ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτόματη κλιμάκωση στο σύστημα υπολογιστικού Νέφους OpenStack

Δημήτριος Α. Ζαμπούρας

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αυτόματη κλιμάκωση στο σύστημα υπολογιστικού Νέφους OpenStack

Δημήτριος Α. Ζαμπούρας

Α.Μ.: 1115200800001

ΕΠΙΒΛΕΠΩΝ: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός συστήματος δυναμικής εξισορρόπησης φόρτου σε περιβάλλον Υπολογιστικού Νέφους χρησιμοποιώντας το ελεύθερο πρόγραμμα ανοικτού κώδικα OpenStack. Αρχικά περιγράφεται το Υπολογιστικό Νέφος, έπειτα η πλατφόρμα OpenStack, εμβαθύνοντας στις υπηρεσίες που χρησιμοποιήθηκαν σε μεγαλύτερο βαθμό για την υλοποίηση της εργασίας, καθώς και η διαδικασία που ακολουθήθηκε για την δημιουργία του συστήματος εξισορρόπησης φόρτου. Τέλος, γίνεται μια αναφορά στην δυνατότητα υψηλής διαθεσιμότητας που προσφέρει το OpenStack και των τεχνικών που παρέχονται για την επίτευξή της.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Νέφους

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: OpenStack, αυτόματη κλιμάκωση, εξισορροπιστής φόρτου

ABSTRACT

The main aim of this dissertation is the implementation of a dynamic autoscaling environment using the OpenStack open source cloud computing software. In the first chapters the dissertation references the fundamental Cloud services and continues addressing the main components of the Openstack architecture with respect to those that were invaluable to the implementation of the project. In the last chapters the implementation guide references the steps required to configure and install the OpenStack platform and the corresponding services in order to achieve the autoscaling environment. Finally the dissertation addresses techniques required to implement high availability.

SUBJECT AREA: Cloud Computing

KEYWORDS: OpenStack, AutoScale, Load Balancer

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια του προπτυχιακού τμήματος Πληροφορικής και Τηλεπικοινωνιών της Σχολής Θετικών Επιστημών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή, κ. Ευστάθιο Χατζηευθυμιάδη, για την συνεργασία και την καθοδήγησή του στην υλοποίηση της εργασίας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

1. Νέφος (Cloud)

Με τον όρο Νέφος (Cloud) εννοούμε την παροχή υπηρεσιών και υπολογιστικών πόρων μέσω ενός κεντροποιημένου δικτύου υπολογιστών (Σύννεφο). Το Υπολογιστικό Νέφος επιτρέπει σε καταναλωτές και επιχειρήσεις να χρησιμοποιούν εφαρμογές και υπηρεσίες, χωρίς να τις έχουν απαραιτήτως εγκαταστήσει, και να έχουν πρόσβαση στους προσωπικούς τους φακέλους μέσω οποιουδήποτε συνδεδεμένου στο διαδίκτυο υπολογιστή. Η ονομασία Νέφος προήλθε από τον τρόπο αναπαράστασης του διαδικτύου σε διαγράμματα ροής ως ένα σύννεφο. Η ονομασία "νέφος" προήλθε από τον τρόπο αναπαράστασης του διαδικτύου σε διαγράμματα ροής ως ένα σύννεφο. Οι υπηρεσίες που προσφέρει το Νέφος εστιάζει στην αποτελεσματική κατανομή των διαμοιραζόμενων πόρων. Οι εν λόγω πόροι ανήκουν σε πολλούς χρήστες και διαμοιράζονται ανάλογα με την ζήτηση [1].

Η βασική τεχνολογία που έκανε δυνατή την δημιουργία του Νέφους είναι η εμφάνιση της εικονικοποίησης (Virtualization). Η εικονικοποίηση δημιουργεί πολλές εικονικές συσκευές μέσα σε μια φυσική συσκευή, καθε μία από τις οποίες μπορεί να εκτελεί διαφορετικές υπολογιστικές εργασίες. Τα φιλοξενούμενα λειτουργικά συστήματα και οι φιλοξενούμενες εφαρμογές εκτελούνται μέσα σε ένα περιβάλλον τέτοιο, ώστε οι χρήστες να έχουν την ψευδαίσθηση ότι αποτελεί το βασικό λειτουργικό σύστημα του υπολογιστή. Με τις αυξανόμενες δυνατότητες των επεξεργαστών σε επίπεδο παραλληλίας μπορούν να δημιουργηθούν εικονικές συσκευές οι οποίες χρησιμοποιούν αποδοτικά τις δυνατότητες των επεξεργαστών στο έπακρο όσο η συσκευή περιμένει κάποια είσοδο από τον χρήστη και βρίσκονται σε κατάσταση αδράνειας(idle). Επομένως, η εικονικοποίηση προσφέρει καλύτερη αξιοποίηση της εκάστοτε υπολογιστικής υποδομής με λιγότερο κόστος [2].Το Υπολογιστικό Νέφος μπορεί να είναι είτε δημόσιο είτε ιδιωτικό. Το δημόσιο Νέφος διαθέτει τις υπηρεσίες του με χρέωση, ενώ το ιδιωτικό έχει αναπτυχθεί για να καλύψει τις ανάγκες ενός φορέα(π.χ. μιας εταιρίας) και λειτουργεί εξυπηρετώντας αποκλειστικά αυτές.

Διάφορες εταιρίες κολοσσοί προσφέρουν τις υπηρεσίες του Υπολογιστικού Νέφους σε ιδιώτες ή σε άλλες εταιρίες χρησιμοποιώντας μεγάλα υπολογιστικά κέντρα(Amazon, Windows κ.ά.). Οποιοσδήποτε απομακρυσμένος πελάτης μπορεί να έχει πρόσβαση στις υπηρεσίες αυτές και χρεώνεται ανάλογα με την χρήση των υπηρεσιών και των πόρων που κάνει. «Πωλείται», δηλαδή, ανάλογα με την ζήτησή του. Ένα επιπρόσθετο χαρακτηριστικό του Υπολογιστικού Νέφους είναι η ελαστικότητά του. Ένας χρήστης, για παράδειγμα, μπορεί να έχει όσες υπηρεσίες χρειάζεται ανά πάσα στιγμή και να τις διαχειρίζεται στο σύνολο τους ο παροχέας. Ο χρήστης δεν χρειάζεται να έχει παρά μόνο ένα τερματικό και σύνδεση στο Ίντερνετ [3].

1.1. Διάκριση των Υπηρεσιών του Νέφους

Οι υπηρεσίες του Νέφους διαχωρίζονται σε τρείς μεγάλες κατηγορίες ανάλογα με το είδος της υπηρεσίας που προσφέρουν:

-***Υπηρεσίες Υποδομής*** (IaaS): Με τον όρο Υπηρεσίες Υποδομής εννούμε την παροχή υλικοτεχνικών υποδομών (εικονικά ή μή) στον χρήστη. Στις υπηρεσίες αυτές περιλαμβάνονται εικονικές μηχανές, εξυπηρετητές(servers), υπηρεσίες αποθήκευσης δεδομένων, υπηρεσίες δικτύων κτλ. Για να αναπτύξουν τις εφαρμογές τους οι χρήστες του Νέφους εγκαθιστούν λογισμικά στις υποδομές που τους παρέχονται και η χρέωση αυτών των υπηρεσιών αντανακλάται συνήθως στον αριθμό των πόρων που έχουν δεσμευτεί.

-Η ***Υπηρεσία Πλατφόρμας*** (PaaS): Στην υπηρεσία αυτή οι πάροχοι προσφέρουν μια ολοκληρωμένη πλατφόρμα με έναν ή περισσότερους διακομιστές δικτύου με λειτουργικό σύστημα, περιβάλλον για την εκτέλεση προγραμμάτων και σύστημα βάσεων δεδομένων. Οι χρήστες μπορούν να εγκαταστήσουν, εκτελέσουν και να εξελίξουν τα προγράμματά τους χωρίς να χρειάζεται να αγοράσουν και να διαχειριστούν υποδομές και προγράμματα που θα χρειαζόντουσαν σε διαφορετική περίπτωση. Τα συστήματα αυτά συνήθως σε περίπτωση αυξημένου φόρτου κλιμακώνονται ανάλογα με τις ανάγκες των εφαρμογών και οι χρήστες δεν χρειάζεται να κατανέμουν πόρους χειροκίνητα.

-Οι ***Υπηρεσίες Λογισμικού*** (SaaS): Πρόκειται για υπηρεσίες στις οποίες δίνεται πρόσβαση σε εφαρμογές λογισμικού και βάσεων δεδομένων . Οι πάροχοι είναι υπεύθυνοι για την υποδομή και την πλατφόρμα στην οποία βασίζεται η υπηρεσία για να λειτουργήσει. Οι χρήστες, επομένως, δεν χρειάζεται να εγκαταστήσουν τα απαραίτητα λογισμικά προγράμματα στον δικό τους υπολογιστή, πράγμα που ελαχιστοποιεί το κόστος συντήρησης και υποστήριξης. Η πρόσβαση στις υπηρεσίες γίνεται μέσω μιας διαδικτυακής πύλης και οι χρήστες δεν γνωρίζουν τι είδους μηχάνημα(εικονικό ή φυσικό) τους εξυπηρετεί καθώς βλέπουν ένα κοινό σημείο πρόσβασης [4] [5] [6] [7].

2. OpenStack

Το OpenStack είναι ένα πρόγραμμα ανοικτού κώδικα για την δημιουργία Νεφών. Η τεχνολογία αυτή αποτελείται από την σύνθεση αλληλοσχετιζόμενων εργασιών οι οποίες συνολικά, είναι υπεύθυνες για την διαχείριση των υπολογιστικών πόρων(resources), των συστημάτων αποθήκευσης, καθώς και των δικτυακών πόρων, παράλληλα δίνοντας στον χρήστη την δυνατότητα να τα χειριστεί μέσω μιας δικτυακής πύλης από την γραμμή εντολών είτε μέσω μιας προγραμματιστικής διεπαφής Restful (Restful API) [8].

2.1. Ιστορική Αναδρομή του OpenStack

Πρωτοπόρος στην ανάπτυξη ενός κεντρικού συστήματος διαχείρισης του συνόλου των προγραμμάτων και εφαρμογών σχετικά με τις διαστημικές εξερευνήσεις υπήρξε η NASA. Με το σχέδιο Nebula η NASA στόχευε να δημιουργήσει μια κοινή διαδικτυακή πλατφόρμα προσφέροντας εργαλεία, εφαρμογές και πόρους στους επιστήμονες με σκοπό την σύγκλιση των εργασιών που επιτελούσαν καθώς και την αύξηση της παραγωγικότητας τους. Με αυτό το σκεπτικό το 2008 η NASA ξεκίνησε να δημιουργήσει την εν λόγω πλατφόρμα. Στη συνέχεια όταν συνειδητοποίησε ότι η υποστήριξη ενός τέτοιου προγράμματος επιζητούσε την ύπαρξη μιας ελαστικής υποδομής σε συνδυασμό με τις ανάγκες της αγοράς, η οποία επιζητούσε λύσεις απευθυνόμενη σε πρώιμα υπολογιστικά νέφη χωρίς ακόμα να έχει παρουσιαστεί κάτι ολοκληρωμένο, επέλεξε να περιστραφεί γύρω από την επίλυση του προβλήματος της ελαστικής υποδομής.

Στην πρώτη παρουσίαση του υπολογιστικού ελεγκτή Nova (Nova controller) βρέθηκαν σε επαφή με την Rackspace η οποία είχε δημιουργήσει εργαζόμενη σε παρόμοια λογική ένα σύστημα διαχείρισης πόρων αποθήκευσης και σκόπευε να αφοσιωθεί στο σύστημα που η NASA είχε μόλις δημιουργήσει, και η NASA αντίστοιχα σκόπευε να δημιουργήσει το σύστημα ελέγχου των αποθηκευτικών πόρων.

Έχοντας την ίδια πορεία, NASA και Rackspace επιλέγουν να ενώσουν τις δυνάμεις και τις γνώσεις τους και να συνεργαστούν προς όφελος και των δύο. Ως συνέπεια τον Ιούλιο του 2010 η NASA σε συνεργασία με την Rackspace Hosting ξεκινούν την δημιουργία ενός λογισμικού διαχείρισης και συντονισμού των πόρων ενός νέφους το οποίο ονομάστηκε OpenStack. Η αρχική ιδέα του λογισμικού ήταν να προσφέρει ολοκληρωμένες υπηρεσίες ως Υπολογιστικό Νέφος και θα βασιζόταν σε συγκεκριμένη υλικοτεχνική υποδομή. Η πρώτη επίσημη έκδοσή του - με την κωδική ονομασία Austin- εμφανίστηκε τέσσερις μήνες μετά [**9**].

Μετά απο ένα χρονικό διάστημα κοινότητες προγραμματιστών πυροδότησαν την εξελικτική πορεία του OpenStack και οι πρώτες εκδόσεις του σε λειτουργικά συστήματα(Ubuntu - NattyNarwal) άρχισαν να εμφανίζονται προσελκύοντας το ενδιαφέρον του κόσμου της πληροφορικής αλλά και εταιριών όπως οι Amazon, Microsoft κ.ά., οι οποίες αναγνώρισαν την ευκαιρία ύπαρξης μιας ολοκληρωμένης

πλατφόρμας παροχής υπηρεσιών Υπολογιστικού Νέφους με αποτέλεσμα να ενεργοποιηθούν στον νεοσύστατο αυτό τομέα [10].
Αυτή την στιγμή το OpenStack με την συνεισφορά της κοινότητας έχει φτάσει σε σημείο να είναι αυτοσυντηρούμενο και αποτελεί τον ακρογωνιαίο λίθο σχεδόν σε όλα τα συστήματα διαχείρισης Υπολογιστικών Νεφών που προσφέρονται σήμερα [9].

2.2. Περιγραφή των συστατικών στοιχείων του OpenStack

Το OpenStack απαρτίζεται από διάφορα αλληλοσχετιζόμενα προγράμματα εργασίες τα οποία διαχωρίζονται συνήθως με βάση τις λειτουργίες που επιτελούν και τον τύπο των υπηρεσιών που προσφέρουν.
Το κομμάτια που απαρτίζουν το OpenStack είναι τα εξής:

- Compute (Nova)
- Object Storage (Swift)
- Block Storage (Cinder)
- Networking (Neutron)
- Dashboard (Horizon)
- Identity Service (Keystone)
- Image Service (Glance)
- Telemetry Service (Ceilometer)
- Orchestration (Heat)
- Database (Trove)

Ακολουθεί αναλυτική περιγραφή του κάθε συστατικού [**11**].

2.2.1. Compute (Nova)

Το Nova είναι το βασικό συστατικό που διαχειρίζεται τους υπολογιστικούς πόρους, δηλαδή τις *Υπηρεσίες Υποδομής* του Υπολογιστικού Νέφους, οι οποίες δεν είναι άλλες από τα φυσικά του εξαρτήματα(επεξεργαστές), ενός ή πολλών μηχανημάτων τα οποία ενορχηστρώνονται υπό την εποπτεία του Nova.
Το Nova είναι υπεύθυνο για τα προγράμματα οδήγησης που αλληλοεπιδρούν με την υποδομή, καθώς και για την λειτουργικότητα μέσω της προγραμματιστικής διεπαφής. Αποτελείται από επτά κύρια εξαρτήματα με τον ελεγκτή να έχει τον ρόλο της αλληλεπίδρασης με τα υπόλοιπα συστατικά [12].

2.2.2. Object Storage (Swift)

Το Swift είναι υπεύθυνο για την δημιουργία μηχανισμών αποθήκευσης που μπορούν να κλιμακωθούν(scale) ανάλογα με τις ανάγκες του συστήματος, χρησιμοποιώντας ένα σύμπλεγμα από εξυπηρετητές. Αποτελεί ένα σύστημα μακροπρόθεσμης αποθήκευσης στατικών, μεγάλων σε έκταση, δεδομένων τα οποία μπορούν ανά πάσα στιγμή να ανακτηθούν, να αξιοποιηθούν και να ενημερωθούν. Το Object Storage χρησιμοποιεί μια κατανεμημένη αρχιτεκτονική προσφέροντας έτσι την δυνατότητα για καλύτερη κλιμάκωση, μονιμότητα και εναλλαξιμότητα.

Τα αντικείμενα αποθηκεύονται σε πολλές συσκευές αποθήκευσης και το πρόγραμμα είναι υπεύθυνο για την συντήρηση και την αξιοπιστία καθόλη την διάρκεια παραμονής τους μέσα στο σύννεφο. Οι αποθηκευτικοί κόμβοι κλιμακώνονται οριζόντια απλά προσθέτοντας έναν νέο κόμβο στην αλυσίδα των ήδη υπαρχόντων κόμβων. Σε περίπτωση που ένας κόμβος υποστεί κάποια βλάβη, το Object Storage είναι υπεύθυνο για την αντιγραφή του περιεχομένου του από τους ενεργούς κόμβους. Το πρόγραμμα αυτό είναι ιδανικό αφού αποτελεί μια οικονομική λύση για αποθήκευση δεδομένων, αρχειοθέτηση και backup με δυνατότητες επεκτασιμότητας [13] [14].

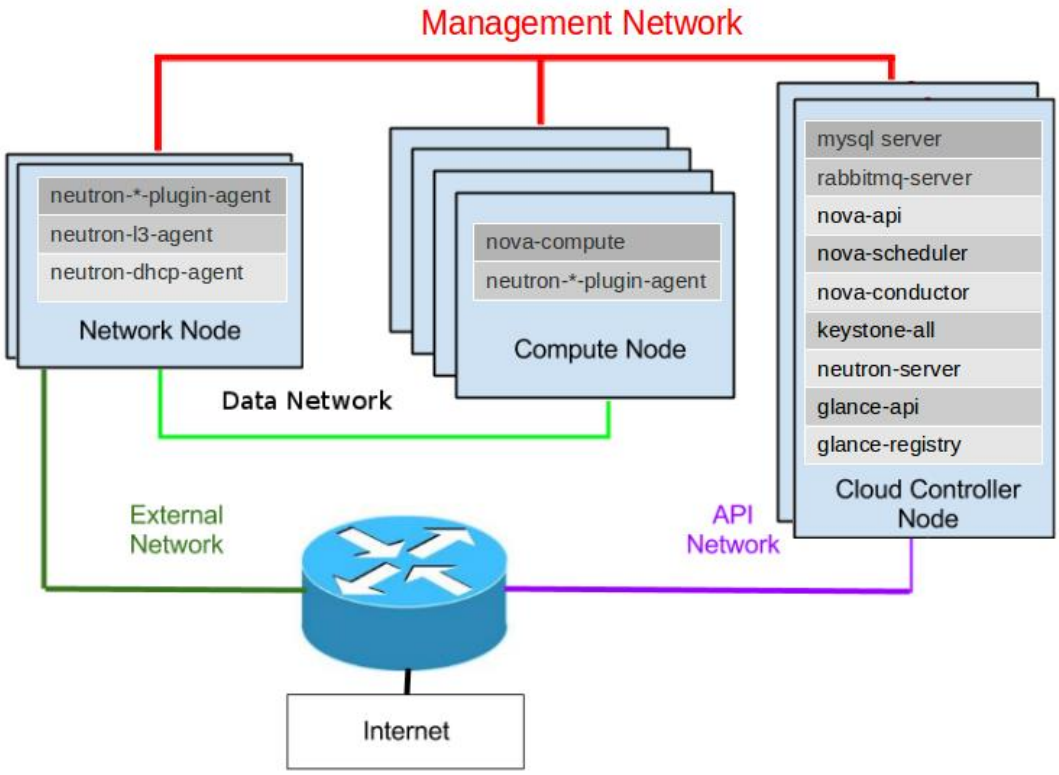
2.2.3. Block Storage (Cinder)

Η υπηρεσία του OpenStack Block Storage λειτουργεί χρησιμοποιώντας μια σειρά απο διεργασίες οι οποίες ανήκουν μόνιμα στο μηχάνημα. Επι της ουσίας η υπηρεσία αυτή μπορεί να παρέχει αποθηκευτικό χώρο επιπέδου μπλοκ στα στιγμιότυπα(instances) που έχουν δημιουργηθεί και δίνει δυνατότητες δημιουργίας στιγμιότυπου στους εικονικούς τόμους που δημιουργεί [15].

2.2.4. Networking (Neutron)

Η υπηρεσία Networking ή αλλιώς Neutron προσφέρει ένα σύνολο από εργαλεία, συναρτήσεις και πρωτόκολλα τα οποία δίνουν την δυνατότητα στον χρήστη να καθορίσει την συνδεσιμότητα του δικτύου και της διευθυνσιοδότησης μέσα στο «Υπολογιστικό Νέφος». Επιπρόσθετα δίνει πρόσβαση σε χειριστές δικτύου οι οποίοι παρέχουν ένα εύρος από εργαλεία με σκοπό οι χρήστες να εφαρμόσουν την δική τους τοπολογία δικτύου καθώς και πλήθος από άλλες υπηρεσίες και πρωτόκολλα όπως προώθηση επιπέδου δικτύου(L3 forwarding), μετάφραση δικτυακών διευθύνσεων(NAT), εξισορροπιστές φόρτου(load balancers), τείχη προστασίας(firewalls), και ασφαλή εικονικά ιδιωτικά δίκτυα(VPN’s). Πιο συγκεκριμένα η υπηρεσία αυτή προσφέρει:

- Δυνατότητα δημιουργίας προηγμένων δικτύων σε πολλαπλά επίπεδα καθώς και μεταφορά εφαρμογών στο «Υπολογιστικό Νέφος» χωρίς να χαθούν οι διευθύνσεις δικτύων που τους έχουν ανατεθεί.
- Ευελιξία στους διαχειριστές να παραμετροποιήσουν το δίκτυο τους.
- Δυνατότητα σε προγραμματιστές να εξελίξουν τα εργαλεία και τα πρωτόκολλα που προσφέρονται ανάλογα με τις δικές τους ανάγκες [16].



Εικόνα 1 – Neutron Service

Υπηρεσία εξισορρόπησης φόρτου - Load-Balancer-as-a-Service (LBaaS)

Στις υπηρεσίες του Neutron προσφέρεται και η υπηρεσία των εξισορροπιστών φόρτου. Η υπηρεσία αυτή δίνει την δυνατότητα κατανομής των αιτήσεων που κατευθύνονται εσωτερικά στο Υπολογιστικό Νέφος, χρησιμοποιώντας αλγόριθμους φόρτου για να παρθούν αποφάσεις προώθησης σε διακομιστές. Η εν λόγω κατανομή διασφαλίζει οτι ο φόρτος εργασίας θα διανεμηθεί με προβλέψιμο τρόπο στα στελέχη του δικτύου. Οι αλγόριθμοι κατανομής φόρτου είναι οι εξής:

Round Robin: Κατανέμει ισάριθμα τις αιτήσεις (περιστροφικά) στα στιγμιότυπα του δικτύου.

Source IP: Οι αιτήσεις από συγκεκριμένες διευθύνσεις(εξωτερικές) προωθούνται μόνιμα και με συνέπεια στο ίδιο στιγμιότυπο.

Least Connections: Προωθεί τις αιτήσεις στο στιγμιότυπο με τις λιγότερες ενεργές συνδέσεις.

Χαρακτηριστικά του LBaaS

Monitors*:* Οι ελεγκτές αυτοί είναι υπεύθυνοι για να αποφανθούν ποιά μέλη μπορούν να δεχτούν συνδέσεις.

Management*:* Οι χρήστες του Νέφους διαχειρίζονται τους εξισορροπιστές φόρτου μέσω του Command Line Interface του Neutron είτε απο το Dashboard είτε απο το REST API.

Connection Limits: Υπάρχει η δυνατότητα εισαγωγής ενός ανώτερου κατωφλίου συνδέσεων, ελέγχοντας κατά συνέπεια τον φόρτο και θωρακίζοντας το σύστημα από επιθέσεις άρνησης υπηρεσίας (Denial of Service - DoS).

Session Persistence*:* Η υπηρεσία υποστηρίζει συνεχείς συνδέσεις προωθώντας αιτήσεις στο ίδιο στιγμιότυπο από το οποίο εξυπηρετήθηκαν προηγουμένως. Επιπρόσθετα υποστηρίζει την χρήση cookies [17].

2.2.5. Dashboard (Horizon)

Το Dashboard, με κωδική ονομασία Horizon, είναι μια διαδικτυακή πύλη μέσω της οποίας ο χρήστης μπορεί να χειριστεί τους πόρους και τις υπηρεσίες του OpenStack. Μέσω του γραφικού περιβάλλοντος που προφέρει οι χρήστες μπορούν να χειριστούν το Νέφος σε ένα φιλόξενο περιβάλλον χωρίς να χρειάζεται να επικεντρωθούν σε άλλα εργαλεία (Command Line Interface) τα οποία προϋποθέτουν μεγαλύτερο βαθμό εμπειρίας και γνώσεων [18].

