UNFINISHED-UNTESTED-STILL ONGOING(there might be mistakes)

תוכן

[כלים לWINDOWS 1](#_Toc159847901)

[**NETWORK MACHINERY** 1](#_Toc159847902)

[PORTS 1](#_Toc159847903)

[PORT ROUTING/FORWARDING 1](#_Toc159847904)

[**USER –> WEB – FLOWCHART** 1](#_Toc159847905)

[**POWERSHELL** 1](#_Toc159847906)

[**מושגים** 1](#_Toc159847907)

[**כלים** 1](#_Toc159847908)

[ERROR CODES 1](#_Toc159847909)

[**SSH** 1](#_Toc159847910)

[**LINUX -.** 1](#_Toc159847911)

[קונפיגורציות לחיבור בין שרתים באותה רשת 1](#_Toc159847912)

[**FILE SYSTEM** /תיקיות חשובות **/** 1](#_Toc159847913)

[פקודות LINUX 1](#_Toc159847914)

[NETWORK COMMANDS 1](#_Toc159847915)

[SYSTEM INFORMATION 1](#_Toc159847916)

[קבצים+דיסק/קונן 1](#_Toc159847917)

[הפניית קבצים: 1](#_Toc159847918)

[קונן/דיסק 1](#_Toc159847919)

[מניפולציות TEXT 1](#_Toc159847920)

[קבצים מכווצים 1](#_Toc159847921)

[הרשאות 1](#_Toc159847922)

[**UFW**- FIREWALL uncomplicated 1](#_Toc159847923)

[העתקת קבצים (גם בין מכונות) 1](#_Toc159847924)

[**BASH-scripting** 1](#_Toc159847925)

[VAGRANT 1](#_Toc159847926)

[CENTOS 1](#_Toc159847927)

[**GIT** 1](#_Toc159847928)

[**APT** 1](#_Toc159847929)

[**PYTHON** 1](#_Toc159847930)

[DOCKER 1](#_Toc159847931)

[DOCKER FILE- 1](#_Toc159847932)

[COMPOSE 1](#_Toc159847933)

[**JENKINS** 1](#_Toc159847934)

[**BLUE OCEAN** 1](#_Toc159847935)

[**STATIC CODE ANALYSIS** 1](#_Toc159847936)

[SONARQube 1](#_Toc159847937)

[JIT Security 1](#_Toc159847938)

[Snyk 1](#_Toc159847939)

[**GIT ACTION** 1](#_Toc159847940)

[**JFrog – Artifactory** 1](#_Toc159847941)

[**NEXUS & JFROG– ARTIFACTORY** 1](#_Toc159847942)

[**Kubernetes** 1](#_Toc159847943)

[KUBERNETES CLUSTER 1](#_Toc159847944)

[**הגדרות של KUBERNETS CLUSTER** 1](#_Toc159847945)

[טכנולוגיות תומכות קוברנטיס 1](#_Toc159847946)

[**HELM** 1](#_Toc159847947)

[**Scenario-How Ingress/Ingress-controller deploys a public app into the LB** 1](#_Toc159847948)

[Kubernetes Setup 1](#_Toc159847949)

[Introducing Ingress 1](#_Toc159847950)

[Public Exposure 1](#_Toc159847951)

[Accessing the Applications 1](#_Toc159847952)

[Conclusion 1](#_Toc159847953)

[**CLOUD** 1](#_Toc159847954)

[**CODE-PIPELINE** 1](#_Toc159847955)

[EKS- Elastic Kubernetes Service 1](#_Toc159847956)

[CERT-MANAGER(UNFINISHED) 1](#_Toc159847957)

[ArgoCD 1](#_Toc159847958)

[TERRAFORM 1](#_Toc159847959)

[**TerraGrunt** 1](#_Toc159847960)

[**ANSIBLE** 1](#_Toc159847961)

[**PROMETHEUS-GRAFANA** 1](#_Toc159847962)

[DEBUGGING 1](#_Toc159847963)

To add: \*\* what is devops?? \*\* what is a CI process? What is a CD process? Difference of fast forward and non-ff(git)?

## כלים לWINDOWS

Windows services and Event viewer/logs:

*Services*: apps that run for the OS or additional, some of it connects to other servers. it manages events and logs

*Event Log/viewer*: logs events of errors and other properties. Also shows what user did what.

*Issues*: anyone can access it/security issue, volume too big, too partial info.

*Application* log- all logs of application-that are used by the system, Partial: not all apps log are connected to event log.

*System* Log – OS logs(services, device drivers, events, operations)

*Security* log – resource using-creating/opening/deleting files/objects. Goes to audit. Valid/invalid login attempts(users)

*Setup* logs – every update, packages, new installation, uninstalling.

*Forward* events- a possible way to get events from other servers, and concentrate it to my server

*Subscriptions*- set up the way to forward events

Usage: filter current logs>filter accordingly to look for something specific

Kind of events: information, Success, Failure, Warning, Verbose, Error

Performance monitor:

Usage: +add>select what to see graphs>can select a time in history

Resource Monitor:

See usage of disk, cpu etc(real time)

Windows manager?

# **NETWORK MACHINERY**

Kinds of Servers:

Client server:multiple clients on one server.(active directory, mail server, print server, web server, storage servers)

Example: [Active directory – server that is responsible for authentication and user accounts, passwords, membership access rights. create Domain inside the organization ]

Client Server: Organized/ secure/has a point of failure X/ high costs

Peer to peer – no dedicated server, networks of computer which are both clients and servers

P2P: unorganized/unsecured/no point of failure V/ low cost

ROLES(of microsoft windows server):

IIS– (internet information services) server software by Microsoft to host websites/apps and services of the OS.

WEB

SMTP (simple mail transfer protocol)

**VMware**

Hypervisor: the software that creates Virtualization over the hardware.

Type1: Bare metal hypervisor(VMware ESXi) -which has independent OS and machine installed over the hardware(good for organizations).

Type2: Hosted hypervison, installed over the hosting machine(hogs resources from the machine)

Vm files:

Disk file(.vmdk): all the OS and files inside the VM, important.

Log file: logs of any VMactions

Configuration file: (.vmx) stores info, machine name, bios, guest OS.

Snapshot: (.vmsd or .vmsn or .vmx) stores a current state of files and OS.

**Data Center**

Data Center: Hardware infrastructure that supports virtualization(instead of loads of pizza servers)

Components: Compute, Storage, Networks

Types of hardware servers:

Blade servers: high class/expensive mounted into an enclosure and very modular-does not disrupt other servers inside

Rack mounted: like pizza servers

Tower server: a single powerful computer

**Storage**

Storage servers:

Some has own OS, has storage for all other servers.

Types of Storage servers:

SAN – Storage Area Network(expensive)-no OS. fast, only local servers can communicate with it(usually connected with fiber, sacsi)

NAS – Network Attached Storage (cheaper)-embedded OS, independent managing of files/”filer”, distributes storage for all servers-not just local.

RAID: redundant Array Independent Disks, Hard drives linked together to create large volume of duplicate(redundant) storage

Mirroring(RAID 1 state) - when one falls. Disk 2 is a mirror of Disk 1- invoked by a RAID card, that does that automatically. Inside the server.

RAID 0 state: no raid, no mirroring. Regular storage.

Data Striping(RAID 5 state): each drive has blocks/segments which can be used as back up for what is stored, instead of whole disk mirroring.

Limited if low in capacity

Block-Level Storage: data is written and accessed from storage volumes/blocks. Does not provide inherent file system. SAN typical.(accessed mostly via client OS)

File-Level Storage: Data is written to disks but accessed from default file system. Common for NAS systems.

Common Data Center Storage Protocls:

iSCSI- internet small computer system interface. Small card that connects to storage(expensive)

FC- fiber channel card that uses fiber cable.

FCoE- Fiber Channel over Ethernet. Expensive

Storage provisioning:

Thick provisioning: fast. pre-allocating disk space. Logical space by partition equal to actual physical space

Thin provisioning: common. not pre allocating. Slowing down the read write because of constant re-assigning .

Virtual Data Center: uses blueprints instead of templates, as virtual machines/resource, and from the blueprint the template is created.

vSphere:

ESXi – type1 hypervisor, software that is installed directly on the physical server to create a virtual layer

Unix component, VMware Kernel component.

vCenter: management software(installed on management server), on a dedicated server to manage ESXi servers and other components of virtual data center

vSphere Client : program that controls host servers and VMs, has GUI that allows administrator of data center to connect to vCenter and ESXi remotely.

NAT- (Network address translation ) single IP for the whole office. That means that you can share localhost from your VM machines(127.0.0.1)

Bridged = different IP but only internal, that means that you have different IP for each V.machine you have, and thus cannot share same localhost(127.0.0.1)

לבדוק יש 2 סוגי VPN – DHCP ועוד משהו

DHCP = (Dynamic Host Configuration Protocol)service or a server/router that configures and creates our IP address, when

unavailable we use static. But still needs convention to apply

MAC address isn’t enough here because we need protocols to communicate the data.

