



ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΜΑΤΙΚΗΣ
ΠΜΣ "Εφαρμοσμένη Πληροφορική"

Διαχείριση Υπολογιστικού Νέφους

Εργασία Εξαμήνου

Καθηγητής: Φραγκιαδάκης Γεώργιος

Φωτεινόπουλος Παναγιώτης - 2021154

Εισαγωγή

Η παρούσα εργασία αφορά τη σχεδίαση, υλοποίηση, παρακολούθηση και κοστολόγηση μιας πολυεπίπεδης (two-tier) εφαρμογής σε υποδομή OpenStack DevStack. Η λύση βασίζεται σε αυτοματοποιημένη προετοιμασία μέσω cloud-init, σωστή δικτύωση (private/floating IP), παρακολούθηση πόρων (monitoring) και τεκμηριωμένη προσέγγιση chargeback.

Η αναπτυγμένη εφαρμογή, Order Details Dashboard System, αποτελείται από backend (Spring Boot με Java 17, RESTful API, PostgreSQL, Hibernate) και frontend (Vue3 σε Node.js 16+), με λειτουργίες authentication, dashboard παραγγελιών, role-based access και blacklisting JWT tokens.

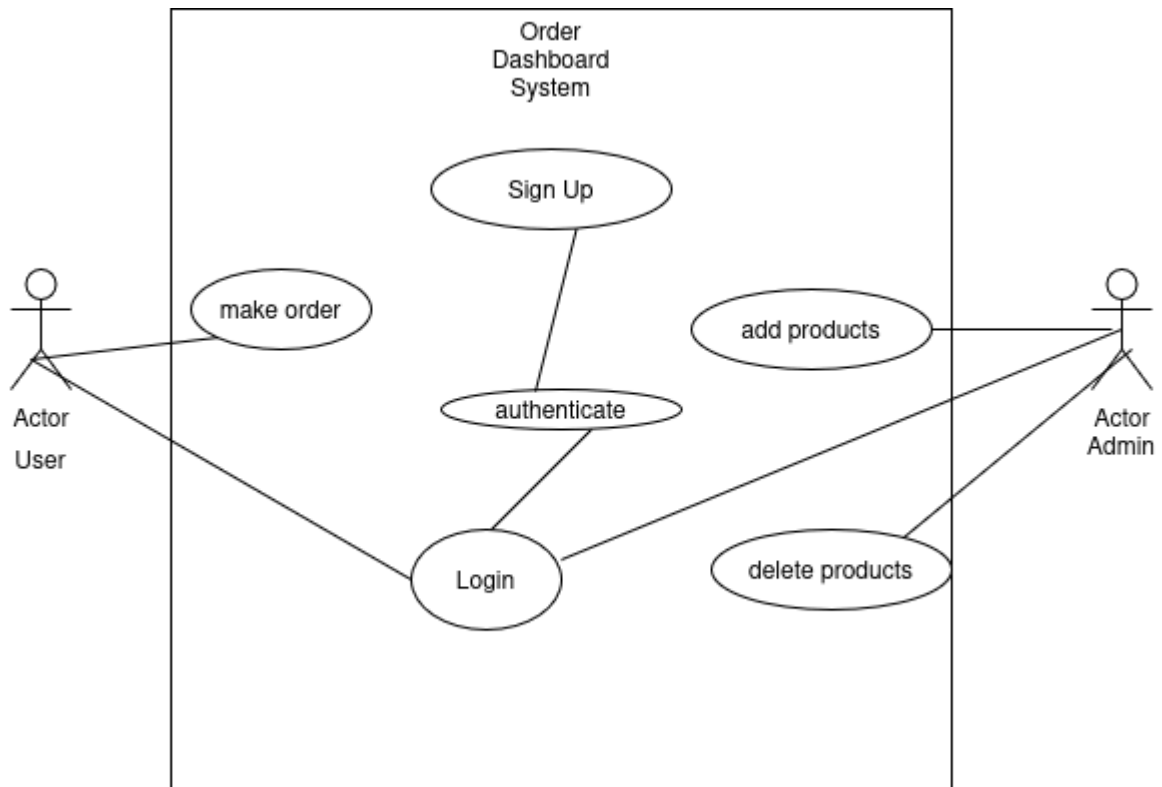
Περιγραφή Εφαρμογής

Η εφαρμογή είναι ένα Order Details Dashboard System το οποίο μπορεί ένας χρήστης (ρόλος User) να διαλέξει προϊόντα και να καταχωρίσει την παραγγελία του έχοντας την δυνατότητα να βλέπει και τι ποσότητα έχει απομείνει. Αντίστοιχα, ο ρόλος του Admin ανεβάζει νέα προϊόντα και ανανεώνει την ποσότητά τους. Το Backend έχει υλοποιηθεί με Spring Boot (Java 17), με χρήση RESTful API, με βάση δεδομένων PostgreSQL. Το Frontend έχει υλοποιηθεί με Vue3 (Node.js 16+). Έτσι, οι κύριες λειτουργίες είναι οι Αυθεντικοποίηση, dashboard παραγγελιών, role-based access, blacklisting JWT tokens.

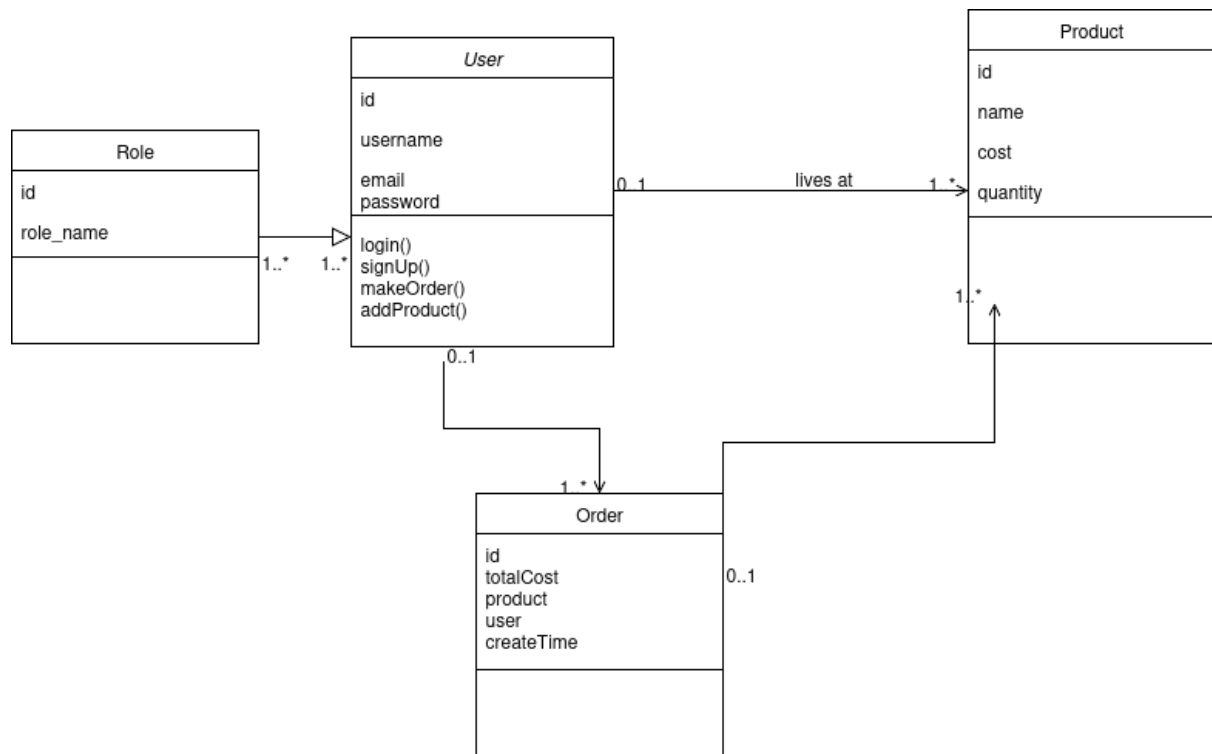
- Αποθετήριο της εφαρμογής: (<https://github.com/Panos994/Order-Dashboard-System.git>)
- Αποθετήριο του μαθήματος: (https://github.com/Panos994/DevStack_DIT266.git)