2.2.6. Identity Service (Keystone)

Το Keystone προσφέρει υπηρεσίες ταυτοποίησης και διαχείρισης ρόλων. Η υπηρεσία χωρίζεται σε τέσσερις μεγάλες υποκατηγορίες:

- Διαχείριση χρηστών
- Διαχείριση υπηρεσιών
- Διαχείριση ομάδων
- Διαχείριση τομέα

Διαχείριση χρηστών

Τα βασικά συστατικά αυτής της υποκατηγορίας είναι:

1. *Χρήστης (User)*: Αντιπροσωπεύει ένα πραγματικό άνθρωπο-χρήστη. Δεσμεύονται σχετικές πληροφορίες όπως user name, password και διεύθυνση ηλεκτρονικού ταχυδρομείου.

2. *Ένοικος (Tenant)*: Ορίζει μια ομάδα ή μια επιχείρηση υπο ένα κοινό έργο(project). Η χρήση του OpenStack επιβάλλει την ύπαρξη ενός ένοικου. Για παράδειγμα όταν κάποιος χρήστης ζητάει απο το Nova(Compute) μια λίστα με τα ενεργά στιγμιότυπα, η λίστα που παίρνει ως απάντηση αφορά το συγκεκριμένο project-tenant.

3. *Ρόλος (Role)*: Συγκρατεί τις διαδικασίες που ένας χρήστης μπορεί να εφαρμόσει σε συγκεκριμένο project-tenant

Το Keystone αναθέτει ένα project-tenant και έναν ρόλο σε κάθε χρήστη. Τα δικαιώματα και οι ρόλοι μπορούν να αλλάξουν μέσω της υπηρεσίας αυτής.
Επιπρόσθετα ένας χρήστης μπορεί να έχει διαφορετικούς ρόλους σε διαφορετικά project-tenants [19].

Διαχείριση Υπηρεσιών

Η Υπηρεσία Ταυτοποίησης (Identity Service) προσφέρει υπηρεσίες τεκμηρίων και πολιτικής. Επιπλέον διατηρεί εναν βασικό χρήστη ο οποίος αντιστοιχεί σε κάθε βασική υπηρεσία του OpenStack, όπως για παράδειγμα ένας χρήστης με το όνομα nova με όλα τα δικαιώματα που χρειάζεται η υπηρεσία.

Διαχείριση Ομάδας

Μια ομάδα αποτελείται από ένα σύνολο χρηστών. Οι διαχειριστές μπορούν να δημιουργήσουν ομάδες και να προσθέσουν χρήστες σε αυτές και αντί να χειρίζονται τα δικαιώματα του κάθε χρήστη μπορούν να τα εφαρμόζουν μαζικά.

Διαχείριση Τομέα

Ένας τομέας καθορίζει τα όρια της διαχείρισης της ίδιας της υπηρεσίας, δηλαδή της Υπηρεσίας Ταυτοποίησης. Κάθε τομέας μπορεί να αντιπροσωπεύει έναν ιδιώτη, μια εταιρία ή το χώρο ενός διαχειριστή. Χρησιμοποιείται για να προσδώσει διοικητικά προνόμια σε χρήστες του συστήματος. Ο τομέας αποτελείται από ένα σύνολο από ενοίκους, χρήστες και ρόλους. Στους χρήστες μπορούν να αποδοθούν τα διοικητικά δικαιώματα ενος τομέα. Στα δικαιώματα αυτά περιλαμβάνονται η δημιουργία νέων χρηστών, ενοίκων και ομάδων καθώς και η απόδοση ρόλων σε αυτά [20].

2.2.7. Image Service (Glance)

Η υπηρεσία αυτή αποθηκεύει εικόνες (Image) των λειτουργικών συστημάτων που χρησιμοποιούνται για την δημιουργία στιγμιότυπων. Οι χρήστες μπορούν να επιλέξουν ποια εικόνα θα μεταφορτώσουν στο glance καθώς και τι είδους διαμόρφωση θα έχουν οι εικόνες αυτές. Προσφέρει υπηρεσίες αποθήκευσης ενός στιγμιότυπου και είναι η υπηρεσία που παρέχει τις εικόνες στην υπηρεσία nova για την εκκίνηση ενός στιγμιότυπου [21] [22].

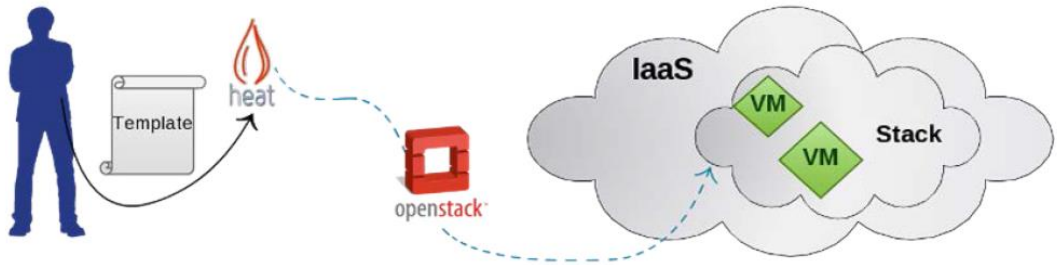
2.2.8. Telemetry Service (Ceilometer)

Η υπηρεσία αυτή δημιουργήθηκε για να καλύψει τις ανάγκες κοστολόγησης των πελατών. Επειδή οι χρήστες νοικιάζουν συγκεκριμένες υπηρεσίες του υπολογιστικού νέφους έπρεπε να δημιουργηθεί μια μέθοδος χρέωσης η οποία να αντανακλά την χρήση των υπηρεσιών από τους πελάτες. Έτσι εμφανίστηκε το ceilometer, το οποίο συλλέγει στατιστικά από τους πόρους που χρησιμοποιούνται και κοστολογεί ανάλογα με την χρήση. Αν αναλογιστούμε το εύρος των υπηρεσιών που μπορεί να προσφέρει ένα Νέφος είναι δύσκολο να υπάρχει μια ενιαία κοστολόγηση των υπηρεσιών και έτσι προσφέροντας στους χρήστες μια χρέωση ανάλογη της χρήσης των πόρων παρέχεται ένα πιο βιώσιμο μοντέλο χρέωσης.

Επομένως η αρχική χρήση της υπηρεσίας να συλλέγει στατιστικά χρήσης εξυπηρετούσε αυτήν την ανάγκη της χρέωσης ωστόσο στην πορεία εμφανίστηκαν σενάρια στα οποία η συλλογή στατιστικών γινόταν για διαφορετικό σκοπό. Για παράδειγμα η αυτόματη κλιμάκωση των πόρων ανάλογα με τον φόρτο εργασίας των στιγμιότυπων με την βοήθεια ειδοποιήσεων-συναγερμών(Alarms) οι οποίοι πυροδοτούνται σε συγκεκριμένα ποσοστά χρήσης (π.χ. φόρτος επεξεργαστή του στιγμιότυπου > 90%). Η εξέλιξη μιας καινούριας υπηρεσίας (Orchestration), η οποία παρουσιάζεται παρακάτω, έδωσε την δυνατότητα εφαρμογής σεναρίων αυτοματοποίησης στα περιβάλλοντα αυτά [23].

2.2.9. Orchestration (Heat)

Η υπηρεσία αυτή προσφέρεται από το OpenStack για την δημιουργία και την εφαρμογή πολύπλοκων συστημάτων δίνοντας στον χρήστη την δυνατότητα να τα περιγράψει με αρχεία κειμένου τα οποία με την σειρά τους μεταφράζονται σε κώδικα. Τα αρχεία αυτά ονομάζονται περιγράμματα(templates). Υπεύθυνη για την «μετάφραση» των αρχείων σε κώδικα είναι η μηχανή Heat(Heat engine).
Αυτή την στιγμή το Heat μπορεί να μεταφράσει δύο τύπους αρχείων ως είσοδο: το Heat Orchestration Template (HOT) και το AWS CloudFormation. Το HOT αποτελεί μια μορφή συντακτικού που το ίδιο το OpenStack έχει δημιουργήσει για τις ανάγκες του, ενώ το AWS CloudFormation ανήκει στην αντίστοιχη υπηρεσία της Amazon για να δίνεται η δυνατότητα σε χρήστες να μπορούν να μετακομίσουν χωρίς να χρειάζεται να δομήσουν από την αρχή το σύστημά τους.



Εικόνα 2 - Η υπηρεσία Heat

Το Heat αποτελείται από εφαρμογές Python και οι δυνατότητες που προσφέρει είναι ο εξής:

- Τα περιγράμματα περιγράφουν την υποδομή και την τοπολογία ενός συστήματος Νέφους σε ένα αρχείο κειμένου το οποίο είναι πλήρως κατανοητό απο τους χρήστες.

- Οι υποδομές που μπορούν να περιγραφούν περιλαμβάνουν: δημιουργία στιγμιότυπου, floating ips, δίσκοι δεδομένων, διαχείριση δικαιωμάτων ομάδας, δημιουργία χρηστών κτλ.

- Στα περιγράμματα δίνεται η δυνατότητα περιγραφής της συσχετίσης των πόρων του συστήματος(π.χ. αυτή η διεύθυνση δικτύου ανήκει σε αυτό το στιγμιότυπο). Για να το πετύχει αυτό και να φτάσει στην επιθυμητή υποδομή, όπως έχει περιγράψει ο χρήστης, το Heat επικοινωνεί με την προγραμματιστική διεπαφή των υπόλοιπων υπηρεσιών.

- Η υπηρεσία Heat είναι υπεύθυνη για όλο τον κύκλο ζωής του συστήματος. Σε περίπτωση που ο χρήστης θελήσει να αλλάξει την υποδομή αρκεί να επεξεργαστεί το περίγραμμα και να το τροφοδοτήσει στην υπηρεσία, η οποία είναι σε θέση να γνωρίζει ποιες είναι οι αλλαγές που πρέπει να εφαρμόσει. Είναι υπεύθυνη για την αποδέσμευση των πόρων [24].

2.2.10. Database (Trove)

Η υπηρεσία αυτή προσφέρει αξιόπιστη και κλιμακωτή λειτουργία για τα διάφορα είδη βάσεων δεδομένων – σχεσιακών ή μή. Οι χρήστες μπορούν με ευκολία και ταχύτητα να χρησιμοποιήσουν λειτουργίες βάσεων δεδομένων και να διαχειριστούν πολλές βάσεις δεδομένων ταυτόχρονα ανάλογα με τις ανάγκες. Η υπηρεσία αυτή προσφέρει ενθυλάκωση των δεδομένων και αυτοματοποιεί πολύπλοκες λειτουργίες όπως η εγκατάσταση, παραμετροποίηση, δημιουργία backup και ελέγχου [25] [26].

3. Αναλυτική περιγραφή του Network as a Service – Neutron

3.1. Περιγραφή του Neutron

3.1.1. Περιγραφή της αρχιτεκτονικής

Η υπηρεσία αυτή του OpenStack είναι ανεξάρτητη και αυτόνομη και δημιουργεί διάφορες διεργασίες στο σύστημα που έχει εγκατασταθεί καθώς και σε διάφορους κόμβους του συστήματος με σκοπό να προσφέρει ολοκληρωμένες υπηρεσίες δικτύου. Οι διεργασίες αυτές δεν επικοινωνούν μόνο μεταξύ τους, αλλά και με άλλες υπηρεσίες του OpenStack ανάλογα με την εργασία που καλούνται να υλοποιήσουν [27].
Οι υπηρεσίες/διεργασίες που χρησιμοποιεί το Neutron είναι οι εξής:

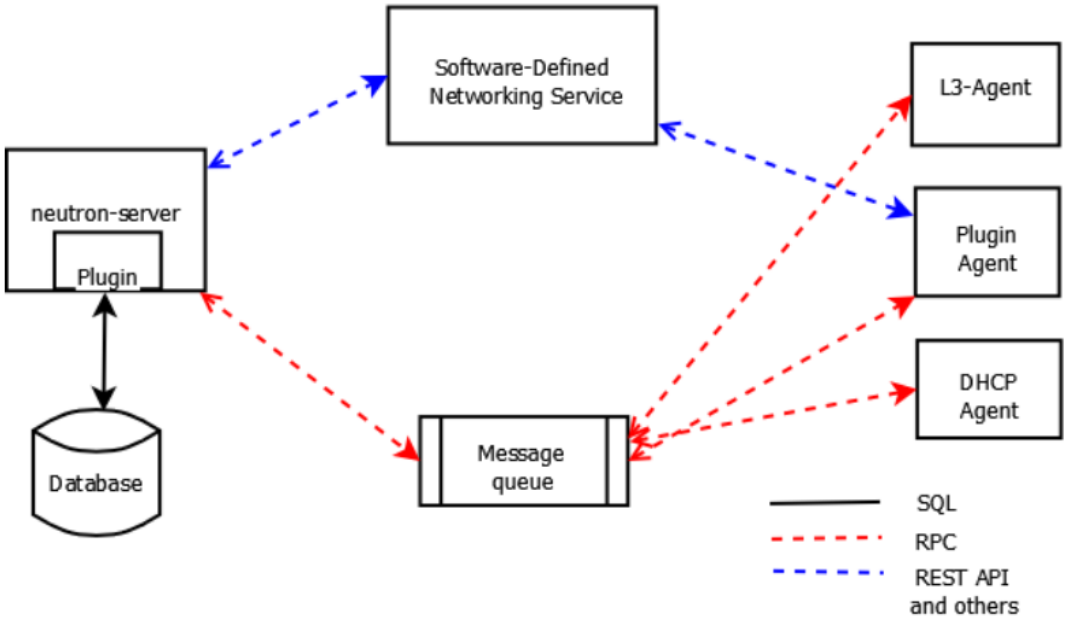
- **neutron server (neutron-server και neutron-*-plugin)**
  Η διεργασία αυτή εκτελείται από τον κόμβο του δικτύου και σκοπός της είναι η υποστήριξη και η λειτουργία του Networking API καθώς και οποιωνδήποτε πρόσθετων λειτουργιών του έχουν ανατεθεί (plugins).
  Επιβάλλει την λειτουργία του δικτύου και είναι υπεύθυνο για την διευθυνσιοδότηση. Οι διεργασίες αυτές πρέπει να έχουν πρόσβαση σε βάση δεδομένων για αποθήκευση καθώς και σε υπηρεσία αποστολής μηνυμάτων(message queues) για την ενδοεπικοινωνία με άλλα συστατικά και διεργασίες του OpenStack

- **plugin agent (neutron-*-agent)**
  Η διεργασία αυτή εκτελείται σε κάθε κόμβο υπεύθυνο για την επεξεργασία (Compute Node) και είναι υπεύθυνη για την διαχείριση του τοπικού εικονικού μεταγωγέα (vswitch). Η παροχή υπηρεσίας αποστολής μηνυμάτων για ενδοεπικοινωνία είναι επίσης απαραίτητη για αυτη την υπηρεσία.

- **DHCP agent (neutron-dhcp-agent)**
  Παρέχει υπηρεσίες DHCP σε δίκτυα ενοίκων (ομάδα χρηστών). Η διεργασία αυτή εκτελεί τις ίδιες λειτουργίες σε όλους τους κόμβους, και η κύρια λειτουργία του είναι να διατηρεί τις προδιαγραφές του DHCP. Για την λειτουργία και αυτής της διεργασίας απαραίτητη προϋπόθεση είναι η ύπαρξη ουράς μηνυμάτων για την επικοινωνία με άλλες υπηρεσίες.

- **L3 agent (neutron-l3-agent)**
  Παρέχει υπηρεσίες επιπεδου δικτύου όπως λογική διευθυνσιοδότηση και δρομολόγηση πακέτων από εξωτερικά δίκτυα που χρησιμοποιούνται από εικονικές μηχανές σε εσωτερικά δίκτυα(tenant networks).

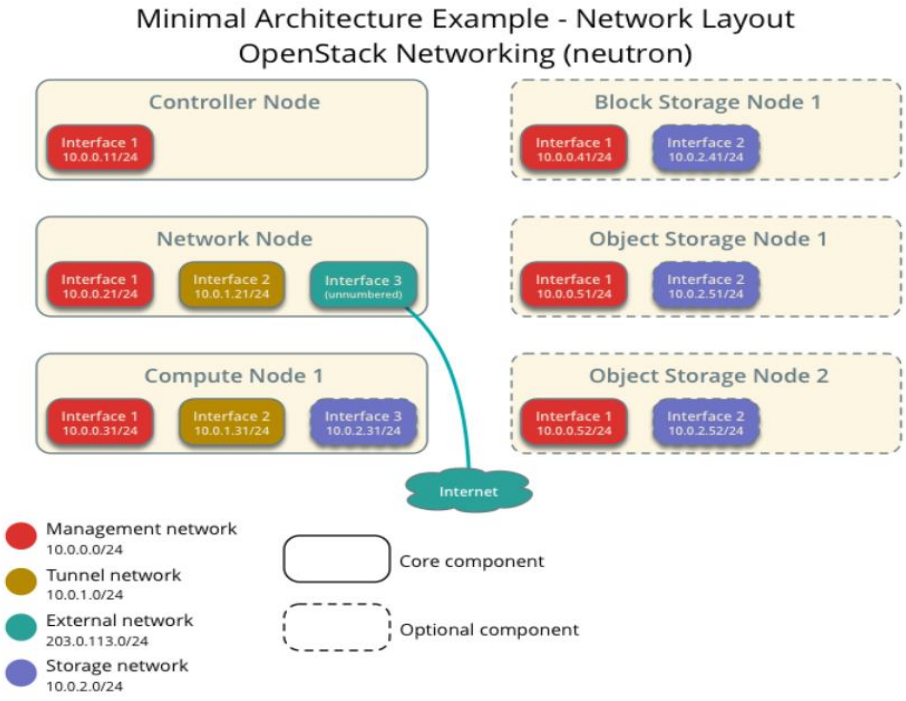
- **network provider services (SDN server/services)**
  Παρέχει επιπρόσθετες λειτουργίες σε tenant networks. Οι διεργασίες αυτές επικοινωνούν με άλλες διεργασίες κάνοντας χρήση διαφόρων προγραμματιστικών διεπαφών τύπου Restful [28].



Εικόνα 3 - Αρχιτεκτονική του Neutron

3.1.2. Εννοιολογική Αρχιτεκτονική

Αξίζει να σημειώσουμε στο σημείο αυτό ότι η προτεινόμενη εγκατάσταση του OpenStack περιέχει στο ελάχιστο τρείς κόμβους και ίδιο αριθμό η μεγαλύτερο δικτύων, ο οποίος εξαρτάται από τον συνολικό σχεδιασμό που έχει προηγηθεί σχετικά με τον φόρτο που θα έχουν τα δίκτυα και οι κόμβοι του συστήματος.
Ενδεικτικά για μικρά clusters προτείνεται η ύπαρξη τριών δικτύων για την επικοινωνία των κόμβων στο σύστημα.



Εικόνα 4 - Προτεινόμενη αρχιτεκτονική Neutron

Εικόνα 4 - Minimal Architecture Neutron

Δίκτυο Management :

Χρησιμοποιείται για την ενδοεπικοινωνία των διεργασιών του OpenStack. Οι διευθύνσεις σε αυτό το δίκτυο θα πρέπει να είναι προσβάσιμες μόνο εσωτερικά.

Δίκτυο Tunnel ή Guest:

Χρησιμοποιείται απο τις εικονικές μηχανές για την μεταξύ τους επικοινωνία, καθώς και με τον network κόμβο που τους παρέχει δρομολόγηση επιπέδου 3 στο εξωτερικό (external) δίκτυο. Οι διευθύνσεις δικτύου που χρησιμοποιούνται εξαρτώνται από την παραμετροποίηση του Neutron και τις επιλογές του κάθε ενοίκου – tenant.

Δίκτυο External:

Χρησιμοποιείται για να παρέχει στις εικονικές μηχανές πρόσβαση στο Διαδίκτυο. Οι διευθύνσεις πρέπει να είναι προσβάσιμες από οποιονδήποτε μέσω του Διαδικτύου.

Δίκτυο API:

Σε διάφορα σενάρια εγκατάστασης προτείνεται η χρήση ενός επιπλέον δικτύου για την επικοινωνία των χρηστών της πλατφόρμας με την διεπαφή εφαρμογής προγραμμάτων [28] [29] [22].

3.2. Χαρακτηριστικά και επεκτάσεις του Neutron

Το OpenStack Neutron παρέχει πολλές υπηρεσίες που θα βρίσκαμε σε ένα κέντρο δεδομένων όπως υπηρεσίες μεταγωγεά, δρομολόγησης, εξισορρόπηση φόρτου, τείχους προστασίας καθώς και εικονικά τοπικά δίκτυα. Τα πρόσθετα(plugins) που προσφέρει το Neutron στοχεύουν στην παραμετροποίηση του δικτύου του χρήστη δημιουργώντας σύνθετες αρχιτεκτονικές δικτύου που χρησιμοποιούνται σε σύγχρονα υπολογιστικά κέντρα. Τα πρόσθετα αυτά χωρίζονται σε Core και Service.

Το βασικό(**Core**) πρόσθετο που παρέχεται με εγκατάσταση του OpenStack είναι το ML2 (Modular Layer Plugin), το οποίο υποστηρίζει λειτουργίες στο επίπεδο ζεύξης δεδομένων (L2), επομένως υποστηρίζει την μεταφορά δεδομένων στο τοπικό δίκτυο. Για την επίτευξη του στόχου αυτού χρησιμοποιεί διευθύνσεις MAC, γέφυρες (bridges) καθώς και μεταγωγείς δικτύου (switches). Υποστηρίζεται απο προγράμματα οδήγησης (type drivers) που είναι υπεύθυνα για την δρομολόγηση

πακέτων στα δίκτυα και την αποθήκευση της κατάστασης του δικτύου, ανάλογα με τον τύπο δικτύου που έχει υλοποιηθεί. Υποστηριζόμενα προγράμματα οδήγησης υπάρχουν για κάθε τύπου εικονικού δικτύου: VLAN, VXLAN, GRE κτλ.

Τα πρόσθετα τύπου **Service** λειτουργούν συνήθως στο επίπεδο δικτύου (L3) και εφαρμόζουν διαφορετικές αρχιτεκτονικές και υπηρεσίες στο επίπεδο αυτό. Ακολουθεί αναλυτική περιγραφή των επεκτάσεων [30].

3.2.1. Υπηρεσίες Μεταγωγέα

Το OpenStack προσφέρει υπηρεσίες δρομολόγησης και διευθυνσιοδότησης χρησιμοποιώντας προώθηση διεύθυνσης δικτύου(Ip forwarding iptables και τεχνικές προσδιορισμού του υποδικτύου με χρήση χώρου ονομάτων (namespaces). Σε κάθε χώρο ονομάτων υπάρχουν διαθέσιμες θύρες και sockets και καθένα από αυτά έχει την δική του δρομολόγηση καθώς και διεργασίες iptables που χρησιμοποιούν φίλτρα και μετάφραση δικτυακών διευθύνσεων(NAT).

3.2.2. Υπηρεσίες Εξισορρόπησης Φόρτου

Η υπηρεσία Load Balancing as a Service ή LBaaS πρωτοεμφανίστηκε μαζί με την έκδοση Grizzly και δίνει την δυνατότητα σε χρήστες να κατανέμουν αιτήματα πελατών σε ένα δίκτυο υπολογιστών. Η έκδοση Havana είχε ως επιλογή την δυνατότητα να χρησιμοποιεί τον HAProxy ως εξισορροπιστή φόρτου.

3.2.3. Υπηρεσίες Τείχους Προστασίας

Στην πλατφόρμα του OpenStack υπάρχουν δύο τρόποι με τους οποίους επιβάλλεται ασφάλεια στο δίκτυο και στα μηχανήματα. Η ομάδα δικαιωμάτων ή ασφάλειας (Security Group) η οποία πρωτοεμφανίστηκε στην αρχική αρχιτεκτονική δικτύου Nova Network που προσέφερε το OpenStack, όταν ακόμα δεν είχε δημιουργηθεί το Network as a Service (Quantum – Neutron) και το οποίο ονομάζεται legacy networking. Η ομάδα δικαιωμάτων η οποία ως μέθοδος ασφαλείας υιοθετήθηκε και απο το neutron χρησιμοποιεί iptables στον Υπολογιστικό Κόμβο (Compute Node) για να ασφαλίσει την σωστή δρομολόγηση και κίνηση μέσα στο δίκτυο.

Με την εμφάνιση του Τείχους Προστασίας ως Υπηρεσία (Firewall as a Service – FWaaS) τα πρωτόκολλα ασφαλείας βρίσκονται στον δρομολογητή και όχι στον Υπολογιστικό Κόμβο.