Analogy: physical address is where I live, and an IP is what my name is.

RAID – redundant array of independent disks. Raid 0 : regular disk. Raid 1: another disks backsup

Raid 5: 3rd disk backs up the system.

NIC – (network Interface control) כרטיס רשת, twisted pair-crossed cable. Very Low freq. elec.

Also called: LAN adaptor, network adaptor.

Twisted pair: כבל מוצלב, דומה לכבל טלפון. אבל יותר רחב עם מעבר נתונים גם.

RJ45 – typical cross cable. More popular between computers, and not in client server scenario.

MAC – (Media Access Control) computer has MAC address when plugged via NIC.

Unique addressכתובת פיזית שצרובה בחומרה ,

Coded in 6 groups of 2 hexadecimals.

לפי תקן IEEE 802

Modulation: הצורה שהתשדורות עוברות דרך הקווים-הסיגנל הדיגיטלי(הנתונים) עוברים המרה לאנלוגי-הגל המחזורי ומתורגמים חזרה לדיגיטלי.

Modulator: המכשיר שאחראי על תרגום האות הדיגיטלי לאנלוגי, הגל המחזורי.

LAN – (*Local*) small private network (home/small office)

WAN – *Wid*e area network, can cover a city or a whole country or more.

Ethernet – חיבור פיזי או וירטואלי למחשבים

Bus network = like single linked list. או חיבור טורי לינארי.

Connects through half-duplex, slow unreliable. Old.

* Host – is a stop at bus connection, a station.
* If a host is injured the whole network can fault.
* We don’t know if the data reached the end.
* Cable is coaxal, 10base5. Un-crossed.

Repeater- enhance range of the network.

Common in hotels, repeats the data and amplifies it.

Hub – מרכזיה , physical layer(layer 1), connects devices to several computers. Like bus network.

מפצל רשת, בעל I/O לכל התקן/מכשיר/מחשב. פחות בשימוש.

Switch – (cant replace router. doesn’t support tunneling, private network) broadcasts frame switching. Common!

Frame switching- frames are data packets, which sliding windows is the managing mechanism that controls the transmission of frames.

* Gets data and returns with a buffer and asks to who to return also(hub doesn’t know who to return)
* Has a list of of all mac addresses of devices he is connected.
* Uses the MAC address to forward data to data link in layer2
* Some switch devices can transfer(?) to layer 3 by adding a routing function, called
* Layer 3 switching/ multilayer switching.

Router- the internet is layer 2, ext. network. The router is what provides IP addresses so we can get out.

* Supports protocols to internet suppliers(switch doesn’t have these)
* Provides traffic directing functions, data that is transferred via packets.
* The router uses the data in the header by reading the table of routers/protocol of routers in the header(of the packet) , and then sends the packet to her next network journey.
* Tracert <your IP address> - provides the journey of the packets through the IPs

Packet – transferred from router to router as a package of the data you send/being sent. also has a-

Header – which tells the packet/packet reader where is the final destination

* When a packet reaches my router, the service(internet) provider checks the address
* And queries if its Global DNS or Local.

FireWall – device or app that blocks or restricts הרשאות according to internal protocols.

Like LAN or to deal with external network (WAN/internet)

-DMZ (DeMilitarized Zone) a buffer area that moves between 2 firewalls.

One border: public {} other border: LAN

Honey Traps – מלכודות לתפוס רמאים באזור הזה

-a computer can become that buffer

- has his own IP unlike NAT. in DMZ there is no NAT.

DPS/IPS – (Intrusion Detection System/Intrusion Prevention System) תוכנות או חומרות שמונעות או מזהות פעילויות חשודות. חוסמות / עוצרות או מדווחות.

**OSI Model**= 7 layer model, Open System Interconnection.

[7:application 6:presentation 5:session 4:transport 3:network 2:data link 1:physical]

Physical(1): switch, router, repeater all the hardware network.

Data-link(2): fiber cable, the signal in the cables, NIC/network adaptors,switch, bridge

Network(3): IP, routers, protocols, connectivity-WAN LAN. Subnetting.

Transport(4): packets, error detection, flow control, monitoring

Session(5): opening a session/user, client, length of session

Presentation(6): output on the browser

Application(7): the app side, “behind the scenes”

(some components can share several layers)

TCP/IP Model: 4 layer model (Transmission Control Protocol) / IP(Internet Protocol)

Layers[ 5-7 Application: http, https, telnet, ftp,TFtp

4:transport(packets): TCP/ UDP

3: network : Router

2:datalink : bridge/switch

1:physical]: Hub

* המודלים מגדיר את הסנכרון הסמנטיקה המתודולוגיה, תהליך אחזור תיקון שגיאות

iSCSI- :רכיב שאחראי על תקשורת בין מחשבים ואחראי על החלק של האחסון. מאפשר הדמיה של אחסון מרחוק-כאילו לוקאלי.פרוש על שכבת TCP ומאפשר תמסורת SCSI בLAN.

**Active Directory** – an ongoing list that refreshes, of users, credentials, policies, group users, servers, DNS and other network resources.

***פקודות בדיקות ,מידע,מניפולציה על NETWORK:***

Ipconfig: מציג מידע על IP של הראוטר, כתובת MAC,

Ipconfig /all : מידע עוד יותר מפורט- רק בPOWERSHELL או CMD, (בGITBASH לא יעבוד אלא אם כן עושים CMD.exe)

Tracert: TRACE ROUTE – מביא את כל תחנות המעבר והפינג אליהן בדרך לכתובת היעד.

Route: מאפשר מניפולציה ושליטה על טבלאות הROUTING. כמו להכריח להשתמש בIPV4 (-4) או 6(-6).

PRINT – מדפיס את הROUTE

ADD: הוספת ROUTE

DELETE, CHANGE

לדוגמא: route print : מראה את כתובות היציאה של הראוטר פיזיות וגם לוגיות(IP).

Arp -a : מציג את כל כתובות הIP מול הכתובות הפיזיות שלהם כDEVICES

Arp -av: מציג פירוט יותר וחלוקה לקטגוריות

Route add: כמו VPN וגם מאפשר הטיה לרשת מסוימת. הטיה של הIP לכתובת שאנחנו בוחרים , לנתב מחדש.(מהDEFAULT GATEWAY)

Netstat: בודק איזה פורטים פתוחים-מה בשימוש.

telnet <ip> <port>: מתחבר לכתובת שנרצה לפי פורט.

Nslookup <domain name>: מציג את הIP של הכתובת דומיין, לדוגמא מה הIP של גוגל

Hostname: מחזיק את שם המכונה/מכשיר/מחשב

# PORTS

IANA: (internet Assinged Numbers Authority) company that responsible for fixed port numbers

Port – logical – not physical (socket is physical)

16bit number 0-65535.

System ports:0-1023 user ports: 1024-49151 dynamic/private ports: 49152-65535

ftp: port 21

ssh: port 22

telnet: 23

25,143: email, SMTP: 25(between mail servers)

53: DNS

487(between clients and Email server)

HTTP: 80

443,4483: https

2375-6:docker (2375 rest API, 2376:https)

SQL : 3306

5432: PostgreSQL

5984: windows RM(remote desktop)

6443: kubernetes

6379: Redis

8080: jenkins

9200,9300: elastic search

9090: prometheus

3389: remote desk protocol(RDP)

443, 4483: HTTPS (secured)

LINUX : \**Ephemeral ports* for linux 32768 – 61000 dynamic(open/closes)

*\*Ephemeral port*: temp port that opens(high numbers 49152-65535) and closes, used by applications or OS.

6443, 10250:Kubernetes: 6443. 10250-kubelet, dashboard: 30000-32767 for nodeport, 10250-10252:kubelet,kube-scheduler, kube-controllermanager.

Jenkins:8080, default agent:50000 (can be configured)

## PORT ROUTING/FORWARDING

איפשור של הIP הפנימי להיות משודר החוצה כדי שיהיה אפשר להתחבר אליו-כמו אפליקציות שעובדות במחשב המקומי ומנתבת גישה דרך FORWARD לפורט.

PORT FORWARDING זה מצב שבו מתעלים/מנתבים כל תנועה שנכנסת (לדוגמא למחשב LOCAL(127.0.0.1)) >> דרך פורט ספציפי- וזה מוגדר על ידי הROUTER או על ידי ROUTER וירטואלי.

משתמשים בזה בגלל שהמחשב המקומי HOST, מפנה את הפורט הספציפי מבפנים, הרשת הפנימית, לבחוץ-לIP החיצוני שלו. אחרת הCLIENT לא יוכל להכנס כי הHOST מאזין רק לפנימי,

וה FORWARDING מאפשר מעבר מהבפנים לבחוץ

לדוגמא 127.0.0.1:2222 כמארח HOST, יאפשר כניסה/תנועה לIP רק דרך פורט 2222. ואם האורח מוגדר כפורט 22, אז כל התנועה שתגיע אך ורק מ2222 תנותב ל22.

