|  |
| --- |
| Χαρα μπουλουγαρη |
| Υπηρεσίες Υπολογιστικού Νέφους |
| Εργασία εαρινού εξαμήνου 2021 |
|  |
| **Τεχνολογίες: ΝοdeRed Docker OpenWhisk MinIO NodeJS Nginx** |
| **25/6/2021** |

ΠΕΡΙΕΧΟΜΕΝΑ

1. Περιγραφή Σεναρίου ………………………………………………………………………… σελ. 3
2. Τεχνολογίες και Εργαλεία που χρησιμοποιήθηκαν …………………………… σελ. 4
3. Setting Up The Environment on Windows 10 ……………………………………. σελ. 4
4. Αρχιτεκτονική ……………………………………………………………………………………. σελ. 5
5. Patterns που χρησιμοποιήθηκαν……………………………………………………….. σελ.6
6. Περιγραφή υλοποίησης.…..………………………………………………………..……... σελ. 7
7. Πλεονεκτήματα / Μειονεκτήματα …………………………………………………….. σελ. 8
8. Χρήσιμες docker commands που χρησιμοποιήθηκαν ……………………….. σελ. 8
9. Credits ………………………………………………………………………………………………. σελ. 9

1. Περιγραφή Σεναρίου

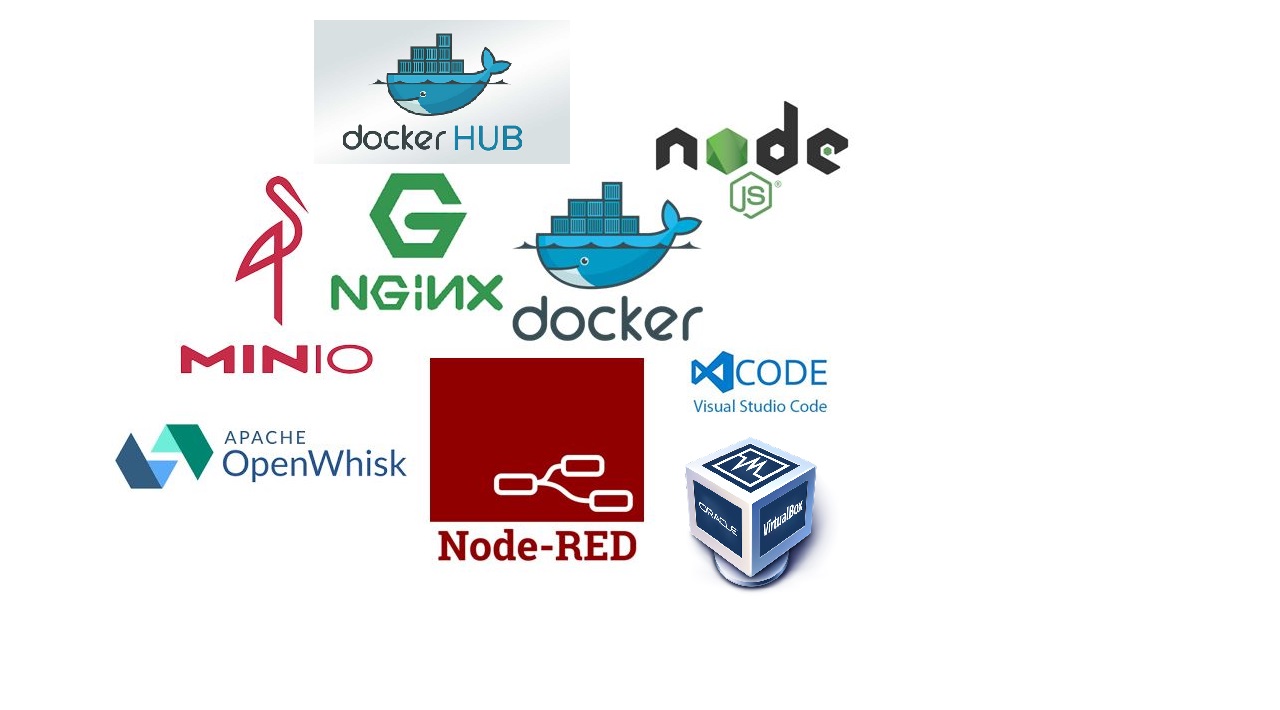
Την εποχή αυτή ολοένα και περισσότεροι άνθρωποι παγκοσμίως ασχολούμαστε με την ψηφιακή τεχνολογία! Αυτό έχει ως συνέπεια να θέλουμε να την χρησιμοποιήσουμε για την διευκόλυνσή μας σε σοβαρά προβλήματα που παρουσιάζονται στην καθημερινότητά μας ακόμα και σε ποιο μικρά και ασήμαντα απλά και μόνο για αισθητικούς λόγους. Πιο συγκεκριμένα χρησιμοποιούμε φίλτρα σχεδόν παντού πλέον σε όλα τα μέσα κοινωνικής δικτύωσης και σε όλα σχεδόν τα προφίλ μας ! Όπως και στις φωτογραφίες έτσι λοιπόν και στα κείμενα χρειαζόμαστε φίλτρα ακόμα και για τέτοιους απλούς λόγους αλλά και πιο σημαντικούς που θα ορίσουμε στην επόμενη παράγραφο !