3.2.4. Υπηρεσίες Εικονικών Ιδιωτικών Δικτύων

Ένα εικονικό ιδιωτικό δίκτυο (VPN) είναι μια εικονική διαδικτυακή σύνδεση μεταξύ οποιουδήποτε σημείου του Διαδικτύου βρίσκεται το μηχάνημα του χρήστη και ενός επιθυμητού εσωτερικού δικτύου. Μέσω του ιδιωτικού δικτύου ο κάθε υπολογιστής μπορεί να στείλει και να λάβει πακέτα χρησιμοποιώντας δημόσια δίκτυα σαν να ήταν συνδεδεμένος σε ένα ιδιωτικό δίκτυο. Το Neutron προσφέρει

υπηρεσίες δίνοντας την δυνατότητα σε tenants να ορίσουν και να δημιουργήσουν τα δικά τους ιδιωτικά δίκτυα χρησιμοποιώντας κατάτμηση και tunneling. Η υπηρεσίας αυτή ονομάζεται **VPNaaS** (**Virtual Private Network as a Service**) [31].

3.3. Εξισσοροπιστής Φόρτου – Load Balancer

Ο εξισσοροπιστής φόρτου όπως έχει ήδη αναφερθεί, είναι ένα πρόσθετο εργαλείο του OpenStack που προσφέρει ένα επιπλέον επίπεδο παροχής υπηρεσιών δίνοντας στον χρήστη την δυνατότητα να εισάγει λειτουργίες εξισορροπιστή φόρτου στο σύστημα. Ο προεπιλεγμένος εξισορροπιστής που προσφέρεται με την βασική εγκατάσταση είναι ο HAProxy ωστόσο είναι εφικτό να προστεθούν διαφορετικοί εξισορροπιστές φόρτου. Συνήθως ο εξισορροπιστής εκτελείται ως υπηρεσία στο κόμβο του δικτύου.

Η σημαντικότητα ύπαρξης ενός τέτοιου συστήματος σε κάθε κέντρο δεδομένων είναι αδιαμφισβήτητη αφού προσφέρει μεγάλη δυνατότητα κλιμάκωσης και προσβασιμότητας.

3.3.1. Επισκόπηση

Η υπηρεσία αυτή δίνει την δυνατότητα σε χρήστες να ισομερίζουν την κίνηση στα μηχανήματα που βρίσκονται μετά τον εξισορροπιστή φόρτου, υποστηρίζει πολλά διαδικτυακά πρωτόκολλα, δυνατότητα παραμένουσας συνόδου (session persistence), την δυνατότητα να ελέγχει την κατάσταση των μελών τα οποία εποπτεύει και στα οποία μοιράζει τον φόρτο καθώς και να τα προστατεύει από επιθέσεις άρνησης υπηρεσίας(DoS) και να συλλέγει στατιστικά για το δίκτυο.

Επιπρόσθετα για να εφαρμοσθεί αυτή η υπηρεσία σε λειτουργικό επίπεδο χρειάζεται να ορισθούν επιπλέον παράμετροι που καθορίζουν τα μέλη υπό την εποπτεία του εξισορροπιστή, τον τρόπο αξιολόγησης της κατάστασης τους καθώς και επιπλέον δεδομένα που χρειάζονται για να λειτουργήσει σωστά ο εξισορροπιστής. Ακολουθεί περιγραφή των συγκεκριμένων παραμέτρων [32].

VIP

Το VIP είναι το πιο βασικό συστατικό παραμετροποίησης του εξισορροπιστή και καθορίζει την θύρα στην οποία όλη η κίνηση από τους πελάτες (Clients) θα συσσωρευτεί καθώς και επιπλέον πληροφορίες σχετικά με το ποια μέθοδος θα χρησιμοποιηθεί, ποιο πρωτόκολλο κ.ά. Είναι συνήθως γνωστό στους εξισορροπιστές φόρτου με την ονομασία “virtual server ”, “vserver” ή “listener”.

Pool

Μία δεξαμενή διευθύνσεων(pool) ενός εξισορροπιστή φόρτου είναι ενα σύνολο απο εξυπηρετητές οι οποίοι συγκροτούνται μαζί με σκοπό να δέχονται και να εξυπηρετούν την κίνηση στο δίκτυο, δηλαδή το σύνολο των αιτημάτων. Ο αλγόριθμος κατανομής των αιτήσεων που ορίστηκε είναι υπεύθυνος για την προώθηση ενός νέου αιτήματος που καταφθάνει στο VIP, στο σωστό μέλος. Μπορεί να υπάρχει μόνο μια δεξαμενή διευθύνσεων για κάθε VIP.

Pool Member

Ένα μέλος της δεξαμενής διευεθύνσεων αντιπροσωπεύει έναν εξυπηρετητή. Ωστόσο το μέλος αυτό μπορεί να καθοριστεί από την διεύθυνση δικτύου που του έχει ανατεθεί είτε απο το όνομά του.

Health monitor

Χρησιμοποιείται για να διαπιστωθεί από τον εξισορροπιστή φόρτου ποια από τα μέλη είναι σε κατάσταση που μπορούν να εξυπηρετήσουν τις εισερχόμενες αιτήσεις. Κάθε ομάδα μπορεί να έχει αρκετούς ελεγκτές ορθής λειτουργίας(Health Monitors) να επιβλέπουν την κατάσταση των εξυπηρετητών. Τα είδη των ελεγκτών ορθής λειτουργίας ποικίλλουν ανάλογα με το πρωτόκολλο που χρησιμοποιούν για να επικοινωνήσουν με τους εξυπηρετητές περιμένοντας να λάβουν απάντηση αν είναι σε κατάσταση λειτουργίας:

- PING: επικοινωνεί χρησιμοποιώντας το πρωτόκολλο ICMP
- TCP: χρησιμοποιεί το πρωτόκολλο TCP.
- HTTP: στέλνει αίτημα τύπου HTTP.
- HTTPS: Στέλνει HTTPS αίτημα.

Όταν μια δεξαμενή διευθύνσεων έχει πολλούς ελεγκτές ορθής λειτουργίας που παρακολουθούν την κατάσταση των μελών, κάθε μέλος παρακολουθείται από όλους τους ελεγκτές. Στην περίπτωση που έστω και ένας αποφασίσει ότι ένας συγκεκριμένος εξυπηρετητής δεν είναι σε κατάσταση να επεξεργαστεί αιτήματα, τότε η κατάσταση του θα αλλάξει σε ανενεργή και θα απομακρυνθεί από το σύνολο των εξυπηρετητών με αποτέλεσμα να μην δέχεται αιτήματα. Επομένως πρέπει όλοι οι ελεγκτές να συμφωνήσουν για κάθε εξυπηρετητή ότι είναι σε ενεργή κατάσταση και μπορεί να δέχεται αιτήματα.

Session Persistence

Η υπηρεσία αυτή είναι υπεύθυνη για την διατήρηση των συνόδων των πελατών. Οποιαδήποτε συνεδρία οι πελάτες έχουν με κάποιον εξυπηρετητή, όσο αυτή η συνεδρία είναι ενεργή, οι αιτήσεις των πελατών θα εξυπηρετούνται από τον αρχικό εξυπηρετητή με τον οποίο ήρθε σε πρώτη επαφή ο πελάτης. Διακρίνεται σε τρείς υποκατηγορίες:

- SOURCE_IP: Με αυτή την ρύθμιση όλες οι συνδέσεις που προέρχονται από την ίδια διεύθυνση δικτύου θα εξυπηρετούνται πάντα από το ίδιο μέλος.

- HTTP_COOKIE: Σε αυτή τη λειτουργία ο εξισορροπιστής φόρτου δημιουργεί ενα cookie στην πρώτη σύνδεση με τον πελάτη. Επόμενες αιτήσεις οι οποίες έχουν το ίδιο cookie θα εξυπηρετούνται από το ίδιο μέλος.

- APP_COOKIE: Με αυτή την λειτουργία ο εξισορροπιστής φόρτου θα χρησιμοποιεί cookies τα οποία έχουν δημιουργηθεί από το κεντρικό σύστημα (backend) της εφαρμογής.

Connection Limits

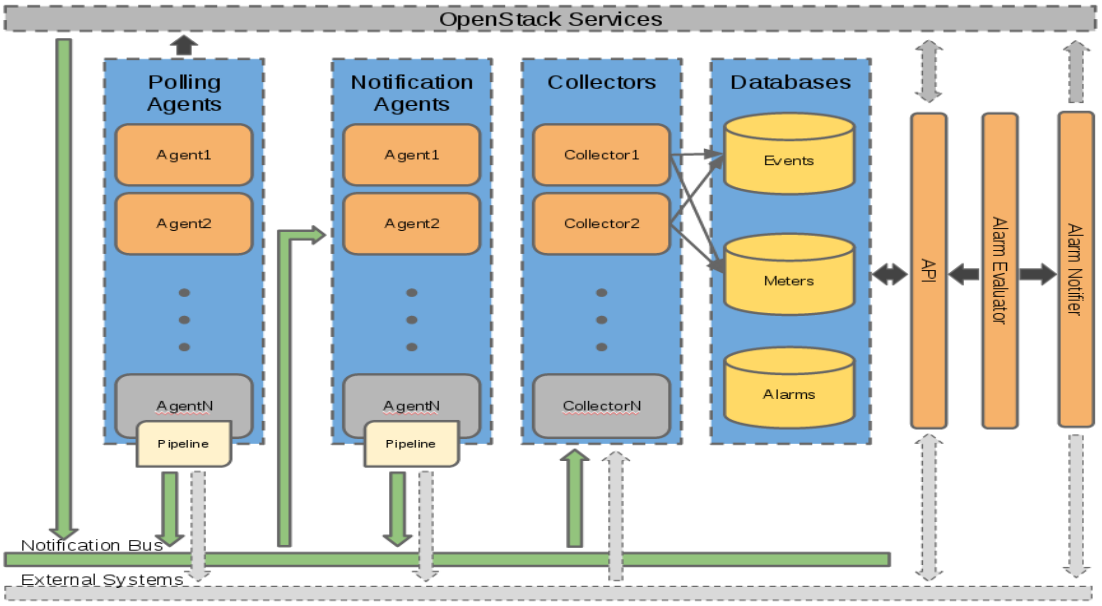
Η ρύθμιση αυτή προσφέρει την δυνατότητα καθορισμού ενός ανώτατου κατωφλίου αιτήσεων που μπορούν να είναι ενεργές συνολικά. Σε περίπτωση που οι συνδέσεις ξεπεράσουν τον αριθμό αυτό ο εξυπηρετητής φόρτου θα απορρίπτει επόμενες συνδέσεις. Σε περίπτωση επίθεσης από DoS (Denial of Service) η λειτουργία εξασφαλίζει ότι τα μέλη δεν θα υπερβούν το όριο εξυπηρέτησης τους [33].

4. Αναλυτική περιγραφή της αρχιτεκτονικής του Ceilometer

4.1. Περιγραφή

Το ceilometer αποτελεί την υπηρεσία του OpenStack που προσφέρει δυνατότητες συλλογής στατιστικών για τους πόρους του συστήματος. Όπως όλες οι υπηρεσίες του OpenStack αποτελείται από ένα σύνολο διεργασιών. Οι διεργασίες του αυτές είναι σχεδιασμένες έτσι ώστε να μπορούν να κλιμακωθούν, επομένως επιπρόσθετοι κόμβοι μπορούν να προστεθούν ανάλογα με τον προσδοκώμενο φόρτο και το σύστημα για το οποίο η υπηρεσία αυτή καλείται να συλλέξει στατιστικά. Οι βασικές διεργασίες του Ceilometer έχουν δημιουργηθεί με τέτοιο τρόπο που μπορούν να λειτουργούν αυτόνομα αλλά και όλοι μαζί προσφέροντας μια ολοκληρωμένη λύση [34]:

- **Πράκτορας Ανίχνευσης(Polling agent):** Ο πράκτορας(agent) αυτός είναι υπεύθυνος για την καταγραφή των υπηρεσιών του OpenStack και για την δημιουργία μετρητών.
- **Πράκτορας Ειδοποιήσεων(Notification Agent):** Πράκτορας ο οποίος είναι σχεδιασμένος με σκοπό να δέχεται ειδοποιήσεις στην ουρά μηνυμάτων και να τα αποθηκεύει ως Events.
- **Collector:** Πράκτορας υπεύθυνος για την συλλογή και την καταγραφή event και δεδομένα μετρήσεων που έχουν δημιουργηθεί από τους notification και polling πράκτορες.
- **Application Programming Interface (API):** Προγραμματιστική διεπαφή σχεδιασμένη για την δημιουργία επερωτημάτων στα δεδομένα που έχει συλλέξει ο collector πράκτορας.
- **Alarming:** Ο πράκτορας αυτος αξιολογεί και ειδοποιεί ανάλογα με τους καθορισμένους συναγερμούς (alarms) [35] [36].



Εικόνα 5 - Αρχιτεκτονική Ceilometer

4.2. Συλλογή Δεδομένων

Η υπηρεσία του Ceilometer χρησιμοποιεί μια βιβλιοθήκη αποστολής μηνυμάτων την Oslo messaging library που παρέχει δυο διεπαφές, την διεπαφή υπεύθυνη για την αποστολή απομακρυσμένων εντολών μεταξύ πελάτη – εξυπηρετητή και την διεπαφή που αφορά την διαχείριση γεγονότων.

Η συλλογή δεδομένων της υπηρεσίας Ceilometer βασίζεται σε δύο τρόπους συλλογής δεδομένων:

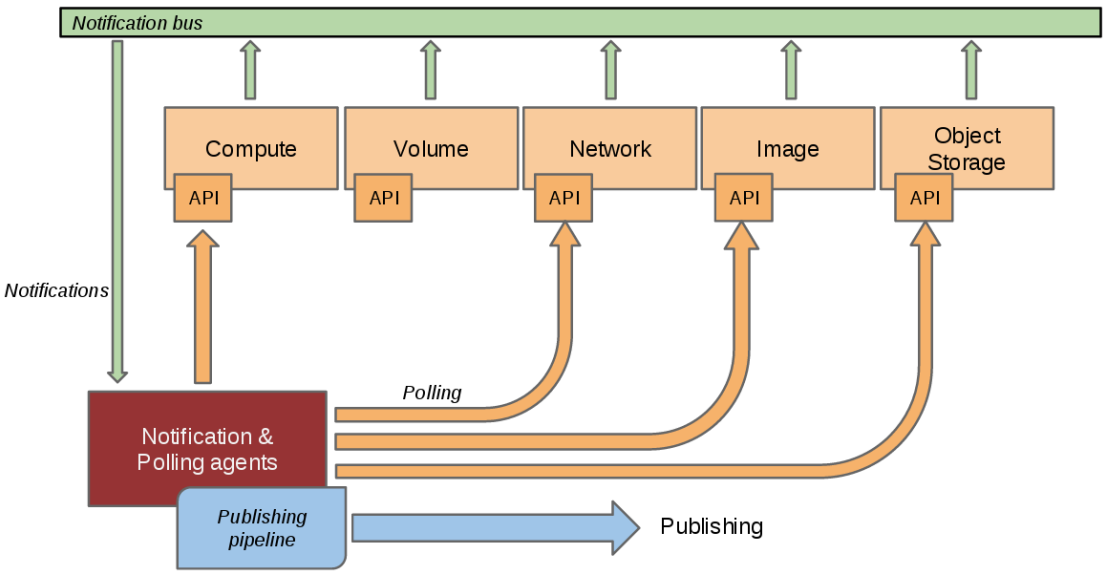
Bus listener agent

Συλλέγει όλα τα γεγονότα που έχουν αποθηκευτεί στο δίαυλο που δέχεται ειδοποιήσεις και τα μετατρέπει σε δεδομένα που χρησιμοποιεί το Ceilometer. Αποτελεί τον βασικό τρόπο συλλογής δεδομένων.

Polling agent

Η μέθοδος αυτή για να συλλέξει τα δεδομένα θα δημιουργήσει επερωτήματα στις προγραμματιστικές διεπαφές και θα συλλέξει δεδομένα ανά ορισμένα χρονικά διαστήματα. Αποφεύγεται να χρησιμοποιείται αφού είναι δυνατόν να προκαλέσει σημαντικό φόρτο στις προγραμματιστικές διεπαφές (APIs).

Η πρώτη μέθοδος χρησιμοποιείται από την διεργασία ceilometer-notification. Η δεύτερη μέθοδος μπορεί να χρησιμοποιηθεί είτε για να συλλέγει στατιστικά από τον Υπερεπόπτη(Hypervisor) είτε απο απομακρυσμένα API [37].



Εικόνα 6 - Συλλογή δεδομένων απο πράκτορες ειδοποίησεων και ανίχνευσης

4.2.1. Πράκτορας Ειδοποιήσεων – Ακρόαση Δεδομένων

Ο πράκτορας ειδοποιήσεων (notification agent) ειναι η πιο βασική διεργασία του ceilometer, η οποία είναι σχεδιασμένη να αναμένει να λάβει κάποια ειδοποίηση στο δίαυλο επικοινωνίας για δεδομένα που έχουν καταφτάσει απο υπόλοιπες υπηρεσίες του OpenStack όπως η Nova, Glance, Cinder, Neutron, Swift, Keystone και Heat.
Ο δαίμονας που φροντίζει για τις ενημερώσεις φορτώνει μία η περισσότερες πρόσθετες υπηρεσίες ακρόασης χρησιμοποιώντας τον χώρο ονομάτων (namespace) `ceilometer.notification`. Κάθε υπηρεσία ακρόασης παρακολουθεί οποιαδήποτε ειδοποίηση ωστόσο είναι προεπιλεγμένη να ακούει στο `notifications.info`. Οι υπηρεσίες αυτές περιμένουν κάποιο μήνυμα να καταφθάσει και τα προωθούν στην υπεύθυνη υπηρεσία με σκοπό να υποστούν επεξεργασία και να μετατραπούν σε συμβάντα (events) ή δείγματα (samples).
Οι πρόσθετες υπηρεσίες που είναι υπεύθυνες για τα συμβάντα προσφέρουν μια μέθοδο για την κατηγοριοποίηση των τύπων συμβάντων, για τα οποία είναι υπεύθυνες, και μια συνάρτηση επιστροφής κλήσης (callback function), για την επεξεργασία των μηνυμάτων. Τα εισερχόμενα μηνύματα αξιολογούνται και φιλτράρονται με βάση την τύπο συμβάντος που περιλαμβάνουν προτού να χρησιμοποιηθούν στην συνάρτηση επιστροφής κλήσης με σκοπό οι αρμόδιες υπηρεσίες να δέχονται μόνο μηνύματα για τα οποία έχουν εκδηλώσει ενδιαφέρον.

Αντίστοιχα ενημερώσεις μετατρέπονται σε συμβάντα τα οποία κατηγοριοποιούνται και φιλτράρονται με βάση το τύπο συμβάντος(event type) που έχει ορισθεί από υπηρεσίες στο σύστημα [38].

4.2.2. Πράκτορας Ανίχνευσης – Συλλογή Δεδομένων

Την ανίχνευση των υπολογιστικών πόρων διαχειρίζονται οι πράκτορες ανίχνευσης οι οποίοι εκτελούνται στον υπολογιστικό κόμβο (compute node) -διότι η επικοινωνία με τον Υπερεπόπτη (Hypervisor) είναι πιο αποτελεσματική- και ονομάζονται υπολογιστικοί πράκτορες. Την ανίχνευση για δεδομένα μη υπολογιστικά την χειρίζονται πράκτορες μέσω μιας προγραμματιστικής διεπαφής που εκτελείται στον κόμβο-ελεγκτή (controller node). Σε μία υλοποίηση ενός Νέφους με έναν μόνο κόμβο ένας μόνο πράκτορας είναι ικανός να εκπληρώσει και τους δύο παραπάνω ρόλους. Αντίστροφα πολλά στιγμιότυπα ενός πράκτορα μπορούν να χρησιμοποιηθούν και ο φόρτος εργασίας τους θα ισομοιραστεί μεταξύ αυτών. Η διεργασία του πράκτορα ανίχνευσης είναι προεπιλεγμένη να εκτελείται σε ένα η περισσότερα pollster plugins χρησιμοποιώντας τους χώρους ονομάτων `ceilometer.poll.compute` ή / και `ceilometer.poll.central`.

Οι πράκτορες ανα τακτά χρονικά διαστήματα δημιουργούν επερωτήματα σε κάθε pollster με σκοπό να συλλέξουν δείγματα(Samples). Το framework των πρακτόρων (agent framework) έπειτα προωθεί τα δεδομένα στον υπεύθυνο αγωγό(pipeline) για επεξεργασία [39].

4.3. Επεξεργασία Δεδομένων

Το ceilometer προσφέρει την δυνατότητα συλλογής δεδομένων με τους τρόπους που αναφέραμε παραπάνω, καθώς και την δυνατότητα επεξεργασίας και δημοσίευσης σε πολλούς αγωγούς.

Τα δεδομένα που συλλέγονται από τους πράκτορες ανίχνευσης και ειδοποίησης περιέχουν συσσωρευμένη πληροφορία και αν συνδυαστούν με προηγούμενα χρονικά δεδομένα είναι βέβαιο ότι θα προκύψει ακόμα μεγαλύτερο πλήθος δεδομένων και πληροφορίας. Η υπηρεσία ceilometer προσφέρει διάφορους μετατροπείς(transformers) οι οποίοι χρησιμοποιούνται για να χειριστούν δεδομένα μέσα σε διασωλήνωση(pipeline).

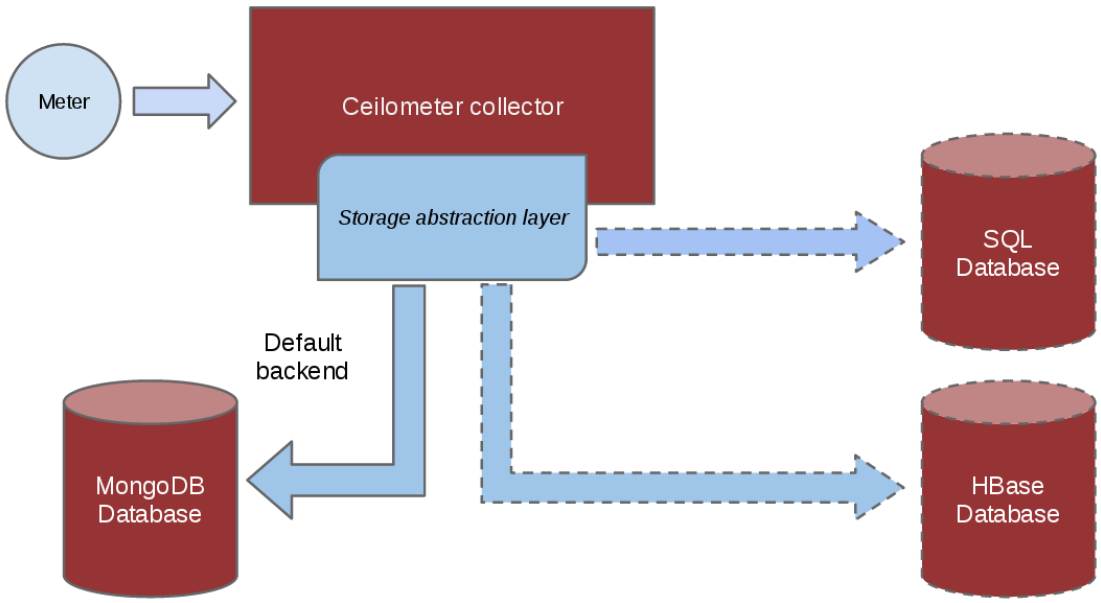
Η δημοσίευση των δεδομένων μπορεί να υλοποιηθεί με διάφορους τρόπους. Οι εκδότες(publishers) είναι ένας notifier, που χρησιμοποιεί ειδοποιήσεις και προωθεί δείγματα(samples) στην ουρά μηνυμάτων, ενας σύγχρονος εκδότης υπεύθυνος για κλήσεις απομακρυσμένων διαδικασιών, ένας UDP εκδότης που χρησιμοποιεί πακέτα UDP για την δημοσίευση και ο εκδότης Kafka, ο οποίος δημοσιεύει τα δεδομένα στην ουρά μηνυμάτων [40].

4.4. Αποθήκευση

4.4.1. Υπηρεσία Συλλογής Δεδομένων

Η υπηρεσία συλλογής δεδομένων συλλέγει τα συμβάντα (events), έπειτα από την επεξεργασία που έχουν υποστεί, και τις μετρήσεις δεδομένων που έχουν συλλέξει οι πράκτορες ειδοποίησης(notification agent) και ανίχνευσης(polling agent). Στη συνέχεια επικυρώνει την εγκυρότητα των εισερχομένων δεδομένων μέσω της υπογραφής(signature) που φέρουν και προωθεί ή αποθηκεύει τα δεδομένα στην υπηρεσία που έχει ορισθεί, είτε είναι μια βάση δεδομένων, ένα αρχείο ή σε http πακέτο.

Στις πιο πρόσφατες εκδόσεις (Juno και Kilo) του OpenStack η βάση δεδομένων που χρησιμοποιεί το ceilometer διαχωρίζεται σε τρείς υποκατηγορίες: συναγερμοί (alarms), συμβάντα (events) και λήψεις μετρήσεων(metering). Οι υποκατηγορίες αυτές δίνουν την δυνατότητα στους χειριστές να χρησιμοποιήσουν είτε μία βάση δεδομένων για να αποθηκεύουν τους συναγερμούς, τα συμβάντα και τις μετρήσεις, είτε να επιλέξουν να αποθηκεύουν σε κάποια άλλη βάση δεδομένων (πχ MongoDb) οποιαδήποτε από τις παραπάνω υποκατηγορίες.