במצב שיש לנו שרת עם SSH שעובד כרגיל על פורט 22, והקליינט רוצה להתחבר לשרת

יכול להיות רק SERVICE/APP אחד על פורט אחד פר SERVER. כי אחרת לא תהיה הפרדה של תנועה לפי האפליקציות, ותנועה שמתאימה לאפליקציה 1 תנותב לאפליקציה 2 -מה שיוצר קונפליקט, ולכן נמנעים מזה ע"י הגדרת פורט אחד פר אפליקציה(על אותו שרת).

## **USER –> WEB – FLOWCHART**

1.User inputs url on browser->

2.DNS resolution-translates to ip->

3.Web server establishes connection, hosting the user->

4.Authentication request(username/password/credentials/token) sent to user/security component->

5.LDAP(lightweight directory access protocol) or Active Directory server established(in case that’s on a server) ->

6.User validation, validates user creds. Against its own server directory database ->

7.Access granted, if authentication successful ->

8.Session management, establishes monitor state of session during interaction with website ->

9. User interaction, user interacts with website.

# **POWERSHELL**

>get-service | where-Object Status -eq ‘Stopped’ : מראה כל השירותים שנעצרו

>get-service |where-Object Status -eq ‘Stopped’ | select-object Name,Status מראה רק עמודות של שם וסטטוס:

>$data = get-service | where-Object Status -eq ‘Stopped’ | select-object Name,Status : שופך את הכל למשתנה DATA

>$data | out-file .\services.csv : שומר בתוך הקובץ של "שירותים" את $דאטה

> notepad .\services.csv : מציג דרך נוטפאד את הקובץ

> $data |export-csv .\services2.csv : מייצא לקובך שירותים2 בפורמט סי אס וי

>get-content .\service2.csv מציג את הקובץ

>get-verb : פקודה עם תת קטגוריות שמראה את כל הפקודות שניתם להפעיל

<get-verb | more : מאפשר דיפדוף עם רווח

>get-verb -Group Security|format-list : מציג בצורה ברורה (פורמט) תת קטגרויות עבור פקודות סקיוריטי

>get-alias –כל קיצורי הדרך לפקודות

>get-help \*service\* : עזרה על כל מילת תוכן שנחפש

>get-command -Commandtype function | measure-object : מראה כמות של פקודות עבור פונקציות

>get-command -name \*ip\* -module NetTCPIP : מראה רק פקודות שקשורות לרשת(כי הגדרנו במודולו)

>get-history : כל הפקודות שהרצתי בסשן

>invoke-history -id 22 : מריץ את הפקודה מס22 לפי ההיסטוריה

>start-transcript -path .\transcripts\transcript1.txt -append : הפעלת שירות לשמור את ההיסטוריה לתוך קובץ אוטמטית

>stop-transcript : עוצר את הרישום בלוג הזה בקובץ, והקובץ מכיל הרבה פרטים על כל הפעולות של המשתמש שעשה

>Ping : בודק כתובת מהירות לייטנסי זמינות וכולי

>Whois <ip> :

>ipconfig , : את האיי.פי שלי ביחס לרכיבי החומרה מראה

>ipconfig /all יותר מידע על התקני רשת , רשת ובמחשב עצמו

<route יכולה לנתב אותי ל DEFAULT GATEWAY אחר

Route add : מאפשר לי לנתב את הטראפיק למקום אחר, נניח DEFAULT GATEWAY

שיעבור דרך גייטוואי אחר עבור איי פי אחרים לדוגמא.

>route print, route add VPN מנתב ליציאה אחרת דרך אותו

<netstat פורטים פתוחים או בשימוש(מקשיב) כלי מרכזי לבדיקת תקשורת עובדת/לאעובדת-

>telnet – מחבר בינך(המחשב) לתחנה מסוימת לפי פורט. יש לציין הוסט ופורט

צריך לאפשר את זה בווינדוז CONTROL PANEL ואז לבחור TELNET CLIENT

<curl : בודק אתר ללא דפדפן אבל דרך קומנד ליין

=PAGE SOURCE

<curl – h עזרה HELP

>curl -o <filename> [www.website.com](http://www.website.com) יוריד את האתר לתוך שם הקובץ הזה

>nslookup : מתרגם דומיין שם אתר רגיל באנגלית לכתובת איי.פי

>arp , arp – a, arp -av : מציג את כל ה IP שמחוברים למחשב

<tracert – מגלה את המסלול של הפאקטץ, מבירות, לייטנסי, ורואים איפה נתקע tracert -4 <ip> נותן גרסא מוכרת יותר של גרסא 4

>hostname – מחזיר את שם המחשב

>env – מציג את כל האינווירמנט וואריאבלז.

>type – כמו אקו או

cat

>netstat -an -p tcp איזה פורטים מקשיב או מקושרים עכשיו –

CRON ב WINDOWS זה task schedular

# **מושגים**

Default Gateway – : השער שממנו הכל קורה. בדרך כלל כתובת השרת/נתב ברירת מחדל שברשת. בלעדיו – לא ניתן לגשת לכולם- לא לרשת פנימית ולא חיצונית.(ראה ROUTE ADD)

BGP (border gateway protocol)רכיב שנמצא בנתב(לא תמיד) ומגלה רשתות אחרת שבתווך. לא צריך לגלות אותו,

לא צריך- להגדיר subnet, כי BGP מגלה בעצמו.

אם לא קיים BGP, צריך להגדיר IP אחד אחד

SSL : secure Sockets Layer תעודה חוקית שמאפשרת זיהוי נכון שהגענו למקום הנכון

TLS: Transport Layer Security- היורש לתעודה שלעיל, והגרסא המודרנית יותר

Tty: teletypewriter- basically a terminal, console.

LVM: logical volume manager (centos) > can grow in size, traditional volume partition-is fixed!

CIDR: Classless inter-Domain Routing. A form of notating ip:192.168.14.0/24 . lookup subnet mask but CIDR is different.

ICMP: the protocol which does ping/telnet/tracert. Internet Control Message Protocol

Subnet: range of IP addresses from your IP, to compromise more IP’s/A sub-network

finOps: optimization of costs. Financial optimization/finance operations

IoT: amazon the protocol/platform which handles physical devices on cloud.

CDN: content delivery network. The service that allows caching websites/apps that are too remote. Making the content available

In closer servers, cutting response times and download times.

Httpd: apacheהתקנה של

SDK: software development kit

Ingress- in gress- inbound rules

Egress – out bound rules from server

Flask – framework to build web apps. Like express.js or node.js

Thick provisioning: fixed storage (faster)

Thin provisioning: dynamic storage allocation(slower)

Internet Gateway VS NAT Gateway: NAT allows outbound traffic only, usually from private servers, while IGW allows both directions(incoming too)

UFI: uniform resource identifier – in aws task definition- put image URI as the pull request(NOT URL).

KMS: Key management service ( aws)

Webhook: trigger like in git action which creates an http POST, can be used for notifications.

RESTful app: Repersentational state transfer (POST, GET req. etc)

Stateless: an app that doesn’t need user data to respond, an app without a “session” data. But it can be dynamic app. (static apps are all stateless)

Stateful: an app that does retain the session data, and need it to respond to user.

NFS: network file sharing , can be a server.

Routetable: איך אפשר לעשות פינג לסבנט1 אם אתה בסאבנט2? הראוט-טייבל מחבר בינהם. וגם נותן לך אפשרות להחליט מי ציבורי ומי פרטי

CRD: costume resource definitions

CDA: Content Delivery Application, it gathers info/data from clients, like transactions.

Extended Regular Expression (EXE): as opposed to BRE( Basic regular expression) certain characters don’t need to escaped via special notation to be read.

**SUBNET**:

נוסחא לחישוב כמות IP עבור מספר SUBNETMASK: 2^(32-subnetmask)

/16 subnet of

Means

2(^32-16)=2^16-2=65,536 addresses

מייצר תרגום בינארי לכתובת IP של הROUTER, ומשם מאפשר לייצר עוד הרבה כתובות

של מחשבים/מכשירים(מדפסות, ראוטר) שמתארחים.

אבל למעשה מיוצג בחוץ כמחשב אחד.

הגדרה של הרבה מחשבים ברשת אחת. ההגדרה מתבצעת על ידי קביעת תחום, נניח 24 סיביות:

10.0.0.0/24

אומר שיש 255 מחשבים ברשת הזו, ואז ניתן לגשת לכל אחד לפי המספר שהוקצה לו ברשת.

ללא הSUBNET MASK המחשב המארח לא יידע לאיזה מחשב לפנות ברשת המקומית שלו(מי זה מי?)-

וגם מאפשר הקצאה של רשתות קטנות יותר כדי שלא יהיה עומס על כל הרשת בשידור.