Το σενάριο βασίζεται στην λογική της εφαρμογής φίλτρων σε αρχεία κειμένου. Πιο συγκεκριμένα ο χρήστης μπορεί να αποθηκεύει σε έναν χώρο στο cloud το αρχείο που επιθυμεί και έπειτα από εκεί να καλεί το φίλτρο μέσω μιας συνάρτησης ώστε να εφαρμόζεται στο αρχείο κειμένου του και να του το μετατρέπει σε μορφή ολόιδια σαν να το είχε γράψει χειρόγραφα. Ο χρήστης με αυτόν τον τρόπο διευκολύνεται στο γεγονός ότι μπορεί να έχει οποιοδήποτε κείμενο αυτός επιθυμεί ανά πάσα ώρα και στιγμή με τέλεια ορθογραφία και σε χειρόγραφη μορφή ενώ το έχει γράψει από το πληκτρολόγιο του υπολογιστή και κατ’ επέκταση έχει επωφεληθεί από όλα τα βοηθητικά ορθογραφικά φίλτρα που του παρέχει ο υπολογιστής. Στην συνέχεια μπορεί να καταθέσει το κείμενό του όπου αυτός επιθυμεί χωρίς να ανησυχεί για την ορθογραφία του.

Ο χρήστης δημιουργεί από μία διεπαφή τους απαραίτητους φακέλους που χρειάζεται στο cloud storage (minIO) ενεργοποιώντας ένα flow της διεπαφής και στην πορεία ανεβάζονται εκεί τα αρχεία κειμένου του μορφής .txt … Στη συνέχεια ένα άλλο flow είναι υπεύθυνο για τη δημιουργία του φακέλου όπου θα αποθηκεύονται τα αρχεία κειμένου στα οποία έχει εφαρμοστεί το φίλτρο και τέλος ένα τελευταίο flow αναλαμβάνει να δημιουργήσει ένα presigned URL ώστε να μπορεί να το στείλει ως παράμετρο στο function το οποίο θα εφαρμόσει το φίλτρο., αφού κατεβάσει πρώτα το αρχείο τοπικά και στο τέλος θα το ανεβάσει στο cloud storage ενημερώνοντας τον χρήστη για την επιτυχή αποθήκευση του αρχείου ή όχι.

Το σενάριο υλοποιείται με τη λογική των serverless apps που υπάρχει και εφαρμόζεται στο cloud οπότε γλυτώνουν τον χρήστη από την εγκατάσταση σε κάποια του συσκευή το λογισμικό του app αυτού έτσι καθώς παρέχεται σαν service διευκολύνει τον χρήστη και στο ενδεχόμενο περιορισμένου αποθηκευτικού χώρου που ίσως διαθέτει ο ίδιος !

1. Τεχνολογίες και Εργαλεία που Χρησιμοποιήθηκαν



1. Setting Up The Environment on Windows 10

(it needs 64 bit system and a very nice RAM)

* Εγκατάσταση του docker toolbox
* Δημιουργία dockerhub account

Σε περίπτωση που κάτι δεν τρέχει ή πετάξει error η version που σίγουρα δουλεύει είναι: Docker version 18.03.0-ce, build 0520e24302 αν πάλι κάτι πάει στραβά τότε εγκαθιστούμε πιο παλιά version του virtual box !

* Εγκατάσταση Vs Code
* Εγκατάσταση του virtual box

Eδώ θα χρειαστεί να πάω στις ρυθμίσεις του vm που έχει φτιάξει το docker toolbox και να ανοίξω τα κατάλληλα ports ώστε να μπορώ να χτυπίσω από windows μέσω port mapping το openwhisk και τον nginx στην πορεία, οπότε πάω ρυθμίσεις -> δίκτυο -> προχωρημένο -> προώθηση θύρας και ορίζω τους κανόνες μου δίνω ένα όνομα στον κανόνα και βάζω για ip οικοδεσπότη το 127.0.0.1 και για θύρα οικοδεσπότη και επισκέπτη τον ίδιο αριθμό αναλόγως τη port ανοίγω για openwhisk το 9000 και για nginx το 80 !

