση αυτή θα πρέπει να υπάρχει αμοιβαία εμπιστοσύνη και αναγνώριση μεταξύ όλων των άκρων. Για να επιτευχθεί όμως αυτή η επικοινωνία στο επίπεδο των εφαρμογών θα πρέπει πρώτα να υπάρχει μια επιτυχής σύνδεση στο επίπεδο του

δικτύου. Μία ακόμα παράμετρος που πρέπει να ληφθεί υπόψιν είναι και η ασφάλεια στην επικοινωνία.

3.2.1 Επίπεδο Δικτύου

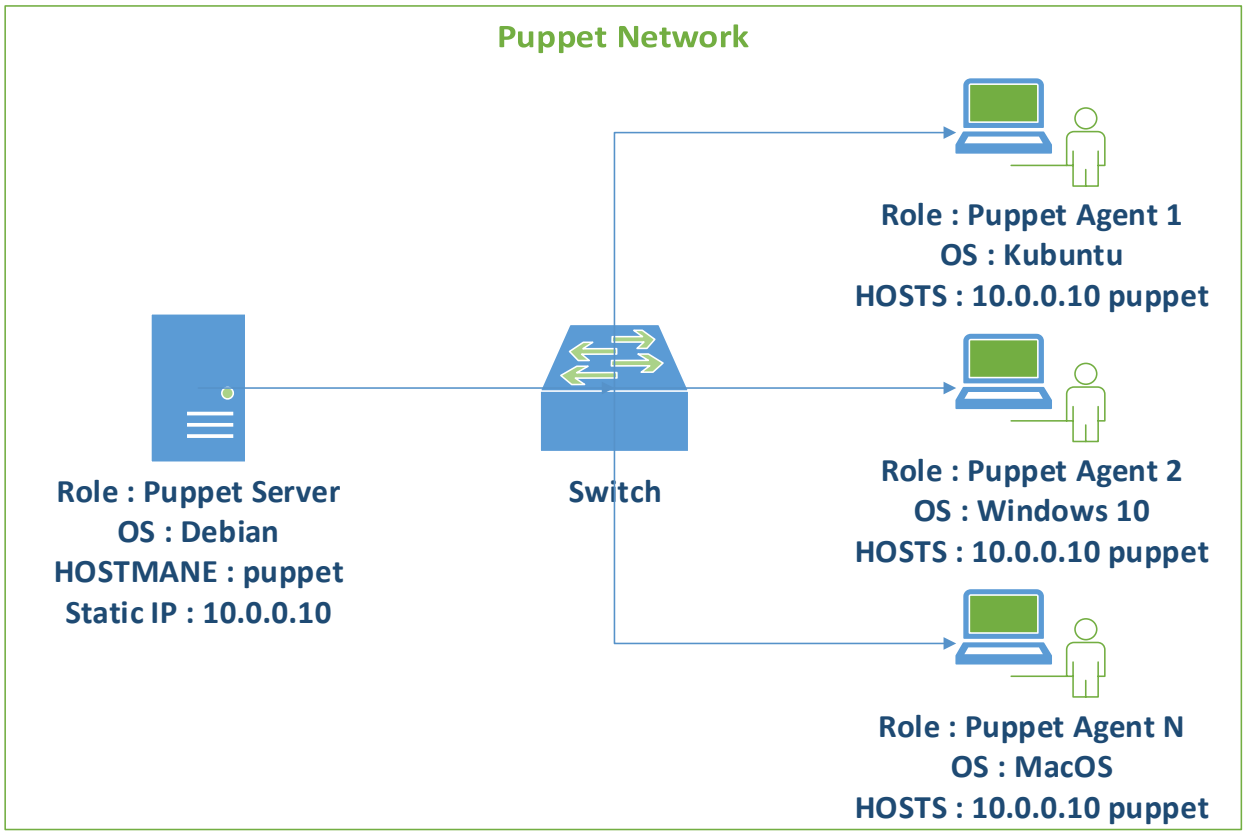
Στην αρχική κατάσταση λειτουργίας του Puppet έπειτα από μία νέα εγκατάσταση, τα αρχεία ρυθμίσεων είναι ενημερωμένα έτσι ώστε ο υπολογιστής σε ρόλο εξυπηρετητή να έχει την ονομασία “puppet”. Αυτή η ρύθμιση είναι φυσικά διαθέσιμη προς παραμετροποίηση μέσω του αρχείου puppet.conf . Μια ρύθμιση ιδιαίτερα σημαντική καθώς το όνομα υπολογιστή μπορεί να διαφέρει σε πολλές περιπτώσεις εφαρμογής. Στην παρούσα κατάσταση όμως και για λόγους διευκόλυνσης της εγκατάστασης του λογισμικού στους επιθυμητούς κόμβους κρίνεται ποιο ορθή η αλλαγή του ονόματος στον ίδιο τον διακομιστή. Για την αλλαγή αυτή θα πρέπει να πραγματοποιηθεί επεξεργασία του αρχείου hostname στην τοποθεσία /etc στο εικονικό μηχάνημα του εξυπηρετητή. Ακολουθεί η ακριβής διαδικασία :

1. Εκτέλεση της εντολής *vi /etc/hostname*
2. Αλλαγή του ονόματος σε puppet
3. Αποθήκευση των αλλαγών
4. Επανεκκίνηση μηχανήματος

Για να μπορεί να παρέχει όμως το επίπεδο δικτύου τις απαραίτητες πληροφορίες στο επίπεδο εφαρμογής θα πρέπει να υπάρχει και ένας συσχετισμός μεταξύ της διεύθυνσης IP και του ονόματος του εξυπηρετητή. Αυτή η ανάγκη θα μπορούσε να καλυφθεί από μία υπηρεσία DNS. Σε περίπτωση απουσίας όμως της υπηρεσίας αυτής, θα μπορούσε εναλλακτικά να ενημερωθεί η κατάλληλη παράμετρος στο λειτουργικό σύστημα των κόμβων. Για τον συσχετισμό μίας διεύθυνσης δικτύου με το όνομα του εξυπηρετητή θα πρέπει να ενημερωθεί

το αρχείο hosts στην τοποθεσία /etc στο κάθε τερματικό. Η διαδικασία για την εισαγωγή αυτής της παραμέτρου έχει ως εξής :

1. Εκτέλεση της εντολής *vi /etc/hosts*
2. Εισαγωγή μια νέας σειράς κειμένου με περιεχόμενο πρώτα την διεύθυνση IP π.χ. (192.168.1.144) του εξυπηρετητή, κενό και το όνομα του εξυπηρετητή (puppet)
3. Αποθήκευση των αλλαγών



Σχήμα 2 : Διάγραμμα δικτύου για σωστή επικοινωνία με τα κατάλληλα ονόματα μηχανών

Για την επιβεβαίωση της επιτυχούς ενημέρωσης, κρίνεται απαραίτητη η εκτέλεση της εντολής ping στο όνομα puppet από την γραμμή εντολών του agent.

```
ping puppet
```

Στην περίπτωση που το σύστημα δεν έχει ενημερωθεί σωστά δεν θα υπάρξει απάντηση από την διεύθυνση αυτή.