(המספר הבינארי שמייצר הMASK מפריד בין מה שמייצג את כתובת הרשת ומה שמייצג הHOST)

the individual number inside the IP address(first 3).

המספר האחרון מייצג כמות המחשבים שחיברת לאותה רשת

כל מספר סאבנט מייצג רשת נפרדת

בייצוג ניתן לכתוב את זה אחרת עבור כל מחשב:

לדוגמא:

10.0.0.1 מייצג מחשב אחד ברשת-ואם נרצה יותר מ255

10.0.1.0-255

הסיביות של הסאבנט:

10.0.0.0/32 = מכונה אחת

10.0.0.0/24 = מדבר עם כל ה255 מחשבים

10.0.0.0/16 = קולט גם את הספרה שאחרי . 255\*255

0.0.0.0/0 : all the address in the whole world!

עולם ה SUBNET מורכב מ- 3 classes

CLASS A: 0-126. 0-255. 0-255.0-255

CLASS B : 128-191….

CLASS C: 192-233…

CLASS A : BIG

:

:

CLASS D: SMALL

UDP לא ייעודי לחיבורים אין האנדשייק שמאשר שהמידע הגיע, טוב לפיר טו פיר, -

TCP כל בקשה או פינג, חוזרת עם תשובה עם הגעה ליעד –

TLS – מוצפן TCP

STDOUT- standard output (מה שרואים כפלט על הקונסולה במסך)

SPOOL- השירות השהיות של המשימות מדפסת, QUEUE

CIDR – classless inter Domain Routing, צורת תיווי שמייצגת טווחי כתובות אייפי, להלן ייצור הסאבנט

לדוגמא 192.168.0.0/16

זה ייתן

SUBNET MASK :255.255.0.0

uWILDCARD MASK: 0.0.255.255

מה שייתן ל192.168 טווח כתובות של 0-255 בשני האוקטטים האחרונים

Std-in std-out -err: standard input(0) standard output(1), error(2). Echo $?לדוגמא בדיקת:

Std-out זה הפלט/מה שמופיע במסך

אם יש מספר חוץ מ-0 זה אומר שלא הצליחה

Piping: להשתמש בפלט כקלט לפקודה הבאה- stdout | command

Cat kern.log | grep 1

Redirecting: command > file, file < command

שנרצה נניח לעקוף/לסנן error

find / -user lskywalker 2> /dev/null : /dev/null שזה שופך את כל השגיאות שגיאה2, לתוך הסיפריה

numbers.txt = “1\n2\n\3\n4\n5”

sort -R numbers.txt > numbers.txt : ייתן קובץ ריק(במקום סידור אקראי )כי השימוש ב < קודם מוחק את הקובץ

Stdin < command:העברת קלט לפקודה

wc -l < numbers.txt

{ls /etc > lsetc.txt} , sort < lsetc.txt is like cat lsetc | sort

Tee : stdin>tee >stdout/file1 : gives stdout in two directions. כלומר יתן לנו גם לשפוך או לעשות מניפולציה וגם יציג במסך מה שעשינו(כי לינוקס לא מציג כלום אחרי פקודה)

Ls /etc | tee file2 file3 : ישפוך את תכולת הרשימה ל2 הקבצים וגם יציג על המסך את מה שהוא שפך. ברידיירקט רגיל אי אפשר לשפוך ליותר מקובץ 1

Xargs: יכול לשמש כפייפ שמפעיל פקודה על סטנדרד אאוט

Cat text.txt | xargs echo

Cat args1 args2 args3 | xargs <command>

# **כלים**

כלים לבדיקת דומיין:

MX TOOLBOX

DOMAINTOOLS

WIRE SHARK-

לוכד את כל התקשורת שיש במחשב

מנתח מידע וגם מוציא כתובות

MAC

יכול לעזור שאין פינג

אפשר לבחור קוניפגוריישן פורפיל ואז לבחור עמודות שאנחנו בוחרים בניתוח

שיהיה רלוונטי לנו

WIRE SHARK סניפר של פאקטץ –

LYNIS: security, vulnerability כלי לבדיקת lynis –check-all –quick רצוי לפני שליחת PROD

REGEX: כלי אונליין לבדיקת, לפי שפת סקריפט SYNTAX של טקסטים

AWSCLI: כלי לאמאזון שעובד כטרמינל

TRICKS AND TIPS

Type “code .” to startup vscode on the same folder you are on gitbash

journalctl –since="today" –until="tomorrow"

ctrl+R: מילת חיפוש מהפקודות האחרונות -אפשר כמה פעמים עליו כדי להגיע לקודם יותר

# ERROR CODES

בפדפדן/CURL:

56: בעיית SSL

**VI- -**

הכלי הרגיל של לינוקס לערוך טקסט.

UNDO: לוחצים esc ואז U

# **SSH**

SECURE SHELL(SSH)תקשורת מוצפנת בין לקוח -שרת

איך זה עובד? לדוגמא:

אם מחשב 1 רוצה לSSH למחשב2, מחשב 2 חייב שיהיה לו את הPUBLIC KEY של מחשב1. ולמחשב1 חייב שהPUBLIC שלו יהיה תואם לKEY PAIR שלו שהוא הPRIVATE.

יצירת SSH בין שרת ללקוח מצריכה שהמפתח הציבורי/ PUBLIC יהיה בשרת שמקבל את הלקוח. וזה מצריך קופי פייסט מהלקוח אליו(של הPUBLIC)

מהמחשב של הלקוח שרוצה להתחבר לשרת, ופייסט לשרת שמארח/HOST.)את הפאבליק של הקליינט)

לכן עובדים עם GITBASH או PUTTY או אחרים, כדי להעתיק את המפתח.

כלומר הPUBLIC זה השם שלך בהזמנה, והPRIVATE KEY זה השומר שמוודא שזה אתה.

שלבים:

התקנה בUBUNTU

Sudo apt install sshd-server

האם פעיל ? systemctl status sshd

לאפשר SSH בUFW: sudo ufe allow ssh

לדעת איזה פורטים פתוחים ציך להתקין NETTOOLS- sudo apt install net-tools

ואז netstat -tanup

SSH דרך נתיב

PRIVATE KEY

Ssh -i “<PATH>/PRIVATE\_KEY” (טל הדר יודע)

לג'נרט SSH בלינוקס: ssh-keygen -t rsa -b 4096

הגדרות נוספות לSSH נמצא ב /etc/ssh/sshd\_config שם אפשר לשנות פרמטרים בכניסות SSH

.ssh/authorized\_keys : קובץ שמכיל את כל הPUBLICS

אם SSH נכשל :host verification failed

אפשר לעשות:

Rm /c/users/ariel/.ssh/known\_hosts

ואז להכנס שוב

Bash linux:

Ssh-keygen -t rsa -b 4096 !!!!

גילוי תקלות בכניסה לSSH

Ssh -v [ariel@127.0.0.1](mailto:ariel@127.0.0.1)

כניסה בSSH לAWS:

Ssh -i “<PATH OF PEM>” username@<ip of server in aws>: שים לב למרכאות. Identity=i

קובץ CONFIG שנמצא בתוך .ssh/~ , מייעל ומפשט תקשורת בSSH:

Host- nickname

HostName- name or IP

User-default user for communication

Port – port to connect

IdentityFile – the private key path

Example:

Host myserver

HostName myserver.example.com

User myusername

Port 2222

IdentityFile ~/.ssh/myserver\_private\_key

Host github

HostName github.com

User meditator3

IdentityFile ~/.ssh/github\_private\_key

**העתקת קבצים בין מחשבים/שרתים דרך SSH :**

scp -P (port num) <file name from computer> <user@ip\_server>:<path>

דוגמא: scp -P 2222 atlassian-bitbucket-8.8.2-x64.bin [ariel@127.0.0.1:/tmp](mailto:ariel@127.0.0.1:/tmp)

בAWS

Scp -i “ariel-ket.pem” <FILENAME> ubuntu@3.17.23.140:/home/ubuntu

Cyberduck מוונידוז למכונה:

בוחרים SFTP, שזה SSH FTP, ממלאים רק שם, ואת IP של השרת. ואז בוחרים בתור KEY את המפתח **האישי,** שנמצא תחת .ssh id\_rsa

# **LINUX -.**

## קונפיגורציות לחיבור בין שרתים באותה רשת

NAT יספק לנו IP חיצוני, יחיד, לכל הרשת, לכן נגדיר רשת פנימית כדי שיהיה לנו כתובות לשאר המחשבים. בכרטיס רשת נוסף.

לדוגמא אם יש לנו לינוקס DESKTOP ולינוקס SERVER. נגדיר STATIC IP לכל אחד מהם (נגיד בהפרש של 1 בIP)

בVM BOX זה internal network ככרטיס רשת נוסף

בלינוקס DESKTOP פשוט מגדירים בכרטיס רשת IP שאנחנו בוחרים וSUBNET נניח 255.255.255.0 , ללא GATEWAY. ואז צריך לרפרש את הכרטיס(לפתוח לסגור).