* Εγκατάσταση NodeJS
* Εγκατάσταση του nodered ακούει στο localhost:1880

Μεταβαίνω στον φάκελο εγκατάστασής του και το τρέχω με την εντολή node-red

* Εγκατάσταση openwhisk ακούει στο localhost:3233

Κατεβάζουμε το wsk cli από εδώ

<https://github.com/apache/openwhisk-cli/releases/tag/1.0.0>

και το βάζουμε στον φάκελο που θα έχουμε την εγκατάσταση του όλου openwhisk, μετά μπορώ ανοίγοντας και από το vs code έναν terminal να γράψω την ακόλουθη εντολή

docker run --name openwhisk -v //var/run/docker.sock:/var/run/docker.sock -p 3233:3233 -p 3232:3232 openwhisk/standalone:66a9417

θα πρέπει να έχω εκκινήσει το docker toolbox πριν εκτελέσω την παραπάνω εντολή! Το docker toolbox δημιουργεί ένα vm εντός του virtual box! H παραπάνω εντολή θα εμφανίσει μία εντολή εντός του terminal με μωβ χρώμα την οποία θα κάνω copy ώστε να εκτελέσω στην πορεία ώστε να πάρω εξουσιοδότηση και να γίνει η σύνδεση μου με το wsk cli που τρέχει στο vm ώστε να μπορώ να εκτελώ το cli με την εντολή wsk. Έτσι λοιπόν μόλις η στοίβα της εγκατάστασης του openwhisk ολοκληρωθεί εκτελώ την εντολή με το μωβ χρώμα.. κάνω cd στον φάκελο όπου έχω το openwhisk και ξεκινώ την εντολή με .\ αν την εκτελώ από τον terminal του vscode και δεν ξεχνώ να σβήσω τα ‘’ από το –apihost και το --auth πάνω στην εντολή πριν την εκτελέσω! Aν τρέχω για 2η φορά το openwhisk δεν εκτελώ την παραπάνω εντολή παρά εκτελώ την ακόλουθη αφού υπάρχει το image

docker run -v //var/run/docker.sock:/var/run/docker.sock -p 3233:3233 -p 3232:3232 f32250d71024 (όπου f32250d71024 είναι το id του image το οποίο μπορώ να βρω τρέχοντας την εντολή docker images)

με την εντολή .\wsk namespace list μπορώ να δω τα namespaces που υπάρχουν ή έχω ορίσει στο openwhisk

τέλος θα χρειαστούμε και το wskdeploy cli το οποίο βρίσκουμε εδώ:

<https://github.com/apache/openwhisk-wskdeploy/releases>

περισσότερες πληροφορίες εδώ:

<https://github.com/apache/openwhisk-wskdeploy#downloading-released-binaries>

αφού το κατεβάσουμε το βάζουμε στον φάκελο εγκατάστασης του openwhisk μας !