Εικόνα 7 - Μοντέλο αποθήκευσης

4.4.2. Πρόσβαση σε Δεδομένα

Ανεξαρτήτως από την βάση δεδομένων που θα επιλεχθεί για την αποθήκευση των δεδομένων είναι πιθανό το σχήμα της βάσης να αλλάξει με την πάροδο του χρόνου. Για τον λόγο αυτό προσφέρεται μια εφαρμογή διασύνδεσης (Rest API) και συνίσταται η συλλογή δεδομένων από τις βάσεις να γίνεται μέσω αυτών και όχι άμεσα μέσω της βάσης δεδομένων. Το Ceilometer ενώ αποτελεί μέρος του OpenStack δεν χρησιμοποιεί ένοικους(tenants) και χρήστες(users) άμεσα όπως οι υπόλοιπες προσφερόμενες υπηρεσίες. Αντί αυτού χρησιμοποιεί ένα πεδίο “source” για να καθορίσει τους χρήστες που χρησιμοποιούν ένα δεδομένο δείγμα (sample). Ο διαχειριστής του OpenStack και οποιοσδήποτε έχει τα κατάλληλα δικαιώματα μπορεί να καθορίσει το πεδίο “source” και να δημιουργήσει παράγοντες για την συλλογή δειγμάτων από καινούριους μετρητές. Ως συνέπεια μέσω της τεχνικής αυτής μπορούν να χρησιμοποιηθούν μετρητές για εφαρμογές που χρησιμοποιούν την υποδομή του OpenStack, σε οποιοδήποτε επίπεδο (PaaS ή IaaS). Επιπρόσθετα οι χρήστες μπορούν να προωθούν τα δικά τους μηνύματα στις βάσεις χρησιμοποιώντας δικούς τους μετρητές [41].

4.5. Αξιολόγηση Δεδομένων

Η υπηρεσία του OpenStack που αφορά τους συναγερμούς(alarms), πρωτοεμφανίστηκε στην έκδοση Havana, δίνοντας την δυνατότητα σε χρήστες να καθορίσουν τους δικούς τους συναγερμούς με τη κατωφλίων(thresholds) στα συλλεγόμενα δείγματα. Ένας συναγερμός μπορεί να ορισθεί υπό την εποπτεία ενός μετρητή ή ενός συνόλου μετρητών λειτουργώντας συνδυαστικά. Για παράδειγμα μπορεί να ορισθεί ένας συναγερμός όταν η κατανάλωση της μνήμης τυχαίας προσπέλασης(RAM) φτάσει στο 80% σε ένα στιγμιότυπο δεδομένου ότι είναι ενεργό περισσότερο από δέκα λεπτά. Για την δημιουργία ενός συναγερμού

χρησιμοποιείται η προγραμματιστική διεπαφή του ceilometer μέσω του οποίου είναι δυνατόν να ορισθούν οι προϋποθέσεις ενεργοποίησης του συναγερμού και ποια είναι η απαραίτητη ενέργεια στην περίπτωση ενεργοποίησης του.

Κάθε χρήστης μπορεί να θέσει δικούς του συναγερμούς για μετρητές που αφορούν τα δικά του στιγμιότυπα και δεδομένα. Οι ενέργειας που έχουν υλοποιηθεί μέχρι στιγμής και μπορούν να εκτελεστούν από έναν συναγερμό είναι δύο:

- **Http callback:** Ο χρήστης δίνει ένα URL για να καλείται κάθε φορά που ενας συναγερμός ενεργοποιείται.
- **Log:** Αποθηκεύει τους συναγερμούς σε αρχείο και χρησιμοποιείται για εντοπισμό σφαλμάτων (debug) [42].

5. Περιγραφή του Heat (Orchestration)

Το Heat είναι η υπηρεσία του OpenStack που προσφέρει δυνατότητες ενορχήστρωσης. Η μηχανή του Heat (heat-engine) δέχεται ένα αρχείο που περιγράφει μια υποδομή και είναι σε θέση να δεσμεύσει τους πόρους που χρειάζεται και να δημιουργήσει την εν λόγω υποδομή. Το περιγραφικό αρχείο ονομάζεται περίγραμμα (template) και η υποδομή που προκύπτει ονομάζεται στοίβα(stack).

Ο σκοπός της υπηρεσίας Heat είναι η κατασκευή μιας υποδομής επικοινωνίας ανθρώπου – μηχανής με σκοπό την δημιουργία και την διαχείριση ενός Υπολογιστικού Νέφους. Το αρχείο που συντάσσει ο χρήστης για την δημιουργία της υποδομής, το περίγραμμα, είναι απολύτως κατανοητό από τον άνθρωπο καθώς περιέχει δομημένη πληροφορία, την οποία η μηχανή του Heat μετατρέπει σε κώδικα μηχανής. Η πληροφορία αυτή δεν είναι άλλη από πόρους που προσφέρονται από τις υπηρεσίες του OpenStack που έχει εγκαταστήσει ο χρήστης στο Νέφος του, όπως υπολογιστικοί(server, floating ip), δικτυακοί(router, networks) κ.ά. Επιπρόσθετα δίνεται η δυνατότητα μέσω των περιγραμματων να περιγραφούν και διασυνδέσεις μεταξύ των στοιχείων που έχουν ορισθεί, όπως για παράδειγμα η διασύνδεση ενός instance με κάποιο δίκτυο που έχει δημιουργήσει ο χρήστης και κατά συνέπεια η λήψη διεύθυνσης δικτύου από το δίκτυο αυτό. Τα περιγράμματα αυτά είναι τύπου YAML(Yaml Aint Another Markup Language), δίνονται ως είσοδο στην μηχανή του Heat και με την εκτέλεση τους παράγουν ένα σύνολο από στοιχεία τα οποία ονομάζονται στοίβα(stack).

Το Heat είναι υπεύθυνο για την διαχείριση όλου του κύκλου ζωής της στοίβας και οποιαδήποτε στιγμή ο χρήστης χρειάζεται να αλλάξει την υποδομή που έχει ορίσει αρκεί να τροποποιήσει το αρχείο-περίγραμμα και η μηχανή του Heat εφαρμόζει τις αλλαγές γνωρίζοντας ποια στοιχεία έχουν τροποποιηθεί και έχουν αλλάξει χωρίς να «ξανακατασκευάσει» την στοίβα από την αρχή. Αντίστοιχα σε περίπτωση διαγραφής της στοίβας το Heat διαγράφει και όλα τα δεδομένα που έχουν δημιουργηθεί/δεσμευθεί.

Επιπρόσθετα το Heat προσφέρει μια υπηρεσία για autoscaling, η οποία αλληλοεπιδρά με το ceilometer, δίνοντας την δυνατότητα δημιουργίας ενός επιπλέον πόρου, το οποίο ονομάζεται autoscaling group. Το autoscaling group είναι ένας πόρος που καθορίζει ένα σύνολο από μέλη, τα οποία μπορούν να ξεκινούν απο ένα μέχρι όσα ορίσει ο χρήστης, και η εισαγωγή ή διαγραφή ενός μέλους εξαρτάται απο τις συνθήκες που διέπουν αυτή την ομάδα (group). Για παράδειγμα για την εισαγωγή ενός καινούριου μέλους στην ομάδα (autoscaling group) των servers, άρα και την δημιουργία ενός καινούριου instance, ενδεχομένως να έχουμε ορίσει να έχει έναν συναγερμό, ο οποίος θα ενεργοποιηθεί σε περίπτωση που η ισχύς των επεξεργαστών των μελών κλιμακωθεί στο 70%. Αυτό θα προκαλέσει την δημιουργία ενός καινούριου server, ο οποίος θα προστεθεί στην ομάδα (autoscaling group).

Το Heat αποτελείται από τέσσερις υπηρεσίες:

- **Heat:** Αποτελεί την διασύνδεση γραμμής εντολών η οποία επικοινωνεί με την προγραμματιστική διεπαφή Heat.

- **Heat-api:** Παρέχει μια προγραμματιστική διεπαφή με την μηχανή του Heat(Heat Engine) μέσω μιας κλήσης απομακρυσμένης διαδικασίας(RPC).

- **Heat-api-cfn:** Συλλέγει δεδομένα που στέλνουν οι εικονικές μηχανές και επιβλέπει την κατάσταση της στοίβας. Επικοινωνεί με την μηχανή του Heat στέλνοντας αιτήματα.

- **Heat-engine:** Αποτελεί την πιο βασική υπηρεσία του Heat καθώς είναι υπεύθυνο για την μετάφραση των περιγραμμάτων σε στοίβες (stacks) και προσφέρει τους πόρους υπολογιστικού νέφους. Μόλις τελειώσει την διαδικασία της μετάφρασης του προτύπου παραμετροποιεί τους πόρους που έχουν δεσμευτεί ανάλογα με την περιγραφή που έχει δοθεί στο πρότυπο [43] [44].

5.1. Περιγράμματα HOT (HOT Templates)

Τα περιγράμματα(templates) περιγράφουν σε απλή γλώσσα μια υποδομή και πως συνδέονται τα δομικά της στοιχεία ακολουθώντας μια συγκεκριμένη μορφή. Οι υποστηριζόμενες μορφές που προσφέρει το OpenStack είναι το HOT(Heat Orchestration Template) και το AWS CloudFormation της amazon έτσι ώστε πολλές υποδομές που έχουν σχεδιαστεί για το Υπολογιστικό Νέφος της amazon να μπορούν να υλοποιηθούν στο OpenStack χωρίς οι χρήστες να αλλάξουν τα περιγράμματα τους. Στα πλαίσια της εργασίας αυτής χρησιμοποιούμε το HOT Template που είναι και η βασική μορφή περιγράμματος που χρησιμοποιεί το OpenStack.
Ακολουθεί αναλυτική περιγραφή του συντακτικού και των κανόνων που χρησιμοποιεί ο συγκεκριμένος τύπος περιγραμμάτων.

5.2. Δομή ενός HOT περιγράμματος

Η βασική δομή ενός περιγράμματος τύπου HOT είναι η εξής:

```
heat_template_version: 2013-05-23

description:
# a description of the template

parameter_groups:
 # a declaration of input parameter groups and order

parameters:
 # declaration of input parameters

resources:
  # declaration of template resources

outputs:
# declaration of output parameters
```

heat_template_version:
Καθορίζει την έκδοση του HOT που χρησιμοποιεί το περίγραμμα. Οι διαθέσιμες εκδόσεις είναι 2013-05-23, όπου υποστηρίζονται χαρακτηριστικά που έχουν υλοποιηθεί κυρίως μέχρι την έκδοση του Icehouse Openstack, η έκδοση 2014-10-16 (Juno) καθώς και η έκδοση 2015-04-30 (Kilo).

description:
Περιγραφή της υποδομής που δημιουργεί το περίγραμμα, των παραμέτρων που δέχεται ως ορίσματα και ό,τι πληροφορία πρέπει να παρέχει στον χρήστη.

parameter_groups:
Το τμήμα αυτό προσδιορίζει τον τρόπο που πρέπει οι παραμέτροι να ομαδοποιηθούν και την σειρά με την οποία πρέπει να δωθούν στην είσοδο. Μπορεί να παραβλεφθεί όταν δεν υπάρχουν ομαδοποιημένες παράμετροι.

parameters:
Το τμήμα αυτό προσδιορίζει ποιες παράμετροι πρέπει να δοθούν στην είσοδο κατά την δημιουργία της στοίβας. Είναι προαιρετικό σε περίπτωση που δεν υπάρχουν παράμετροι.

resources:
Το τμήμα αυτό καθορίζει την δήλωση των πόρων που θα δεσμεύσει το περίγραμμα. Κάθε περίγραμμα πρέπει να δηλώνει κατ' ελάχιστον έναν πόρο, αλλιώς το περίγραμμα δεν θα δημιουργεί κάτι κατά την εκτέλεση του.

outputs:
Καθορίζει ποιοι παράμετροι εξόδου θα είναι διαθέσιμοι στον χρήστη ως πληροφορία με το πέρας της δημιουργίας όλης της υποδομής που περιγράφει το περίγραμμα. Οι πληροφορίες αυτές μπορεί να είναι διευθύνσεις δικτύου και θύρες που έχουν δεσμευτεί ακόμα και ονόματα στιγμιότυπων. Αυτό το τμήμα είναι προαιρετικό και μπορεί να παραληφθεί σε περίπτωση που δεν απαιτείται κάποια πληροφορία στο τέλος της εκτέλεσης [45] [46].

5.2.1. Ομάδα παραμέτρων (parameters_group)

Όπως αναφέραμε η ομάδα παραμέτρων καθορίζει τον τρόπο με τον οποίο ένα σύνολο από παραμέτρους ομαδοποιείται, καθώς και την σειρά με την οποία πρέπει να δοθούν από την είσοδο. Κάθε ομάδα προσδιορίζεται από μία λίστα η οποία περιέχει τις παραμέτρους που πρέπει να ορισθούν για την ομάδα. Κάθε παράμετρος μπορεί να είναι μέλος μόνο μιας ομάδας.

```
parameter_groups:
 - label: <human-readable label of parameter group>
  description: <description of the parameter group>
   parameters:
   - <param name>
```

```
    - <param name>
```

label:
Μια επιγραφή που καθορίζει την ομάδα των παραμέτρων.

description:
Περιγραφή της ομάδας παραμέτρων.

parameters:
Η λίστα των παραμέτρων που ανήκουν σε αυτή την ομάδα.

Param name:
Όνομα της κάθε παραμέτρου [47].

5.2.2. Παράμετροι (parameters)

Το τμήμα των παραμέτρων καθορίζει την πληροφορία που εισάγει ο χρήστης για την δημιουργία της υποδομής που περιγράφει το περίγραμμα. Επι της ουσίας αποτελούν μεταβλητές που καθορίζουν πόρους του συστήματος οι οποίοι πρέπει να δεσμευτούν. Για παράδειγμα μια παράμετρος θα μπορούσε να είναι το ID που χαρακτηρίζει ένα εσωτερικό δίκτυο που έχει φτιάξει ο χρήστης και θέλει η υποδομή να το χρησιμοποιήσει ως το εσωτερικό δίκτυο πάνω στο οποίο θα βρίσκονται τα στιγμιότυπα. Με την δυνατότητα εισαγωγής παραμέτρων ο χρήστης μπορεί να φτιάξει μια υποδομή βασιζόμενος σε ήδη υπάρχοντες πόρους που έχει δημιουργήσει ο ίδιος είτε σε πόρους που προϋπήρχαν, χωρίς να είναι αναγκασμένος κάθε φορά να δημιουργεί όλη την υποδομή από την αρχή. Επομένως μέσω των παραμέτρων του δίνεται η δυνατότητα επαναχρησιμοποίησης(reusability) των περιγραμμάτων σε οποιαδήποτε υλοποιημένη υποδομή που ήδη έχει το σύστημα του.

Κάθε παράμετρος ορίζεται σε ξεχωριστό τμήμα (μπλοκ) του αρχείου με το όνομα της παραμέτρου να αποτελεί την πρώτη γραμμή και οι επιπλέον παράμετροι ορίζονται αμέσως μετά την πρώτη γραμμή.

```
parameters:
 <param name>
   type: <string | number | json | comma_delimited_list |
 boolean>
     label: <human-readable name of the parameter>
     description: <description of the parameter>
     default: <default value for parameter>
     hidden: <true | false>
     constraints:
       <parameter constraints>
```

param name:
Το όνομα της παραμέτρου.

type:
Ο τύπος της παραμέτρου. Οι τύποι που υποστηρίζονται είναι `string`, `number`, `comma_delimited_list`, `json` and `boolean`. Υποχρεωτικό πεδίο.

description:
Περιγραφή της παραμέτρου και σε τι αποσκοπεί. Προαιρετική.

default:
Προεπιλεγμένη τιμή για την παράμετρο σε περίπτωση που ο χρήστης δεν προσδιορίσει την τιμή της κατά την εκτέλεση. Προαιρετικό.

hidden:
Καθορίζει την ασφάλεια της παραμέτρου και αν θα είναι ορατή σε περίπτωση που κάποιος χρήστης ζητήσει πληροφορίες σχετικά με το περίγραμμα. Είναι προαιρετικό πεδίο και η προεπιλεγμένη του τιμή είναι false. Συνήθως χρησιμοποιείται για να αποκρύψει κωδικούς που πρέπει να δοθούν ως παράμετροι στο περίγραμμα.

constraints:
Μια λίστα από περιορισμούς που πρέπει να εφαρμοστούν, οι οποίοι εφαρμόζονται απο την μηχανή του Heat κάθε φορά που ο χρήστης δημιουργεί μια στοίβα. Σε περίπτωση που δεν τηρείται κάποιος από τους περιορισμούς η δημιουργία της στοίβας αποτυγχάνει. Το πεδίο αυτό είναι προαιρετικό. Οι δυνατοί περιορισμοί που μπορούν να επιβάλλουν οι χρήστες είναι οι εξής:

length: { min: <lower limit>, max: <upper limit> }
range: { min: <lower limit>, max: <upper limit> }
allowed_values: [<value>, <value>, ...]
allowed_pattern: <regular expression>

Επιπρόσθετα η μηχανή του Heat δίνει την δυνατότητα δημιουργίας δικών του περιορισμών [48].

Παράδειγμα:

```
parameters:
   user_name:
     type: string
     label: User Name
     description: User name to be configured for the application
     constraints:
      - length: { min: 6, max: 8 }
        description: Name must be between 6 and 8characters
      - allowed_pattern: "[A-Z]+[a-zA-Z0-9]*"
        description: Name must start with an uppercase character
   port_number:
     type: number
     label: Port Number
     description: Port number
```

5.2.3. Πόροι (Resources)

Το τμήμα των πόρων (resources) καθορίζει πόρους που πρέπει να δεσμευτούν απο το νέφος για την δημιουργία της στοίβας (στιγμιότυπα, εικονικά δίκτυα, δίσκοι αποθήκευσης). Κάθε πόρος ορίζεται σε ξεχωριστό τμήμα(μπλόκ) και συντάσσεται ως εξής:

```
resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
```

resource ID:
Μία ονομασία-κλειδί που καθορίζει τον πόρο και πρέπει να είναι μοναδική για όλους τους πόρους που έχουν δηλωθεί.

type:
Το είδος του πόρου, δηλαδή από ποια υπηρεσία προέρχεται και τι αποτελεί. Για παράδειγμα, OS::Nova::Server ή OS::Neutron::Port. Το πεδίο αυτό είναι υποχρεωτικό.

properties:
Μια λίστα με χαρακτηριστικά των πόρων προς δέσμευση. Τα χαρακτηριστικά αυτά μπορούν να προσδιοριστούν με την χρήση κάποιων εγγενών συναρτήσεων που προσφέρονται. Το τμήμα αυτό είναι προαιρετικό.

metadata:
Προσδιορισμός μετα-δεδομένων που αφορούν τους πόρους. Το πεδίο είναι προαιρετικό.

depends_on:
Εξαρτήσεις των πόρων. Προαιρετικό πεδίο.
Παράδειγμα:
depends_on: [server2, server3], όπου server2 και server3 είναι πόροι που ορίζονται μέσα στο περίγραμμα.

update_policy:
Η πολιτική που πρέπει να ακολουθηθεί σε περίπτωση που πρέπει να γίνουν πρόσθετες ενημερώσεις μετά από την δημιουργία της στοίβας.

deletion_policy:
Η πολιτική που πρέπει να ακολουθηθεί για την διαγραφή του πόρου. Ανάλογα με τον τύπο του πόρου προσφέρονται και διαφορετικές πολιτικές. Προαιρετικό πεδίο.

Παράδειγμα ενός πόρου (resource) [49]:

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: F18-x86_64-cfntools
```

5.2.4. Έξοδος (Outputs)

Το τμήμα της εξόδου(outputs) αφορά παραμέτρους και μεταβλητές που πρέπει να είναι διαθέσιμα στον χρήστη με το πέρας της δημιουργίας της στοίβας, όπως οι διευθύνσεις δικτύου των στιγμιότυπων που δημιουργήθηκαν ή URLs από εφαρμογές διαδικτύου που χρησιμοποιούν τα στιγμιότυπα για να εξυπηρετούν.

Κάθε μεταβλητή εξόδου καθορίζεται και δηλώνεται σε ξεχωριστό τμήμα (μπλοκ) του αρχείου ακολουθώντας συγκεκριμένο συντακτικό [50]:

```
outputs:
  <parameter name>:
    description: <description>
    value: <parameter value>
```

parameter name:
Το όνομα της παραμέτρου προς εκτύπωση.

description:
Περιγραφή της παραμέτρου εξόδου. Υποχρεωτικό πεδίο.

Parameter value:
Η τιμή της παραμέτρου εξόδου. Η τιμή αυτή συνήθως προέρχεται από κάποια συνάρτηση που προσφέρεται από τα περιγράμματα και την μηχανή του Heat. Οι συναρτήσεις αυτές ονομάζονται intrinsic functions(Εγγενείς συναρτήσεις) και αναλύονται παρακάτω.

Το παράδειγμα που ακολουθεί δείχνει πως η διεύθυνση δικτύου ενός στιγμιότυπου μπορεί να ορισθεί ως παράμετρος εξόδου:

```
outputs:
  instance_ip:
    description: IP address of the deployed compute instance
    value: { get_attr: [my_instance, first_address] }
```

5.3. Εγγενείς Συναρτήσεις (Intrinsic functions)

Η μηχανή του Heat προσφέρει ένα σύνολο από συναρτήσεις που μπορούν να χρησιμοποιηθούν στο περίγραμμα για να εκτελέσουν συγκεκριμένες εργασίες, όπως για παράδειγμα να επιστρέψουν την τιμή ενός χαρακτηριστικού στην διάρκεια της εκτέλεσης του περιγράμματος. Ακολουθεί περιγραφή των συναρτήσεων αυτών, του συντακτικού τους καθώς και τις λειτουργίες που επιτελούν [51].

5.3.1. get_attr

Η συνάρτηση αυτή χρησιμοποιείται για να επιστρέψει μια τιμή ενός πόρου (resource). Η τιμή αυτή αποκτάται κατά την εκτέλεση του περιγράμματος από τον αντίστοιχο πόρο που μόλις δημιουργήθηκε. Το συντακτικό της εντολής αυτής είναι το εξής:

```
get_attr:
   - <resource ID>
   - <attribute name>
   - <key/index 1> (optional)
   - <key/index 2> (optional)
   - ...
```

resource ID:
Το αναγνωριστικό του πόρου στο οποίο αναφερόμαστε, το οποίο πρέπει να έχει ορισθεί στο τμήμα των πόρων του περιγράμματος.

attribute name:
Το όνομα της παραμέτρου που αφορά τον πόρο. Αν το χαρακτηριστικό αυτό έχει κάποια πολύπλοκη δομή, έχει δηλαδή μορφή λίστας ή ζεύγος κλειδιού-τιμής, τότε μπορούν να καθοριστούν επιπρόσθετα αναγνωριστικά τα οποία διατρέχουν την δομή δεδομένων με σκοπό να αποκτηθεί η τιμή που ζητείται.

Το ακόλουθο παράδειγμα επιδεικνύει τον τρόπο χρήσης της συνάρτησης get_attr:

```
resources:

   my_instance:
     type: OS::Nova::Server
     # ...
 outputs:
   instance_ip:
     description: IP address of the deployed compute instance
     value: { get_attr: [my_instance, first_address] }
```

```
   instance_private_ip:
     description: Private IP address of the deployed compute
instance
    value: { get_attr: [my_instance, networks, private, 0] }
```

Στο παράδειγμα αυτό η δομή δεδομένων του instance_private_ip έχει την μορφή:
`{"public": ["2001:0db8:0000:0000:0000:ff00:0042:8329","1.2.3.4"], "private": ["10.0.0.1"]}`
Η δομή είναι ζεύγος κλειδιού-τιμή με την κάθε τιμή να είναι δομή λίστας. Για να επιστραφεί η private ip του my_instance που ανήκει στο δεύτερο στοιχείο κλειδιού-τιμής και είναι το πρώτο στοιχείο της λίστας δίνεται ως όρισμα στην get_attr το `[my_instance, networks, private, 0]`.

5.3.2. get_file

Η συνάρτηση get_file επιστρέφει το περιεχόμενο ενός αρχείου στο περίγραμμα. Χρησιμοποιείται συνήθως για να συμπεριλάβει αρχεία scripts. Συντάσσεται ως εξής:
`get_file:<content key>`

Το content key αποτελεί μια διεύθυνση URL (relative or absolute) που δείχνει την διαδρομή για το συγκεκριμένο αρχείο (path).

Παράδειγμα:

```
resources:
   my_instance:
     type: OS::Nova::Server
     properties:
       # general properties ...
       user_data:
         get_file: my_instance_user_data.sh

   my_other_instance:
     type: OS::Nova::Server
     properties:
       # general properties ...
       user_data:
         get_file:
http://example.com/my_other_instance_user_data.sh
```

5.3.3. get_param

Η συνάρτηση get_param χρησιμοποιείται για να επιστρέψει τιμές των παραμέτρων που έχουν ορισθεί κατά την εκτέλεση του περιγράμματος. Το συντακτικό της συνάρτησης:

```
get_param:
  - <parameter name>
  - <key/index 1> (optional)
  - <key/index 2> (optional)
  - ...
```

parameter name:

Το όνομα της παραμέτρου που πρέπει να επιστραφεί. Σε περίπτωση που η παράμετρος είναι κάποια πολύπλοκη δομή δεδομένων επιπλέον χαρακτηριστικά καθορίζονται όπως και στη συνάρτηση get_attr.

Για παράδειγμα:

```
parameters:
    instance_type:
     type: string
     label: Instance Type
     description: Instance type to be used.
   server_data:
     type: json

 resources:
   my_instance:
     type: OS::Nova::Server
     properties:
       flavor: { get_param: instance_type}
       metadata: { get_param: [ server_data, metadata ] }
       key_name: { get_param: [ server_data, keys, 0 ] }
```

Στο παράδειγμα αυτό οι παράμετροι instance_type και server_data περιέχουν τα ακόλουθα δεδομένα:

```
{"instance_type": "m1.tiny",
{"server_data": {"metadata": {"foo": "bar"},
                 "keys":["a_key","other_key"]}}}
```

5.3.3. get_resource

Η συνάρτηση get_resource χρησιμοποιείται για να αναφερθεί κατα την διάρκεια της εκτέλεσης σε κάποιον πόρο που έχει ήδη ορισθεί στο περίγραμμα.

Συντάσσεται ως εξής:

```
get_resource:<resource ID>
```

Για παράδειγμα:

```
resources:
  instance_port:
    type: OS::Neutron::Port
    properties: ...
  instance:
    type: OS::Nova::Server
    properties:
      ...
      networks:
        port: { get_resource: instance_port }
```

5.3.4. str_replace

Η συνάρτηση αυτή χρησιμοποιείται για να δημιουργήσει δυναμικά συμβολοσειρές δεχόμενο ως είσοδο ενα πρότυπο. Συνήθως χρησιμοποιείται για να περάσει σε κάποιο στιγμιότυπο ένα σύνολο εντολών που πρέπει να εκτελέσει μόλις τελειώσει η αρχικοποίηση του. Συντάσσεται ως εξης:

```
str_replace:
  template: <template string>
  params: <parameter mappings>
```

Για παράδειγμα:

```
parameters:
   DBRootPassword:
     type: string
     label: Database Password
     description: Root password for MySQL
     hidden: true

 resources:
   my_instance:
     type: OS::Nova::Server
     properties:
       # general properties ...
       user_data:
         str_replace:
           template: |
             #!/bin/bash
             echo "Hello world"
             echo "Setting MySQL root password"
             mysqladmin -u root password $db_rootpassword
```

```
            # do more things ...
          params:
            $db_rootpassword: { get_param: DBRootPassword }
```

Στο παράδειγμα αυτό παρατηρούμε ότι το περίγραμμα δέχεται ως είσοδο έναν κωδικό για την βάση δεδομένων και έπειτα στους πόρους του περιγράμματος χρησιμοποιεί την str_replace για να περάσει κάποιες εντολές τύπου bash στον πόρο (OS::NOVA::SERVER), και τον κωδικό που ζητείται στην εκτέλεση του περιγράμματος.

6. Εγκατάσταση OpenStack

Το κεφάλαιο αυτό έχει ως σκοπό την περιγραφή του τρόπου εγκατάστασης του OpenStack με τις απαραίτητες υπηρεσίες που χρειάζεται για να επιτευχθεί εξισορρόπηση φόρτου και autoscaling καθώς και της αρχιτεκτονικής που επιλέχθηκε.

6.1. Περιγραφή της υποδομής

Η εγκατάσταση του OpenStack έγινε σε έναν κόμβο(All in one deployment) σε προσωπικό υπολογιστή με τις εξής προδιαγραφές:
CPU: Intel Core i5-4440 @ 3.2 GHz(4 CPU's)
RAM: 16 GB DDR3

Το λειτουργικό σύστημα του μηχανήματος είναι Windows 7 SP1. Η υλοποίηση του OpenStack έγινε σε αυτό το μηχάνημα με σκοπό όλο το σύστημα του νέφους να βρίσκεται υπό την εποπτεία του VMWare. Η υλοποίηση αυτή δεν έχει σκοπό να μπει σε περιβάλλον παραγωγής, ωστόσο επιλέχθηκε για την δυνατότητα που προσφέρεται από τα λογισμικά εικονικοποίησης να αποθηκεύσουμε στιγμιότυπα νεός λειτουργικού συστήματος και έπειτα να δίνεται η δυνατότητα να γυρίσουμε σε μια προηγούμενη κατάσταση. Για την καλύτερη κατανόηση της αρχιτεκτονικής στη συνέχεια το λειτουργικό Windows 7 θα ονομάζεται host.

Για controller node του OpenStack επιλέχθηκε το λειτουργικό Fedora 20. Το Fedora 20 στη συνέχεια θα ονομάζεται controller και σε αυτό το λειτουργικό εγκαταστάθηκε το OpenStack με όλες τις υπηρεσίες που χρειάστηκε για την εκπόνηση της εργασίας αυτής.

Η μέθοδος εγκατάστασης που υλοποιήθηκε είναι μέσω του packstack. Πρόκειται για ένα πρόγραμμα της RedHat που χρησιμοποιώντας αρθρώματα(modules) Puppet εγκαθιστά το OpenStack σε μία μονάδα ή σε μια ολόκληρη υποδομή ανάλογα με τις παραμέτρους εγκατάστασης που δίνονται στην είσοδο του packstack.

Στο σημείο αυτό πρέπει να αναφέρουμε ότι προσφέρονται πολλοί τρόποι εγκατάστασης του OpenStack, όπως από το ίδιο το OpenStack που έχει εκτεταμένους οδηγούς εγκατάστασης ή από την Canonical που προσφέρει το MaaS(Metal-as-a-Service) και τα juju charms.

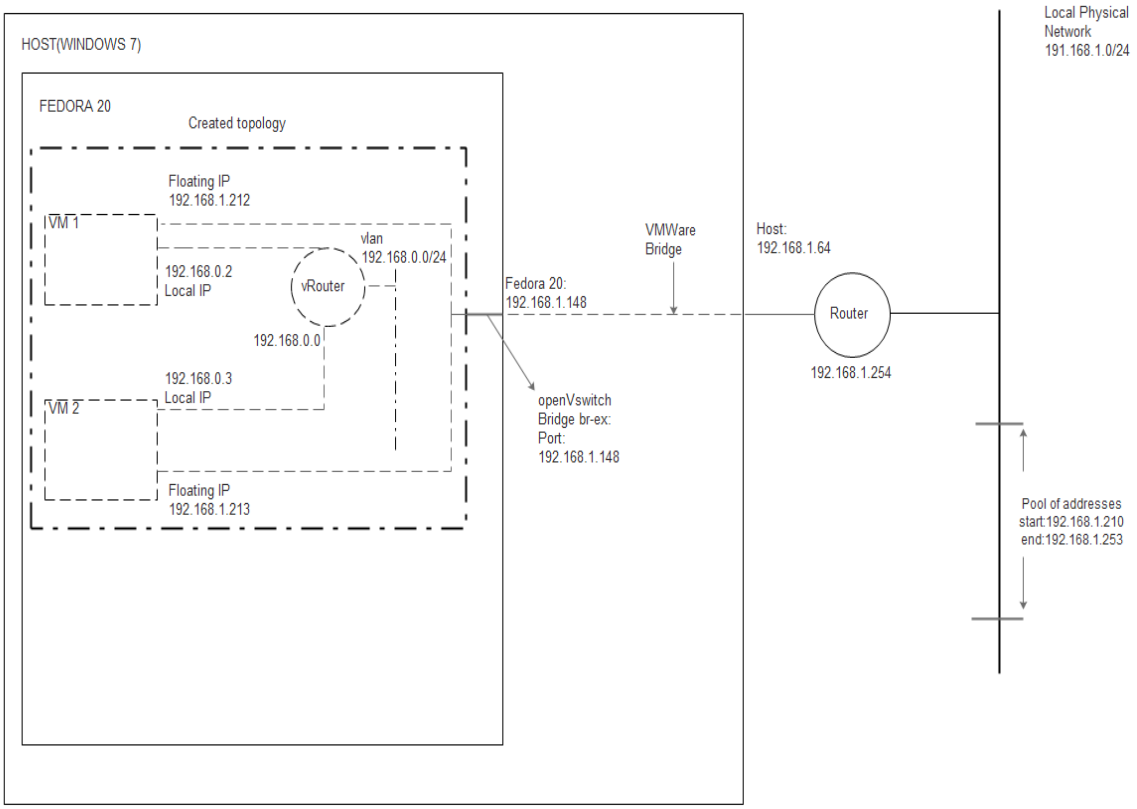
6.2. Περιγραφή της αρχιτεκτονικής

Η αρχιτεκτονική που επιλέχθηκε βασίστηκε στη διαθέσιμη υποδομή για την δημιουργία του Νέφους. Η υποδομή περιλαμβάνει ως φυσικά μέσα ένα δίκτυο (Home Network) με διευθύνσεις 192.168.1.0/24 και έναν δρομολογητή με gateway: 192.168.1.254. Επομένως το εξωτερικό δίκτυο το οποίο πρέπει να χρησιμοποιούν

οι εικονικές μηχανές είναι το Home Network και μέσω αυτού θα έχουν πρόσβαση στο Διαδίκτυο.

Το λογισμικό παρουσίασης VMWare που χρησιμοποιείται είναι υπεύθυνο για την γεφύρωση (bridging) του controller (fedora 20) με το λειτουργικό σύστημα του host(windows 7). Έπειτα από την εγκατάσταση του OpenStack στον controller χρησιμοποιείται πάλι η τεχνική της γεφύρωσης για να δρομολογείται πληροφορία από τον controller στα στιγμιότυπα που θα δημιουργούνται (OpenVSwitch). Επιπρόσθετα δίνεται στο neutron η πληροφορία οτι υπάρχει εξωτερικός δρομολογητής (Router) ο οποίος είναι υπεύθυνος για την δρομολόγηση του εξωτερικού δικτύου (Home Network), καθώς και ένα σύνολο διευθύνσεων τις οποίες μπορεί το neutron να χρησιμοποιεί για τα στιγμιότυπα.

Η αρχιτεκτονική απεικονίζεται στο παρακάτω σχεδιάγραμμα:



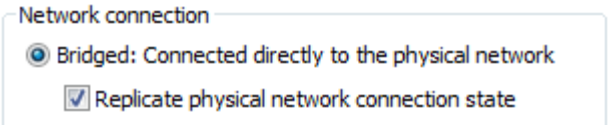
Εικόνα 8 - Αρχιτεκτονική εγκατάστασης

6.3. Διαδικασία εγκατάστασης

Περιγραφή της διαδικασίας εγκατάστασης του controller καθώς και του packstack.

6.3.1. Εγκατάσταση του controller στον host

Αρχικά για την εγκατάσταση του controller (fedora 20) είναι σημαντικό μέσω του VMWare πρέπει να κάνουμε κάποιες απαραίτητες ρυθμίσεις. Η διεύθυνση δικτύου του controller δεν πρέπει να είναι ίδια με την διεύθυνση του host. Επομένως επιλέγουμε στις ρυθμίσεις την επιλογή bridged connection (με την επιλογή Replicate physical network connection state.



Εικόνα 9 - Επιλογή bridged connection σε VMWare

Εξίσου σημαντικό είναι να έχει η μηχανή αυτή τους απαραίτητους πόρους για να μπορεί να αντέξει την υποδομή του νέφους. Επομένως επιλέχθηκε να δοθεί όλη η υπολογιστική ισχύς στον controller καθώς και αρκετή μνήμη RAM για να καλύψει τις ανάγκες(περίπου 12 GB).

Συνολικά η διαδικασία εγκατάστασης δεν διαφέρει από οποιαδήποτε εγκατάσταση λειτουργικού σε σύστημα εικονικής παρουσίασης. Το γραφικό περιβάλλον δεν είναι απαραίτητο να προστεθεί δεδομένου ότι οποιοδήποτε υπολογιστής στο δίκτυο μπορεί να έχει πρόσβαση στον controller στο dashboard που προσφέρει το OpenStack, ακόμα και ο host.

6.3.2. Προετοιμασία εγκατάστασης του Packstack

Μετά την εγκατάσταση του Fedora 20 πάνω στον controller έπεται να αλλάξουν ορισμένες ρυθμίσεις για να ξεκινήσει η διαδικασία εγκατάστασης του OpenStack μέσω του packstack.

Αρχικά ανοίγουμε ένα τερματικό και αποκτούμε προνόμια υπερχρήστη. Στη συνέχεια σταματάμε και απενεργοποιούμε τον διαχειριστή δικτύου (Network Manager), το τείχος προστασίας (Firewall) και ενεργοποιούμε το δίκτυο.

```
#sudo su
#systemctl stop NetworkManager
#systemctl disable NetworkManager
#systemctl enable network
#systemctl disable firewalld
```

Επιπρόσθετα πρέπει να αλλάξουμε την πολιτική selinux:

```
#setenforse permissive
```

Και για να παραμείνει η αλλαγή μετά από επανεκκινήσεις πρέπει να αλλάξουμε και

στο αρχείο `/etc/sysconfig/selinux` την τιμή SELINUX σε `permissive`.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Εικόνα 10 - Selinux Policy

Στη συνέχεια πρέπει να δημιουργήσουμε στατική διεύθυνση δικτύου για τον controller.Αλλάζουμε τα περιεχόμενα του αρχείου `/etc/sysconfig/network-scripts/ifcfg-<your interface name>` . Το όνομα του περιβάλλοντος διασύνδεσης στην περίπτωση μας ήταν `eno16777736`. Αλλάζουμε τις τιμές των μεταβλητών `BOOTPROTO` σε `static` , `ONBOOT` σε `yes` και συμπληρώνουμε τις πληροφορίες σχετικά με το δίκτυο μας. Στη περίπτωση μας το δίκτυο είχε διεύθυνση `192.168.1.0/24`, και η πύλη την `192.168.1.254`. Επιλέχθηκε να δοθεί στον controller η διεύθυνση `192.168.1.148`. Επομένως το αρχείο πρέπει είναι παρόμοιο με την εικόνα που ακολουθεί:

```
TYPE="Ethernet"
BOOTPROTO=static
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="eno16777736"
UUID=3b101482-8db3-45bc-b71e-f8ed7fb24e9f
ONBOOT="yes"
HWADDR=00:0C:29:7B:47:D1
IPADDR=192.168.1.148
PREFIX=24
GATEWAY=192.168.1.254
DNS1=8.8.8.8
DNS2=192.168.1.254
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
```

Στη συνέχεια πρέπει να αλλάξουμε το αρχείο /etc/hosts στο οποίο προσθέτουμε την διεύθυνση δικτύου του μηχανήματος, τον ιδιωτικό τομέα και το όνομα που έχει στο δίκτυο ο υπολογιστής.

```
127.0.0.1 localhost localhost.localdomain localhost4
localhost4.localdomain4

::1   localhost localhost.localdomain localhost6
localhost6.localdomain6

192.168.1.148 localhost localhost.localdomain
```

Η τελευταία γραμμή του αρχείου είναι και εκείνη που πρέπει να προσθέσουμε.

Επίσης αλλάζουμε και το όνομα κόμβου(hostname) σε localhost στο αρχείο `/etc/hostname`.
Τέλος προχωράμε σε επανεκκίνηση του controller [52].

6.3.3. Εγκατάσταση του Packstack

Για την εγκατάσταση του packstack εκτελούμε τις εξής εντολές αφού πρώτα αποκτήσουμε δικαιώματα υπερχρήστη:

```
#yum update -y
#yum install -y http://rdo.fedorapeople.org/rdo-release.rpm
#yum install -y openstack-packstack
#packstack --allinone --provision-all-in-one-ovs-bridge=n\
 --os-heat-install=y  --os-neutron-lbaas-install=y\
–nagios-install=n --os-cinder-install=n\
--os-heat-cfn-install=y
```

Στη συνέχεια για να εγκαταστήσουμε το OpenStack πρέπει να εκτελέσουμε το packstack με κάποιες παραμέτρους:

Στο σημείο αυτό παραμετροποιούμε την εγκατάσταση του packstack δηλώνοντας ότι δεν επιθυμούμε να επιχειρήσει να παραμετροποιήσει την γέφυρα δικτύου OpenVSwitch, να μην εγκαταστήσει την υπηρεσία cinder και nagios και να εγκαταστήσει την πρόσθετη υπηρεσία του εξισορροπιστή φόρτου, την υπηρεσία Heat και την πρόσθετη υπηρεσία του heat-cfn η οποία είναι υπεύθυνη για την διαχείριση των συναγερμών.Η εγκατάσταση του packstack διαρκεί αρκετή ώρα και ενδεχομένως να παρουσιάσει προβλήματα, τα οποία ωστόσο εμφανίζει και μπορεί να γίνει εντοπισμός σφαλμάτων. Σε περίπτωση που η εγκατάσταση αποτύχει δεν πρέπει να εκτελεσθεί πάλι το packstack με τις αρχικές παραμέτρους αλλά πρέπει να εκτελεστεί δίνοντας του ως είσοδο ένα αρχείο packstack-answers-*.txt το οποίο δημιουργείται με την πρώτη εκτέλεση του packstack στον φάκελο στον οποίο εκτελείται. Η εντολή για να δοθεί το αρχείο αυτό ως είσοδος στο packstack είναι:

```
#packstack  --answer-file=packstack-answers-*.txt
```

Όταν η εγκατάσταση του packstack ολοκληρωθεί επιτυχώς εμφανίζει στο τερματικό κάποιες πληροφορίες σχετικά με την εγκατάσταση. Το αρχείο που περιέχει αυτές τις πληροφορίες είναι το keystonerc_admin και περιέχει κυρίως μεταβλητές περιβάλλοντος αλλά και τον κωδικό του διαχειριστή του OpenStack. Το αρχείο αυτό καλούμαστε να το εκτελούμε (#source keystonerc_admin) κάθε φορά που θέλουμε να εκτελέσουμε λειτουργίες του OpenStack μέσω τερματικού [53] [54].

6.3.4. Παραμετροποίηση της γέφυρας (OpenVSwitch)

Με το πέρας της εγκατάστασης του packstack σειρά έχει η παραμετροποίηση της γέφυρας του OpenVSwitch. Με την εντολή ifconfig βλέπουμε οτι έχει δημιουργηθεί ένα επιπλέον interface, το br-ex. Στο σημείο αυτό πρέπει να παραμετροποιήσουμε τα αρχεία `/etc/sysconfig/network-scripts/ifcfg-br-ex` και `/etc/sysconfig/network-scripts/ifcfg-eno16777736`.

Πιο συγκεκριμένα το αρχείο `ifcfg-br-ex` πρέπει να παραμετροποιηθεί πρώτο και να έχει την μορφή:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=192.168.1.148
NETMASK=255.255.255.0
GATEWAY=192.168.1.254
DNS1=192.168.1.254
ONBOOT=yes
```

Στο σημείο αυτό το br-ex παίρνει την διευθυνση δικτύου του interface eno16777736 και στη συνέχεια θα παραμετροποιήσουμε το eno16777736 ετσι ώστε να είναι μια πορτα στο interface br-ex. Φέρνουμε το αρχείο ifcfg-eno16777736 στην εξής μορφή:

```
TYPE=OVSPort
DEVICE=eno16777736
DEVICETYPE=ovs
BOOTPROTO=none
OVS_BRIDGE=br-ex
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
NAME=eno16777736
UUID=3b101482-8db3-45bc-b71e-f8ed7fb24e9f
ONBOOT=yes
HWADDR=00:0C:29:7B:47:D1
```

```
PEERDNS=yes
PEERROUTES=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
```

Εκτελώντας `ovs-vsctl show` βλέπουμε τις δικτυακές γέφυρες που έχουμε δημιουργήσει με το OpenVSwitch:

```
Bridge br-ex
    Port br-ex
        Interface br-ex
            type: internal
    Port "qg-b4813382-e2"
        Interface "qg-b4813382-e2"
            type: internal
    Port "qg-d91bf0ff-91"
        Interface "qg-d91bf0ff-91"
            type: internal
    Port "eno16777736"
        Interface "eno16777736"
ovs_version: "2.3.1-git3282e51"
```

Όπως φαίνεται στο σημείο αυτό το eno16777736 ανήκει στο br-ex και είναι αυτό που θέλουμε να πετύχουμε.

Στη συνέχεια στο αρχείο `/etc/neutron/plugin.ini` κάνουμε τις εξής αλλαγές:

```
network_vlan_ranges = physnet1
bridge_mappings = physnet1:br-ex
```

και προχωράμε κάνοντας επανεκκίνηση της υπηρεσίας του δικτύου [54] [55] [56] [57].

6.3.5. Ρύθμιση του εξωτερικού δικτύου του OpenStack.

Το OpenStack και πιο συγκεκριμένα το Neutron χρειάζεται ένα εξωτερικό δίκτυο από το οποίο θα μπορεί να δίνει διευθύνσεις στα στιγμιότυπα και τα στιγμιότυπα μέσω αυτού θα μπορούν να συνδεθούν στο διαδίκτυο. Στη συγκεκριμένη υλοποίηση το εξωτερικό δίκτυο του OpenStack είναι το προσωπικό δίκτυο 192.168.1.0/24 με δρομολογητή ο οποίος έχει την πύλη 192.168.1.254. Στο σημείο αυτό πρέπει να παραμετροποιήσουμε το neutron έτσι ώστε να δεχτεί το ήδη υπάρχον φυσικό δίκτυο, καθώς και τον δρομολογητή ως το εξωτερικό δίκτυο του. Αρχικά με την εγκατάσταση του packstack το Neutron δημιουργεί ένα εξωτερικό δίκτυο με τυχαίες διευθύνσεις και έναν εικονικό δρομολογητή για το δίκτυο αυτό. Ο εικονικός δρομολογητής πρέπει να διαγραφεί γιατί στην θέση του θα δηλώσουμε το εξωτερικό δίκτυο καθώς και τον δρομολογητή που είναι υπεύθυνος για το δίκτυο.

Στη συνέχεια καθορίζουμε ένα σύνολο διευθύνσεων τις οποίες μπορεί να χρησιμοποιεί το Neutron για να δίνει διευθύνσεις δικτύου (Floating IP's) στα στιγμιότυπα. Αρχικά εκτελούμε το αρχείο keystonerc_admin και στη συνέχεια:

```
#neutron router-gateway-clear router1
#neutron subnet-delete public_subnet
#neutron router-delete router1
#neutron net-create public --shared --router:external=True
```

Στο σημείο αυτό έχουμε αφαιρέσει τον εικονικό δρομολογητή και το συσχετιζόμενο δίκτυο με αυτό και δημιουργήσαμε ένα καινούριο εξωτερικό δίκτυο με την επιλογή ότι έχει εξωτερικό δρομολογητή. Τώρα θα πρέπει να καθορίσουμε το υποδίκτυο και το εύρος διευθύνσεων για το υποδίκτυο του εξωτερικού δικτύου. Υπενθυμίζω οτι το εξωτερικό μας δίκτυο έχει τις διευθύνσεις 192.168.1.0/24 και πύλη δρομολογητή 192.168.1.254. Έτσι έχουμε:

```
#neutron subnet-create public --name public_subnet \
  --allocation-pool start=192.168.1.210,end=192.168.1.253\
  --disable-dhcp --gateway 192.168.1.254 192.168.1.0/24
```

Ολοκληρώνοντας τη διαδικασία αυτή το OpenStack θεωρεί ως εξωτερικό το home network και οποιαδήποτε διεύθυνση δοθεί σε κάποιο στιγμιότυπο αυτό μπορεί να έχει πρόσβαση στο διαδίκτυο [53] [58].

6.3.6. Παραμετροποίηση του Ceilometer

Έπειτα προχωράμε στην παραμετροποίηση του Ceilometer για τους συναγερμούς. Συγκεκριμένα το ceilometer έχει ένα αρχείο παραμετροποίησης στο οποίο ορίζει τα χρονικά διαστήματα ανάμεσα στα οποία συλλέγει δεδομένα από τους συναγερμούς. Στο αρχείο `/etc/ceilometer/pipeline.yaml` αλλάζουμε τα interval σε 60.