בשרת UBUNTU אנחנו נכנסים לקובץ שנמצא תחת /etc/netplan/ בAWS יש לו שם עם CLOUD, בUBUNTU רגיל בד"כ 00-netplanner. זה קובץ YAML ושם עורכים ומוסיפים בזהירות

את הנתונים של הכרטיס רשת החדש, נניח enp0s8. באותו גובה שדה של הכרטיס שכבר שם. ומוסיפים באותו גובה dhcp4: false ותחת אותו גובה שדה של DHCP4, מוסיפים addresses:

מתחת שדה אחד קדימה עם – את הכתובת שאנחנו בוחרים. וגם את הSUBNET /24 שזה נותן את הסאבנט של 255.255.255.0

אחרי שסיימנו את ההגדרות יש להחיל את השינויים sudo netplan apply

## **FILE SYSTEM** /תיקיות חשובות **/**

BOOT: קבצי KERNEL מערכת. אם נגעתי בקובץ-המערכת לא תעלה

TMP: קבצים זמניים.

צבע זוהר-כל ההרשאות אפשריות בתיקיה זו

DEV: קבצי DEVICE קבצי התקנים. קבצים צבועים בצהוב-כי הם התקנים.

ETC: תיקיה מפעילה. קבצי מערכת , כמו SYS32

MEDIA: תיקיית CD או USB

MNT: שמעלים נניח כונן פיזי נוסף

OPT: תיקיה שמחלצים בה אפליקציות, כמו PROGRAM FILES

PROC: PROCESSES , יותר עולם של SYS ADMIN

RUN: כלים שניתן להשתמש בהם + משימות שרצות בBOOT. SERVICES ועוד.

SWAP: שימור גרסאות קודמות של התקנות

SYS: PLACEMENT לשירותי מערכת, אין קבצים – או רק מצביעים שהם בשימוש המערכת.

USR: **לא** תיקיית USER. תיקיה משותפת של המערכת קבצים שנשתמש בהם. יש כאן תוכנות,תוספות,סיפריות,סוג של APP DATA?

VAR: גם יש תיקיית עם שיתוף, אבל לא כמו USR. חלק מותקן בVAR ועובד דרך USR(ההפעלה), לדוגמא.

תיקיית LOGS: /var/log/ --

* כל הלוגים של המערכת. DPKG-התקנות. DMESG-התקנים שמחברים בBOOT לינוקס. AUTH-היסטורית כניסות יוזרים ומה שקשור לאוטנטיקציה. SYSLOG: כמו EVENT VIEWER. FAILEDLOG-לוג כשלונות.
* כשנרצה לראות רק את השורות האחרונות -כי זה האחרון מבחינת הזמן – נעשה tail

## פקודות LINUX

## NETWORK COMMANDS

-netstat מציג פורטים פתוחים netstat -tanup

-ipconfig מציג IP קיימים בהתקני רשת

-ping בודק תקינות תקשורת עם כתובת IP

-route: ניתוב מחדש לאותו דומיין route add -net <ip that you want to reroute> netmask 255.255.255.255 gw 192.168.0.1 <<your router gateway

-nslookup : תרגום כתובת של אינטרנט כמו DOMAIN NAME, לכתובת IP

Arp : מראה את כתובות הMAC – הכתובת הפיזית של ההתקנים arp -a

Tracert: מעקב אחרי הPACKETS -כל המסלול שלה, איזה IP היא עוברת, ואיפה יכולה להיות איטיות אם המסלול או נקודה מסוימת איטית מדי. ראה BGP במושגים.

Curl [telnet://<IP>:<PORT](telnet://%3cIP%3e:%3cPORT)> -vvv : shows telnet communication over a designated IP, incase telnet isn’t installed

## SYSTEM INFORMATION

Uname: מצציג מידע על השרת/מחשב uname -i איזה מע, הפעלה והאם 32 או 64. Uname -a גרסאת המכונה.

Parted: מציג PARTITION, parted -l מציג רשימת PARTITION

לסדר אחכ לקטגוריות:הרשאות-משתמשים

Last : כל המשתמשים האחרונים

Grep -i <username> /etc/passwd : מציג את כל היוזרים

Grep passwd -S <username> : מציג פרטי היוזר – אם נעול וכולי

Echo “hello there” |wc -w : סופר כמה מילים יש //2

לגלות IP של המכונה: hostname -I איי גדולה

שם המכונה: Hostname

לדעת איזה גרסא של UBUNTU LINUX :lsb\_release -a , אם זה 20 מייצג את השנה(2020). אח"כ זה החודש, והגרסא. לדוגמא 20.4.5 זה שנת 2020 מחודש אפריל גרסא 5,

איזה גרסא של CENOTS: cat /etc/os-release (ID=)

איזה פורטים פתוחים ב FIREWALL : sudo ufw status

איזה פורטים מקשיבים : netstat -tanup

התקנת כלים לתקשורת : sudo apt install net-tools ( אם אין NETSTAT)

איזה IP פתוחים עכשיו: ip addr show

בודק קבצים ותהליכים פתוחים לפי פורט/סוקט : lsof

מראה רכיבי חומרה: lshw

Process כמו : task manager top

מה עובד ברקע? Ps -aux

להרוג PROCESS: kill <PID(process id)>

האם PROCESS זה רץ? Ps -ef | grep <name of process> | grep -v grep

[סקריפט ל PID](https://github.com/meditator3/devops_scripts/blob/main/linux_bash_demos/sleepy.sh)

כמה זכרון בדיסקים, מידע על בלוק LEVEL: lsblk

איזה SERVICES רצים עכשיו : systemctl list-units --type=service --state=running

כמה מעבדים יש לי? Lscpu

כמה דיסק פנוי? Df -h / -i(inode)

סוג המערכת קבצים: df -T

מה גודל/כובד הספריה du -sh <name of folder>

גודל כל התיקיות בתיקיה הנוכחית? Dh -h –max-depth=1 ./

כמה RAM פנוי? Free, free -h(page file), watch free

ERROR שהיו לנו: journalctl -xe או systemctl status <name of service>

רק סיפריות/תיקיות: ls -d \*/

Sleepy – הפעלת סקריפט ברקע

איזה SERVICES עובדים ברקע? Systemctl list-unit-files

איזה גרסא השרת? Uname -a

איפה ממוקמת הפקודה – MAN WHICH

Useradd: הוספת יוזר, יותר מתאים לסקריפט. עם -m נייצר תיקיה ליוזר . -d נקבע שזו התיקית HOME של היוזר: sudo useradd -d /home/dtrump -m dtrump

Adduser – אם צריך רק להוסיף קצת

Passwd - שינויי סיסמא ליוזר

Whoami – איזה יוזר עכשיו?

איזה קבוצות המשתמש נמצא בהן – cat etc/group או groups

או עבור משתמש ספציפי cat etc/group | grep ariel

כל הפקודות האחרונות – history

Usermod – שינוי דברים בתוך יוזר, לדוגמא – הוספה לקבוצה

Usermod -a(append) -G(group) sudo lskywalker – יכניס את המשתמש הזה לקבוצת ה סודו

נעילת משתמש – sudo usermod -L dnltrump

שחרור משתמש – U-

רשימת משתמשים בעלי HOME- cat /etc/passwd |grep home

נעילת יוזר : passwd -l <username>

איזה יוזר נעול? : sudo passwd -s <username>

נעילת כל המשתמשים: @

היסטורית כל הפקודות: cat ~/.bash\_history

מחיקת יוזר – deluser לסקריפט, או USERDEL לשימוש בSHELL.

רפרוש של הרשת, שלא צריך לאתחל את המחשב: sudo systemctl restart networking

הצג רק סיפריות בתיקיה הנוכחית: ls -d

רק קבצים : ls -p -type f

הצג ממוין לפי גודל קבצים: ls -lS

הצגת שגיאה מפקודה אחרונה : echo $?

**RUN IN THE BACKGROUND**הוספת פרוסס ברקע/סקריפט שירוץ ברקע: פקודת הרצה ו &

Zgrep – חיפוש בקבצי gz

Egrep – אפשר לשים כמה ביטויים בבת אחרת בהפרדה של |, לדגומא: ‘ads|ddd|123|’

איזה קבוצות המשתמש: cat etc/group |grep <username>

העתקת פלט לקובץ : <command, ls or other> **>** <name of file>

העתקה לקובץ מ-פלט בלי לדרוס: <command, ls or other> **>>** <name of file>

Wc- ספירת מילים, -l כסוויץ מציג כמה שורות יש בקובץ

Head – הצגת 10 השורות הראשונות

Sort – סידור קובץ לפי ABC או אפשר גם סויץ -r ברוורס. -nלפי מספרים

להוציא מתוך קובץ ולהחזיר מסודר: sort -r **<** <filename> **>** reverse\_order

Tee – מעתיקה כמו המשפך. Ls | tee file.txt שומר את תכולת שמות הקבצים בספריה לתוך file.txt

Xargs- לוקח פלט ומעביר אותו לפקודה שלידו: ls | xargs echo יעביר את ls שיודפס דרך echo

[סקריפט XARGS](https://github.com/meditator3/devops_scripts/blob/main/linux_bash_demos/xargs_demo.txt)

Lsof – איזה קבצים פתוחים עכשיו -u מציג על פי USER שבחרנו. בלי כלום זה מציג את הכל.