3.2.2 Επίπεδο εφαρμογών

Εφόσον έχει διασφαλιστεί η δικτυακή επικοινωνία μεταξύ των δύο άκρων το μόνο που απομένει είναι η ένωση των λογισμικών πελάτη και εξυπηρετητή. Το λογισμικό του Puppet Server αναγνωρίζει και αλληλοεπιδρά μόνο με κόμβους τους οποίους εμπιστεύεται. Πρόκειται για μία διαδικασία αυθεντικοποίησης με πιστοποιητικά. Ο agent αναρτά μία αίτηση για σύνδεση στον server και ο server με την σειρά του θα πρέπει να αποδεχτεί την αίτηση αυτή και να προσθέσει τον agent στην λίστα των ενεργών κόμβων με τους οποίους συνεργάζεται. Η διαδικασία έχει ως εξής :

1. Ο Puppet Agent αιτείται σύνδεση με τον εξυπηρετητή με όνομα puppet στο δίκτυο

```
puppet agent -tv
```

2. Ο Puppet Server δημιουργεί μία λίστα αιτημάτων για σύνδεση, η οποία είναι ορατή με την εντολή

```
puppet cert list
```

3. Έχοντας αναγνωρίσει το όνομα των κόμβων (hostname) από την λίστα ο διαχειριστής μπορεί να αποδεχτεί χειροκίνητα την σύνδεση με τους και να ολοκληρώσει την διαδικασία.

```
puppet cert sign “hostname_of_the_agent”
```

Σε περίπτωση ανάγκης εισαγωγής πολλών κόμβων εναλλακτικά μπορεί να εκτελεστεί υπογραφή των πιστοποιητικών μαζικά με την ανάλογη εντολή.

```
puppet cert sign -a
```

Η διαδικασία αυτή βασίζεται πάνω στο πρότυπο ανταλλαγής κλειδιού *Diffie-Hellman*. Πρόκειται για μία πολύ γνωστή και παλαιά μέθοδο ασφαλούς ανταλλαγής κλειδιού μέσα σε ένα δημόσιο δίκτυο [26]. Τον τελευταίο χρόνο όμως έρευνες έχουν δείξει ότι μέχρι και αυτή η μέθοδος που θεωρείτο ασφαλή εδώ και πολλά χρόνια έχει ελαττώματα. Η επίθεση που μπορεί να δημιουργήσει προβλήματα σε αυτή την διαδικασία ανταλλαγής κλειδιού ονομάζεται Logjam Attack [27]. Το Puppet αντιμετωπίζει αυτήν την αδυναμία εφαρμόζοντας ένα κλειδί με μέγεθος 2048 bit στην βασική του λειτουργία.

3.3 Κώδικας εφαρμογής

Σε αυτό το σημείο το σύστημα είναι έτοιμο να εκτελέσει λειτουργίες παραμετροποίησης στους κόμβους. Το μόνο που απομένει είναι ο καθορισμός και προγραμματισμός των λειτουργιών αυτών μέσα στον Puppet Server. Όταν οι λειτουργίες όμως είναι πολλές, όπως σε αυτήν την περίπτωση εφαρμογής, ενδείκνυται ένα πρότυπο ιεραρχίας και ελέγχου. Τις απαιτήσεις αυτές καλύπτει η δυνατότητα δημιουργίας module από το Puppet. Έτσι για την κάθε μία λειτουργία εφαρμογής παραμετροποιήσεων, δημιουργήθηκε ξεχωριστό module με την ανάλογη κλάση. Η διαχείριση και κλήση των κλάσεων ορίζεται μέσα από το κεντρικό μανιφέστο.

3.3.1 Κεντρικό σημείο αναφοράς

Το κεντρικό μανιφέστο όπως έχει αναφερθεί και στα προηγούμενα κεφάλαια, έχει ονομασία site.pp και βρίσκεται μέσα στο μονοπάτι φακέλου :

/etc/puppetlabs/code/environments/production/manifests/site.pp

Είναι το πρώτο σημείο που το Puppet κοιτά για αλλαγές και εφαρμογές νέων ενημερώσεων στην βασική του λειτουργία. Σε αυτό το manifest θα πρέπει να οριστούν και τα κατάλληλα module προς εφαρμογή.

Οι λειτουργίες που πρέπει να εφαρμοστούν χωρίζονται σε πέντε διαφορετικά modules. Για την παραμετροποίηση των λειτουργιών της υπηρεσίας LDAP θα δημιουργηθεί το module με όνομα ldap. Για την παραμετροποίηση των λειτουργιών της υπηρεσίας NSLCD θα δημιουργηθεί το module με όνομα nslcd. Για την εκτέλεση της εντολής Getent θα δημιουργηθεί το module με όνομα getent. Για την εγκατάσταση του λογισμικού VIM θα δημιουργηθεί το module με όνομα vim. Το τελευταίο module θα έχει σκοπό την εκτέλεση εντολών για την ενημέρωση λογισμικού και το module που θα δημιουργηθεί θα έχει ονομασία update.

Η κλήση των module μέσα από το κεντρικό manifest γίνεται με την παράμετρο include και το ακριβές όνομα του module. Για την δημιουργία του site.pp και χρησιμοποιώντας τον κειμενογράφο εκτελείται η εντολή :

```
vi /etc/puppetlabs/code/environments/production/manifests/site.pp
```

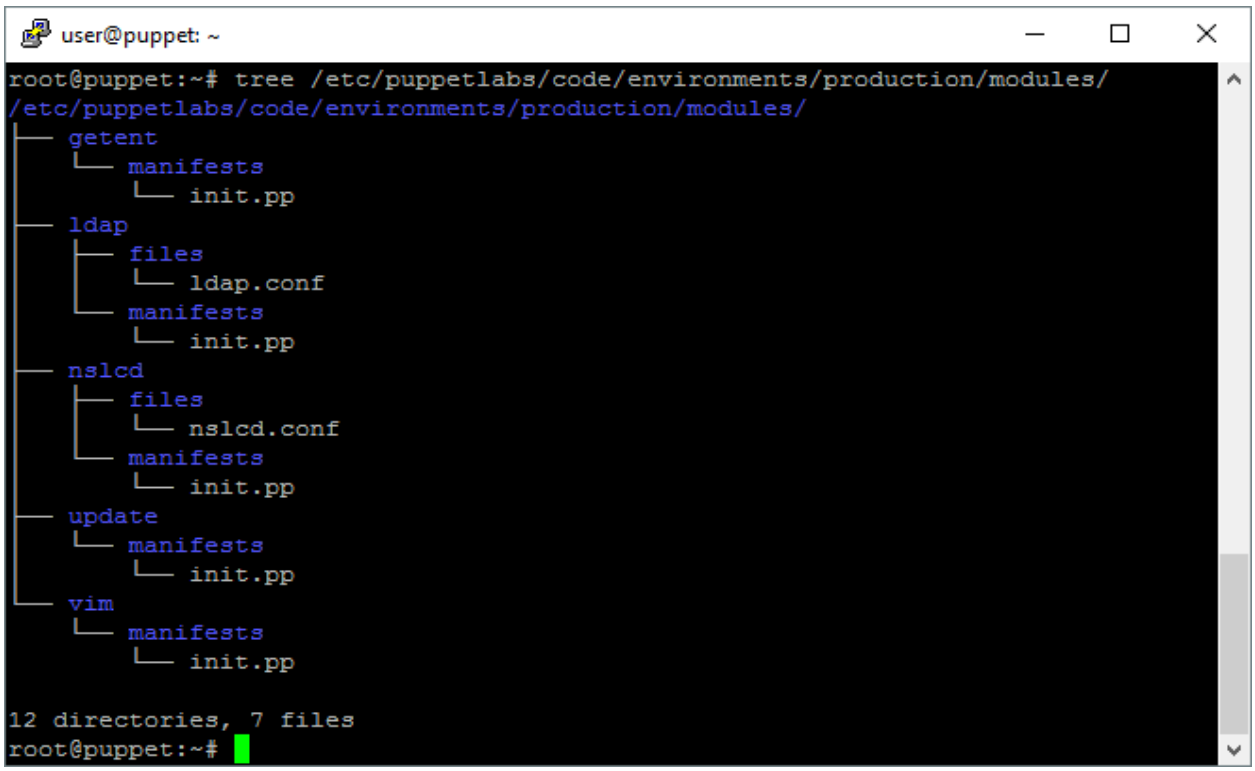
Τα περιεχόμενα του αρχείου θα πρέπει να είναι σύμφωνα με τα μέχρι τώρα δεδομένα τα ακόλουθα.

```
include ldap
include nslcd
include getent
include vim
include update
```

Ύστερα από την αποθήκευση του κεντρικού manifest, το επόμενο βήμα είναι η δημιουργία των module.