Στη συνέχεια κάνουμε επανεκκίνηση των υπηρεσιών του ceilometer:

```
#service openstack-ceilometer-api restart;
#service openstack-ceilometer-alarm-evaluator restart;
#service openstack-ceilometer-alarm-notifier restart;
#service openstack-ceilometer-central restart;
#service openstack-ceilometer-collector restart;
#service openstack-ceilometer-compute restart;
#service openstack-ceilometer-notification restart;
```

6.3.7. Εγκατάσταση του Ubuntu 14.04 στην υπηρεσία Glance

Για την εγκατάσταση του Ubuntu 14.04 στην υπηρεσία Glance πρέπει να εκτελέσουμε την εντολή wget και να κατεβάσουμε από ένα repository την εικόνα του λειτουργικού που θέλουμε να έχουμε διαθέσιμο στο Glance. Επομένως:

```
#wget http://cloud-
images.ubuntu.com/releases/14.04/release/ubuntu-14.04-server-
cloudimg-amd64-disk1.img
```

Και στη συνέχεια πρέπει να προσθέσουμε την εικόνα στην υπηρεσία του glance με το κατάλληλη μορφή έτσι ώστε να μπορεί να το χρησιμοποιήσει το nova για να δημιουργήσει στιγμιότυπα [59].

```
glance image-create --name="Ubuntu 14.04" --disk-format=qcow2  -
-container-format=bare  --is-public=true  <  ubuntu-14.04-server-
cloudimg-amd64-disk1.img
```

Η εγκατάσταση και παραμετροποίηση του OpenStack έχει ολοκληρωθεί. Μπορούμε να δούμε την κατάσταση των υπηρεσιών εκτελώντας σε τερματικό

```
[root@localhost ~(keystone_admin)]# openstack-status
== Nova services ==
openstack-nova-api:                     active
openstack-nova-cert:                    active
openstack-nova-compute:                 active
openstack-nova-network:              inactive  (disabled on boot)
openstack-nova-scheduler:               active
openstack-nova-conductor:               active
== Glance services ==
openstack-glance-api:                   active
openstack-glance-registry:              active
== Keystone service ==
openstack-keystone:                  inactive  (disabled on boot)
== Horizon service ==
openstack-dashboard:                    active
== neutron services ==
neutron-server:                         active
neutron-dhcp-agent:                     active
neutron-l3-agent:                       active
neutron-metadata-agent:                 active
neutron-lbaas-agent:                    active
neutron-openvswitch-agent:              active
== Swift services ==
openstack-swift-proxy:                  active
openstack-swift-account:                active
openstack-swift-container:              active
openstack-swift-object:                 active
== Ceilometer services ==
openstack-ceilometer-api:               active
openstack-ceilometer-central:           active
openstack-ceilometer-compute:           active
```

```
openstack-ceilometer-collector:          active
openstack-ceilometer-alarm-notifier:     active
openstack-ceilometer-alarm-evaluator:    active
openstack-ceilometer-notification:       active
== Heat services ==
openstack-heat-api:                      active
openstack-heat-api-cfn:                  active
openstack-heat-api-cloudwatch:     inactive  (disabled on boot)
== Support services ==
libvirtd:                                active
openvswitch:                             active
dbus:                                    active
rabbitmq-server:                         active
memcached:                               active
```

Σε περίπτωση που το `openstack-ceilometer-api` βρίσκεται σε κατάσταση failed αρκεί να επανεκκινήσουμε τον δαίμονα της mongodb και το `ceilometer-api`.

```
#service mongod start
#service openstack-ceilometer-api restart
```

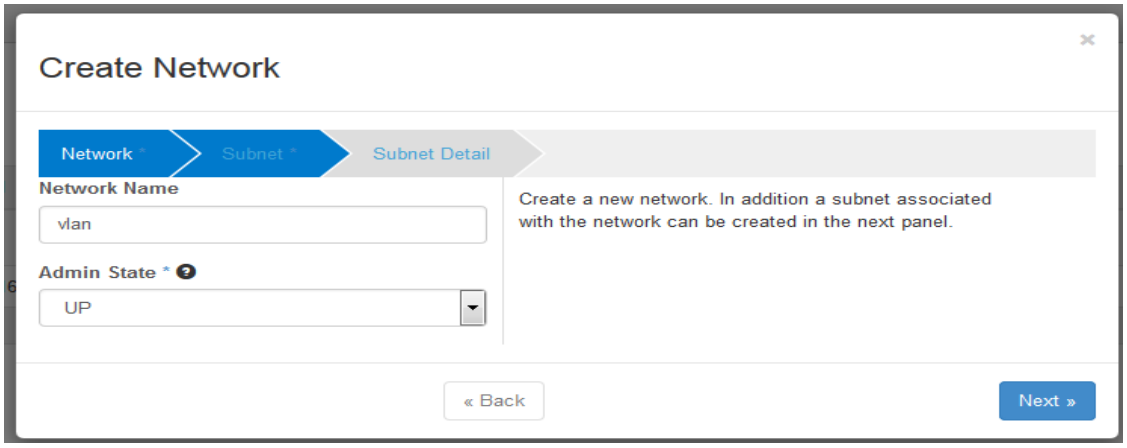
7. Διαμόρφωση της υποδομής σε OpenStack

Στο κεφάλαιο αυτό αναλύουμε την υποδομή που χρειάζεται να δημιουργηθεί, τα όποια εργαλεία και υπηρεσίες του OpenStack χρησιμοποιούνται για να πετύχουμε τον σκοπό του autoscaling.Στην υλοποίηση αυτή χρησιμοποιήθηκε το γραφικό περιβάλλον του Horizon για την δημιουργία της υποδομής, ωστόσο είναι δυνατόν να χρησιμοποιηθούν και οι command clients που προσφέρει το OpenStack. Ενώ και τα δύο μπορούν να φτάσουν στο ίδιο αποτέλεσμα επιλέξαμε να ασχοληθούμε με το Horizon για να είναι πιο κατανοητή και ευκολομνημόνευτη η διαδικασία. Το Horizon δέχεται αιτήσεις στην διεύθυνση δικτύου του controller, επομένως ανεξάρτητα αν ο controller έχει γραφικό περιβάλλον μπορούμε να συνδεθούμε στο Horizon από έναν υπολογιστή του δικτύου και μέσω ενός περιηγητή να αποκτήσουμε πρόσβαση.

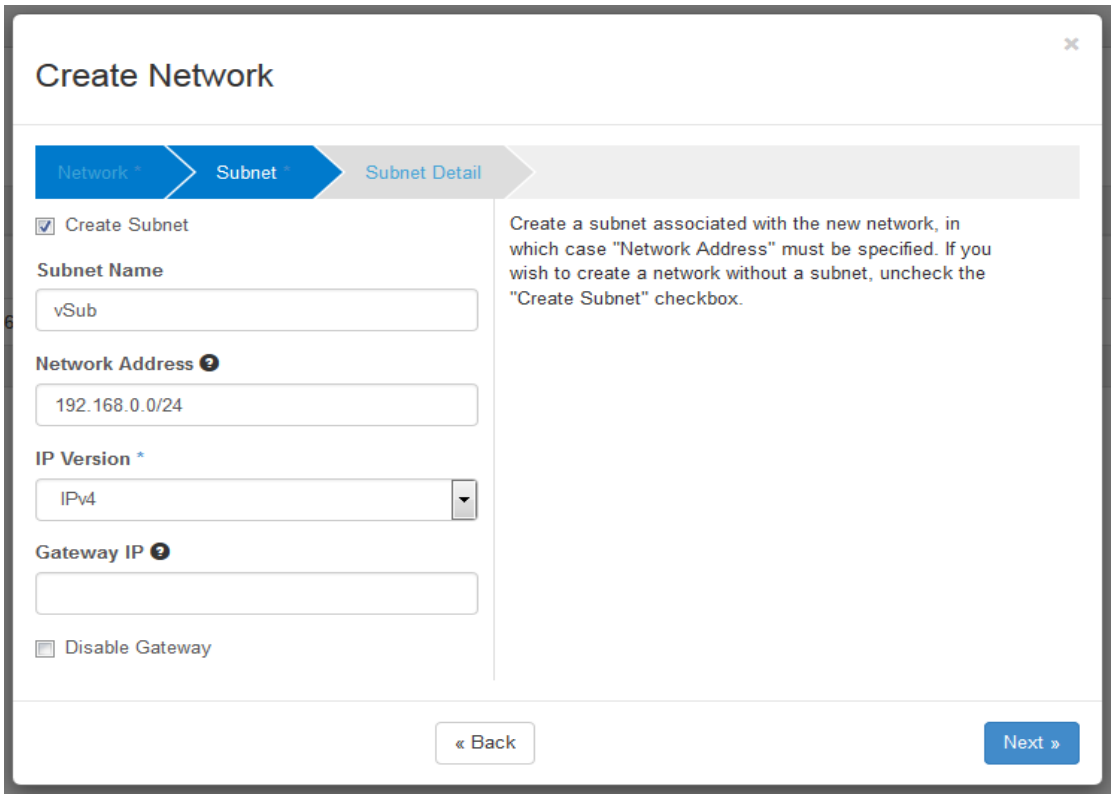
Επομένως ανοίγουμε έναν περιηγητή στην διεύθυνση που έχει ο controller είτε σε κάποιο άλλο μηχάνημα που ανήκει στο ίδιο εξωτερικό δίκτυο, και πληκτρολογούμε την διεύθυνση http://<controller_IP>/dashboard, όπου στην περίπτωση μας η controller_IP είναι 192.168.1.148. Στην σελίδα σύνδεσης πληκτρολογούμε στο πεδίο χρήστη admin και για συνθηματικό πληκτρολογούμε τον κωδικό που βρίσκεται στο keystonerc_admin.

7.1. Δημιουργία εσωτερικού εικονικού δικτύου.

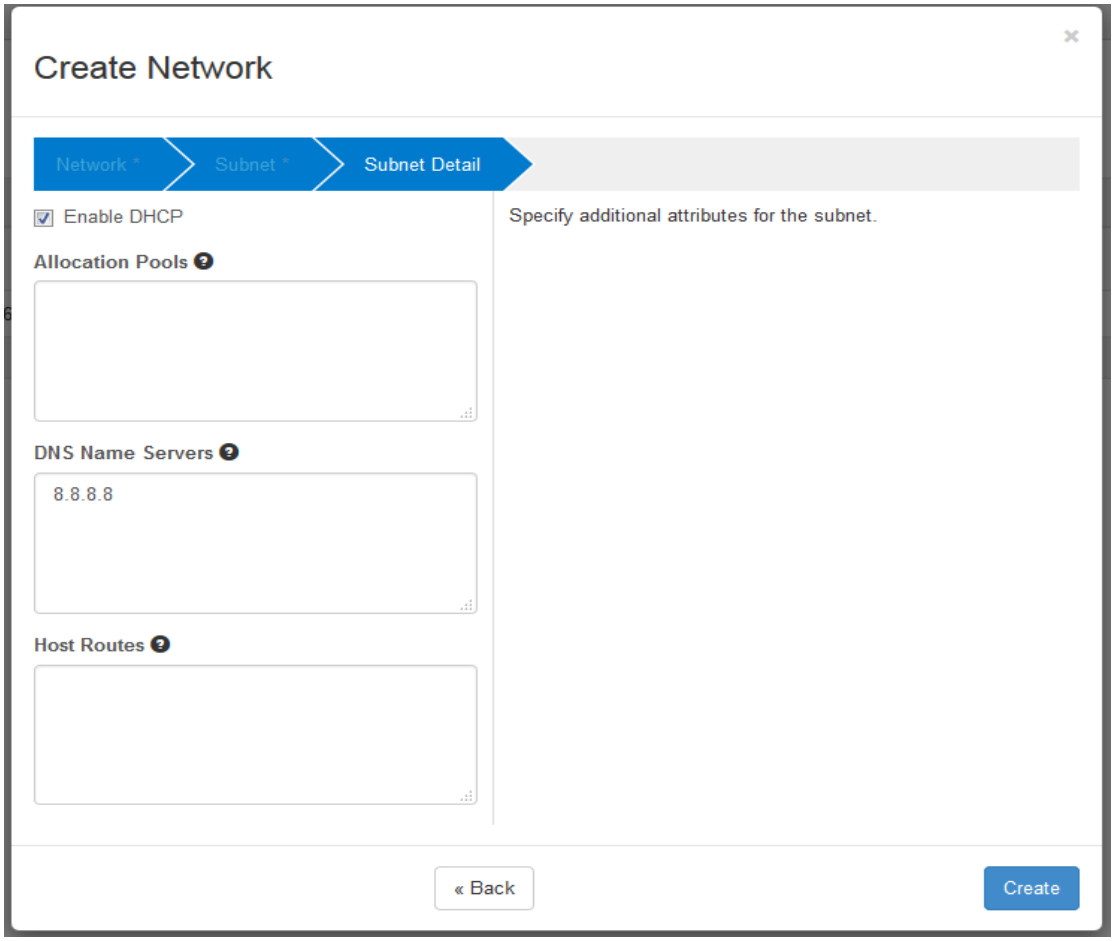
Το OpenStack χρησιμοποιεί εικονικά εσωτερικά δίκτυα για να επικοινωνούν τα στιγμιότυπα μεταξύ τους εφαρμόζοντας πρόσθετες λειτουργίες σε αυτά. Στο Horizon κατευθυνόμαστε στο Project και μετά στο Network->Networks->Create Network στην σελίδα που ανοίγει. Στο παράθυρο που ανοίγει ονομάζουμε το δίκτυο vlan και πατάμε Next(βλ. εικόνα 10). Στο επόμενο παράθυρο ονομάζουμε το υποδίκτυο vSub και για network addresses δίνουμε όποια τιμή θέλουμε, για παράδειγμα 192.168.0.0/24 (βλ. εικόνα 11). Συνεχίζουμε πατώντας next και στο επόμενο παράθυρο δίνουμε στο πεδίο DNS Name servers το 8.8.8.8 (βλ. εικόνα 12).



Εικόνα 10 - Δημιουργία εσωτερικού εικονικού δικτύου απο το Horizon

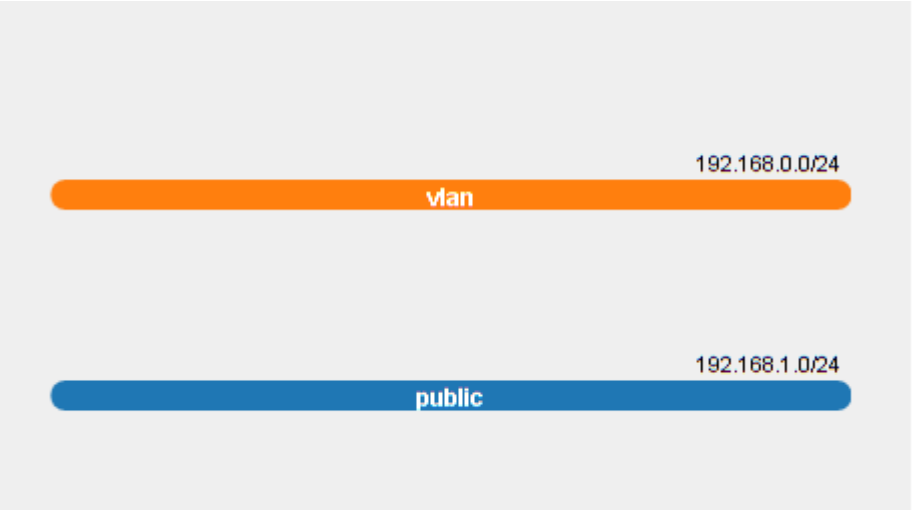


Εικόνα 11 - Δημιουργία υποδικτύου απο Horizon



Εικόνα 12 - Ρυθμίσεις Υποδικτύου

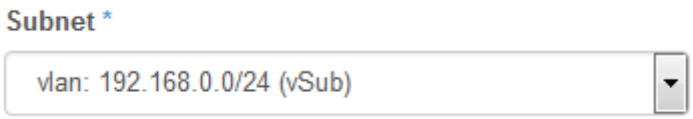
Τελειώνοντας την δημιουργία του δικτύου θα πρέπει στο Network tab->Network Topology να έχουμε αυτή την τοπολογία(βλ. εικόνα 15).



Εικόνα 13 - Τοπολογία εικονικού και δημόσιου δικτύου

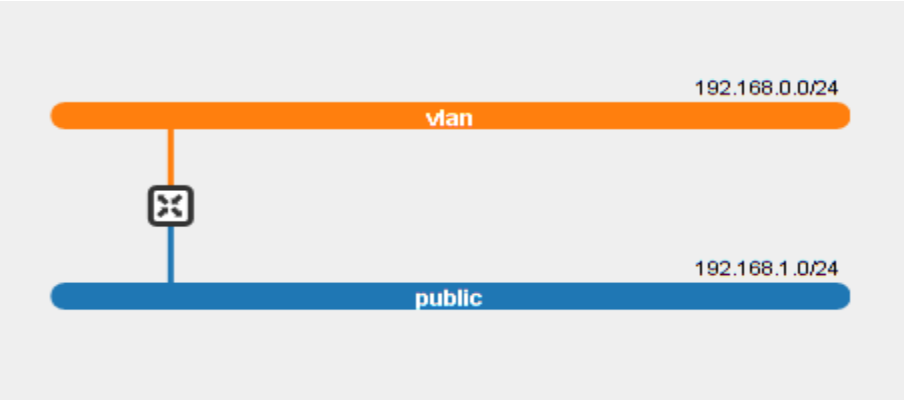
7.2. Δημιουργία εικονικού δρομολογητή.

Στη συνέχεια στο Network tab-> Routers πατάμε το Create Router. Δίνουμε όνομα στον δρομολογητή και έπειτα τον επιλέγουμε. Στο επόμενο παράθυρο πατάμε το κουμπί Add Interface και επιλέγουμε το προηγούμενο δίκτυο που είχαμε δημιουργήσει.



Εικόνα 14 - Απόδοση δικτύου σε Δρομολογητή

Αφού δημιουργήσουμε το interface επιλέγουμε το Set Gateway και επιλέγουμε το δημόσιο δίκτυο που έχουμε. Η τοπολογία της υποδομής πρέπει να έχει τη εξής μορφή:



Εικόνα 15 - Τοπολογία δικτύου με δρομολογητή

Στο σημείο αυτό το δίκτυο που θα φιλοξενήσει τον εξισορροπιστή φόρτου είναι έτοιμο.

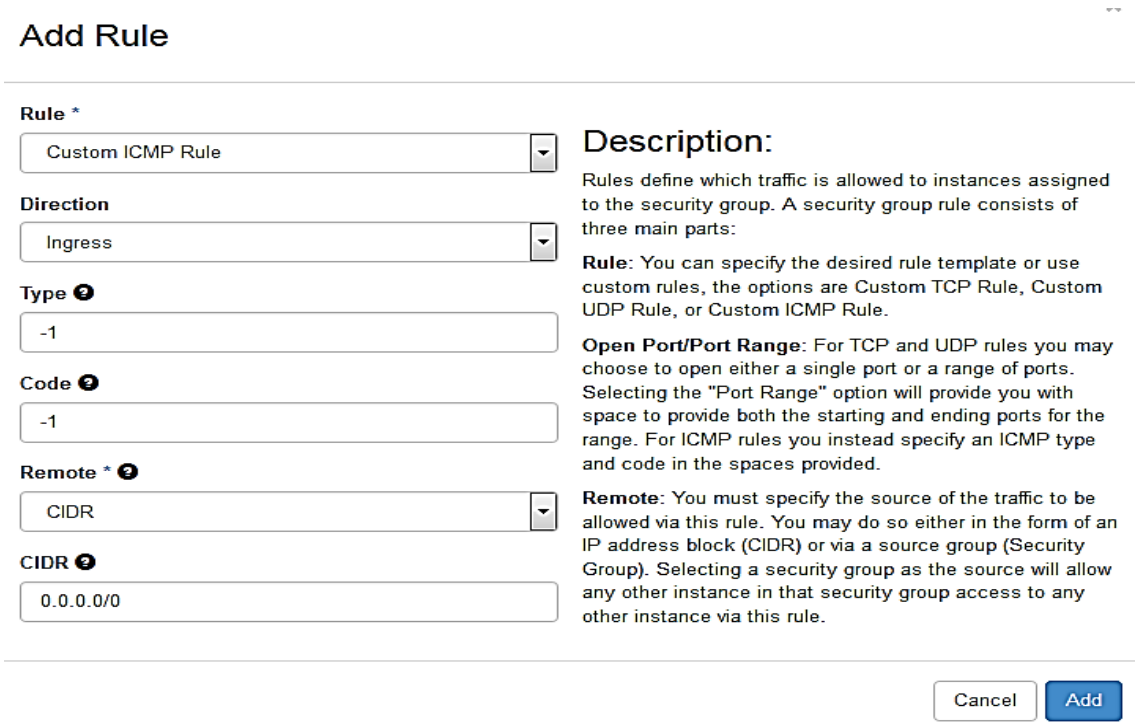
7.3. Δημιουργία κλειδιού.

Έπειτα από την δημιουργία του δικτύου πρέπει να ρυθμίσουμε το nova έτσι ώστε να μπορούμε να έχουμε πρόσβαση στα στιγμιότυπα που δημιουργούμε. Στο Project>Compute>Access & Security επιλέγουμε το tab Key Pairs. Επιλέγουμε να δημιουργήσουμε καινούριο κλειδί πατώντας το Create Key Pair και δίνουμε ονομασία στο κλειδί(πχ cloudKey). Στη συνέχεια θα δημιουργηθεί ένα κλειδί το οποίο κατεβάζουμε και αποθηκεύουμε σε μια τοποθεσία γιατί θα χρησιμοποιηθεί για να αποκτήσουμε πρόσβαση στα στιγμιότυπα.

7.4. Δημιουργία κανόνων πρόσβασης.

Το επόμενο βήμα είναι να επιλέξουμε το tab Security Rules στο οποίο ρυθμίζουμε τι τύπου διαδικτυακές αιτήσεις αποδέχονται τα στιγμιότυπα και ποιές θύρες θα

χρησιμοποιούνται. Το προεπιλεγμένο σύνολο κανόνων που υπάρχει είναι το default στο οποίο επιλέγουμε το Manage Rules. Πρέπει να δημιουργήσουμε έξι καινούριους κανόνες για αυτή την ομάδα ασφαλείας. Πατώντας το Add Rule επιλέγουμε το Custom ICMP Rule , Direction Ingress, Type -1 και Code -1. Τα υπόλοιπα μένουν ως έχουν (βλ. εικόνα 17)



Εικόνα 16 - Κανόνας ICMP

Ο κανόνας αυτός δημιουργείται για να μπορούμε να στείλουμε ping στα στιγμιότυπα και αυτά να απαντούν. Με τον τρόπο αυτό βεβαιωνόμαστε ότι τα στιγμιότυπα και το δίκτυο λειτουργούν σωστά. Τον συγκεκριμένο κανόνα χρειάζεται και ο εξισορροπιστής φόρτου διότι ανάλογα την υλοποίηση μας μπορεί να χρησιμοποιεί ping για να βεβαιωθεί ότι ένα στιγμιότυπο είναι ενεργό και σε θέση να εξυπηρετεί.

Στη συνέχεια προσθέτουμε έναν ακόμα κανόνα επιλέγοντας το SSH για να αποδέχονται τα μηχανήματα ασφαλή απομακρυσμένη σύνδεση στο τερματικό τους.

Έπειτα πρέπει να δημιουργήσουμε άλλους τέσσερις κανόνες στην θύρα που θα επιλέξουμε να χρησιμοποιήσουν τα στιγμιότυπα για να εξυπηρετούν. Στην περίπτωση μας έχουμε επιλέξει την θύρα 8080. Επομένως δημιουργούμε δύο κανόνες τύπου TCP στην θύρα 8080 με διαφορετική κατεύθυνση ο καθένας (Ingress & Egress) και δύο παρόμοιους κανόνες UDP [60].

7.5. Προετοιμασία των στιγμιότυπων.

Στη συνέχεια έχουμε ως σκοπό να προετοιμάσουμε τα στιγμιότυπα και να τα ρυθμίσουμε ετσι ώστε να είναι σε θέση να εξυπηρετούν αιτήσεις. Επομένως στα

Ubuntu 14.04 αφού κάνουμε τις απαραίτητες ενημερώσεις θα εγκαταστήσουμε tomcat και θα φτιάξουμε μια διαδικτυακή εφαρμογη.

Προχωράμε στο tab Project > Compute > Instances και επιλέγουμε το launch instance. Δίνουμε όνομα στο στιγμιότυπο, επιλέγουμε για flavor το m1.small(ή tiny ανάλογα με τους διαθέσιμους πόρους του συστήματος.), επιλέγουμε Boot from image και μετά διαλέγουμε το Ubuntu 14.04 που είχαμε προσθέσει χειροκίνητα στο glance. Έπειτα επιλέγουμε το tab Access & Security και διαλέγουμε την default ομάδα και το ζεύγος κλειδιού που δημιουργήσαμε πριν. Στο Networking tab επιλέγουμε το εσωτερικό ιδιωτικό δίκτυο vlan.