Sudo lsof -n -i4TCP:22 מציג ללא הוסט, על פרוטוקול אינטרנט4, בTCP, מה עובד/רץ על פורט 22(ואם הוא בכלל פתוח)

Ls -p – מציג ספריות עם /

Ls -p ~ | grep -v / - יציג רק קבצים כי הgrep מביא רק מה שבלי ה / -p מכליל את הספריות

Grep -q יביא רק מה שאין לו כזה, ויתן EXIT 0 אם אין כלום.

Grep “^<string>” יביא EXACTLY את הSTRING הזה (בגלל ה ^)

עם Xargs אפשר לייצר הרבה מאוד פקודות על הרבה קבצים ls -p ~ |grep -v / | xargs cp <filename>

יעתיק את כל קבצי הסיפריה למה שנגיד לו בבת אחת.

watch -n 1 df -h : ניטור DISKSPACE בזמן אמת, עדיף בחלון נפרד בזמן התקנות-שנגמר המקום כל פעם

watch -n 1 'df -m' : to see same thing but in Mb usage

journalctl -n: נותן את הלוג האחרון ביותר(רק JOURNALCTL לא ייתן את האחרונים)

משתני סביבה ENVIRONMENT VARIABLES לLINUX: export

מהם משתני הסביבה? env : מציג את כולם

לדוגמא שינויי פורט או איי פי משתני סביבה:

export

FLASK\_RUN\_HOST=0.0.0.0

export

FLASK\_RUN\_PORT=3000

Nmcli (ראה CENTOS)

TIME:

Cron \* \* \* \* \*

Min(0-59) hour(0-23) day(1-31) month(1-12) day of the week(0-6 or sun -sat)

Crontab -e to setup the scheduler for the script.

Use < \* \* ..> bash /<path of script>

\*= all values

Example : (in crontab -e script) 0 20 4 \* \*/path to script/ arg1 arg2 - this will run script every day 4 of month at 20:00.

Date: נותן תאריך ושעה , יום חודש ושנה

Date “+%d/%b/%Y-%R:%S” : 20/Feb/2024-16:33:51 נותן פירמוט לפי מה שהגדרנו

## קבצים+דיסק/קונן

השוואה בין קבצים diff <filename1> <filename2> , שימוש: לדוגמא קבצי קונפיגורציה שגיבינו וערכנו להם שינוי אח"כ.

sudo du -h --max-depth=1 / | sort -rh : מציג את הקבצים הגדולים לפי הסדר

Hier - מראה היראכיה של הקבצים

Tree – מראה היראכיה בצורה גרפית(בד"כ דורש apt install tree) tree -d -L 2: ייתן רמת פירוט LEVEL 2, אפשר לתת LEVEL 3 או יותר כדי לקבל יותר עומק בענפים

Tail – מציג רק את ה10 שורות האחרונות של הקובץ, בד"כ שימושי בLOGS. אם נרצה לשנות כמות שורות נוסיף FLAG -n : tail -n 12 <filename>, יציג 12 שורות אחרונות

Head – מראה את ה10 ראשונות. גם שימושי בSCRIPTING

Lsof – מראה איזה קבצים פתוחים. Ls open files

Du -sh <name of folder> - מציג כמות דיסק בשימוש על פי הסיפריה(כמה שוקלת הסיפריה)

[סקריפט אתראת מקום פנוי בדיסק](https://github.com/meditator3/devops_scripts/blob/main/monitor-disk-space.txt)

[סקריפט קבלת שמות תיקיות והצגת LS בכולן, לא משנה כמות התיקיות ששמים](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe5.sh)

**העתקת קבצים בין מחשבים/שרתים :**

scp -P (port num) <file name from computer> <user@ip\_server>:<path>

דוגמא: scp -P 2222 atlassian-bitbucket-8.8.2-x64.bin [ariel@127.0.0.1:/tmp](mailto:ariel@127.0.0.1:/tmp)

Scp -i “ariel-ket.pem” ubuntu@3.17.23.140:/home/ubuntu/<FILENAME>

^^ למשוך קבצים מהענן אל המכונה שאני נמצא בה.

Rsync(יש להתקין APT):

Shred: גריסה של קובץ-אבל לא מחיקה שלו. סוג של ENCRYPT אבל לא ניתן לשחזור! שימושי במצבים שהאקרים מנסים לגעת בקבצים חשובים.

Wipe: מוחק לגמרי מהקונן ברמה הפיזית/המגנטית.

Locate {שם קובץ שמחפשים}

צריך להתקין דרך mlocate

או

Find / -name {name of file} 2>devnull

2>devnull removes error of permissions.(error 2)

Rsync : אולי צריך להתקין, יודע לשים לב לשינויי קבצים בספריות שנגדיר לו- ואז ישמור אותם

Rsync -avz <name of folder> <name of target>: if we add –delete ימחקו קבצים לא קשורים בתיקיית היעד. FLAGS:z מכווץ v verbose, a ARCHIVE-משמר את כל תכונות הקובץ מבחינת הרשאות תכונות ובעלות

Rsync -avz –delete /root/ tmp/sync-target/ : מוחק בתיקיית היעד את הקבצים הלא קשורים.

דוגמא: ייצר ספריה: dir1 dir2, צור קבצים בdir1, פקוד RSYNC ואז שנה בDIR2 הוסף קובץ לא קשור, ותבדוק אם הDELETE עבד.

[סקריפט RSYNC](https://github.com/meditator3/devops_scripts/blob/main/rsync.txt)

## הפניית קבצים:

אפשרות 1:

SYMBOLIC LINK:

מבוסס inode(סוג של כתובת-כל קובץ הוא בעל פוינטר לכתובת של הקובץ, הפניה). Ls -li ייתן לנו את הקידוד של inode ברשימת הקבצים. לדוגמא RENAME לא משנה inode.

כדי לבדוק כמה נשארו לנו inode: Df -h / -i

אם יש יותר מדי SYMBOLIC LINK, אז יכולים להגמר הכמות של הinode ולכן לא יהיה אפשר לייצר עוד קבצים

* softlink: כמו SHORTCUT בווינדוז. Ln -s <filename path(link to what)> <link to where> . -s softlink. כאן משתנה הinode, לא זהה לקובץ המקורי.ולכן שמוחקים זה לא מוחק את המקור.
* Hardlink: ln <file name path(link to what)> <link -the new path> ללא -s זה HARDLINK. כלומר אם נמחוק את הLINK זה מוחק גם את הקובץ כי זה אותו הinode

אפשרות 2:

/usr/local/bin

אפשרות 3 -זמנית:

Export PATH=<folder path of executable>

אפשרות 4:

לשים EXPORT כמו באפשרות 3 ב ~/.bashrc

## קונן/דיסק

Partition (part) זה אומר FIXED – גודל קבוע, לא ניתן לשנות. אלא אם כן זה LVM, וזה מצויין שם.

ללא LVM לא ניתן להוסיף כונן לשרת והוא קרס סופית.

Block device: מכשיר שמאפשר כתיבה וקריאה של DATA/מידע בBLOCKS, ניתן עדיין להציל דרך BLOCKS.

TRADITIONAL PARTITIONS: קבועים וכבולים לנפח, ולא ניתנים לשינויי.

:LVM: מגיע NATIVE עם CENTOS, LOGICAL VOLUME MANAGER, שמאפשר הגדלה של נפח ככל שנרצה ונוכל.

PV: physical volume, כל הארד דיסק פיזי נחשב PV.

VG: volume group, מחיצה/קופסא/קבוצה שבה יש PV.הקבוצה היא לוגית/דימיונית. לדוגמא PV1, PV2,PV3 בתוך הקבוצה.

ואפשר לייצר כמה מחיצות לוגיות שנרצה בתוך השרת. אי אפשר לשים אותו PV בכמה VG שונים.

LV: logical volume המצב שאנחנו מגדירים שבו לוקחים את הVG כמיכל אחד. ואז למעשה הVG יכול להפוך לLV וכל השטח אחסון הפנוי יהיה זמין כווליום ממשי.

בזכות LV וVG, אנחנו יכולים לעשות EXTENSION ולהוסיף עוד PV, וכך להגדיל אינסופית את הLV...הוא הוא היכולות של ה LVM

ואין צורך אפילו לעשות RESTART לשרת.