3.3.2 Χρήση Modules

Η δημιουργία ενός module μπορεί να γίνει με δύο τρόπους, είτε αυτόματα μέσα από το Puppet είτε χειροκίνητα. Για την συγκεκριμένη εργασία τα module δημιουργήθηκαν χειροκίνητα και όσον αφορά τους φακέλους και την δόμηση αλλά και τα αρχεία. Τα module μπορούν να περιέχουν μεγάλος πλήθος πληροφοριών ανάλογα και με την πολυπλοκότητα τους. Σε γενικές γραμμές και για τις ανάγκες αυτής της εργασίας όλα τα module πρέπει περιέχουν το manifest με ονομασία init.pp, μέσα στο οποίο ορίζονται και οι λειτουργίες τους. Εξαίρεση αποτελούν μόνο τα module ldap και nslcd, στα οποία θα πρέπει να υπάρχουν και τα αρχεία ldap.conf και nslcd.conf . Αυτό συμβαίνει διότι είναι πολύ ποιο απλή και εύχρηστη η αντιγραφή ενός αρχείου, σε σχέση με την παραμετροποίηση ενός ήδη υπάρχοντος. Τα module θα πρέπει να δημιουργηθούν σε μία συγκεκριμένη τοποθεσία και οι υπό φάκελοι αυτών θα πρέπει να έχουν συγκεκριμένη ιεραρχία έτσι ώστε το Puppet να τα αναγνωρίζει. Τα κατάλληλα εργαλεία για την ανάγκη αυτή δεν είναι τίποτα περισσότερο από την εντολή δημιουργίας φακέλων mkdir και τον κειμενογράφο VI. Με τα μέχρι τώρα δεδομένα η δομή των modules θα πρέπει να είναι όπως στην εικόνα που ακολουθεί.



Εικόνα 14 : Δομή φακέλου των modules

Για την προβολή της δομής των module χρησιμοποιήθηκε η εντολή:

```
tree /etc/puppetlabs/code/environments/production/modules/
```

Το όνομα κλήσης της εντολής αυτής είναι tree και ορίζοντας έναν συγκεκριμένο φάκελο, προβάλει όλα τα περιεχόμενα του κατηγοριοποιώντας τα βάση επιπέδου. Όπως παρατηρείται οι υπό φάκελοι με ονομασία files υπάρχουν μόνο μέσα στα module ldap και nslcd. Με την δομή ξεκάθαρη ακολουθεί ο προγραμματισμός των αρχείων και manifest.

3.3.2.1 Module ldap

Η παραμετροποίηση των υπηρεσιών του LDAP απαιτεί διαμόρφωση του αρχείου ldap.conf και επανεκκίνηση της διεργασίας του. Για την διαμόρφωση του αρχείου όμως θα πρέπει να γίνει είτε επεξεργασία κειμένου είτε αντικατάσταση. Βέλτιστη οδό στο συγκεκριμένο θέμα αποτελεί πρώτα η αποθήκευση ενός αντιγράφου ασφαλείας του αρχείου αυτού και ύστερα η αντικατάσταση του.

Για την δημιουργία του module ldap λοιπόν είναι απαραίτητα, ο προγραμματισμός του manifest με ονομασία site.pp και του αρχείου ldap.conf στην κατάλληλη τοποθεσία. Έτσι με την χρήση του κειμενογράφου το αρχείο ldap.conf διαμορφώθηκε ως εξής :

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE dc=hua,dc=gr
URI ldap://10.100.51.119

#SIZELIMIT 12
#TIMELIMIT 15
#DEREF never

# TLS certificates (needed for GnuTLS)
TLS_CACERT /etc/ssl/certs/ca-certificates.crt

# Search timelimit.
# In a local network we can set this to a low value. ~5s is good enough.
timelimit 5

# Bind/connect timelimit
# In a local network we can set this to a low value.
bind_timelimit 1

# Reconnect policy:
#  hard_open: reconnect to DSA with exponential backoff if
#           opening connection failed
#  hard_init: reconnect to DSA with exponential backoff if
#           initializing connection failed
#  hard:      alias for hard_open
#  soft:      return immediately on server failure
bind_policy soft
```

Για τον προγραμματισμό του manifest του συγκεκριμένου module θα πρέπει να οριστεί η ανάλογη κλάση και οι τρεις απαιτούμενες διαδικασίες.

1. Δημιουργία αντιγράφου ασφαλείας του αρχείου ldap.conf στον κόμβο
2. Αντικατάσταση αρχείου με αυτού που περιέχεται μέσα στο module
3. Επανεκκίνηση διεργασίας LDAP

Αρχικά λαμβάνει θέση ο ορισμός της κλάσης με το ανάλογο όνομα ldap (class ldap {}). Μέσα στα άγκιστρα θα πρέπει να οριστούν οι τρεις προαναφερθείσες διαδικασίες. Για την δημιουργία του αντιγράφου ασφαλείας χρησιμοποιείτε η εντολή cp στο λειτουργικό σύστημα του κόμβου. Έτσι θα πρέπει να οριστεί ότι πρόκειται για μία εντολή (exec) να δηλωθεί ένα όνομα ενδεικτικά για την διαδικασία (ldap) και να οριστεί επακριβώς η εντολή (/bin/cp -rp /etc/ldap/ldap.conf /etc/ldap/ldap.conf.backup) προς εκτέλεση. Στην εντολή ορίζεται ότι το αρχείο προς αντιγραφή είναι στο τοπικό μηχάνημα το /etc/ldap/ldap.conf και ότι θα πρέπει να αντιγραφεί στην τοποθεσία /etc/ldap/ με όνομα ldap.conf.backup

Για την διαδικασία αντικατάστασης του αρχείου θα πρέπει να ληφθούν υπόψιν και τα δικαιώματα του αρχείου. Κάτι απολύτως αναγκαίο αφού αν το αρχείο δεν έχει τα απαραίτητα δικαιώματα το λογισμικό που το διαχειρίζεται μπορεί να σταματήσει να έχει πρόσβαση σε αυτό. Για την απόκτηση των απαραίτητων στοιχείων των δικαιωμάτων του αρχείου αρκεί η εντολή ls με την παράμετρο -n στο φάκελο που περιέχει το αρχείο αυτό. Για το αρχείο ldap.conf ο κάτοχος θα πρέπει να είναι ο χρήστης root, θα πρέπει να ανήκει στην ομάδα root και να έχει δικαιώματα πρόσβασης ορισμένα με κωδικό 664. Ο προγραμματισμός της διαδικασίας αυτής ξεκινάει με την δήλωση ότι πρόκειται για διαχείριση αρχείου (file {}) και την δήλωση μίας ενδεικτικής ονομασίας ('/etc/ldap/ldap.conf':). Ακολουθεί ο ορισμός της τοποθεσίας του αρχείου προς αντικατάσταση (path => '/etc/ldap/ldap.conf',) και του αρχείου που θα το αντικαταστήσει (source => 'puppet:///modules/ldap/ldap.conf',). Σε περίπτωση που το αρχείο υπάρχει με μορφή συντόμευσης, το Puppet μπορεί να το αναγνωρίσει και να εκτελέσει την αντικατάσταση στην πραγματική τοποθεσία αυτού (links => 'follow',). Ο ορισμός των δικαιωμάτων γίνεται με την ακριβή δήλωσή αυτών μέσω των παραμέτρων owner, group και mode (owner => 'root', group => 'root', mode => '644',).

Για την διαδικασία επανεκκίνησης της διεργασίας θα πρέπει αρχικά να δηλωθεί ότι πρόκειται για διαχείριση διεργασίας (service {}) με ένα ενδεικτικό όνομα (ldap:). Για την εκτέλεση αρκεί η χρήση της παραμέτρου restart και ο ορισμός αυτής ως αληθή (restart => 'true',).

Το manifest στην τελική του μορφή περιέχει τα ακόλουθα.