Διαγράμματα Αρχιτεκτονικής

Use Case Εφαρμογής

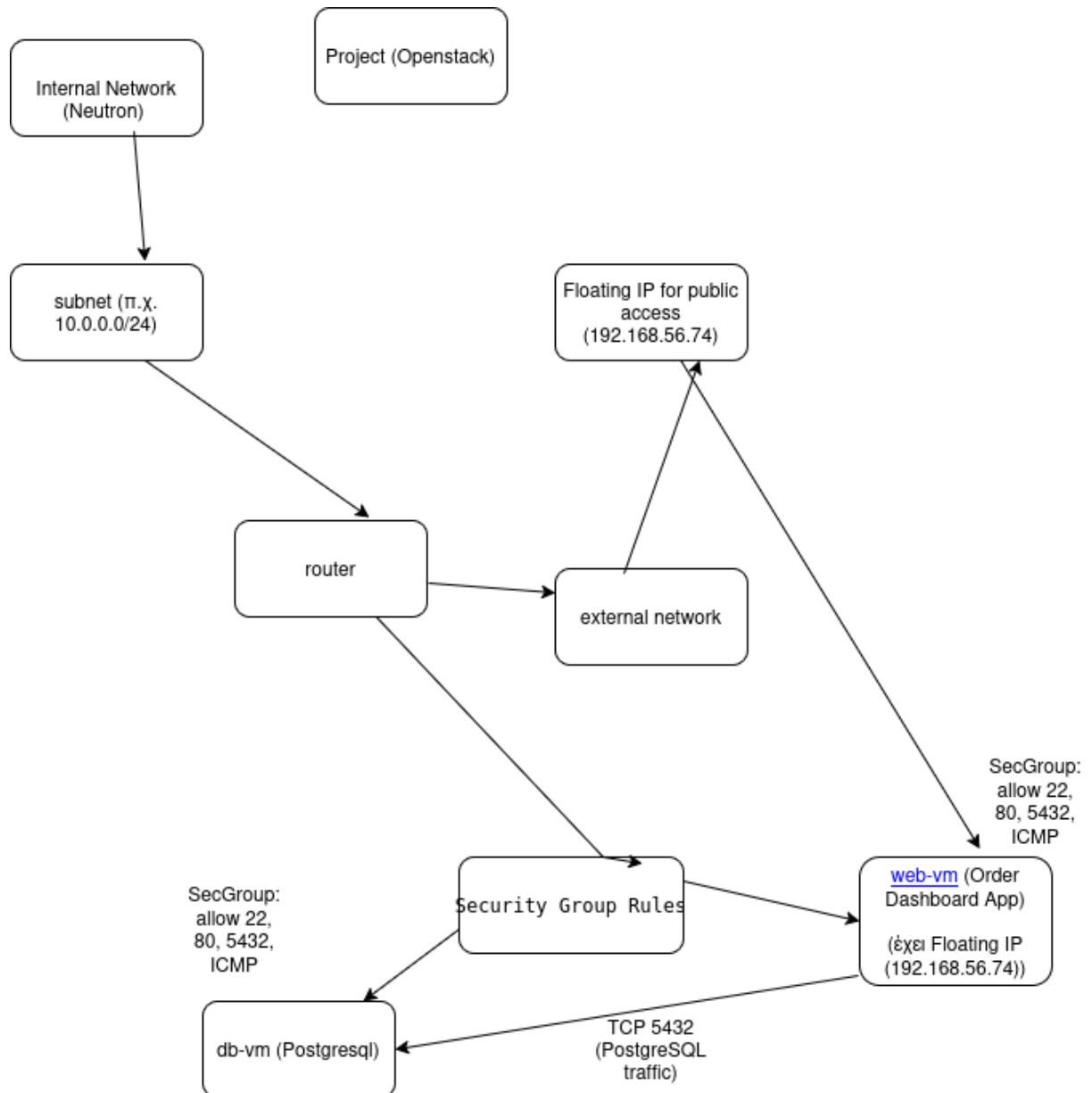


Class Diagram Εφαρμογής



Διάγραμμα Αρχιτεκτονικής για Openstack

- Project/tenant → Network → Subnet → Router → VMs (Web/API, DB)
- Floating IP προς Web VM
- Security group rules



Υλοποίηση Βημάτων

Το αρχείο local.conf:

```
[[local|localrc]]

# Host IP address
HOST_IP=$(hostname -I | awk '{print $1}')

# Floating IPs: make sure this subnet is not in use on your local
network
FLOATING_RANGE=192.168.56.0/24
PUBLIC_NETWORK_GATEWAY=192.168.56.1
Q_FLOATING_ALLOCATION_POOL=start=192.168.56.50,end=192.168.56.200
PUBLIC_INTERFACE=enp0s8
Q_USE_PROVIDERNET_FOR_PUBLIC=True

# Internal (fixed) network for VM communication
FIXED_RANGE=10.0.0.0/24

# Set common passwords
ADMIN_PASSWORD=stackpass
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD

# Enable services
enable_service h-eng h-api h-api-cfn h-api-cw
enable_service horizon
enable_service s-account s-container s-object s-proxy

# Reduce disk usage for logs
LOGDAYS=1

# Disable rate limiting
API_RATE_LIMIT=False
```

```
# Destination directory for DevStack
DEST=/opt/stack

# Swift specific config
SWIFT_HASH=$(echo $RANDOM | md5sum | head -c 30)
SWIFT_REPLICAS=1

# Logging (optional)
LOGFILE=$DEST/logs/stack.sh.log

# Enable Gnocchi plugin (storage backend)
enable_plugin gnocchi https://opendev.org/openstack/gnocchi.git

# Enable Ceilometer devstack plugin
enable_plugin ceilometer https://opendev.org/openstack/ceilometer.git

# (ΠΡΟΣΟΧΗ: το παρακάτω είναι απαραίτητο)
CEILOMETER_BACKEND=gnocchi

# (Προαιρετικό, Aodh για alarms)
# enable_plugin aodh https://opendev.org/openstack/aodh.git
```