Στο σημείο αυτό μπορούμε να προχωρήσουμε με δύο τρόπους. Αρχικά μπορούμε να δώσουμε ένα script ως είσοδο στο στιγμιότυπο το οποίο θα αρχίσει να εκτελείται αυτό δημιουργηθεί και αρχικοποιηθούν διάφοροι παράμετροι, όπως το δίκτυο για παράδειγμα, είτε να επιλέξουμε να συνδεθούμε μέσω ssh χρησιμοποιώντας το κλειδί που δημιουργήσαμε και να εκτελέσουμε τις εντολές που χρειάζεται για να κάνουμε ενημερώσεις στο στιγμιότυπο, να εγκαταστήσουμε τον tomcat και να φτιάξουμε μια δικτυακή εφαρμογή. Όποιον τρόπο και να επιλέξουμε το στιγμιότυπο πρέπει να αποκτήσει μια floating ip για να μπορέσει να έχει πρόσβαση στο διαδίκτυο [61].

7.5.1. Παραμετροποίηση μέσω post creation script.

Πριν επιλέξουμε να ξεκινήσει η δημιουργία του στιγμιότυπου επιλέγουμε το tab Post-Creation και δίνουμε ως είσοδο ένα αρχείο το οποίο είναι τύπου bash και έχει τις εξής εντολές:

```
#!/bin/bash -v
sudo apt-get update -y &;
wait;
sudo apt-get install tomcat7 -y &;
wait;
echo "installed tomcat7";
sudo mkdir /usr/share/tomcat7-myapp;
sudo mkdir /usr/share/tomcat7-myapp/myapp;
sudo echo "<!DOCTYPE html>
<html>
<body>
<h1>Load balancer test</h1>
<h3>Each server generates a random number</h3>
</body>
</html>
 " >> /usr/share/tomcat7-myapp/myapp/index.html;
echo "@@created the folder in /usr/share ";
#sudo nano /etc/tomcat7/Catalina/localhost/myapp.xml
sudo        echo        "        <Context        path=\"/myapp\"
docBase=\"/usr/share/tomcat7-myapp/myapp\"      />      "      >>
/etc/tomcat7/Catalina/localhost/myapp.xml;
sudo  sed  -i  's/JAVA_OPTS="-Djava.awt.headless=true  -Xmx128m  -
XX:+UseConcMarkSweepGC"/JAVA_OPTS="-
Djava.security.egd=file:\/dev\/.\/urandom                          -
```

```
Djava.awt.headless=true    -Xmx1024m    -XX:MaxPermSize=512m    -
XX:+UseConcMarkSweepGC"/g' /etc/default/tomcat7
#JAVA_OPTS="-Djava.awt.headless=true          -Xmx128m          -
XX:+UseConcMarkSweepGC"
sudo service tomcat7 restart;
echo "READY TO SERVE";
```

Οι εντολές κάνουν ενημερώσεις για το στιγμιότυπου που μόλις δημιουργήθηκε και στη συνέχεια εγκαθιστούν τον tomcat δίνοντας του μια παραμετροποίηση για να έχει πιο γρήγορη εκκίνηση. Έπειτα δημιουργείται ένας φάκελος στο `/usr/share/tomcat7-myapp` ο οποίος περιέχει την εφαρμογή. Η διαδικτυακή εφαρμογή αποτελείται απο ενα απλό index page.

Μόλις δώσουμε ως είσοδο το αρχείο επιλέγουμε το Launch για να ξεκινήσει η δημιουργία του στιγμιότυπου. Στη συνέχεια εμφανίζεται το μηχάνημα που έχουμε δημιουργήσει στο παράθυρο Instances και απο εκεί επιλέγουμε το Assign Floating IP και επιλέγουμε να δώσουμε μια διεύθυνση δικτύου απο το δημόσιο δίκτυο μας. Είναι σημαντικό να δώσουμε floating ip άμεσα στο στιγμιότυπο για να μπορέσει να εκτελέσει τις εντολές. Επιλέγοντας το όνομα του στιγμιότυπου στο tab Log μπορούμε να δούμε την έξοδο του τερματικού του.

7.5.2. Παραμετροποίηση με σύνδεση μέσω SSH.

Στην περίπτωση αυτή επιλέγουμε την δημιουργία του στιγμιότυπου χωρίς να δώσουμε κάποια είσοδο στο Post-Creation. Μετά την δημιουργία του στιγμιότυπου και αφού του παραχωρήσουμε μια διεύθυνση δημόσιου δικτύου (Floating IP) μπορούμε να συνδεθούμε στο στιγμιότυπο μέσω τερματικού χρησιμοποιώντας το κλειδί που έχουμε δημιουργήσει και έχουμε αποθηκεύσει. Επομένως σε τερματικό εκτελούμε:

```
#ssh –i cloudKey.pem ubuntu@<floating_IP>
```

Με την floating ip να είναι η διεύθυνση που μόλις παραχωρήθηκε. Στη συνέχεια χειριζόμαστε το στιγμιότυπο όπως θα χειριζόμασταν ένα οποιοδήποτε μηχάνημα μέσω τερματικού και εκτελούμε τις εντολές του παραπάνω αρχείου [62].

7.5.3. Αποθήκευση του στιγμιότυπου

Όταν ολοκληρώσουμε την διαδικασία παραμετροποίησης του στιγμιότυπου και βεβαιωθούμε ότι αυτό στέλνει απάντηση στην διεύθυνση http//:<floating_IP>:8080/myapp από το Horizon επιλέγουμε να αποθηκεύσουμε το στιγμιότυπο πατώντας Create Snapshot δίπλα από τις πληροφορίες του στιγμιότυπου στο Instances tab. Διαλέγουμε ένα όνομα (πχ ubuntuTomcat) και αποθηκεύουμε.

8. Υλοποίηση του autoscaling

Το κεφάλαιο αυτό περιγράφει λεπτομερώς την υλοποίηση του autoscaling χρησιμοποιώντας έναν εξισορροπιστή φόρτου ο οποίος μοιράζει τις εισερχόμενες αιτήσεις που έρχονται απο πελάτες σε ενα ενεργό στιγμιότυπο. Σε περίπτωση που η υπολογιστική ισχύς φτάσει σε κάποιο όριο που έχει προκαθοριστεί ο υπεύθυνος συναγερμός ενεργοποιείται και δημιουργείται ενα καινούριο στιγμιότυπο, κλώνος του αρχικού, με σκοπό να προσφέρει τις υπηρεσίες του.

Στην υλοποίηση χρησιμοποιούνται περιγράμματα και η μηχανή του Heat για την δημιουργία ενός autoscaling group. Το autoscaling group είναι μια λειτουργία που προσφέρεται από το Heat και είναι μια ομάδα διαχείρισης ενός συνόλου στιγμιότυπων που έχουν ως στόχο να προσφέρουν λειτουργίες κλιμάκωσης. Δέχονται διάφορες παραμετροποιήσεις που καθορίζουν το είδος της κλιμάκωσης που προσφέρουν καθώς και τα χρονικά διαστήματα που πρέπει να μεσολαβήσουν για να δεχτούν κάποια αλλαγή.

Ας δούμε τα αρχεία περιγράμματος που χρησιμοποιήθηκαν:
`AutoscalingGroup.yaml`

```
heat_template_version: 2013-05-23

description: The autoscaling group

parameters:

  key_name:
    type: string
    description: Name of an existing key pair to use
    default: cloud
    constraints:
      - custom_constraint: nova.keypair

  flavor:
    type: string
    description: Flavor for the server to be created
    default: m1.custom
    constraints:
      - custom_constraint: nova.flavor

  image:
    type: string
    description: Image ID or image name to use for the server
    default: a32e8cbb-e12d-4678-8d86-bdd2701eafd6
    constraints:
      - custom_constraint: glance.image


  net:
    description: name of local network used to launch instance.
    type: string
    default: vlan
```

```
  subnet_id:
    type: string
    description: net on which the load balancer will be located
    default: 08aedc30-a179-4fcd-8069-8b0a9608cffa


  external_network_id:
    type: string
    description: UUID of a Neutron external network
    default: 6483ce9a-be4f-402d-b279-ad4701a7bbc3

resources:
  auto_scaling_group:
    type: OS::Heat::AutoScalingGroup
    properties:
      min_size: 1
      max_size: 2
      resource:
        type: lb_server.yaml
        properties:
          flavor: {get_param: flavor}
          image: {get_param: image}
          key_name: {get_param: key_name}
          network: {get_param: net}
          pool_id: {get_resource: lb_pool}
          metadata:{"metering.stack":{get_param:"OS::stack_id"}}

  web_server_scaleup_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: auto_scaling_group}
      cooldown: 60
      scaling_adjustment: 1

    web_server_scaledown_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: {get_resource: auto_scaling_group}
      cooldown: 60
      scaling_adjustment: -1

  cpu_alarm_high:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-up if the average CPU > 50% for 1
minute
      meter_name: cpu_util
      statistic: avg
      period: 60
      evaluation_periods: 1
      threshold: 70
      alarm_actions:
        - {get_attr: [web_server_scaleup_policy, alarm_url]}
      matching_metadata:{'metadata.user_metadata.stack':
```

```
{get_param: "OS::stack_id"}}
      comparison_operator: gt

  cpu_alarm_low:
    type: OS::Ceilometer::Alarm
    properties:
      description: Scale-down if the average CPU < 10% for 10
minutes
      meter_name: cpu_util
      statistic: avg
      period: 60
      evaluation_periods: 1
      threshold: 3
      alarm_actions:
        - {get_attr: [web_server_scaledown_policy, alarm_url]}
      matching_metadata:         {'metadata.user_metadata.stack':
{get_param: "OS::stack_id"}}
      comparison_operator: lt

  health_monitor:
    type: OS::Neutron::HealthMonitor
    properties:
      type: HTTP
      delay: 30
      max_retries: 6
      timeout: 30
      http_method: GET
      url_path: /myapp/


  lb_pool:
    type: OS::Neutron::Pool
    properties:
      protocol: HTTP
      monitors: [{get_resource: health_monitor}]
      subnet_id: {get_param: subnet_id}
      lb_method: ROUND_ROBIN
      vip:
        protocol_port: 8080

  load_balancer:
    type: OS::Neutron::LoadBalancer
    properties:
      protocol_port: 8080
      pool_id: {get_resource: lb_pool}

  load_balancer_floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network_id: {get_param: external_network_id}
      port_id: {get_attr: [lb_pool, vip, port_id]}

outputs:
  scale_up_url:
    description: >
      This  URL  is  the  webhook  to  scale  up  the  autoscaling
```

```
group.Youcan invoke the scale-up operation by doing an HTTP POST
to this URL; no body nor extra headers are needed.
    value: {get_attr: [web_server_scaleup_policy, alarm_url]}
  scale_dn_url:
    description: >This URL is the webhook to scale down the
autoscaling group.You can invoke the scale-down operation by
doing an HTTP POST tothis URL; no body nor extra headers are
needed.
    value: {get_attr: [web_server_scaledown_policy, alarm_url]}

  pool_ip_address:
    value: {get_attr: [lb_pool, vip, address]}
    description: The IP address of the load balancing pool

  website_url:
    value:
      str_replace:
        template: http://host:8080/myapp
        params:
          host:{get_attr:[load_balancer_floating_ip,
floating_ip_address] }
    description: >
      This URL is the "external" URL that can be used to access
the website.
```

Το περίγραμμα αυτό δέχεται ως παραμέτρους την εικόνα του στιγμιότυπου, το flavor, το ιδιωτικό δίκτυο, το εξωτερικό δίκτυο και τον κωδικό του διαχειριστή. Σε κάθε μια από τις παραμέτρους έχουν οριστεί και προεπιλεγμένες τιμές για να γίνεται να δώσουμε το περίγραμμα ως είσοδο χωρίς καμία παράμετρο.

Στο τμήμα των πόρων (resources) δημιουργείται ένα autoscaling group στο οποίο μπορούμε να ορίσουμε το μέγεθος που θα μπορεί να λάβει (min, max size). Τον τύπο του, δηλαδή ποια θα είναι τα στιγμιότυπα που θα αναλάβει να διαχειρίζεται. Το αρχείο lb_server.yaml δηλώνει στο autoscaling group το στιγμιότυπο που θα αναπαράγει. Παρακάτω εξηγείται αναλυτικά το αρχείο αυτό.

Στη συνέχεια ορίζονται οι πολιτικές του autoscaling group, δηλαδή τι τύπου αλλαγές θα πρέπει να εφαρμόζει σε περίπτωση που οι συνθήκες που έχουμε θέσει πραγματοποιηθούν. Εδώ βλέπουμε ότι έχουμε δύο πολιτικές, scale up και scale down policy που καθορίζουν ότι οι αλλαγές που θα γίνονται είναι ποσοτικές και μάλιστα με βήμα ±1.

Ακολουθούν οι συνθήκες των συναγερμών οι οποίοι καθορίζουν τι είδους συναγερμό θα χρησιμοποιήσουμε, κάθε πότε θα γίνονται μετρήσεις και την ίδια την συνθήκη. Επομένως ορίζονται δύο συναγερμοί για scale up και scale down οι οποίοι συλλέγουν στατιστικά από τα στιγμιότυπα και σε περίπτωση που η συνθήκη που έχει τεθεί πραγματοποιηθεί, δηλαδή ο συναγερμός ενεργοποιηθεί εφαρμόζεται η κατάλληλη πολιτική. Η περίοδος που καθορίζουμε είναι για πόσο πρέπει κατ' ελάχιστο να ισχύει η συνθήκη για να θεωρηθεί ότι πρέπει να εφαρμοστεί η αντίστοιχη πολιτική. Η περίοδος αξιολόγησης αυξάνει πολλαπλασιαστικά την περίοδο αφού για μεγαλύτερο της μονάδας πρέπει να περάσει n κύκλους αξιολόγησης για να εφαρμοστεί η πολιτική που χρειάζεται.

Στη συνέχεια καθορίζεται ο ελεγκτής ορθής λειτουργίας που θα είναι υπεύθυνος να αξιολογεί τα στιγμιότυπα στέλνοντας τους αιτήματα, των οποίων τον τύπο και τις παραμέτρους καθορίζουμε, με σκοπό να αποφανθεί για την κατάσταση των στιγμιοτύπων (ACTIVE/INACTIVE). Στο σημείο αυτό ορίζουμε εναν ελεγκτή ορθής λειτουργίας(health monitor) ο οποίος θα στέλνει HTTP requests τύπου GET με πρόσθετο path το /myapp/ . Στο σημείο αυτό ο ελεγκτής ορθής λειτουργίας με την εκκίνηση της στοίβας θεωρεί οτι δεν έχει ενεργά μέλη. Με την πάροδο του κύκλου αξιολόγησης θα αποφανθεί αν τα στιγμιότυπα είναι ενεργά και θα αλλάξει την κατάσταση τους.

Έπειτα ακολουθεί η δημιουργία του χώρου συγκέντρωσης των διευθύνσεων στις οποίες οι εξισορροπιστής φόρτου θα προωθεί τα αιτήματα, τον αλγόριθμο προώθησης που θα εφαρμοστεί καθώς και η θύρα στην οποία εξυπηρετούν τα μηχανήματα, αλλά και ο ίδιος.

Τέλος δημιουργείται ο εξισορροπιστής φόρτου και καθορίζεται μια διεύθυνση δημόσιου δικτύου για να χρησιμοποιεί.

Το τμήμα των εκτυπώσεων φροντίζει να εμφανίσει σημαντικές πληροφορίες στον χρήστη για στοιχεία της στοίβας, όπως την διεύθυνση του εξισορροπιστή φόρτου καθώς και τα url που στέλνονται οι αιτήσεις για scale up/ down έτσι ώστε να έχει την δυνατότητα να χειριστεί την στοίβα εκείνος στέλνοντας αίτημα [63] [64].

lb_server.yaml

```
heat_template_version: 2013-05-23
description: A load-balancer server
parameters:

  image:
    type: string
    description: Image used for servers
    default: a32e8cbb-e12d-4678-8d86-bdd2701eafd6

  key_name:
    type: string
    description: SSH key to connect to the servers
    default: cloud

  flavor:
    type: string
    description: flavor used by the servers
    default: m1.custom

  pool_id:
    type: string
    description: Pool to contact

  metadata:
    type: json

  network:
    type: string
    description: Network local used by the server
```

```
    default: 9ff62a20-e22f-4ebe-9f65-0dfc19896f6c

  external_network_id:
    type: string
    description: UUID of a Neutron external network
    default: 9ca875b5-729f-4bc9-8260-572e29ba6e1a

resources:
  hostname:
    type: OS::Heat::RandomString


  server:
    type: OS::Nova::Server
    properties:
      flavor: {get_param: flavor}
      image: {get_param: image}
      key_name: {get_param: key_name}
      metadata: {get_param: metadata}
      networks: [{network: {get_param: network} }]
      user_data_format: RAW
      user_data:
        str_replace:
          template: |
            #!/bin/bash -v
            RANGE=500
            number=$RANDOM
            let "number %= $RANGE"
            echo "Random number less than $RANGE  ---  $number"
            echo "hostname is $hostname"
            sudo     sed     -i     "7i     <p>random     number:
<h2>$number</h2></p>" /usr/share/tomcat7-myapp/myapp/index.html
            sudo sed -i "6i <p>Hello World from $hostname</p>"
/usr/share/tomcat7-myapp/myapp/index.html
          params:
            $hostname: {get_attr: [hostname, value]}
  pool_member:
    type: OS::Neutron::PoolMember
    properties:
      pool_id: {get_param: pool_id}
      address: {get_attr: [server, first_address]}
      protocol_port: 8080

 #Load balancer should function without a floating ip at their
members
  floating_ip:
    type: OS::Neutron::FloatingIP
    properties:
      floating_network: public


  association:
    type: OS::Neutron::FloatingIPAssociation
    properties:
      floatingip_id: { get_resource: floating_ip }
      port_id:{get_attr:[server,addresses,{get_param:   network},
```

```
0, port]}
```

Το αρχείο αυτό περιέχει όλη την πληροφορία που χρειάζεται για να δημιουργούνται τα στιγμιότυπα του AutoScaling group. Στην είσοδο δίνονται πληροφορίες για το δίκτυο που θα χρησιμοποιούν, το ζεύγος κλειδιού, η εικόνα, ο χώρος διευθύνσεων (pool) που χρησιμοποιείται από τον εξισορροπιστή φόρτου και το ιδιωτικό και δημόσιο δίκτυο.

Στο τμήμα των πόρων καθορίζεται το στιγμιότυπο με τις παραμέτρους του και δίνεται ως είσοδος ένα σύνολο εντολών, οι οποίες προσθέτουν μια καινούρια γραμμή στο index page της διαδικτυακής μας εφαρμογής η οποία περιέχει έναν τυχαίο αριθμό. Κάθε φορά που δημιουργείται ένα στιγμιότυπο θα έχει τον δικό του τυχαίο αριθμό να εμφανίζεται στο index page επομένως θα ξεχωρίζει από τα υπόλοιπα. Έτσι θα μπορούμε να εξακριβώσουμε ότι ο εξισορροπιστής φόρτου λειτουργεί σωστά και κατευθύνει κάθε καινούρια αίτηση σε διαφορετικό στιγμιότυπο.

Στη συνέχεια προστίθεται στο σύνολο των μελών του εξισσοροπιστή φόρτου και του παραχωρείται μια διεύθυνση δημόσιου δικτύου (floating ip) [65] [66] [67].

8.1. Εκτέλεση του περιγράμματος

Στη συνέχεια για την εκτέλεση του περιγράμματος μεταφέρουμε το AutoscalingGroup.yaml και lb_server.yaml στον controller. Παράλληλα στον περιηγητή(του host ή του controller) έχουμε κάνει είσοδο στο Horizon. Σε τερματικό του controller στον φάκελο που βρίσκονται τα αρχεία yaml και αφού εκτελέσουμε το keystonerc_admin δίνουμε το περίγραμμα στην μηχανή του Heat:

```
#heat stack-create mystack --template-file AutoscalinGroup.yaml
```

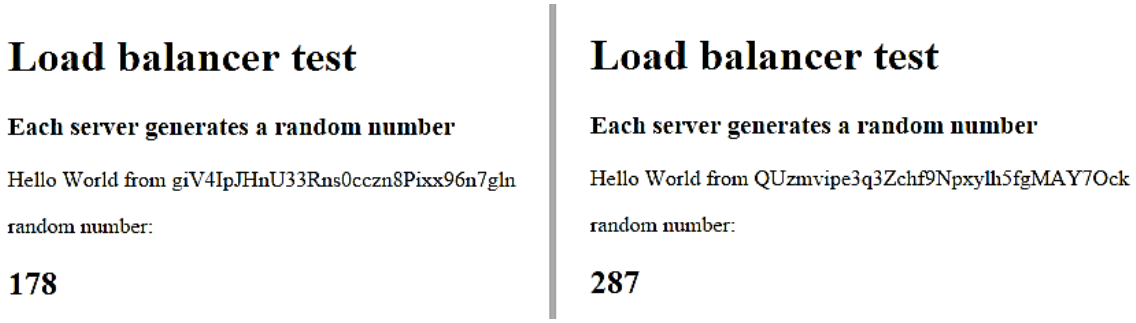
Με την εκτέλεση της εντολής παρατηρούμε στο Orchestration tab -> Stacks του Horizon ότι εμφανίστηκε μια στοίβα και είναι In Progress. Όταν δημιουργηθεί η στοίβα μπορούμε επιλέγοντας την να δούμε την τοπολογία της καθώς και πληροφορίες σχετικά με τις εκτυπώσεις, τους πόρους που χρησιμοποιεί και τα συμβάντα της. Παράλληλα μπορούμε να παρατηρούμε τους συναγερμούς που έχουν συνδεθεί με την στοίβα εκτελώντας σε τερματικό:

```
#ceilometer alarm-list
```

Η εντολή εκτυπώνει το όνομα, την κατάσταση, το είδος του συναγερμού και τους περιορισμούς που έχουμε θέσει. Οι συναγερμοί έχουν τρείς δυνατές καταστάσεις: insufficient data, ok και alarm. Η τιμή insufficient data εμφανίζεται στην αρχή της δημιουργίας της στοίβας όπου οι συναγερμοί δεν έχουν ακόμα συλλέξει τα απαραίτητα δεδομένα για να αποφανθούν αν οι συνθήκες ενεργοποίησης ισχύουν η όχι. Στη συνέχεια αποκτούν τιμή ανάλογα με τις συνθήκες. Η τιμή ok δηλώνει ότι ο συναγερμός δεν έχει ενεργοποιηθεί και η τιμή alarm δηλώνει το αντίθετο.

Παράλληλα μπορούμε να εισάγουμε την διεύθυνση του εξισορροπιστή φόρτου στον περιηγητή για να προωθήσει την αίτηση στο στιγμιότυπο που έχει δημιουργηθεί και να λάβουμε απάντηση. Σε περίπτωση που δεν λάβουμε απάντηση το στιγμιότυπο δημιουργείται και δεν είναι έτοιμο να εξυπηρετήσει αιτήσεις.

Σε περίπτωση που ο συναγερμός για scale up ενεργοποιηθεί, δηλαδή η υπολογιστική ισχύς του στιγμιότυπου ξεπεράσει το 70% τότε αν επιλέξουμε το tab Project-> Compute -> Instances μπορούμε να δούμε το καινούριο στιγμιότυπο που μόλις δημιουργήθηκε. Μετά την αρχικοποίηση και του δεύτερου στιγμιότυπου έχοντας την σελίδα του εξισορροπιστή φόρτου ανοικτή, αν πατήσουμε ανανέωση θα βλέπουμε κάθε φορά μια παρόμοια απάντηση (το index page του κάθε στιγμιότυπου) με διαφορετικό τυχαίο αριθμό να εμφανίζεται.



Εικόνα 17 - Απάντηση εξυπηρετητών με διαφορετικό τυχαίο αριθμό

Στο σημείο αυτό έχουμε καταφέρει να πετύχουμε αυτόματη κλιμάκωση στα στιγμιότυπα μας, που ήταν και ο σκοπός της πτυχιακής εργασίας. Η υποδομή που έχουμε δημιουργήσει ανταποκρίνεται στον φόρτο εργασίας των στιγμιότυπων επιλέγοντας να αυξήσει ή να μειώσει τον αριθμό τους ανάλογα με τον φόρτο που αντιμετωπίζουν.

9. Υψηλή διαθεσιμότητα

Ο όρος υψηλή διαθεσιμότητα [68] αναφέρεται σε μια υπολογιστική υποδομή και καθορίζει την δυνατότητα του συστήματος να είναι ενεργό και να εξυπηρετεί τους χρήστες χωρίς να υπάρχει διακοπή της λειτουργίας του και απώλεια δεδομένων. Η διακοπή της λειτουργίας μιας υποδομής χαρακτηρίζεται σε:

- **Προγραμματισμένη διακοπή λειτουργίας:** Είναι η προγραμματισμένη διακοπή της λειτουργίας η οποία έχει ως στόχο την συντήρηση ή ενημέρωση των υπηρεσιών που προσφέρονται.
- **Μη προγραμματισμένη διακοπή λειτουργίας:** Η οποία μπορεί να προκύψει από διάφορους παράγοντες, όπως από αύξηση της τάσης του ρεύματος, από πρόβλημα στις υπηρεσίες που χρησιμοποιούνται, από φυσικές καταστροφές κ.ά [69].