```
class ldap {
      exec { «LDAP»:
            command => "/bin/cp -rp /etc/ldap/ldap.conf /etc/ldap/ldap.conf.backup",
      }
      file { '/etc/ldap/ldap.conf':
            path => '/etc/ldap/ldap.conf',
            source => 'puppet:///modules/ldap/ldap.conf',
            links => 'follow',
            owner => 'root',
            group => 'root',
            mode => '644',
      }
      service { ldap :
            restart => 'true',
      }
}
```

3.3.2.2 Module nslcd

Η παραμετροποίηση των υπηρεσιών του NSLCD απαιτεί διαμόρφωση του αρχείου nslcd.conf και επανεκκίνηση της διεργασίας του. Για την δημιουργία του module nslcd είναι απαραίτητα, ο προγραμματισμός του manifest με ονομασία site.pp και του αρχείου nslcd.conf

στην κατάλληλη τοποθεσία. Έτσι με την χρήση του κειμενογράφου το αρχείο nslcd.conf διαμορφώθηκε ως εξής:

```
# /etc/nslcd.conf
# nslcd configuration file. See nslcd.conf(5)
# for details.
# The user and group nslcd should run as.
uid nslcd
gid nslcd

# The location at which the LDAP server(s) should be reachable.
uri ldap://10.100.51.119

# The search base that will be used for all queries.
base dc=hua,dc=gr

# The LDAP protocol version to use.
ldap_version 3

# The DN to bind with for normal lookups.
#binddn cn=annonymous,dc=example,dc=net
#bindpw secret
binddn cn=username,ou=users,ou=dit,dc=hua,dc=gr
bindpw *******

# The DN used for password modifications by root.
#rootpwmoddn cn=admin,dc=example,dc=com
# SSL options
#ssl off
#tls_reqcert never
tls_cacertfile /etc/ssl/certs/ca-certificates.crt
# The search scope.
scope sub

pagesize 1000
referrals off
```

```
#filter passwd (&(objectClass=user)(uidNumber=*))
filter passwd
(&(objectClass=user)(!(objectClass=computer))(uidNumber=*)(unixHomeDirectory=*))
map    passwd uid            sAMAccountName
map    passwd homeDirectory    unixHomeDirectory
map    passwd gecos          displayName
filter shadow
(&(objectClass=user)(!(objectClass=computer))(uidNumber=*)(unixHomeDirectory=*))
map    shadow uid            sAMAccountName
map    shadow shadowLastChange pwdLastSet
filter group  (objectClass=group)
```

Για τον προγραμματισμό του manifest του συγκεκριμένου module θα πρέπει να οριστεί η ανάλογη κλάση και οι τρεις απαιτούμενες διαδικασίες, περίπου με τον ίδιο τρόπο όπως και στο module ldap.

Αρχικά λαμβάνει θέση ο ορισμός της κλάσης με το ανάλογο όνομα nslcd (class nslcd {}). Μέσα στα άγκιστρα θα πρέπει να οριστούν οι τρεις προαναφερθείσες διαδικασίες. Για την δημιουργία του αντιγράφου ασφαλείας χρησιμοποιείτε η εντολή cp στο λειτουργικό σύστημα του κόμβου. Έτσι θα πρέπει να οριστεί ότι πρόκειται για μία εντολή (exec) να δηλωθεί ένα όνομα ενδεικτικά για την διαδικασία (nslcd) και να οριστεί επακριβώς η εντολή (/bin/cp -rp /etc/nslcd.conf /etc/nslcd.conf.backup) προς εκτέλεση. Στην εντολή ορίζεται ότι το αρχείο προς αντιγραφή είναι στο τοπικό μηχάνημα το /etc/nslcd.conf και ότι θα πρέπει να αντιγραφεί στην τοποθεσία /etc/ με όνομα nslcd.conf.backup .

Για την διαδικασία αντικατάστασης του αρχείου παρομοίως θα πρέπει να ληφθούν υπόψιν και τα δικαιώματα του αρχείου. Για το αρχείο nslcd.conf ο κάτοχος θα πρέπει να είναι ο χρήστης root, θα πρέπει να ανήκει στην ομάδα nslcd και να έχει δικαιώματα πρόσβασης ορισμένα με κωδικό 771. Ο προγραμματισμός της διαδικασίας αυτής ξεκινάει με την δήλωση ότι πρόκειται για διαχείριση αρχείου (file {}) και την δήλωση μίας ενδεικτικής ονομασίας ('/etc/nslcd.conf':). Ακολουθεί ο ορισμός της τοποθεσίας του αρχείου προς αντικατάσταση (path => '/etc/nslcd.conf',) και του αρχείου που θα το αντικαταστήσει (source =>

'puppet:///modules/nslcd/nslcd.conf',). Σε περίπτωση που το αρχείο υπάρχει με μορφή συντόμευσης, το Puppet μπορεί να το αναγνωρίσει και να εκτελέσει την αντικατάσταση στην πραγματική τοποθεσία αυτού (links => 'follow',). Ο ορισμός των δικαιωμάτων γίνεται με την ακριβή δήλωσή αυτών μέσω των παραμέτρων owner, group και mode (owner => 'root', group => 'nslcd', mode => '711',).

Για την διαδικασία επανεκκίνησης της διεργασίας θα πρέπει αρχικά να δηλωθεί ότι πρόκειται για διαχείριση διεργασίας (service {}) με ένα ενδεικτικό όνομα (nslcd:). Για την εκτέλεση αρκεί η χρήση της παραμέτρου restart και ο ορισμός αυτής ως αληθή (restart => 'true',).

Το manifest στην τελική του μορφή περιέχει τα ακόλουθα.

```
class nslcd {
	exec { "nslcd":
		command => "/bin/cp -rp /etc/nslcd.conf /etc/nslcd.conf.backup",
	}
	file { '/etc/nslcd.conf':
		path => '/etc/nslcd.conf',
		source => 'puppet:///modules/nslcd/nslcd.conf',
		links => 'follow',
		owner => 'root',
		group => 'nslcd',
		mode => '711',
	}
	service { nslcd :
		restart => 'true',
	}
}
```

3.3.2.3 Module getent

Για την δημιουργία του module getent, είναι αναγκαία η δημιουργία μόνο του manifest. Έτσι θα πρέπει να τροποποιηθεί το αρχείο init.pp μέσα σε αυτό το module, αρχικά ορίζοντας την κλάση με το ακριβές όνομα της (class getent {}). Η getent όμως είναι μια εντολή για λογισμικό στο μηχάνημα του κόμβου και γιαυτό τον τρόπο θα πρέπει να εκτελεστεί ανάλογα. Για την κλήση μίας εντολής προς εφαρμογή θα πρέπει να οριστεί ο τύπος λειτουργίας με την ανάλογη παράμετρο και μία ενδεικτική ονομασία (exec {“GETENT PASSWD” : }). Για την εκτέλεση της εντολής θα πρέπει να δηλωθεί και η ακριβής τοποθεσίας αυτής καθώς και των παραμέτρων κλήσης (command => “/usr/bin/getent passwd”,). Η συγκεκριμένη εντολή όμως κατά την εκτέλεση της εξάγει κάποιες πληροφορίες. Στην περίπτωση αυτή, για να φανούν αλλά και συνάμα να καταγραφούν τα αποτελέσματα αυτά θα πρέπει να χρησιμοποιηθεί και η ανάλογη παράμετρος από το Puppet. Η παράμετρος αυτή ονομάζεται logoutput και για την χρήση της αρκεί να δηλωθεί μία αληθής ή ψευδής κατάσταση για αυτή (logoutput => 'true',). Το manifest για το module getent έχει την εξής τελική μορφή :