Lsblk: מציג מכשירי אחסון. גם וירטואלים כמו קובץ שמנגיש אחסון. )loop( SDA זה בד"כ כונן. SR0 זה CD ROM

Pvdisplay: מציג פרטים על קונן/אחסון פיזי

Vgdisplay: מציג קבוצות ווליום, אם נרצה להרחיב אחסון-נוסיף אותו לפי הקבוצה מהאחסון החדש.

Lvdisplay: קבוצות אחסון לוגיים(לא פיזיים)

Lvcreate: יצירת VOLUME חדש לוגי.

Pvcreate: מייצר אחסון חדש פיזי, אבל לא ALLOCATED

Pvcreate /dev/sdb

כדי למפות אותו לאחסון הקודם שיגדיל אותו:

Vgextend: מעמיס/מוסיף עוד שטח אחסון על פי הקבוצה שהגדרנו

Vgextend ubuntu-vg /dev/sdb

או: vgextend -L +10G /dev/ubuntu-vg/ubuntu-lv

Lvextend: הוספת נפח (עם הגדרת כמות) לLV, מתוך הVG.כלומר, מתוך הנפח שנותר בVG שבו נמצא הLV.

Lvextend -L 1.5g /dev/centos/root

בתוך הVG יש לנו את הLV, ושם אנחנו יכולים לבחור לאיזה PATH אנחנו רוצים להגדיל את הנפח מתוך הVG הקיים

(שאליו משוייך הLV). יש לשים לב להבדל שלמרות שיש נפח אחסון פנוי, הכוונה היא לALLOCATED SPACE. ולא לDATA

שמאוחסן בדיסק.

Df -T : מציג את סוגי מערכת הקבצים נניח ext4, xfs, tmpfs

עבור מערכת קבצים XFS

Xfs\_growfs <path of mapped storage, such as /dev/ubuntu/>

ירחיב לנו את הדיסק לתוך מערכת הקבצים

אחרי שווידאנו שאנחנו לא הולכים לפרמט את הדיסק שלנו שקיימם. בודקים היטב שאנחנו מפרטים דיסק חדש שלא קשור לכלום:

Fdisk /dev/sdb

L : מציג את כל סוגי מערכות הקבצים שיש(עם הקוד הרלוונטי שלהם)

P: מציג את פרטי האחסון הנוכחי שאנחנו מפרמטים

N: יצירת PARTITION

XFS לעומת EXT4 יותר טוב כי יש לו (XFS) גודל PARTITION גדול יותר וגם גודל קובץ גדול יותר. וגם יותר INODES

Mkfs.xfs : make filesystem ושמים את השם שלה/קיצור.

Mkfs.xfs /dev/sdb

Lsblk -f : מציג גם את UUID וגם את הסוג מערכת הקבצים

כדי להוסיף אתחול של מכשיר נוסף לBOOT: יש לערוך את /etc/fstab בזהירות!!!! קובץ רגיש מאוד!!

**הרמת VOLUME נוסף LVM למכונה AWS אחרי שהצמדנו אליה VOLUME:**

-זיהוי הכונן lsblk, יצירת ווליום פיזי- pvcreate (כי הוא עוד לא מזוהה),

לדוגמא: pvcreate /dev/sdb

- יצירת VOLUME GROUP, vgcreate כדי שנוכל להכניס את PV ל"קרטון"/קבוצת ווליום לטובת הLV שמייד יגיע.

- במידה ואנחנו רוצים לעשות EXTEND, להוסיף נפח, נעשה vgextend

יצירת ווליום לוגי- lvcreate, יצירת FILESYSTEM mkfs.ext4 לדוגמא, MOUNT של הFILESYSTEM לסיפריה, עדכון /etc/fstab להוסיף שהכונן יעלה בBOOT

-הוספת נפח לLV קיים מתוך הVG שלו. Lvextend -L +1.5g /dev/centos/root

בדיקה: LVDISPLAY שאכן הנפח השתנה בPATH המצויין.

- עכשיו צריך ליידע את FILESYSTEM על השינויי, וצריך לבדוק איזה סוג FILESYSTEM הדיסק כדי לסדר את האיחויי הסופי.

בדיקה איזה סוג מערכת קבצים: df -T

Xfs\_growfs /dev/mapper/centos-root

**הרמת MOUNT לATTACHED DISK**

* Fdisk /dev/sdc כאשר SDC הוא הדיסק החדש.. להיזהר שלא מוחקים כלום, זה פירמוט.
* בprompt, בוחרים M כדי לראות מה האפשרויות לביצוע של הפירמוט. הצגת מחיצות קיימות דרך P
* נייצר מחיצה חדשה, ראשית -שממנה אפשר לייצר עוד מחיצות. (עוגה שלמה ואז חתיכות).
* בוחרים N, PARTITION-1, בחירת גודל רצוי(הכל או חלק)-ENTER מייצג את הכל.
* כדי לשמור את השינויים בוחרים W וזהו(או מתקנים או יוצאים עם Q)
* בודקים LSBLK, ורואים שהדיסק מחולק(אך לא ASSIGNED), כרגע יש רק מחיצה מוגדרת.
* צריך לפרמט את הדיסק ולסדר לו FILESYSTEM
* הכנת FILESYSTEM: mkfs.xfs /dev/sdc1 הנתיב של הדיסק החדש, אחרי שבנינו לו מחיצה שאפשר לפרמט אותו במערכת קבצים הזו.
* בודקים אם אכן יש FS לדיסק החדש: lsblk -fs וזה מציג איזה FS יש לכל דיסק
* עכשיו צריך לשייך אותו לPATH , נניח בMNT/ mount /dev/sdc1 /mnt
* הבעיה שבLINUX באיתחול מחדש, הMOUNT לא תופס לתמיד, וצריך לקנפג כדי שיעלה תמיד בBOOT
* תיקון- דרך הקובץ etc/fstab , ועורכים אותו VI /etc/fstab וצריך להוסיף את הנתיב. קובץ רגיש מאוד -לגיבויי- יכול להרוס את ההפעלה מחדש!
  + /dev/sdc1 /mnt xfs defaults 0 0 וזה אמור לשמור את זה. DEFAULTS או AUTOMATIC אם זה מצבי READONLY.בDEFAULTS זה כולל כל ההרשאות READ WRITE ETC. ברגע שיש 0 בראשון, זה אומר שאין גיבויי. כל מספר אחר מפעיל גיבויי. 0 האחרון זה לבדיקות.

-יש לשים לב לתוכן שקיים במחיצה/דיסק וחייבים לשמור בסיפרייה נפרדת למנוע מצב שזה נמחק

### מניפולציות TEXT

Wc: ספירת מילים, -w רק מילים , -l מס' שורות, -m מס' אותיות

Grep: grep <word> <filename>= cat <filename> | grep <word>

Grep -v <word> <filename> : נותן את כל מה שאין בשורה שלו את word

Grep -i <word> <filename> : מתעלם מ upper/lower case

Grep ‘^Feb 15’ auth.log : ה-^ מגדיר תחילת שורה, כל מה שבתחילת שורה יש את הטקסט הזה. ולכן ייתן כל הכניסות של פברואר 15.

Grep ‘host$’ auth.log : $ מציין סוף השורה, כל מה שיש לו טקסט בסוף השורה הוא ייתן.

Zgrep - חיפוש רק בקבצי ZIP או GZ

Egrep – חיפוש בכמה מילות/טקסטים, \* egrep ‘10|127|1’ יחפש את כל אחד מהמספרים, ויציג את התוצאות עבור כולם.

Pgrep : לדוגמא pgrep httpd יציג אם יש תהליך שנקרא HTTPD

Awk : מאפשר PIPING של עמודות, בין השאר. לדוגמא ls |awk ‘{print $2, $4}’ ידפיס רק את עמודה 2 ו4 של הLS

Sed: streamEditorמאפשר חילוץ והטמעה של טקסט מתוך קובץ , sed -i “s/<pattern>/<replacement>/g” <filename> יש לשים לב -i מתכוון לin place שמטמיע בתוך הקובץ, כאשר ללא -i זה רק יציג כSTDOUT

Example: sed -i "s/ansible ansible\_host=[0-9.]\+/ansible ansible\_hosts=${PRIVATE\_IP}/g" /etc/ansible/hosts.

This will replace whatever ip is after ansible\_host. s/ is to search and substitute.