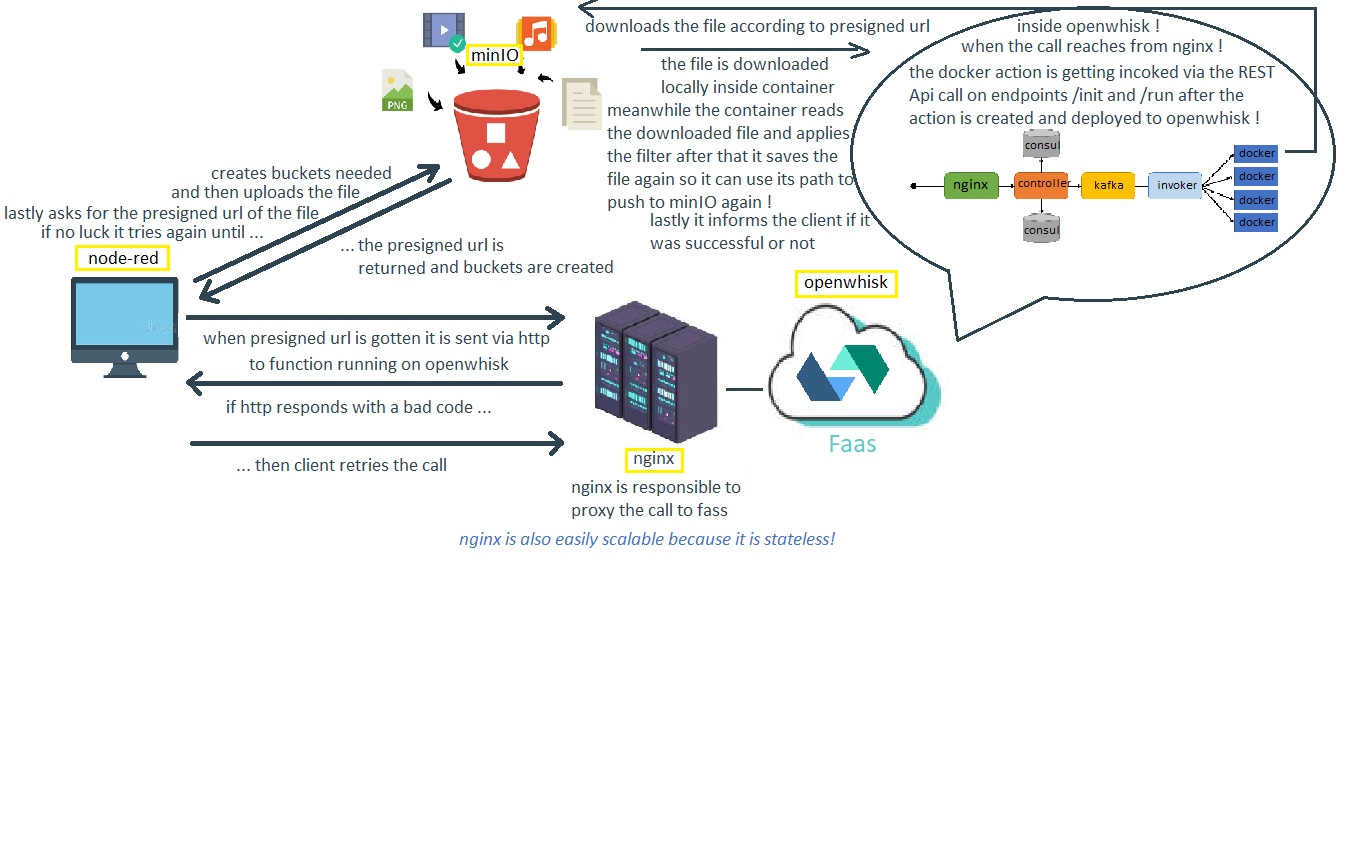
* Εγκατάσταση minIO ακούει στο localhost:9000

To κατεβάζω από τον σύνδεσμο:

https://dl.min.io/server/minio/release/windows-amd64/minio.exe

μετά σε ξεχωριστό terminal μεταβαίνω με cd στον φάκελο της εγκατάστασης του minIO όπου υπάρχει και το cli του ώστε να μπορέσω να τρέξω την εντολή .\minio.exe server +the name of the folder where minIO saves objects this name will create a folder locally on my machine