1. Ρύθμιση Project και Δικτύου

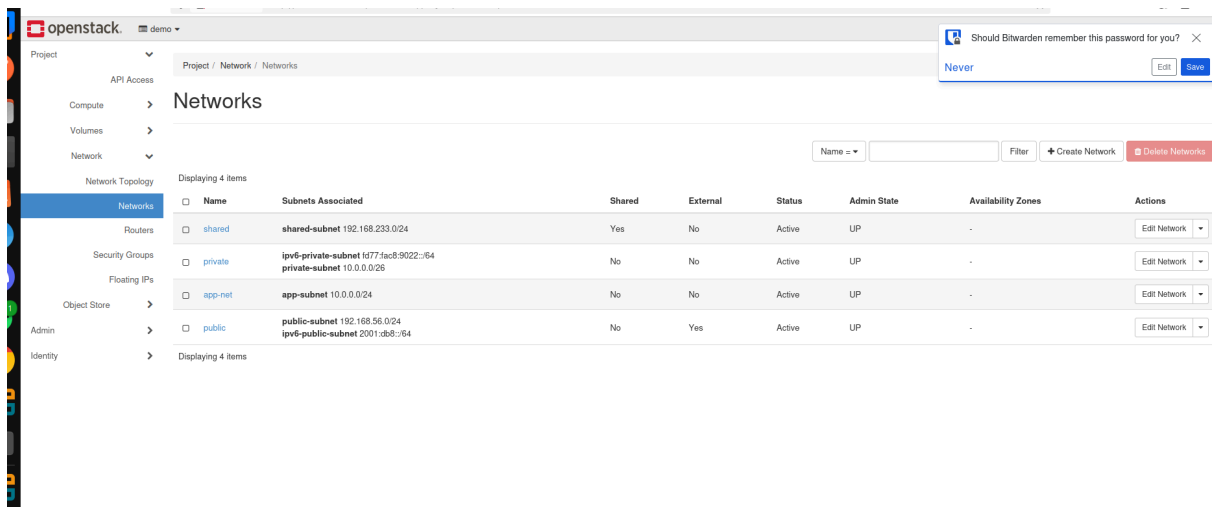
Ορίστηκε ιδιωτικό δίκτυο και subnet αλλά και router για έξοδο στο public/external δίκτυο:

```
openstack project create cloudapp

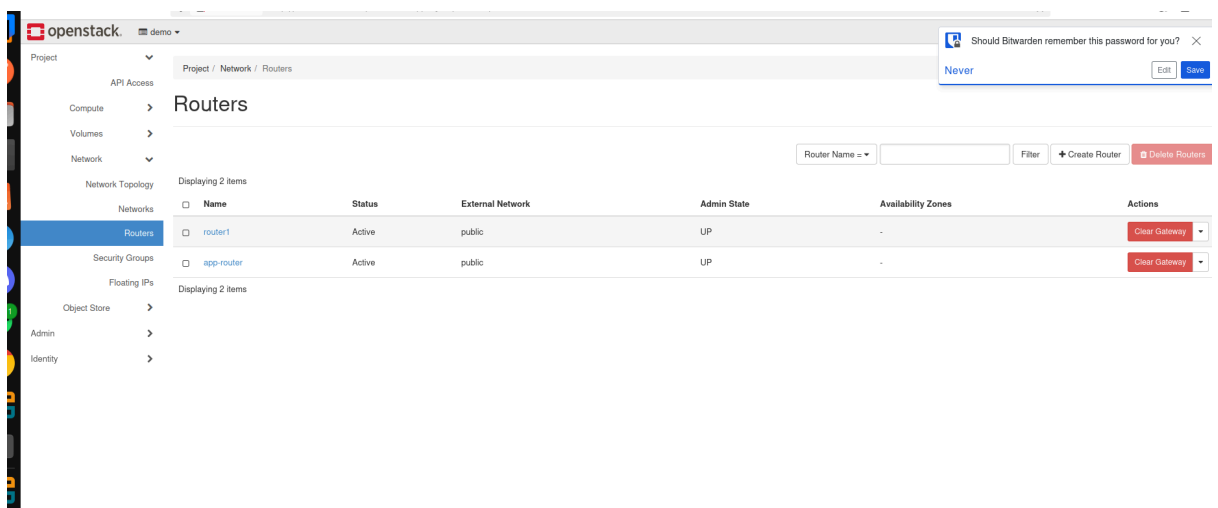
# Δημιουργία δικτύου & subnet
openstack network create app-net
openstack subnet create --network app-net --subnet-range 10.0.0.0/24
app-subnet

# Router & σύνδεση στο public/external
openstack router create app-router
openstack router set app-router --external-gateway public
openstack router add subnet app-router app-subnet
```

Με αυτό τον τρόπο Διαχωρίζουμε το εσωτερικό δίκτυο (app-net) από το public, διασφαλίζοντας πρόσβαση μόνο μέσω floating IP και δρομολογητή (router).



Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
shared	shared-subnet 192.168.233.0/24	Yes	No	Active	UP	-	Edit Network
private	ipv6-private-subnet fd77:fac8:9022::/64 private-subnet 10.0.0.0/26	No	No	Active	UP	-	Edit Network
app-net	app-subnet 10.0.0.0/24	No	No	Active	UP	-	Edit Network
public	public-subnet 192.168.56.0/24 ipv6-public-subnet 2001:db8::/64	No	Yes	Active	UP	-	Edit Network



Name	Status	External Network	Admin State	Availability Zones	Actions
router1	Active	public	UP	-	Clear Gateway
app-router	Active	public	UP	-	Clear Gateway

Ορίστηκε security group με τα απαραίτητα rules (SSH, HTTP, PostgreSQL, ICMP):

```
openstack security group create webapp-sg --description "Allow SSH, HTTP, PostgreSQL, Ping"
openstack security group rule create --proto tcp --dst-port 22 webapp-sg
```



```
openstack security group rule create --proto tcp --dst-port 80 webapp-sg
openstack security group rule create --proto tcp --dst-port 5432
webapp-sg
openstack security group rule create --proto icmp webapp-sg
```

Επεξήγηση:

Ορίζουμε πολιτικές ασφάλειας (firewall) για να επιτρέπονται μόνο οι απαιτούμενες υπηρεσίες.

The screenshot shows the OpenStack dashboard interface. The left sidebar contains navigation links for Project, API Access, Compute, Volumes, Network, Network Topology, Networks, Routers, Security Groups, Floating IPs, Object Store, Admin, and Identity. The main content area is titled 'Manage Security Group Rules: webapp-sg (219eb038-a28b-4d2e-a35b-5f5d2ed6a8be)'. It features a table with 6 rules and buttons for '+ Add Rule' and 'Delete Rules'.

