**Cyberbit CloudOps TL Exercise**

התרגיל משלב ידע בפיתוח ואינטגרציה בטכנולוגיות הבאות:

1. Git
2. Docker
3. Jenkins
4. Jfrog Artifactory
5. Python

לטובת ביצוע התרגיל, יש להתקין סביבה הכוללת:

1. Docker Host
2. Jenkins
3. Jfrog Artifactory OSS

דגשים לביצוע התרגיל:

* יש להשתמש בDocker Images להתקנת מערכות.
* את קבצי הDocker, Python, Jenkinsfile יש לשים בשרת SourceControl (GIT) כלשהו ולהביא אותו כחלק מהpipeline
* ביצירת Jenkins pipeline, יש להשתמש כמה שיותר בפלגינים של Jenkins, לדוגמא Docker,Artifactory,etc ופחות בcommand line
* בסיום התרגיל, יש לשלוח את התוצרים הבאים במייל חוזר:

1. Dockerfile
2. Jenkinsfile\_scripted
3. Jenkinsfile\_declarative
4. zip\_job.py
5. תיאור קצר בנקודות עיקריות של השלבים בהם בחרת לביצוע של כל תת משימה בתרגיל.
6. צילומי מסך של Jenkins jobs ו Artifactory.
7. Create Dockerfile for Docker image build as follows:

a.      Based on Ubuntu latest image

b.      Define environment variable VERSION=1.2.0

c.      Install python

d.      Install vim

e.      Install zip

f.        Install unzip

g.      Copy zip\_job.py into the image's /tmp folder

h.      Once docker container is up, run a command which will print OS type and architecture + verify /tmp/[zip\_job.py](http://zip_job.pl/) exists

2.       Create [zip\_job.py](http://zip_job.pl/) python script as follows:

a.       Create an array of a,b,c,d,e,f….z

b.       Based on this array create txt files (a.txt,b.txt….)

c.       Make sure all txt files created and if not - fail the script

d.       Create zip files with names based on array + VERSION environment variable, that each one will have one txt file inside (a\_1.2.0.zip should include a.txt, b\_1.2.0.zip should include b.txt and so on)

e.       Make sure all zip files created and if not - fail the script

f. all logs should be logged into console (stdout)

3.      Create 2 Jenkinsfiles pipeline jobs with the same logic, one Declarative and one Scripted:

1. Agent should be based on the Dockerfile you created in step 1

        i.            it should run in a privileged mode with label zip-job-docker

b.       Build stage should execute the [zip\_job.py](http://zip_job.pl/) you've created in step 2

c.       Publish stage should upload all the zip files created (only in case build stage succeeded) to Artifactory you installed using the following properties:

      i.         Artifactory server "https://artifactory-xx" (dns name defined by you)

    ii.        Artifactory user "superman"

    iii.        Artifactory password "P@ssw0rd123$"

    iv.       Artifactory repository to upload to "store-artifacts/{VERSION env variable}"

d.       Report stage - send email with job status in the subject to requestor address

e.       Cleanup stage - delete the workspace

1. Feel free to add some more stages that can be more efficient\intresting
2. Let us know if you have any additional questions.