1. Αρχιτεκτονική



1. Patterns που χρησιμοποιήθηκαν

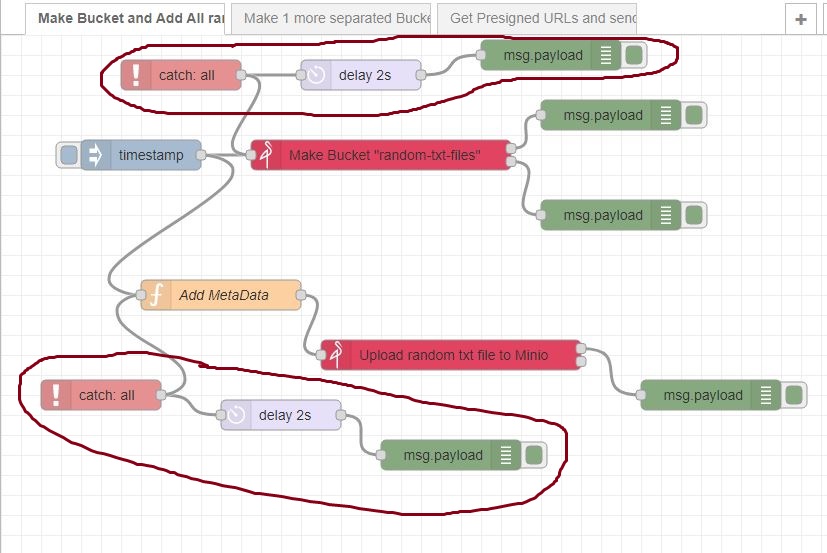
*Retry Pattern*

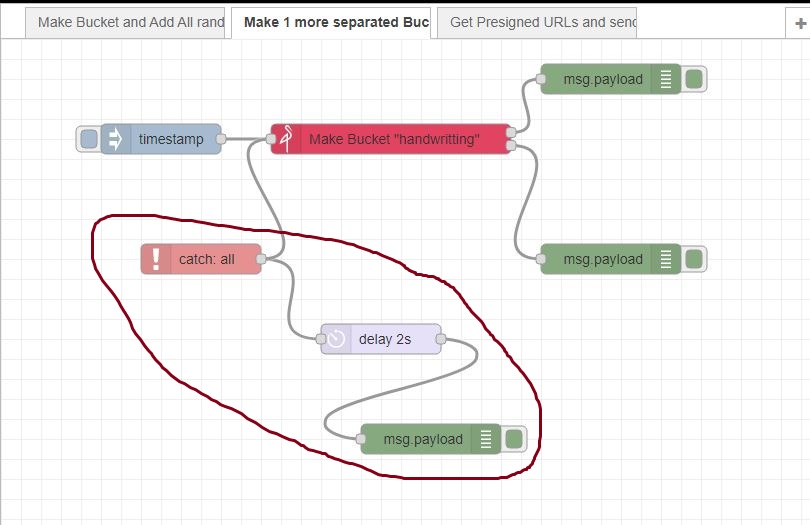
Ποιο πρόβλημα λύνει ?

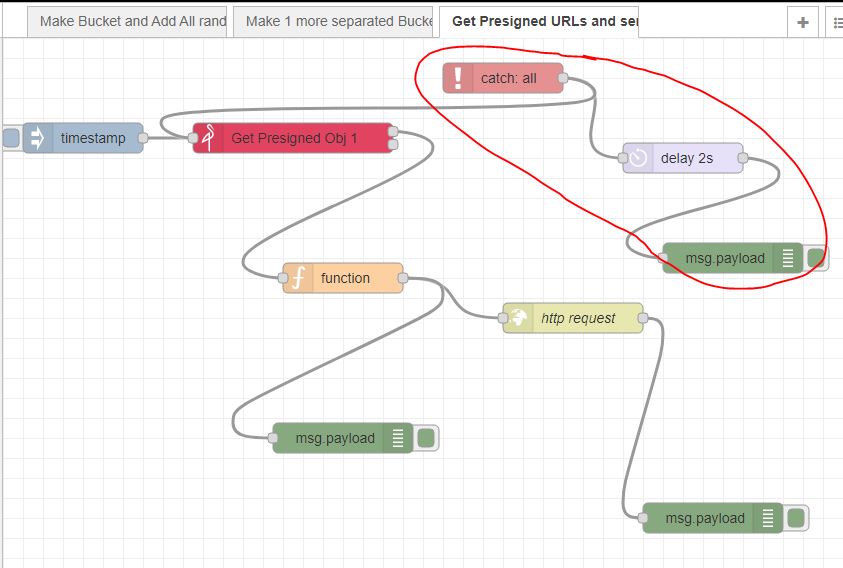
* Όταν έχουμε να κάνουμε με εξωτερικά microservices τα οποία καλούμε μέσω διαδικτύου και κατ’επέκταση αν έχουμε μια διακοπή διαδικτύου πολύ πιθανό να χάσουμε το αποτέλεσμα του call που κάναμε. Αν ξαναδοκιμάσουμε να πραγματοποιήσουμε το ίδιο call μπορεί να παρατηρήσουμε πως ίσως τη δεύτερη φορά όλα να πάνε καλά !

Πότε το χρησιμοποιούμε ?

* Όταν έχουμε αστοχίες που αργούν να διορθωθούν μόνες τους







Οι τρεις παραπάνω εικόνες εξηγούν που και πως εφαρμόστηκε το retry pattern που σύμφωνα με το παρακάτω σύνδεσμο υλοποιείται με τον catch node ακολουθούμενο από ένα delay

<https://cookbook.nodered.org/basic/retry-on-error>

Οι κόμβοι κάνουν αυτό ακριβώς που αναφέρεται στο όνομά τους περισσότερες πληροφορίες για το περιεχόμενο του καθενός θα αναφερθούν σε επόμενη παράγραφο με συνοδεία εικόνων και επεξηγήσεις όπου είναι απαραίτητο !

1. Περιγραφή υλοποίησης
2. Πλεονεκτήματα / Μειονεκτήματα
3. Χρήσιμες docker commands που χρησιμοποιήθηκαν
4. Credits