```
class getent {
      exec { "GETENT PASSWD":
      command => "/usr/bin/getent passwd",
      logoutput => 'true',
      }
}
```

3.3.2.4 Module vim

Για την εγκατάσταση του λογισμικού VIM θα χρησιμοποιηθεί το module vim, με περιεχόμενο μόνο ένα manifest. Στην περίπτωση που κάποιο τερματικό δεν περιέχει την εφαρμογή, τότε η λειτουργία της εγκατάστασης είναι απλή. Όμως στην περίπτωση που σε κάποιο από αυτά έχει ήδη εγκατασταθεί σε κάποια τυχαία έκδοση, θα μπορούσε να δημιουργηθεί πρόβλημα στην εφαρμογή των παραμετροποιήσεων. Το Puppet μπορεί να ελέγξει την έκδοση και τον τρόπο εφαρμογής έτσι ώστε να καλυφθεί κάθε ενδεχόμενο κατάστασης λογισμικού, με τον ακόλουθο τρόπο εφαρμογής. Αρχικά δηλώνεται η κλάση με το ακριβές όνομα της, το οποίο θα πρέπει να είναι ίδιο με το όνομα του module όπως και στα υπόλοιπα module (class vim {}). Στην συνέχεια θα πρέπει να δηλωθεί ο τύπος χρήσης που σε αυτή την περίπτωση είναι ένα πακέτο λογισμικού, με το ακριβές όνομα του (package { 'vim':}) και να δηλωθεί η προτιμώμενη κατάσταση αυτού (ensure => 'latest',). Με τον τρόπο αυτό σε κάθε περίπτωση κατάστασης του κόμβου, θα διασφαλιστεί από το Puppet η ύπαρξη του πακέτου λογισμικού στην τελευταία έκδοση του. Συνολικά το manifest για το module vim θα πρέπει να περιέχει τα ακόλουθα :

```
class vim {
 package { 'vim':
      ensure => 'latest',
          }
}
```

3.3.2.5 Module update

Η διαδικασία ελέγχου και διαχείρισης ενημέρωσης του λογισμικού σε ένα λειτουργικό σύστημα, μπορεί πολλές φορές να αποτελεί μεγάλη πρόκληση για κάθε διαχειριστή. Με την μέχρι τώρα εφαρμογή του Puppet θα μπορούσε να δημιουργηθεί ένα module ανάλογο με αυτό του module vim, για κάθε πακέτο λογισμικού που ενδιαφέρει τον διαχειριστή για την

διασφάλιση της διαθεσιμότητας του στην ποιο ενημερωμένη έκδοση. Ένα λειτουργικό σύστημα όμως μπορεί να περιέχει πολλά πακέτα λογισμικού. Σε ένα περιβάλλον όπως αυτό του εργαστηρίου του ιδρύματος οι ανάγκες είναι πολλές. Το λειτουργικό σύστημα Xubuntu, διαθέτει ήδη την εφαρμογή Amptitude που έχει την δυνατότητα ελέγχου και ενημέρωσης όλων των πακέτων λογισμικού που βρίσκονται σε χρήση στο κάθε μηχάνημα. Για την λειτουργία αυτή θα πρέπει πρώτα να ενημερωθούν όλα τα αποθετήρια από το διαδίκτυο και ύστερα να εκτελεστεί η λειτουργία της ενημέρωσης.

Έτσι δηλώνοντας ακριβώς την κλάση και το όνομα της στο αρχείο του βασικού manifest για το συγκεκριμένο module (class update {}), θα πρέπει να ακολουθήσουν και οι δύο εντολές για την ενημέρωση. Με την ίδια σειρά θα πρέπει πρώτα να γίνει η ενημέρωση των αποθετηρίων και ύστερα η ενημέρωση των πακέτων λογισμικού. Δηλώνοντας αρχικά τον τύπο και ένα ενδεικτικό όνομα (exec{ “APT UPDATE”: }), ορίζονται οι λειτουργίες εκτέλεσης της συγκεκριμένης εντολής (command => "/usr/bin/apt-get update",) και της καταγραφής των στοιχείων που προβάλει αυτή (logoutput => 'true',). Για την λειτουργία ενημέρωσης των πακέτων λογισμικού ομοίως γίνεται κλήση, δήλωση τύπου και ορισμός των επιθυμητών λειτουργιών (exec { "APT UPGRADE": command => "/usr/bin/apt-get -y upgrade", logoutput => 'true',}}). Η τελική μορφή του manifest θα πρέπει να είναι η ακόλουθη :

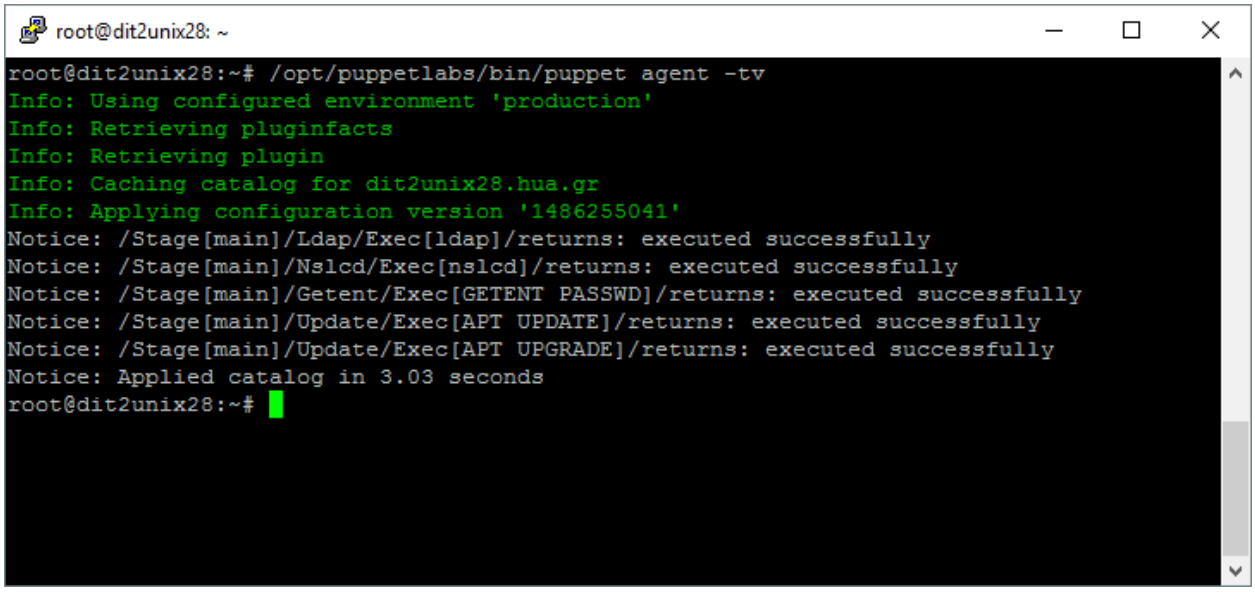
```
class update {
      exec { "APT UPDATE":
            command => "/usr/bin/apt-get update",
            logoutput => 'true',
            }
      exec { "APT UPGRADE":
            command => "/usr/bin/apt-get -y upgrade",
            logoutput => 'true',
            }
}
```

3.4 Εκτέλεση

Σε αυτό το σημείο ο Puppet Server περιέχει όλα τα κατάλληλα στοιχεία για την εφαρμογή των επιθυμητών παραμετροποιήσεων στον agent. Οι λειτουργίες θα εφαρμοστούν με την ίδια σειρά που ακολουθεί η κλήση του κάθε module στο βασικό manifest. Η σειρά εκτέλεσης έχει μεγάλη σημασία για ορισμένες εφαρμογές. Στην περίπτωση αυτής της εργασίας η εκτέλεση του module getent δεν θα είχε νόημα εφαρμογής αν πρωτύτερα δεν είχαν αλλάξει τα δεδομένα για τις υπηρεσίες LDAP και NSLCD. Η προϋπόθεση για την εφαρμογή των λειτουργιών στα τερματικά του εργαστηρίου δεν είναι άλλη από την εκτέλεση της εντολής :