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	:::0	-	-	Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	5432	0.0.0.0/0	-	-	Delete Rule

1.2 Μετέπειτα πρέπει να έχει δημιουργηθεί ένα Keypair για SSH στα Instances αν δεν υπάρχει:

```
# Δημιουργία SSH key
ssh-keygen -t rsa -b 4096

# Προσθήκη του δημόσιου κλειδιού στο OpenStack ώστε να τοποθετείται αυτόματα
στα νέα VMs
openstack keypair create --public-key /opt/stack/.ssh/id_rsa.pub mykey
```

Επεξήγηση:

Το keypair επιτρέπει ασφαλή SSH πρόσβαση στα VMs που θα δημιουργηθούν. Στόχος είναι να διαχωριστεί η εσωτερική υποδομή από το public network, και να επιτρέπεται μόνο η ελάχιστη απαιτούμενη πρόσβαση, εξασφαλίζοντας ασφάλεια.

2. Εξυπηρετητής Βάσης (DB)

Δημιουργία Instances με cloud-init (Automation Scripts)

Δημιουργήθηκαν δύο cloud-init scripts:

- db-init.yaml: Αυτόματη εγκατάσταση PostgreSQL, δημιουργία βάσης και χρήση, άνοιγμα θύρας 5432.
- web-init.yaml: Εγκατάσταση Java/Maven/Node.js, git clone & build εφαρμογής, εκκίνηση backend/frontend, άνοιγμα θυρών 80/8080.

Τα scripts παρατίθενται στο παράρτημα και τοποθετήθηκαν στο repo για αναπαραγωγή.

Να τα στείλω στο git hub repo (Τα αρχεία αυτά παρατίθενται στο Παράρτημα.*) !!!!!!!!!

Εκκίνηση VM (Database Server)

```
# Δημιουργία volume για επίμονη αποθήκευση (optional)
openstack volume create --size 5 db-volume

# Εκκίνηση DB instance με cloud-init
openstack server create \
  --flavor m1.small \
  --image ubuntu-22.04 \
  --network app-net \
  --key-name mykey \
  --security-group webapp-sg \
  --user-data db-init.yaml \
  db-vm

# Επισύναψη volume (optional)
openstack server add volume db-vm db-volume
```

Επεξήγηση:

To script `db-init.yaml` αυτοματοποιεί πλήρως την εγκατάσταση PostgreSQL και τις αρχικές ρυθμίσεις.

```
#cloud-config
package_update: true
package_upgrade: true
packages:
  - postgresql
  - postgresql-contrib
runcmd:
  - sudo -u postgres psql -c "CREATE DATABASE order_system_db;"
  - sudo -u postgres psql -c "CREATE USER dbuser WITH ENCRYPTED PASSWORD 'pass123';"
  - sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE order_system_db TO dbuser;"
  - sudo sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*' /g" /etc/postgresql/*/main/postgresql.conf
  - echo "host all all 10.0.0.0/24 md5" | sudo tee -a
```

```
/etc/postgresql/*/main/pg_hba.conf
- sudo systemctl restart postgresql
- sudo ufw allow 5432/tcp
- sudo ufw --force enable
```

Επεξήγηση εντολών:

- #cloud-config

Δηλώνει ότι το αρχείο είναι τύπου cloud-init και θα ερμηνευτεί αυτοματοποιημένα από το cloud-init agent του Ubuntu κατά το πρώτο boot.

- package_update: true

Εκτελεί `apt update` για να ενημερώσει τα repositories.

- package_upgrade: true

Εκτελεί `apt upgrade` για να αναβαθμίσει τα ήδη εγκατεστημένα πακέτα στην τελευταία έκδοση.

- packages:

Εγκαθιστά τα πακέτα `postgresql` και `postgresql-contrib` (την επίσημη βάση δεδομένων PostgreSQL και επιπλέον εργαλεία).

- runcmd:

Εκτελεί μια σειρά από εντολές bash (μετά το τέλος της εγκατάστασης πακέτων και πριν ολοκληρωθεί το cloud-init):

1. `sudo -u postgres psql -c "CREATE DATABASE order_system_db;`

Δημιουργεί μια νέα βάση δεδομένων με όνομα `order_system_db` ως χρήστης postgres.

2. `sudo -u postgres psql -c "CREATE USER dbuser WITH ENCRYPTED PASSWORD 'pass123';"`

Δημιουργεί χρήστη `dbuser` με password `pass123` για την πρόσβαση στη βάση.

3. `sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE order_system_db TO dbuser;`

Δίνει όλα τα δικαιώματα στη βάση για τον χρήστη `dbuser`.

4. `sudo sed -i "s/^#listen_addresses =.*/listen_addresses = '*'/"
/etc/postgresql/*/main/postgresql.conf`

Επεξεργάζεται το αρχείο ρυθμίσεων της PostgreSQL για να δέχεται συνδέσεις από οποιαδήποτε διεύθυνση IP (`listen_addresses = '*').

5. `sudo bash -c "echo \"host all all 0.0.0.0/0 md5\" >> /etc/postgresql/*/main/pg_hba.conf"`

Προσθέτει στο `pg_hba.conf` κανόνα που επιτρέπει σε οποιονδήποτε host να συνδέεται στη βάση, αρκεί να δίνει έγκυρο username/password (md5).

6. `sudo systemctl restart postgresql`

Κάνει επανεκκίνηση της υπηρεσίας PostgreSQL ώστε να εφαρμοστούν οι αλλαγές στις ρυθμίσεις.

3. Εξυπηρετητής Web/API (web)

Εκκίνηση VM (Web/API server)

```
openstack server create \  
  --flavor m1.small \  
  --image ubuntu-22.04 \  
  --network app-net \  
  --key-name mykey \  
  --security-group webapp-sg \  
  --user-data web-init.yaml \  
  web-vm
```

Επεξήγηση:

To script `web-init.yaml` εγκαθιστά Java, Maven, Node.js, κάνει clone το repo, κάνει build και ξεκινά backend/frontend.

```
#cloud-config
package_update: true
package_upgrade: true
packages:
  - openjdk-17-jdk
  - maven
  - git
  - curl
  - npm
runcmd:
  - git clone https://github.com/Panos994/Order-Dashboard-System.git
  - cd Order-Dashboard-System && mvn spring-boot:run &
  # Εναλλακτικά, build και run jar αν θέλουμε
  # - cd Order-Dashboard-System && mvn package && java -jar target/*.jar
  &
  # Προσαρμογή PORT αν χρειάζεται
  - cd Order-Dashboard-System/order_system_frontend && npm install &&
  npm run serve &
  - sudo ufw allow 80/tcp
  - sudo ufw allow 8080/tcp
  - sudo ufw --force enable
```

Επεξήγηση εντολών:

- #cloud-config
Δηλώνει αρχείο cloud-init.
- package_update: true
Εκτελεί `apt update` για να ενημερώσει τα πακέτα.
- package_upgrade: true
Εκτελεί `apt upgrade`.