Οι λύσεις που προσφέρονται για την αντιμετώπιση των μη προγραμματισμένων διακοπών λειτουργίας διαφέρουν ανάλογα με τον σχεδιασμό της εκάστοτε υποδομής αλλά και της υπηρεσίας που προσφέρουν. Οι πρακτικές που μπορούν να οδηγήσουν σε συστήματα υψηλής διαθεσιμότητας κατηγοριοποιούνται σε λύσεις που προσφέρουν υψηλή διαθεσιμότητα σε ένα κέντρο δεδομένων καθώς και σε λύσεις που αφορούν φυσικές καταστροφές, στις οποίες συνήθως χρησιμοποιούνται συστήματα σε διαφορετικές γεωγραφικές περιοχές.

Για την υλοποίηση ενός συστήματος με υψηλή διαθεσιμότητα επιπλέον τεχνολογίες και τεχνικές είναι απαραίτητο να χρησιμοποιηθούν, με τον πιο σημαντικό ρόλο να έχει ο πλεονασμός συστημάτων. Με την ύπαρξη πλεοναζόντων συστημάτων και εξαρτημάτων εφαρμόζονται τεχνικές που σε περιπτώσεις διακοπής λειτουργίας της υποδομής είτε αναλαμβάνουν τον επιπρόσθετο φόρτο υπάρχοντα δευτερεύοντα συστήματα, είτε ενεργοποιούνται για τον σκοπό αυτό.

Τα περισσότερα συστήματα υψηλής διαθεσιμότητας εγγυώνται προστασία σε περίπτωση διακοπής της λειτουργίας και απώλειας δεδομένων, ωστόσο αναμένονται να προστατεύουν συστήματα στα οποία μία αποτυχία μπορεί να οδηγήσει σε ενα σύνολο από αλληλεπικαλυπτόμενες αποτυχίες. Κάθε σύστημα αναλαμβάνει να εξουδετερώσει όλα τα διακριτά σημεία στα οποία μπορεί να εμφανιστεί μια αποτυχία.Σε περιπτώσεις που πολλές αποτυχίες συμβούν μαζί συνήθως επέρχεται διακοπή της λειτουργίας και το σύστημα θα φροντίσει να μην υπάρξει απώλεια δεδομένων.

Στο κεφάλαιο αυτό θα ασχοληθούμε με τις κυριότερες τεχνικές που χρησιμοποιούνται στο OpenStack για να επιτευχθεί υψηλή διαθεσιμότητα.

9.1. Υψηλή διαθεσιμότητα στο OpenStack.

Η υλοποίηση ενός συστήματος υψηλής διαθεσιμότητας που χρησιμοποιεί OpenStack επαφίεται στου σχεδιαστές του, καθώς το OpenStack δεν προσφέρει ολοκληρωμένες λύσεις μαζί με την εγκατάσταση του. Ωστόσο με την κατάλληλη

υποδομή και τεχνικές μπορεί να γίνει σύστημα υψηλής διαθεσιμότητας με χρόνο λειτουργίας χωρίς αποτυχίες σε ποσοστό 99.99% που αντιστοιχεί σε μία ώρα μη λειτουργίας σε διάστημα ενός έτους [70].

Κάθε σύστημα το οποίο θέλει να πετύχει υψηλή διαθεσιμότητα πρέπει να εκμηδενίσει σημεία στην υποδομή, τα οποία σε περίπτωση βλάβης είναι ικανά να προκαλέσουν την κατάρρευση του. Η δυνατότητα εκμηδένισης τον «κρίσιμων» σημείων εξαρτάται αν η υπηρεσία τους είναι stateless ή stateful.

Μια υπηρεσία ονομάζεται **stateless** [71] όταν, ενώ εξυπηρετεί αιτήματα, δεν συγκρατεί κάποιου είδους πληροφορία από προηγούμενα αιτήματα και κάθε καινούριο αίτημα έχει εξ ’ολοκλήρου τα δεδομένα που χρειάζονται για να εξυπηρετηθεί. Για να λειτουργήσουν οι υπηρεσίες αυτές σε υψηλή διαθεσιμότητα αρκεί να υπάρχει πλεονασμός σε υλικό της υποδομής, καθώς σε περίπτωση που κάποιο υλικό παρουσιάσει βλάβη, μπορεί κάποιο οποιοδήποτε άλλο υλικό με την υπηρεσία αυτή να συνεχίσει να εξυπηρετεί.

Αντίστοιχα μια υπηρεσία ονομάζεται **stateful** [72] όταν η εξυπηρέτηση των αιτημάτων που δέχεται καθορίζεται από παραμέτρους που έχουν ορισθεί από προηγούμενα αιτήματα, είτε από άλλους παράγοντες. Οι υπηρεσίες αυτές είναι πιο δύσκολο να λειτουργήσουν σε κατάσταση υψηλής διαθεσιμότητας καθώς σε περίπτωση βλάβης ενός υλικού είναι αναγκαίο το υλικό που θα αναλάβει την εξυπηρέτηση να γνωρίζει και την κατάσταση του υλικού πριν υποστεί βλάβη. Παραδείγματα των υπηρεσιών αυτών στο OpenStack είναι η βάση δεδομένων του και η υπηρεσία ανταλλαγής μηνυμάτων.

Για είδος υπηρεσίας υπάρχουν διαφορετικές τεχνικές που αν εφαρμοστούν μπορούν να οδηγήσουν σε υψηλή διαθεσιμότητα [73].

9.1.1. Αρχιτεκτονική active/passive

Στην αρχιτεκτονική active/passive [74], συστήματα εφεδρικά ενεργοποιούνται σε περίπτωση που τα βασικά συστήματα βρεθούν σε κατάσταση βλάβης. Σε περίπτωση που η υπηρεσία είναι stateless καθορίζονται κάποια μηχανήματα τα οποία θα αρχίσουν να λειτουργούν σε περίπτωση βλάβης. Σε stateful υπηρεσίες χρειάζεται να καθοριστεί ένα εφεδρικό μηχάνημα το οποίο θα πρέπει να βρίσκεται υπό την εποπτεία επιπρόσθετων λογισμικών όπως είναι το Pacemaker και Corosync.
Στην αρχιτεκτονική αυτή ο χρόνος που θα χρειαστεί για επέλθει σε κατάσταση ετοιμότητας φαίνεται στον χρήστη ως καθυστέρηση εξυπηρέτησης του αιτήματος του.

9.1.2. Αρχιτεκτονική active/active

Στην αρχιτεκτονική αυτή [75] τα εφεδρικά συστήματα βρίσκονται σε κατάσταση λειτουργίας και ο χρήστης σε περίπτωση βλάβης δεν θα συνειδητοποιήσει κάποια αλλαγή στην ποιότητας της υπηρεσίας. Σε περίπτωση που η υπηρεσία είναι τύπου stateless αρκεί η ύπαρξη ενός εφεδρικού υλικού. Αντίθετα για τις stateful υπηρεσίες τα μηχανήματα εφεδρικά η μη πρέπει να βρίσκονται σε πανομοιότυπη κατάσταση. Αυτό σημαίνει ότι για παράδειγμα μια αλλαγή στη βάση δεδομένων ενός μηχανήματος πρέπει να αντικατοπτρίζεται και στα υπόλοιπα εφεδρικά.

9.1.3. Υψηλή διαθεσιμότητα και Heat

Σε συστήματα υπολογιστικών νεφών κυριαρχεί η φιλοσοφία οτι τα στιγμιότυπα είναι αναμενόμενο κάποια στιγμή να εμφανιστεί μια βλάβη και να τερματιστεί η λειτουργία τους. Επομένως δεν επιστρατεύονται τεχνικές για την διατήρηση ενός συγκεκριμένου στιγμιότυπου, δεδομένου ότι αυτό το στιγμιότυπο μπορεί εύκολα να αναπαραχθεί.

Η μηχανή του Heat (Heat engine) σε συνεργασία με την υπηρεσία ceilometer μπορούν να προσφέρουν υπηρεσίες υψηλής διαθεσιμότητας στην υποδομή. Πιο συγκεκριμένα στα περιγράμματα μπορούν να περιγραφούν συναγερμοί οι οποίοι ενεργοποιούνται όταν το στιγμιότυπο σταματήσει να απαντάει σε αιτήματα που δέχεται. Η πολιτική που θα ακολουθηθεί σε περίπτωση που ο συναγερμός ενεργοποιηθεί μπορεί να είναι είτε η επανεκκίνηση του στιγμιότυπου είτε ο τερματισμός του και η δημιουργία ενός καινούριου [76].

10. Συμπεράσματα

Τα τελευταία χρόνια η τεχνική της εικονικοποίησης έχει δημιουργήσει νέους ορίζοντες στον κόσμο της πληροφορικής. Απόγειο αυτής της τεχνικής είναι και η υπολογιστική νέφους η οποία έχει κάνει δυναμική είσοδο αλλάζοντας τα δεδομένα στα υπολογιστικά κέντρα, δημιουργώντας νέες υπηρεσίες και κατά συνέπεια καινούριες αγορές.

Η προσπάθεια αποσύνδεσης (abstraction) υλικού και λογισμικού προσφέρει σταθερότητα σε ένα υπολογιστικό κέντρο, δυνατότητα εύκολης και γρήγορης αντικατάστασης υλικών χωρίς να διακόπτεται η λειτουργία τους καθώς και καλύτερη εποπτεία των πόρων και της κατάστασης του συστήματος από τους διαχειριστές. Πλέον με τις τεχνικές αυτές το λογισμικό που εκτελείται και προσφέρει τις υπηρεσίες του σε χρήστες δεν συνδέεται άμεσα με το υλικό και κατά συνέπεια μπορούν εξυπηρετητές με διαφορετική τεχνολογία να συνυπάρχουν και να συγκροτούν την υποδομή του νέφους. Πριν την εμφάνιση της εικονικοποίησης μια εφαρμογή που είχε δημιουργηθεί σε έναν εξυπηρετητή ήταν σχεδόν αδύνατον να μεταφερθεί σε κάποιον άλλον και να λειτουργεί, πόσω μάλλον όταν ο εξυπηρετητής χρησιμοποιούσε διαφορετική τεχνολογία.

Ακόμα πιο σύνθετο και δύσκολο πρόβλημα ήταν η συντήρηση των εξυπηρετητών, αφού έπρεπε να γίνουν συγκεκριμένες χρονοβόρες διαδικασίες για τον σκοπό αυτό (να αρνούνται νέες συνεδρίες και να περιμένουν να λήξουν οι παλιές για να ξεκινήσει η διαδικασία της συντήρησης). Πλέον με την δυνατότητα μεταφοράς συνεδριών μέσα στο νέφος σε διαφορετικό εξυπηρετητή το πρόβλημα έχει γίνει πολύ πιο απλό.

Βλέποντας τα οφέλη της εικονικοποίησης ήταν απολύτως λογικό η προσπάθεια εικονικοποίησης του υλικού να μην σταματούσε στο σημείο αυτό. Η εικονικοποίση των δικτύων και υλικού δικτύου ήταν φυσικό επακόλουθο και επομένως τα δίκτυα βασισμένα σε λογισμικό (SDN’s -- software defined networks) προστέθηκαν στις δυνατότητες της εικονικοποίησης. Πλέον τα υπολογιστικά κέντρα είναι σε θέση να προσφέρουν σύνθετες αρχιτεκτονικές δικτύου στις υπηρεσίες τους.

Επιπρόσθετα η εμφάνιση της αυτόματης κλιμάκωσης των εικονικών συσκευών αυτοματοποίησε ακόμα περισσότερο τις διαδικασίες για την ανταπόκριση στις απαιτήσεις των πελατών για αναβάθμιση των πακέτων υπηρεσιών τους. Αυτή η δυνατότητα από την μεριά των υπολογιστικών κέντρων έχει ως αποτέλεσμα τη μείωση της καταναλώμενης ενέργειας, όταν δεν υπάρχει μεγάλος φόρτος εργασίας, και παράλληλα την κλιμάκωση σε περίπτωση που χρειαστεί περισσότερη υπολογιστική ισχύς. Οι πελάτες απο την μεριά τους με τα νέα μοντέλα κοστολόγησης δεν δεσμεύουν πόρους που ενδεχομένως δεν θα χρειαστεί να χρησιμοποιήσουν.

Τα οφέλη της εικονικοποίησης είναι πολλά, ωστόσο ας μην ξεχνάμε την συνεισφορά του OpenStack στην υπολογιστική νέφους διότι ενδεχομένως χωρίς αυτό να μην υπήρχε το νέφος όπως το γνωρίζουμε σήμερα. Η εμφάνιση των υπολογιστικών κέντρων με τις υπηρεσίες που προσφέρουν οφείλουν σε έναν βαθμό την εξέλιξη τους σε αυτό.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	Ελληνικός όρος
Virtualization	Εικονικοποίηση
Restful API	Προγραμματιστική διεπαφή Restful
Instances	Στιγμιότυπα
L3 forwarding	Προώθηση επιπέδου δικτύου
Virtual Private Network(VPN)	Εικονικό Ιδιωτικό Δίκτυο
Network address translation(NAT)	Μετάφραση δικτυακών διευθύνσεων
Load Balancer	Εξισορροπιστής φόρτου
Tenant	Ένοικος
Plugin	Πρόσθετο
Application Programming Interface(API)	Προγραμματιστική Διεπαφή
Switch	Μεταγωγέας
Session persistence	Παραμένουσα σύνοδος
Ip address pool	Δεξαμενή διευθύνσεων
Denial of Service(DoS)	Επίθεση άρνησης υπηρεσίας
Health monitor	Ελεγκτής ορθής λειτουργίας
Backend	Κεντρικό σύστημα
Event	Συμβάν
Alarms	Συναγερμοί
Collector agent	Πράκτορας Συλλογής Πληροφορίας
Hypervisor	Υπερεπόπτης
Daemon	Δαίμων
Namespace	Χώρος ονομάτων
Callback function	Συνάρτηση επιστροφής κλήσης
Pipeline	Διασωλήνωση
Metering	Λήψη μετρήσεων
Threshold	Κατώφλι
Samples	Δείγματα
Remote Procedure Call(RPC)	Κλήση απομακρυσμένης διαδικασίας
Resource	Πόρος
Lifecycle	Κύκλος ζωής
Scale	Κλιμάκωση
Autoscale	Αυτόματη κλιμάκωση
Metadata	Μετα-δεδομένα
Intrinsic functions	Εγγενείς συναρτήσεις
Puppet modules	Αρθρώματα puppet
Superuser	Υπερχρήστης
Template	Περίγραμμα
Agent	Πράκτορας
Domain	Τομέας
Hostname	Όνομα κόμβου
Network bridge	Διαδικτυακή γέφυρα
Secure Shell(SSH)	Ασφαλής απομακρυσμένη σύνδεση
Repository	Χώρος εναπόθεσης
Security Groups	Ομάδες Ασφαλείας
Bash script	Σενάριο Bash
Message queues	Υπηρεσία αποστολής μυνημάτων σε ουρά
Publisher	Εκδότης
Event	Συμβάν
Transformers	Μετατροπείς
Stack	Στοίβα

reusability	Επαναχρησιμοποίηση

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
VPN	Virtual Private Network
HOT	Heat Orchestration Template
NAT	Network Address Translation
VPNaaS	Virtual Private Network as a Service
DoS	Denial of Service
RPC	Remote Procedure Call
API	Application Programming Interface
YAML	Yaml Ain’t another Markup Language

ΑΝΑΦΟΡΕΣ

[1] Margaret Rouse. TechTarget. [Online]. http://searchcloudcomputing.techtarget.com/definition/cloud-computing

[2] Galvin, Gagne Silberschatz, *Operating System concepts*.

[3] Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia Michael Armbrust. (2010, April) Communications of the ACM. [Online]. http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext

[4] Aissaoui, Sefraoui Eleuldj. (2012, Οκτώβριος) [Online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.245.229&rep=rep1&type=pdf

[5] Grance Mell. [Online]. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[6] Rackspace Support. (2013, Οκτώβριος) Rackspace. [Online]. http://www.rackspace.com/knowledge_center/whitepaper/understanding-the-cloud-computing-stack-saas-paas-iaas

[7] appenda. [Online]. http://apprenda.com/library/paas/iaas-paas-saas-explained-compared/

[8] Opensource. [Online]. http://opensource.com/resources/what-is-openstack

[9] Ali Llewellyn. (2012, Ιούνιος) Open Nasa. [Online]. http://open.nasa.gov/blog/2012/06/04/nebula-nasa-and-openstack/

[10] Wikipedia. [Online]. https://en.wikipedia.org/wiki/OpenStack

[11] Bhanu Bhushan Parashar Rakesh Kumar. [Online]. http://www.researchgate.net/profile/Rakesh_Kumar175/publication/268405246_Dynamic_Resource_Allocation_and_Management_Using_OpenStack/links/546af6290cf2397f783021b9.pdf

[12] OpenStack. (2012) [Online]. http://docs.openstack.org/developer/nova/nova.concepts.html

[13] SwiftStack. SwiftStack. [Online]. https://swiftstack.com/openstack-swift/

[14] OpenStack - Swift. OpenStack. [Online]. http://docs.openstack.org/admin-guide-cloud/content/section_objectstorage-intro.html

[15] Openstack - Cinder. [Online]. http://docs.openstack.org/admin-guide-cloud/content/managing-volumes.html

[16] Openstack - Neutron. [Online]. http://docs.openstack.org/admin-guide-cloud/content/section_networking-intro.html

[17] OpenStack. Openstack Admin Guide -- LBAAS. [Online]. http://docs.openstack.org/admin-guide-cloud/content/section_lbaas-overview.html

[18] Openstack - Horizon. Openstack developer's guide. [Online]. http://docs.openstack.org/developer/horizon/

[19] Openstack - Keystone. Openstack Developer's guide. [Online]. http://docs.openstack.org/developer/keystone/architecture.html#application-construction

[20] Openstack Keystone. Openstack Keystone Admin Guide. [Online]. http://docs.openstack.org/admin-guide-cloud/content/keystone-user-management.html

[21] Openstack Glance. Glance Developer's guide. [Online]. http://docs.openstack.org/developer/glance/

[22] Tom Fiefield et al., *Openstack Operations Guide*.: O'reilly.

[23] Openstack Ceilometer. Openstack Admin guide Ceilometer. [Online]. http://docs.openstack.org/admin-guide-cloud/content/section_telemetry-introduction.html

[24] Openstack Heat. Wiki Heat. [Online]. https://wiki.openstack.org/wiki/Heat

[25] Openstack Trove. Wiki Trove. [Online]. https://wiki.openstack.org/wiki/Trove

[26] Openstack Trove Architecture. openstack. [Online]. https://wiki.openstack.org/wiki/TroveArchitecture

[27] Openstack neutron wiki. wiki.openstack. [Online]. https://wiki.openstack.org/wiki/Neutron#What_is_Neutron.3F

[28] Openstack Neutron Architecture. Openstack. [Online]. http://docs.openstack.org/security-guide/content/networking-architecture.html

[29] Openstack example Arch Neutron. openstack. [Online]. http://docs.openstack.org/openstack-ops/content/example_architecture.html

[30] Openstack ML2. Wiki ML2. [Online]. https://wiki.openstack.org/wiki/Neutron/ML2

[31] James Denton, *Learining Openstack Networking (Neutron)*., 2014.

[32] Openstack LbaaS Overview. openstack. [Online]. http://docs.openstack.org/admin-guide-cloud/content/section_lbaas-overview.html

[33] Openstack LbaaS API. wiki.openstack.org. [Online]. https://wiki.openstack.org/wiki/Neutron/LBaaS/API

[34] Openstack. docs.openstack. [Online]. http://docs.openstack.org/developer/ceilometer/overview.html#objectives

[35] Ruslan Kiyanchuk. (2013, June) Mirantis. [Online]. https://www.mirantis.com/blog/openstack-metering-using-ceilometer/

[36] Ceilometer. ceilometer.readthedocs.org. [Online]. https://ceilometer.readthedocs.org/en/latest/architecture.html

[37] Openstack Ceilometer Collecting Data. docs.openstack.org. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#how-is-data-collected

[38] Openstack Ceilometer Listening for data. docs.openstack.org. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#notification-agents-listening-for-data

[39] Openstack Ceilometer collecting data. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#polling-agents-asking-for-data

[40] Openstack Ceilometer prosessing data. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#processing-the-data

[41] Openstack Ceilometer Storing data. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#storing-the-data

[42] Openstack Ceilometer evaluating data. [Online]. http://docs.openstack.org/developer/ceilometer/architecture.html#evaluating-the-data

[43] Openstack Heat. docs.opnestack.org. [Online]. http://docs.openstack.org/developer/heat/architecture.html

[44] Openstack Wiki Heat. [Online]. https://wiki.openstack.org/wiki/Heat

[45] Openstack Heat Hot guide. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

[46] Openstack Heat Template structure. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#template-structure

[47] Openstack Heat Parameters Group. [Online]. docs.openstack.org/developer/heat/template_guide/hot_spec.html#parameter-groups-section

[48] Openstack Ceilometer Parameter Constraints. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#parameter-constraints

[49] Openstack Heat Resources. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#resources-section

[50] Openstack Heat Outputs. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#outputs-section

[51] Openstack Heat Intrinsic Functions. [Online]. http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#hot-spec-intrinsic-functions

[52] Packstack Installation Guide. [Online]. https://rdoproject.org/Quickstart

[53] Pacstack with existing external Network. [Online]. https://www.rdoproject.org/Neutron_with_existing_external_network

[54] Openstack Community. [Online]. https://ask.openstack.org/en/question/24719/how-to-install-packstack-allinone-on-fedora-20-with-an-existing-external-network/

[55] AllThingsOpen. [Online]. http://allthingsopen.com/2013/08/23/openstack-packstack-

installation-with-external-connectivity/

[56] packstack Forum. [Online]. https://www.rdoproject.org/forum/discussion/comment/2007#Comment_2007

[57] Packstack Fedora 20 with External Network. [Online]. https://www.rdoproject.org/Fedora_20_with_existing_network

[58] Openstack Community. [Online]. https://ask.openstack.org/en/answers/25145/revisions/

[59] Openstack Install Guide Glance. [Online]. http://docs.openstack.org/havana/install-guide/install/apt/content/glance-verify.html

[60] Marco Berube. (2014, June) [Online]. http://www.marcoberube.com/archives/76

[61] Packstack Deploying Instances. [Online]. https://www.rdoproject.org/Running_an_instance

[62] Floating Ip. [Online]. https://www.rdoproject.org/Difference_between_Floating_IP_and_private_IP

[63] Sheetal Taneja, Aparna Datt Pratibha. [Online]. http://jimsindia.org/8i_Journal/VolII/Autoscaling-in-OpenStackCRC.pdf

[64] RedHat LBaaS Setup. (2013, Αύγουστος) myopensourcelife.com. [Online]. http://myopensourcelife.com/2013/08/08/openstack-load-balancer-as-a-service-lbaas/

[65] Openstack Heat Autoscaling. [Online]. https://wiki.openstack.org/wiki/Heat/AutoScaling

[66] Red Hat. (2013) myopensourcelife.com. [Online]. http://myopensourcelife.com/2014/09/13/autoscaling-in-openstack-using-heat-and-ceilometer-part-1/

[67] RedHat Autoscaling Part2. (2015, January) myopensourcelife.com. [Online]. http://myopensourcelife.com/2015/01/02/autoscaling-in-openstack-using-heat-and-ceilometer-part-2/

[68] Oracle High availability. [Online]. https://docs.oracle.com/cd/E15586_01/core.1111/e10106/intro.htm

[69] Openstack Hgh Availability. [Online]. http://docs.openstack.org/high-availability-guide/content/ch-intro.html

[70] Openstack Stateless Stateful. docs.openstack.org. [Online]. http://docs.openstack.org/high-availability-guide/content/stateless-vs-stateful.html

[71] TechTarget Stateless Stateful. [Online]. http://whatis.techtarget.com/definition/stateless

[72] Guillermo Antonio. (2015, Μάρτιος) [Online]. http://galvarado.com.mx/en/stateless-and-stateful-services-in-high-availability/

[73] RedHat HA in OpenStack. redhatstackblog. [Online]. http://redhatstackblog.redhat.com/2014/04/16/the-road-to-high-availability-for-openstack/

[74] Openstack Active Pasive. [Online]. http://docs.openstack.org/high-availability-guide/content/ap-intro.html

[75] Openstack Active Active. docs.openstack.org. [Online]. http://docs.openstack.org/high-availability-guide/content/aa-intro.html

[76] Piotr Siwczak. (2012, Σεπτέμβριος) mirantis. [Online]. https://www.mirantis.com/blog/understanding-options-deployment-topologies-high-availability-ha-openstack/