```
puppet agent -tv
```

Στην εντολή μετά την κλήση του απαραίτητου λογισμικού (Puppet) ορίζετε η επιλογή λειτουργίας της (agent) και οι επιθυμητές παράμετροι. Με την παράμετρο -t το λογισμικό εκτελεί πρώτα δοκιμή του κώδικα εφαρμογής με βασικές ρυθμίσεις και εφόσον η δοκιμή είναι επιτυχής και δεν περιέχει σφάλματα τότε εκτελείτε και η πραγματική εφαρμογή του. Η παράμετρος -v χρησιμοποιείτε για την αναπαραγωγή αναφορών κατά την εκτέλεση. Στην εικόνα που ακολουθεί προβάλλονται τα αποτελέσματα εφαρμογής του καταλόγου από την πλευρά του Puppet Agent χωρίς την χρήση της παραμέτρου logoutput.



Εικόνα 15 : Εφαρμογή καταλόγου σε Puppet Agent

Η εκτέλεση της εντολής αυτής για την εφαρμογή των παραμετροποιήσεων μπορεί να γίνει είτε χειροκίνητα είτε αυτόματα στα τερματικά του εργαστηρίου. Η χειροκίνητη εκτέλεση απαιτεί την σύνδεση σε κάθε τερματικό με τον λογαριασμό διαχειριστή. Για την αυτόματη εκτέλεση της όμως θα πρέπει να έχει δημιουργηθεί σε κάθε τερματικό μία προγραμματισμένη διεργασία με χρονοδιάγραμμα. Για την δημιουργία της διεργασίας αυτής δεν υπάρχει πιο κατάλληλο εργαλείο από το Cron. Το χρονοδιάγραμμα θα μπορούσε να οριστεί σε μία ώρα που το εργαστήριο δεν χρησιμοποιείτε, όπως το βράδυ στις 03:00. Με αυτό τον τρόπο κάθε βράδυ τα μηχανήματα του εργαστηρίου θα μπορούσαν να εκτελούν αυτόματα τις διεργασίες που έχουν οριστεί στον Puppet Server. Για την δημιουργία μίας τέτοιας προγραμματισμένης διεργασίας στο Cron θα πρέπει μετά με την εγκατάσταση του Puppet Agent να εκτελεστούν και οι ανάλογες ενέργειες. Ο προγραμματισμός του Cron πολλές φορές μπορεί να μπερδεύει του χρήστες, αλλά με την βοήθεια του Puppet σαν διαθέσιμο λογισμικό στου υπολογιστές του εργαστηρίου μπορεί να απλοποιηθεί αρκετά. Ακολουθεί η εντολή χρήσης του λογισμικού Puppet σε ένα τερματικό του εργαστηρίου για την δημιουργία ενός Cron job για αυτό.

```
puppet resource cron puppet-agent ensure=present user=root hour=3
command='/opt/puppetlabs/bin/puppet agent -tv --splay --splaylimit 60'
```

Το παράμετροι της παραπάνω εντολής θα μπορούσαν και να παρουσιαστούν και με την μορφή ενός manifest. Κάτι που προβάλει και το ίδιο το Puppet κατά την εκτέλεση της.