- packages:

Εγκαθιστά τα παρακάτω πακέτα:

- openjdk-17-jdk: Java 17 (απαιτείται για το backend/Spring Boot)
- maven: Εργαλείο build για Java/Maven project
- git: Για να γίνει clone το repository της εφαρμογής
- curl: Χρήσιμο για debugging/HTTP requests
- nodejs και npm: Απαιτούνται για το frontend (Vue3 app)

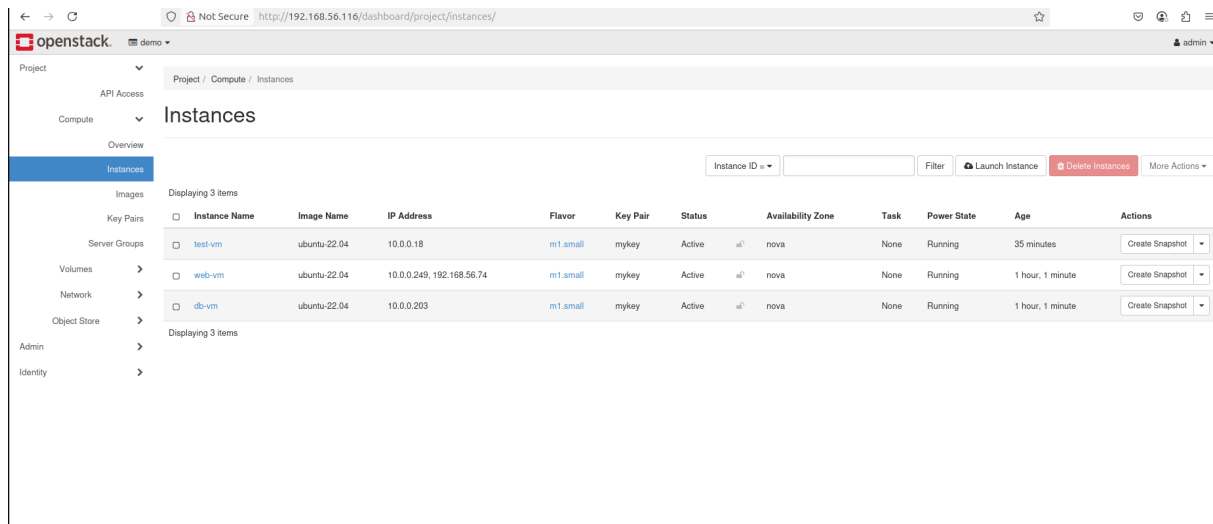
- runcmd:

Εκτελεί εντολές για να φέρει και να εκκινήσει την εφαρμογή:

1. `git clone https://github.com/Panos994/Order-Dashboard-System.git`
`/home/ubuntu/order-dashboard`
clone το repository της εφαρμογής σε τοπικό φάκελο.

2. ``cd /home/ubuntu/order-dashboard && mvn spring-boot:run &``
Μεταβαίνει στον φάκελο backend και ξεκινά την Java Spring Boot εφαρμογή στο background.

3. ``cd /home/ubuntu/order-dashboard/order_system_frontend && npm install && npm run serve &``
Μεταβαίνει στον φάκελο frontend, εγκαθιστά όλες τις εξαρτήσεις (``npm install``) και εκκινεί το Vue3 frontend server (``npm run serve``) επίσης στο background.



4. Απόδοση Floating IP

```
# Δημιουργία floating IP
FLOATING_IP=$(openstack floating ip create public -f value -c
floating_ip_address)

# Αντιστοίχιση floating IP στο web-vm
openstack server add floating ip web-vm $FLOATING_IP
```

Επεξήγηση:

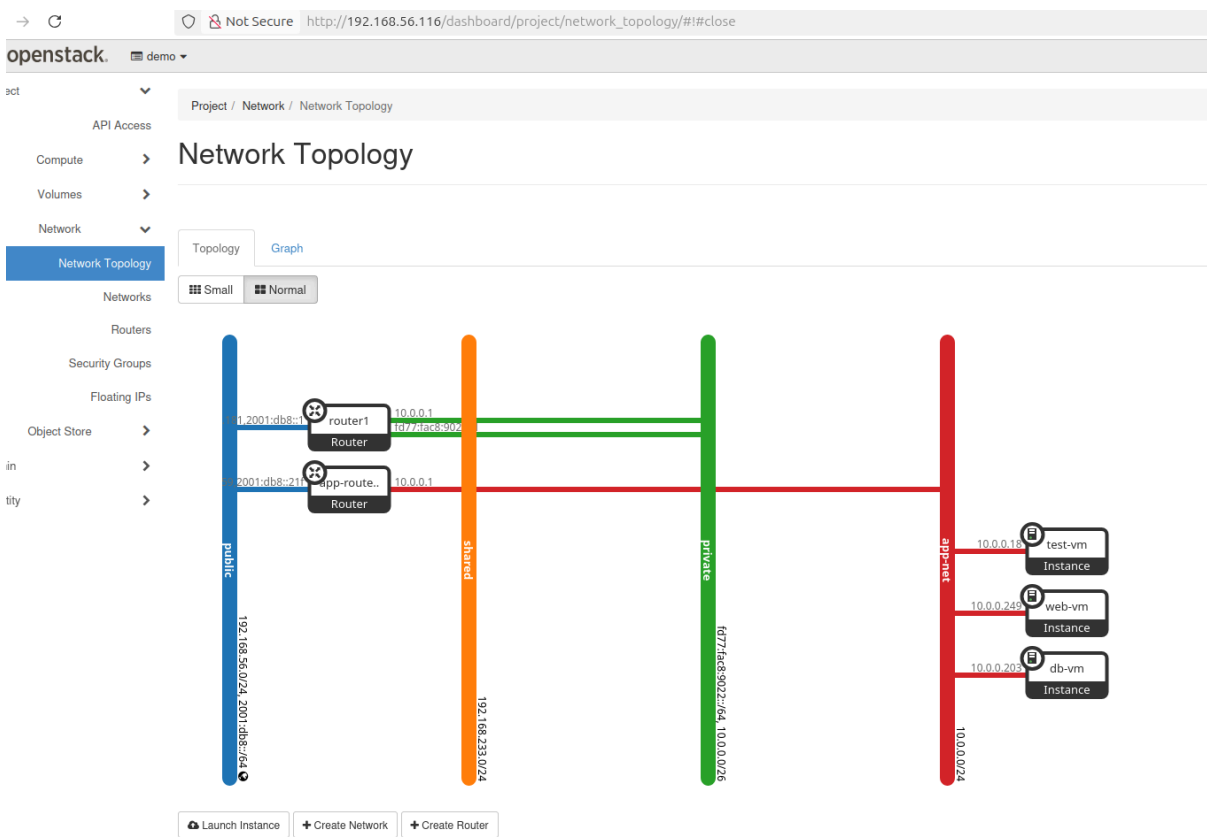
Το web-vm γίνεται προσβάσιμο από το Internet μέσω της floating IP.