```
cron { 'puppet-agent':
  ensure  => 'present',
  command => '/opt/puppetlabs/bin/puppet agent -tv --splay --splaylimit 60',
  hour    => ['3'],
  target  => 'root',
  user    => 'root',
}
```

Ουσιαστικά πραγματοποιείται η χρήση της υπηρεσίας Cron για την διασφάλιση της παρουσίας μία προκαθορισμένης διεργασίας. Η εντολή που εκτελείται σε αυτή την διεργασία είναι /opt/puppetlabs/bin/puppet agent -tv --splay --splaylimit 60. Πρόκειται για την χρήση της εφαρμογής Puppet σε μορφή agent με τις παραμέτρους -t και -v για την δοκιμή και προβολή των λειτουργιών που θα εφαρμοστούν. Η παράμετρος --splay και splaylimit 60 χρησιμοποιείται για την αποφυγή υπερφόρτωσης του Pupper Server σε περίπτωση που πολλοί agent ζητήσουν εφαρμογή του καταλόγου του. Για τον ορισμό της διεργασίας θα πρέπει να δηλωθεί και το ποθητό χρονοδιάγραμμα εφαρμογής (hour   => ['3'],) που στην συγκεκριμένη περίπτωση όπως έχει προαναφερθεί θα είναι στις 03:00 κάθε μέρα. Για την χρήση του Cron είναι υποχρεωτικό να καθοριστεί ένας λογαριασμός χρήσης της εντολής αλλά και ιδιοκτησίας (target => 'root', user   => 'root',).

Το σύστημα σε αυτή την κατάσταση είναι έτοιμο για χρήση και εφαρμογή των παραμετροποιήσεων αυτόματα αν και εφόσον τα μηχανήματα των agent είναι ενεργά οποιαδήποτε μέρα στις τρεις η ώρα το πρωί. Σε περίπτωση απουσίας παραμετροποιήσεων δεν θα εκτελεστεί καμία αλλαγή.

3.5 Έλεγχος και καταγραφή

Για την καταγραφή και έλεγχο των διεργασιών το Puppet στην ανοιχτή έκδοση του δεν προσφέρει αρκετές λειτουργίες σε σχέση με την εμπορική έκδοση. Το κομμάτι καταγραφής γεγονότων είναι πολύ σημαντικό σε πολλές περιπτώσεις διαχείρισης λειτουργικών συστημάτων και λογισμικών. Ειδικά σε μία περίπτωση εφαρμογής όπως αυτή της παρούσας εργασίας είναι πολύ σημαντικό να υπάρχει ένα σημείο για τον έλεγχο των αποτελεσμάτων της εφαρμογής του.

Στα τερματικά με ρόλο Puppet Agent η καταγραφή γεγονότων είναι διαθέσιμη μόνο για την τελευταία εφαρμογή καταλόγου. Η τοποθεσία αποθήκευσης αυτών είναι /opt/puppetlabs/puppet/cache/state/ και το αρχείο έχει ονομασία last_run_report.yaml. Το συγκεκριμένο αρχείο περιέχει αρκετές πληροφορίες για την τελευταία εφαρμογή καταλόγου, συμπεριλαμβανομένου και των εξαγόμενων κειμένων πληροφοριών κατά την εκτέλεση. Έτσι για την περίπτωση χρήσης του module getent που περιέχει την παράμετρο logoutput θα πρέπει να έχουν καταγραφεί τα εμφανιζόμενα αποτελέσματα κατά την εκτέλεση της εντολής. Η καταγραμμένη πληροφορία στα αρχεία αυτά μπορεί να είναι δυσανάγνωστη αλλά παρόλα αυτά είναι αρκετά πλήρης.

Στην πλευρά του Puppet Server όμως τα πράγματα είναι πολύ διαφορετικά αφού υπάρχει πλήρες αρχείο καταγραφής γεγονότων εφαρμογής καταλόγων για κάθε Puppet Agent. Η τοποθεσία των αρχείων αυτών είναι /opt/puppetlabs/server/data/puppetserver/reports/ και μέσα σε αυτή υπάρχει ξεχωριστός φάκελος για τον κάθε agent με ονομασία το όνομα υπολογιστή. Σε κάθε φάκελο για τον κάθε agent υπάρχουν πολλά αρχεία, όπου το κάθε ένα αφορά και την εφαρμογή ενός καταλόγου. Τα αρχεία διαχωρίζονται βάση ονομασίας και κάθε μία περιέχει μια σειρά αριθμών που υποδηλώνουν με ακριβή σειρά το έτος, μήνα, ημέρα, ώρα και λεπτό καταγραφής.

ΚΕΦ.4: Συνολικά

Σε αυτό το κεφάλαιο παρουσιάζεται συνολικά η λειτουργία του συστήματος αυτόματης παραμετροποίησης, με τα θετικά και τα αρνητικά σημεία στην όλη διαδικασία. Επισημαίνονται τα σημεία που χρήζουν ιδιαίτερης προσοχής κατά την εφαρμογή του. Τέλος πραγματοποιείτε αναφορά για μελλοντικές εφαρμογές του συστήματος.

4.1 Συμπεράσματα

Μέσα από τις αυξημένες απαιτήσεις για παραμετροποίηση και διαχείριση μεγάλου πλήθους μηχανημάτων γεννήθηκε και η ιδέα δημιουργίας ενός συστήματος που θα αυτοματοποιεί αυτές τις εργασίες. Στον κλάδο των υπηρεσιών DevOps οι εφαρμογές και υπηρεσίες έχουν πολλές δυνατότητες. Από το 2011 έως σήμερα, ένα εργαλείο του χώρου κάνει την διαφορά και κρατάει πολύ ψηλά τον πήχη. Πρόκειται για το λογισμικό που βασίζεται και αυτή η εργασία, το Puppet. Με την δυνατότητα ευελιξίας και διαθεσιμότητας του ελεύθερου λογισμικού τόσο στο ίδιο το λογισμικό αλλά και στο λειτουργικό σύστημα, επιτεύχθηκε η δημιουργία μιας αυτοματοποιημένης εργασίας για την διαχείριση και παραμετροποίηση των υπολογιστών του εργαστηρίου του τμήματος Πληροφορικής και Τηλεματικής.

Για την εγκατάσταση, προετοιμασία και εφαρμογή της υπηρεσίας έπαιξε πολύ μεγάλο ρόλο η δυνατότητα απομακρυσμένης σύνδεσης. Ο απαιτούμενος χρόνος για την φυσική μετάβαση στο κάθε μηχάνημα εκμηδενίστηκε με την βοήθεια των υπηρεσιών SSH και VPN. Η διαδικασία εγκατάστασης του λογισμικού Puppet τόσο στον Server, όσο και στους Client υστερεί πάρα πολύ σε θέματα ευχρηστίας. Η παράλληλη διαθεσιμότητα του λογισμικού τόσο από τα αποθετήρια του κάθε λειτουργικού συστήματος όσο και από τα αποθετήρια της

εταιρίας του λογισμικού αυξάνουν το ενδεχόμενο σφάλματος. Ένα ακόμα σημείο που αφορά την εγκατάσταση του λογισμικού είναι η έλλειψη ενημέρωσης του λειτουργικού συστήματος για την τοποθεσία διαθεσιμότητας του λογισμικού. Είναι πολύ ποιο εύκολη η εκτέλεση ενός λογισμικού απλά με την χρήση του ονόματος του στην γραμμή εντολών από την χρήση του ονόματος μαζί με την τοποθεσία του.

Το σημαντικότερο μειονέκτημα του Puppet στην ελεύθερη έκδοση του είναι όμως η έλλειψη ενός εύχρηστου και πλήρους μηχανισμού ενημερώσεων. Σε σενάρια εφαρμογής του σε μεγάλο πλήθος μηχανημάτων είναι αρκετά δύσκολο κάθε φορά που εφαρμόζεται ένας νέος κατάλογος να πρέπει ο διαχειριστής να ανατρέχει όλα τα τελευταία αρχεία ένα ένα για να ελέγξει για τυχόν σφάλματα σε κόμβους. Ένα στοιχείο όμως του Puppet και στον παρών τρόπο εφαρμογής του είναι ότι όταν εκτελείτε μια ενημέρωση καταλόγου, πρώτα γίνεται έλεγχος ορθότητας εφαρμογής και ύστερα εφαρμόζονται οι αλλαγές. Πέρα από τα αρνητικά του στοιχεία και τις ελλείψεις του όμως, το Puppet μπορεί να αποτελέσει ένα σπουδαίο εργαλείο καθώς με σωστή σκέψη και εφαρμογή του, μπορούν να αντιμετωπιστούν αποτελεσματικά αρκετά από τα μειονεκτήματα του.

Η παροχή των συγκεκριμένων υπηρεσιών αυτόματης παραμετροποίησης, μπορεί εκ πρώτης όψεως να φαίνεται απλή και εύκολη προς εφαρμογή, όμως στην ουσία για την εφαρμογή τους απαιτείται βαθιά γνώση και κατανόηση των λειτουργικών συστημάτων αλλά και των υπηρεσιών. Σε περίπτωση χρήσης χωρίς την πλήρη γνώση των ενεργειών και των αποτελεσμάτων τους σε όλο το λειτουργικό σύστημα τα προβλήματα που μπορούν να προκύψουν είναι σοβαρά και ίσως μη αναστρέψιμα με αυτόματο τρόπο.

Πέρα από τα θετικά, αρνητικά, τις δυσκολίες και τις προϋποθέσεις χρήσης όμως, το τμήμα Πληροφορικής και Τηλεματικής του Χαροκοπείου Πανεπιστημίου πλέον διαθέτει ένα εργαλείο ικανό να αυτοματοποιήσει αρκετές λειτουργίες. Οι αλλαγές στην υποδομή πλέον θα μπορούν να ενημερωθούν ποιο γρήγορα και εύκολα στα λειτουργικά συστήματα. Η διαθεσιμότητα νέου λογισμικού αλλά και η ενημέρωση των ήδη υπαρχόντων θα αποτελεί διαδικασία λιγότερο χρονοβόρα. Τα οφέλη είναι πολλά και μεταξύ αυτών είναι και η ενίσχυση της ασφαλείας στις παρεχόμενες υπηρεσίες. Η συνεχής και άμεση ενημέρωση στο λογισμικό

ενός λειτουργικού συστήματος μειώνει αρκετά τα κενά ασφαλείας. Ένα ακόμα στοιχείο του Puppet που μπορεί να χρησιμοποιηθεί για λόγους ασφαλείας είναι η διασφάλιση συγκεκριμένης κατάστασης λειτουργίας υπηρεσιών.

4.2 Μελλοντική έρευνα

Το Puppet μπορεί να έχει πολλούς τρόπους εφαρμογής και να εξυπηρετεί διάφορους σκοπούς, ανάλογα με τις ανάγκες. Η εφαρμογή του σε αυτή την εργασία αφορούσε μόνο ένα από τα εργαστήρια του πανεπιστημίου, αλλά θα μπορούσε να εφαρμοστεί σε όλα, χρησιμοποιώντας την δυνατότητα λειτουργίας με πολλούς υπό-εξυπηρετητές για την διαχείριση διαφορετικών περιβαλλόντων εφαρμογής. Πιο συγκεκριμένα θα μπορούσε να δημιουργηθεί ένας υπό-εξυπηρετητής για κάθε ομάδα υπολογιστών ή και για κάθε εργαστήριο ξεχωριστά. Ο κάθε υπό-εξυπηρετητής θα είναι υπεύθυνος για την εξυπηρέτηση της ομάδας που του έχει ανατεθεί για διαχείριση, ενώ ταυτόχρονα όλοι οι υπό-εξυπηρετητές θα διαχειρίζονται από τον κεντρικό Puppet Server. Με αυτόν τον τρόπο εφαρμογής θα είναι δυνατή η διαχείριση όλων των υπολογιστών του Χαροκοπείου Πανεπιστημίου από έναν εξυπηρετητή.

Ένα ακόμα σενάριο εφαρμογής του Puppet που θα μπορούσε να αποτελέσει θέμα μελλοντικής μελέτης θα ήταν στην δημιουργία ενός συστήματος αποτροπής κακόβουλων επιθέσεων. Θα μπορούσαν να εντοπιστούν όλα τα κρίσιμα σημεία του δικτύου και των υπηρεσιών του ιδρύματος σε περίπτωση κακόβουλης επίθεσης και να οριστεί η προτιμώμενη κατάσταση λειτουργίας τους. Επιπλέον να δημιουργηθούν με το Puppet τα κατάλληλα module για την διασφάλιση της ορθής κατάστασης λειτουργίας των συστημάτων αυτών και υπηρεσιών αυτών. Τέλος σε συνδυασμό με την χρήση μηχανισμών ανίχνευσης κακόβουλων επιθέσεων θα μπορούσε να ενεργοποιηθεί η διαδικασία εφαρμογής καταλόγου στο Puppet και τα συστήματα να προσαρμοστούν στην επιθυμητή κατάσταση λειτουργίας.

4.3 Επίλογος

Εν κατακλείδι το μέλλον της οργάνωσης των λειτουργικών συστημάτων βρίσκεται σε εργαλεία σαν το Puppet. Εργαλεία που παρέχουν την δυνατότητα αυτόματης μαζικής παραμετροποίησης, σε περιβάλλοντα με ιδιαίτερες απαιτήσεις. Περιβάλλοντα όπου πολλές φορές οι χρήστες μη σκεπτόμενοι, προκαλούν προβλήματα. Προβλήματα που ο διαχειριστής σκεπτόμενος ορθά μπορεί να λύσει με εύκολο και γρήγορο τρόπο. Διότι όπως είπε και ο Πλάτων «Ένας που σκέφτεται σωστά είναι καλύτερος από μύριους που δεν σκέφτονται».

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Debian, "Debian," [Online]. Available: https://www.debian.org/index.el.html. [Accessed 16 January 2017].

[2] Wikipedia, the free encyclopedia, "List of Linux distributions," [Online]. Available: https://en.wikipedia.org/wiki/List_of_Linux_distributions. [Accessed 16 January 2017].

[3] Xubuntu, "Xubuntu," [Online]. Available: https://xubuntu.org/tour/. [Accessed 16 January 2017].

[4] Google Trends, "Compare," [Online]. Available: https://www.google.com/trends/explore?cat=5&date=2010-01-03%202017-01-15&q=puppet%20devops,docker%20devops,chef%20devops,ansible%20devops. [Accessed 13 January 2017].

[5] Ansible, "Ansible," [Online]. Available: https://www.ansible.com/. [Accessed 15 January 2017].

[6] Docker, "Docker," [Online]. Available: https://www.docker.com/. [Accessed 15 January 2017].

[7] Wikipedia, the free encyclopedia, "Active Directory," [Online]. Available: https://en.wikipedia.org/wiki/Active_Directory. [Accessed 24 January 2017].

[8] M. Butcher, Mastering OpenLDAP Configuring, Securing and Integrating Directory Services, 2007.

[9] Puppet, "Puppet," [Online]. Available: https://puppet.com/. [Accessed 15 January 2017].

[10] National Vulnerability Database, "Vulnerability Summary for CVE-2015-4000," 20 May 2015. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-4000. [Accessed 23 January 2017].

[11] Wikipedia, the free encyclopedia, "Diffie–Hellman key exchange," [Online]. Available: https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange. [Accessed 22 January 2017].

[12] Wikipedia, the free encyclopedia, "Virtual Private Network," [Online]. Available: https://en.wikipedia.org/wiki/Virtual_private_network. [Accessed 20 January 2017].

[13] Harokopio University, "VPN Service," [Online]. Available: https://www.hua.gr/index.php/el/services80/user-account52. [Accessed 15 January 2017].

[14] Puppet, "Resource Type Reference," [Online]. Available: https://docs.puppet.com/puppet/latest/type.html. [Accessed 20 December 2016].

[15] ruby-lang, "Ruby," [Online]. Available: http://www.ruby-lang.org/en/. [Accessed 12 December 2016].

[16] Puppet, "Reference," [Online]. Available: https://docs.puppet.com/puppet/latest/system_requirements.html. [Accessed 03 December 2016].

[17] Wikipedia, the free encyclopedia, "macOS," [Online]. Available:

https://en.wikipedia.org/wiki/MacOS. [Accessed 05 December 2016].

[18] vmwware, "Fusion for Mac," [Online]. Available: http://www.vmware.com/products/fusion.html. [Accessed 01 December 2016].

[19] Puppet, "Repository," [Online]. Available: https://apt.puppetlabs.com/. [Accessed 4 January 2017].

[20] Chef, "Chef," [Online]. Available: https://www.chef.io/. [Accessed 15 January 2017].

[21] Puppet, "Puppet Enterprise and Open Source Puppet," [Online]. Available: https://puppet.com/product/puppet-enterprise-and-open-source-puppet. [Accessed 3 January 2017].

[22] Puppet, "Open source projects," [Online]. Available: https://puppet.com/product/open-source-projects. [Accessed 20 December 2016].

[23] Debian, "Aptitude," [Online]. Available: https://wiki.debian.org/Aptitude. [Accessed 19 January 2017].

[24] Wikipedia, the free encyclopedia, "Vim (text_editor)," [Online]. Available: https://en.wikipedia.org/wiki/Vim_(text_editor). [Accessed 19 January 2017].

[25] Wikipedia, the free encyclopedia, "Getent," [Online]. Available: https://en.wikipedia.org/wiki/Getent. [Accessed 19 January 2017].

[26] die.net, "nslcd(8) - Linux man page," [Online]. Available: https://linux.die.net/man/8/nslcd. [Accessed 19 January 2017].

[27] Ubuntu Documentation, "LDAPClientAuthentication," [Online]. Available: https://help.ubuntu.com/community/LDAPClientAuthentication. [Accessed 19 January 2017].

[28] Wikipedia, the free encyclopedia, "Cron," [Online]. Available: https://en.wikipedia.org/wiki/Cron. [Accessed 18 January 2017].

[29] Wikipedia, the free encyclopedia, "Vi," [Online]. Available: https://en.wikipedia.org/wiki/Vi. [Accessed 17 January 2017].

[30] Wikipedia, the free encyclopedia, "Secure Shell," [Online]. Available: https://en.wikipedia.org/wiki/Secure_Shell. [Accessed 17 January 2017].

[31] Wikipedia, the free encyclopedia, "DevOps," [Online]. Available: https://en.wikipedia.org/wiki/DevOps. [Accessed 12 January 2017].

Γλωσσάρι

DevOps……Κατεύθυνση πληροφορικής που συνδυάζει τον προγραμματισμό και την διαχείριση

Puppet……..Λογισμικό διαχείρισης λειτουργικών συστημάτων

Chef……..Λογισμικό διαχείρισης λειτουργικών συστημάτων

Docker……..Λογισμικό διαχείρισης λειτουργικών συστημάτων

Ansible……..Λογισμικό διαχείρισης λειτουργικών συστημάτων

LDAP……..Υπηρεσία ενεργού καταλόγου

NSLCD……..Διεργασία του LDAP

VIM……..Κειμενογράφος

GETENT……..Εντολή ενημέρωσης δεδομένων

Updates……..Ενημέρωση συστήματος με τις τελευταίες εκδόσεις λογισμικού

Linux……..Λειτουργικό σύστημα

SSH……..Πρωτόκολλο δικτύου

Ruby……..Γλώσσα προγραμματισμού

Manifest……..Κατάλογος δηλώσεων

Module……..Ενότητα

Forge……..Αποθετήριο ενοτήτων για το Puppet

Hiera……..Εργαλείο αναζήτησης και ελέγχου για το Puppet

Logs - Logging……Μητρώα καταγραφής πληροφοριών

VMware Fusion……Λογισμικό εικονικών υπολογιστικών μηχανών

Cloud……Τεχνολογία απομακρυσμένων και αυτοματοποιημένων υπηρεσιών

VI……Κειμενογράφος

Windows……Λειτουργικό σύστημα

Xubuntu……Διανομή λειτουργικού συστήματος Linux

Repositories……Αποθετήρια

Debian……Διανομή λειτουργικού συστήματος Linux

Xfce……Γραφικό περιβάλλον για Linux

Configuration……Διαμόρφωση

Cron……Λογισμικό προγραμματισμένων εργασιών με χρονοδιάγραμμα

Snapshot……Στιγμιότυπο

Trusty……Έκδοση λειτουργικού συστήματος βασισμένο σε Debian

Willy……Έκδοση λειτουργικού συστήματος βασισμένο σε Debian

Xenial……Έκδοση λειτουργικού συστήματος βασισμένο σε Debian

Diffie-Hellman……Διαδικασία ανταλλαγής κλειδιού

DNS……Σύστημα συσχετισμού ονόματος με μηχανή

VPN……Τεχνολογία δημιουργίας ενός εικονικού προσωπικού δικτύου

Install……Εγκατάσταση

Command.....Εντολή

Server.....Εξυπηρετητής

Agent.....Τερματικό πελάτη

Ευρετήριο