SSH Tunnel για PostgreSQL

- Για απομακρυσμένη πρόσβαση στη βάση, δημιουργείται SSH tunnel:
sh

```
ssh -L 5432:localhost:5432 ubuntu@<web-vm-floating-ip>
```

Έτσι, η τοπική πόρτα 5432 δρομολογείται μέσω του web-vm στο db-vm.



5. Παρακολούθηση (Διάγνωση και Επίλυση Προβλημάτων SSH και cloud-init)

Πρόβλημα:

Κατά την προσπάθεια σύνδεσης μέσω SSH στο instance (π.χ. `ssh ubuntu@192.168.56.74`), λάμβανα το μήνυμα:

```
ssh: connect to host 192.168.56.74 port 22: Connection refused
```

Το συγκεκριμένο μήνυμα σημαίνει ότι το VM απαντά στην IP, αλλά η υπηρεσία SSH (sshd) δεν είναι ενεργή ή δεν έχει εγκατασταθεί καθόλου. Στην δικιά μου περίπτωση δεν ίσχυε τίποτα από τα παραπάνω.

Έλεγχος cloud-init logs:

Ελέγχοντας τα αρχεία `/var/log/cloud-init.log` και `/var/log/cloud-init-output.log` μέσω της κονσόλας Horizon, διαπίστωσα τα παρακάτω:

- Εμφανίζεται το μήνυμα:

```
Datasource DataSourceNone.
```

- Δεν εκτελείται κανένα custom cloud-init script (`--user-data`), δηλαδή δεν εγκαθίσταται ούτε ενεργοποιείται ο SSH server, ούτε άλλα πακέτα/ρυθμίσεις.

Πιθανή Αιτία:

Το cloud-init δεν έλαβε καθόλου user-data script, επειδή:

- Το flag `--user-data` είτε δεν χρησιμοποιήθηκε είτε δόθηκε λάθος path κατά τη δημιουργία του instance.
- Το αρχείο cloud-init δεν στάλθηκε ή δεν βρέθηκε κατά το deployment.
- Ενδεχομένως χρησιμοποιήθηκε image που δεν υποστηρίζει cloud-init.

Τι σημαίνει το DataSourceNone:

Το μήνυμα `DataSourceNone` στα logs του cloud-init σημαίνει ότι το instance δεν έλαβε καθόλου metadata/user-data κατά το startup. Αυτό έχει ως αποτέλεσμα να μην εκτελούνται οι αυτοματοποιημένες ενέργειες ρύθμισης (όπως η εγκατάσταση και ενεργοποίηση SSH server).

Επίλυση:

1. Σωστή δημιουργία cloud-init αρχείου:

```
#cloud-config
package_update: true
packages:
  - openssh-server
runcmd:
  - systemctl enable ssh
  - systemctl start ssh
```

2. Κατά τη δημιουργία του instance, χρήση του σωστού `--user-data` (με ολόκληρο το path του αρχείου):

```
openstack server create \
  --flavor m1.small \
  --image ubuntu-22.04 \
  --network app-net \
  --key-name mykey \
  --security-group webapp-sg \
  --user-data /home/stack/ssh-init.yaml \
  test-vm
```

3. Επιβεβαίωση εκτέλεσης cloud-init:

- Έλεγχος των logs (`cloud-init-output.log`) μετά το boot για να διαπιστωθεί ότι εκτελέστηκαν οι εντολές εγκατάστασης και ενεργοποίησης SSH.
- Όλα εκτελέστηκαν σωστά, η σύνδεση SSH λειτούργησε κανονικά.

```
stack@it2021154:~/devstack$ ssh ubuntu@192.168.56.74
The authenticity of host '192.168.56.74 (192.168.56.74)' can't be established.
ED25519 key fingerprint is SHA256:/BYX4xSzJNE3FF5a7Y6Q/kS1UjXg9kum/w3eea0xIN8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.74' (ED25519) to the list of known hosts.

Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information disabled due to load higher than 1.0

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@web-vm:~$
```

Ωστόσο, κάθε φορά που γινόταν η σύνδεση δυστυχώς στα instances γινόταν freeze της κονσόλας χωρίς να μπορώ να κάνω κάποια επόμενη ενέργεια. Αυτό με έφτασε στο σημείο για την συνέχεια της εργασίας να εργαστώ με υποθετικά σενάρια.

Έτσι, ανεξαρτήτως των δικτυακών προβλημάτων μπορούμε να δούμε αφού είναι Running τα VMs μου μετρήσεις όπως παρακάτω με το ceilometer:

<input type="checkbox"/> web-vm	ubuntu-22.04	10.0.0.249, 192.168.56.74	m1.small	mykey	Active		nova	None	Running	1 week, 3 days	Create Snapshot ▾
<input type="checkbox"/> db-vm	ubuntu-22.04	10.0.0.203	m1.small	mykey	Active		nova	None	Running	1 week, 3 days	Create Snapshot ▾

Displaying 3 items

Πληροφορίες Instance → web-vm

- Resource ID: 043bd164-775d-4c3c-a2be-5dc467bb59fb

```
(venv) stack@lt2021154:~/devstack$ openstack metric resource show 043bd164-775d-4c3c-a2be-5dc467bb59fb
```

Field	Value
id	043bd164-775d-4c3c-a2be-5dc467bb59fb
creator	60e5fcf042714ce8bc75ce4cb367fe4c:c7c61e5acf7b4d2f8e7ca398e6f8ddb1
started_at	2025-06-29T12:53:40.637922+00:00
revision_start	2025-06-29T13:00:55.635983+00:00
ended_at	None
user_id	1794d0665bc14296b28497fec75cedff
project_id	7685d1ca8303462697641865e3243c9b
original_resource_id	043bd164-775d-4c3c-a2be-5dc467bb59fb
type	instance
revision_end	None
metrics	compute.instance.booting.time: 4d6a7e5e-6f5e-440e-bc53-e796e4f86f6e cpu: f9a540b3-937c-4ec9-b6dd-945924bddd91 disk.ephemeral.size: 5973484f-a18f-4c2f-bd28-369392854be7 disk.root.size: 94ad8977-dc6c-4b6c-9e2d-387b872d90e5 memory.resident: 47c598c8-144c-419c-9d56-b796665cef3a memory.swap.in: 8e1c3dce-4ee6-45c1-b188-039f3cc61bc2 memory.swap.out: 8bca2da9-4e51-455d-9ff7-41a7a0a7f908 memory.usage: 827cddf3-e083-4cc6-b786-97ece4902bf4 memory: 59379c98-d1df-4fa5-b71d-a4c3050a72b power.state: eef1b051-ad7d-4891-a3ce-24306ff67a4e vcpus: 48169e5e-1257-401f-8600-cd4496f55a63
created_by_user_id	60e5fcf042714ce8bc75ce4cb367fe4c
created_by_project_id	c7c61e5acf7b4d2f8e7ca398e6f8ddb1

CPU (vCPUs)

Για real-time παρακολούθηση του instance `web-vm`, χρησιμοποιήθηκε το resource id:

****043bd164-775d-4c3c-a2be-5dc467bb59fb****

Δείγματα CPU usage (με Gnocchi CLI)

openstack metric measures show f9a540b3-937c-4ec9-b6dd-945924bddd91

timestamp	granularity	value
2025-06-29T12:55:00+00:00	300.0	286820000000.0
2025-06-29T13:00:00+00:00	300.0	577640000000.0
2025-06-29T13:05:00+00:00	300.0	872810000000.0
2025-06-29T13:10:00+00:00	300.0	960030000000.0

RAM (MB)

Δείγματα χρήσης μνήμης

openstack metric measures show 827cddf3-e003-4cc6-b786-97ece4902bf4

timestamp	granularity	value
2025-06-29T12:55:00+00:00	300.0	250.91796875
2025-06-29T13:00:00+00:00	300.0	273.09765625
2025-06-29T13:05:00+00:00	300.0	275.4609375
2025-06-29T13:10:00+00:00	300.0	304.8984375

```
### Storage (GB) disk.root.size
openstack metric measures show 94ad8977-dc6c-4b6c-9e2d-387b872d90e5
```

timestamp	granularity	value
2025-06-29T12:50:00+00:00	300.0	20.0
2025-06-29T12:55:00+00:00	300.0	20.0
2025-06-29T13:00:00+00:00	300.0	20.0
2025-06-29T13:05:00+00:00	300.0	20.0
2025-06-29T13:10:00+00:00	300.0	20.0

Real-time παρακολούθηση (ανά 10 δευτερόλεπτα)

```
watch -n 10 "openstack metric measures show
f9a540b3-937c-4ec9-b6dd-945924bddd91"
```

```
Every 10.0s: openstack metric measures s... it2021154: Sun Jun 29
13:22:16 2025
```

timestamp	granularity	value
2025-06-29T12:55:00+00:00	300.0	286820000000.0
2025-06-29T13:00:00+00:00	300.0	577640000000.0
2025-06-29T13:05:00+00:00	300.0	872810000000.0
2025-06-29T13:10:00+00:00	300.0	960030000000.0
2025-06-29T13:15:00+00:00	300.0	969970000000.0

- Το OpenStack Ceilometer καταγράφει επακριβώς τους πόρους κάθε VM: vCPU, RAM, Disk.
- Τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν για monitoring ή chargeback/billing.

Τα παραπάνω outputs προέρχονται από πραγματικό περιβάλλον DevStack με ενεργό Ceilometer/Gnocchi.

6. Ζητήματα Συνδεσιμότητας & Υποθετικό Σενάριο Δοκιμής Εφαρμογής

Πίνακας κόστους (Chargeback)

Κατά τη διάρκεια των δοκιμών με το OpenStack DevStack περιβάλλον, αντιμετώπισα σημαντικά προβλήματα συνδεσιμότητας λόγω έλλειψης πρόσβασης στο Internet από τα instances. Αυτό είχε ως αποτέλεσμα:

- Η σύνδεση SSH προς τα instances ήταν εφικτή ελάχιστες φορές, συχνά διακοπτόταν ή δεν ήταν δυνατή.
- Η απουσία Internet εντός των VMs δυσκόλεψε ή ακύρωσε την εγκατάσταση απαραίτητων πακέτων και εργαλείων (π.χ. μέσω `apt`, `git`, `npm`).
- Δοκιμές που απαιτούσαν κατέβασμα εξαρτήσεων ή updates (όπως build εφαρμογής, λήψη βιβλιοθηκών ή ενημερώσεις λειτουργικού) δεν ήταν δυνατό να πραγματοποιηθούν απευθείας μέσα στα instances. Έτσι, προσπάθησα να δημιουργήσω ένα υποθετικό σενάριο για την συνέχεια της εργασίας.

Υποθετικό σενάριο πλήρους δοκιμής (αν υπήρχε Internet):

Εάν το περιβάλλον ήταν πλήρως λειτουργικό και υπήρχε απρόσκοπτη πρόσβαση στο Internet, το σενάριο δοκιμής της εφαρμογής μου θα περιλάμβανε τα εξής βήματα:

1. Απομακρυσμένη σύνδεση στα instances μέσω SSH (με σταθερή σύνδεση).
2. Ενημέρωση των λειτουργικών συστημάτων:


```
sudo apt update  
sudo apt upgrade -y
```

3. Εγκατάσταση όλων των απαραίτητων πακέτων (π.χ. Java, Maven, Node.js, npm, git, PostgreSQL κ.λπ.).

4. clone του αποθετηρίου της εφαρμογής από το GitHub:

```
git clone https://github.com/Panos994/Order-Dashboard-System.git
```

5. Εγκατάσταση εξαρτήσεων frontend/backend και εκτέλεση build:

- Για το backend (Spring Boot):

```
cd Order-Dashboard-System  
mvn spring-boot:run &
```

- Για το frontend (Vue.js):

```
cd order_system_frontend  
npm install  
npm run serve &
```

6. Ρύθμιση σύνδεσης με βάση δεδομένων..

7. Έλεγχος λειτουργικότητας εφαρμογής μέσω web browser ή με curl.

8. Παρακολούθηση συστήματος (monitoring):Χρήση εντολών όπως `top`, `vmstat`, `free -h`, `df -h` για έλεγχο πόρων.

9. Τελική επιβεβαίωση σωστής λειτουργίας όλων των υπηρεσιών και δυνατότητας απομακρυσμένης διαχείρισης.

Η απουσία πρόσβασης στο Internet και οι διακοπές στη συνδεσιμότητα αποτέλεσαν σημαντικό περιορισμό για την ολοκλήρωση της εργασίας στο πραγματικό DevStack περιβάλλον. Τα παραπάνω βήματα περιγράφουν το ιδανικό σενάριο λειτουργίας και δοκιμής της εφαρμογής, εφόσον το περιβάλλον δικτύου ήταν πλήρως προσβάσιμο.

Υποθετικό Chargeback – Πίνακας Κόστους

Εφόσον έχω ήδη τα real metrics για Disk, RAM, και CPU usage από τα OpenStack Gnocchi measures προχώρησα σε προσομοίωση χρέωσης (chargeback) με script που διαβάζει αυτά τα δεδομένα.

Κάποιες Υποθέσεις:

- **Disk: 20 GB (σταθερό)**
- **RAM: θα υπολογιστεί ο μέσος όρος των values (σε MB, θα μετατραπεί σε GB)**
- **CPU: η τιμή (value) είναι cumulative CPU time**
- **Οι τιμές μονάδας:**
 - **vCPU: 0.05 €/ώρα (θα υποθέσουμε 1 vCPU)**
 - **RAM: 0.01 €/GB/ώρα**
 - **Storage: 0.005 €/GB/ώρα**
 - **Floating IP: 0.02 €/ώρα**

```
import re
```

```
# Τα metrics μου ως raw strings (copy-paste όπως τα έχω στο report  
παραπάνω εδώ)
```

```
disk_data = """
```

```
| 2025-06-29T12:50:00+00:00 |          300.0 | 20.0 |  
| 2025-06-29T12:55:00+00:00 |          300.0 | 20.0 |  
| 2025-06-29T13:00:00+00:00 |          300.0 | 20.0 |  
| 2025-06-29T13:05:00+00:00 |          300.0 | 20.0 |  
| 2025-06-29T13:10:00+00:00 |          300.0 | 20.0 |
```

```
"""
```

```
ram_data = """
```

```
| 2025-06-29T12:55:00+00:00 |          300.0 | 250.91796875 |  
| 2025-06-29T13:00:00+00:00 |          300.0 | 273.09765625 |  
| 2025-06-29T13:05:00+00:00 |          300.0 | 275.4609375 |  
| 2025-06-29T13:10:00+00:00 |          300.0 | 304.8984375 |
```

```
"""
```

```
cpu_data = """
```

```
| 2025-06-29T12:55:00+00:00 |          300.0 | 286820000000.0 |  
| 2025-06-29T13:00:00+00:00 |          300.0 | 577640000000.0 |  
| 2025-06-29T13:05:00+00:00 |          300.0 | 872810000000.0 |  
| 2025-06-29T13:10:00+00:00 |          300.0 | 960030000000.0 |
```

```
"""
```

```
# Τιμές μονάδας
```

```
vcpu_count = 1
```

```
vcpu_price = 0.05      # €/vCPU/ώρα
```

```
ram_price = 0.01       # €/GB/ώρα
```

```
storage_price = 0.005  # €/GB/ώρα
```

```
fip = 1                # Αλλαξε αν δεν έχεις FIP
```

```
fip_price = 0.02       # €/ώρα
```

```
# Εξαγωγή values
```

```

def extract_values(data):
    return [float(x) for x in
re.findall(r"\\|s*[\d\-\T\:\+\.\.]+\s*\\|s*[\d\.\.]+\s*\\|s*([\d\.\.]+)",
data)]

# Δίσκος (GB) - σταθερό
disk_gb = extract_values(disk_data)[0]

# RAM (MB) -> GB
ram_vals = extract_values(ram_data)
ram_avg_gb = sum(ram_vals)/len(ram_vals)/1024 # από MB σε GB

# CPU (cumulative value, πχ nanoseconds) - Δείγμα
cpu_vals = extract_values(cpu_data)
cpu_delta = cpu_vals[-1] - cpu_vals[0] # μεταβολή στη διάρκεια
intervals = len(cpu_vals)-1
hours = intervals * 300.0 / 3600 # Κάθε 300s = 5min, το
σύνολο σε ώρες

# Υπολογισμός κόστους
cost_vcpu = vcpu_count * vcpu_price * hours
cost_ram = ram_avg_gb * ram_price * hours
cost_storage = disk_gb * storage_price * hours
cost_fip = fip * fip_price * hours

total = cost_vcpu + cost_ram + cost_storage + cost_fip

print(f"--- Προσομοίωση Χρέωσης (για διάστημα {hours:.2f} ώρες) ---")
print(f"vCPU ({vcpu_count} x {vcpu_price} €/ώρα): {cost_vcpu:.4f} €")
print(f"RAM ({ram_avg_gb:.2f} GB x {ram_price} €/GB/ώρα): {cost_ram:.4f} €")
print(f"Storage ({disk_gb} GB x {storage_price} €/GB/ώρα): {cost_storage:.4f} €")
print(f"Floating IP ({fip} x {fip_price} €/ώρα): {cost_fip:.4f} €")
print(f"Συνολικό κόστος: {total:.4f} €")

```

```
(venv) stack@it2021154:~/devstack$ python chargeback.py
--- Προσομοίωση Χρέωσης (για διάστημα 0.25 ώρες) ---
vCPU (1 x 0.05 €/ώρα): 0.0125 €
RAM (0.27 GB x 0.01 €/GB/ώρα): 0.0007 €
Storage (20.0 GB x 0.005 €/GB/ώρα): 0.0250 €
Floating IP (1 x 0.02 €/ώρα): 0.0050 €
Συνολικό κόστος: 0.0432 €
```

8. Αποθετήριο κώδικα

https://github.com/Panos994/DevStack_DIT266.git

Επίλογος

Η εργασία αυτή αποτέλεσε μια ολοκληρωμένη προσπάθεια κατανόησης και εφαρμογής αρχών cloud υποδομών με χρήση του OpenStack/DevStack, εστιάζοντας στη σχεδίαση, αυτοματοποίηση, παρακολούθηση και κοστολόγηση μιας σύγχρονης πολυεπίπεδης εφαρμογής. Η αξιοποίηση εργαλείων όπως το cloud-init για αυτοματοποιημένη προετοιμασία VMs, η ορθή δικτύωση (με χρήση floating IPs και αυστηρών security groups), αλλά και η παρακολούθηση των πόρων μέσω CLI και OpenStack telemetry (Ceilometer), συνέβαλαν στη διαμόρφωση ενός περιβάλλοντος που προσομοιώνει πραγματικές ανάγκες παραγωγής.

Στην πράξη, η υλοποίηση ήρθε αντιμέτωπη με ρεαλιστικές προκλήσεις, κυρίως όσον αφορά τη συνδεσιμότητα (πρόσβαση SSH, Internet, floating IP routing). Η καταγραφή των βημάτων, η διάγνωση σφαλμάτων (π.χ. cloud-init, DatasourceNone, security groups), καθώς και η τεκμηρίωση υποθετικών σεναρίων ελέγχου, προσφέρουν πρακτική γνώση για μελλοντικές αναπτύξεις σε cloud περιβάλλοντα.

Παρότι οι τεχνικοί περιορισμοί (έλλειψη Internet, αποσυνδέσεις SSH, αδυναμία πλήρους τεστ end-to-end) δεν επέτρεψαν την ολοκλήρωση όλων των σταδίων σε πραγματικό περιβάλλον, το σύνολο της εργασίας παρέχει μια σαφή μεθοδολογία που μπορεί να αναπαραχθεί σε cloud πλατφόρμα με αντίστοιχη αρχιτεκτονική. Η αναλυτική τεκμηρίωση, η αξιολόγηση κόστους (chargeback) και η χρήση αυτοματοποιημένων εργαλείων αποτελούν πολύτιμες γνώσεις για μελλοντικές υλοποιήσεις.

Συνοψίζοντας, η εργασία ανέδειξε τόσο τις δυνατότητες όσο και τις προκλήσεις του OpenStack, προσφέροντας ουσιαστική εμπειρία στις τεχνολογίες cloud και στην πρακτική διαχείριση πολύ-επίπεδων εφαρμογών σε δυναμικά υπολογιστικά περιβάλλοντα.