Remove a line with the word web in it: sed -i '/web/d' hosts\_new

add web to start of existing line : sed -i '/development is fun/ s/^/web /' hosts\_new

cut -

*cut -d’%’ -f1*: removes %, it needs -f# to know which field to cut also

[SED GUIDE](https://www.linkedin.com/pulse/mastering-sed-comprehensive-guide-text-manipulation-linux-manav/)

ping:

להשתמש בפלט כקלט לפקודה הבאה- stdout | command

Cat kern.log | grep 1

Redirecting:

command > file, file < command

שנרצה נניח לעקוף/לסנן error

find / -user lskywalker 2> /dev/null : /dev/null שזה שופך את כל השגיאות שגיאה2, לתוך הסיפריה

numbers.txt = “1\n2\n\3\n4\n5”

sort -R numbers.txt > numbers.txt : ייתן קובץ ריק(במקום סידור אקראי )כי השימוש ב < קודם מוחק את הקובץ

Stdin < command:העברת קלט לפקודה

wc -l < numbers.txt

{ls /etc > lsetc.txt} , sort < lsetc.txt is like cat lsetc | sort

Tee : stdin>tee >stdout/file1 : tee gives stdout in two directions. כלומר יתן לנו גם לשפוך או לעשות מניפולציה וגם יציג במסך מה שעשינו(כי לינוקס לא מציג כלום אחרי פקודה)

Ls /etc | tee file2 file3 : ישפוך את תכולת הרשימה ל2 הקבצים וגם יציג על המסך את מה שהוא שפך. ברידיירקט רגיל אי אפשר לשפוך ליותר מקובץ 1

שימוש בPIPE כדי לסנן תיקיות או רק קבצים:

Ls -p יציג תיקיות עם /

Ls -p | grep / יציג רק תיקיות

Ls -p | grep -v / יציג רק קבצים-כי זה הפוך INVERSE

Rename manipulation, several files: xargs -n1 -I{} mv {} {}.jpg | ls or echo

Awk : מאפשר הרצת פקודות מתוך STDIN בעזרת ‘{}’ שהפקודה בתוך. לדוגמא כדי לראות רק עמודות 1,2,אחרונה בתהליכים : ps aux |awk ‘{print $1, $2, $11}’ | tail -n 10 |grep ariel

יציג 10 שורות אחרונות רק של המשתמש ARIEL, ורק העמודות שמצוינת בAWK

## קבצים מכווצים

לראות רק תכולת הARCHIVE: tar -tf bundle.tar

# הרשאות

Drw-r—r—

D- מייצג DIRECTORY כלומר תיקייה

קבוצת ה3 הראשונים-עבור הרשאות היוזר שבעל הקובץ

קבוצת ה3 האמצעים עבור ההרשאות של הקבוצה שקשורה לקובץ

והאחרונים זה איזה הרשאות יש לשאר העולם.

Chmod u-w מוריד הרשאת כתיבה עבור בעלי הקובץ הUSER

+w יוסיף הרשאת כתיבה.

U: זה עבור יוזר

G: עבור הGROUP

O: עבור OTHERS, כל השאר.לדוגמא:

Chmod uo+x נותן הרשאת הרצה EXECUTE לקובץ עבור הuser והother

usermod -aG <name of group> : מכניס את המשתמש הנוכחי לקבוצה

- groupadd GROUP\_NAME: קובע קבוצה חדשה

Chown – שינויי הרשאה לסיפריה/קובץ : chown :GROUP FILE\_NAME

Useradd – הוספת יוזר, יותר מתאים לסקריפט

Adduser – אם צריך רק להוסיף קצת

Passwd - שינויי סיסמא ליוזר

Whoami – איזה יוזר עכשיו?

איזה קבוצות המשתמש נמצא בהן – cat etc/group

Groups: מציג את כל הקבוצות שיש למכונה

או עבור משתמש ספציפי cat etc/group | grep ariel

איזה משתמשים יש: cd home או cat /etc/passwd

מה קורה שקובץ עם הרשאות מסוימות עובר לתיקיה עם הרשאות אחרות?

* כל ההרשאות הולכות איתם לא משנה איזה הרשאות יש בתיקיה. גם אם עשינו העברה בSUDO, זה נשאר אותם הרשאות לקובץ שהעברנו. כלומר בשום מצב לא ישתנה!

כל הפקודות האחרונות – history

Usermod – שינוי דברים בתוך יוזר, לדוגמא – הוספה לקבוצה

Usermod -a(append) -G(group) sudo lskywalker – יכניס את המשתמש הזה לקבוצת ה סודו

נעילת משתמש – sudo usermod -L dnltrump

שחרור משתמש – U-

רשימת משתמשים בעלי HOME- cat /etc/passwd |grep home

נעילת יוזר : passwd -l <username>

איזה יוזר נעול? : sudo passwd -S <username>

נעילת כל המשתמשים: @

מחיקת יוזר – deluser לסקריפט, או USERDEL לשימוש בSHELL.לא מוחק סיפריות של הנמחק.

* אפשר גם ללכת ל/etc/group ולמחוק ידנית מהקובץ

איזה קבוצות המשתמש נמצא בהן – cat etc/group

או עבור משתמש ספציפי cat etc/group | grep ariel

דוגמאות:

הוספת קבוצה:

sudo addgroup rebel-alliance

הוספת יוזר לקבוצה

sudo usermod -a -G rebel-alliance lskywalker

שינוי קבוצה לקובץ ספציפי מ-יוזר ספציפי

chown lskywalker:rebel-alliance secret-plans.txt

לדארת' וויאדר יש קובץ שלו אישית:

Chown dvader my-stuff

ורק לקבוצה שלו מותר לקרוא או לפתוח

Chmod o-r my-stuff

^^^ זה הייתה הורדת הרשאה לOTHER

[סקריפט שבודק אם יוזר נעול או לא+קיים](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe3.sh)

[סקריפט שנועל תיקיה לבקשת המשתמש](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe4.sh)

## **UFW**- uncomplicated FIREWALL

Sudo ufw status

Sudo ufw allow ssh

Sudo ufw deny incoming: חוסם כניסה של משתמשים (כל הפורטים)

Sudo ufw allow outgoing: פותח לכל המשתמשים בשרת לצאת החוצה מכל פורט.

Sudo ufw enable

לדעת מה החוקים של הFIREWALL כאשר הוא INACTIVE(למקרה שSSH נניח סגור ונרצה לוודא שפתוח) : sudo cat /etc/ufw/user.rules

ב centOS זה firewalld

# העתקת קבצים (גם בין מכונות)

Scp: (SECURE COPY)

דוגמא-

scp username@linux-vm-ip:/path/to/source/file.txt C:\path\to\destination\

ENVIRONMENT VARIABLES

export PATH=/my/custom/bin:$PATH : מאפשר הרצה של מה שנמצא בתיקיה למשתנה הסביבה PATH ואז אפשר להריץ כל EXEC שנמצא בסיפריות שהגדרנו שם.

סדרך כלל שומרים את הקובצים הבינארים/EXECUTABLES בסיפריית BIN. זה יכול להיות כל תיקיה אבל.

SECURITY: ההרשאות לא עוברות עם ההעתקה. אבל למשתמש במכונה המקבלת יש הרשאות לעשות בקובץ כרצונו.

כאשר מעבירים קבצים בין מכונות יש תמיד סיכון גם אם לשבריר שניה שיהיה אפשר לדוג את המידע הרגיש.

# **BASH-scripting**

עבור BASH: שבנג, #! /bin/bash

ב shell : #! /bin/sh

Chmod u+x <filename> : כדי לאפשר הרצה של הקובץ x . אחרת אין הרשאה

Sleep 90 - מחכה 90 שניות לא ניתן להשתמש בקונסולה. אלא אם כן נריץ ברקע עם &

variables

MY\_SHELL=”BASH” באותיות גדולות מקובל. $MY\_SHELL כדי לשלוף אותו

Echo $? מציג את הפלט שגיאה של הפקודה..

Echo $! מציג את הPROCESS האחרון שנפתח..לדוגמא סקריפט שהרצנו

Wait <process ID> מחכה עד שיגמר הPID, ואז ממשיך את הסקריפט (ראה סקריפט לדוגמא)

Pidof <name of script/process> נותן את מספר הPID של מה שכתבנו בשם, לדוגמא: pidof sleep

Ps -pf <PID> נותן פירוט לPROCESS ספציפי, (-p), עם פירוט מלא (-f)

[סקריפט דוגמא עם VARIABLE](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/variabl2.sh)

[סקריפט קריאה מקובץ וספירת שורות CHARACTERS](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe2.sh)

Conditionals

-e: נכון אם קובץ קיים

-d : נכון אם זה תיקיה

-f : נכון אם זה קובץ רגיל וגם קיים

-r: נכון אם יש לך הרשאה לקרוא ממנו

-s: נכון אם הקובץ קיים ולא ריק

-w: נכון אם יש לך הרשאה לכתוב על הקובץ

-x: נכון אם יש לך הרשאה להריץ את הקובץ

-z: אין כלום בSTRING

If [ -e /etc/passwd ]

Then

Echo $

Fi

= או -eq אם יש שיוויון

-lt קטן מ-

Elif כמו בפייתון

-o : OR לדוגמא $SIDE1 -eq $SIDE2 -o $SIDE1 -eq $SIDE3

|| : גם OR בצורה אחרת [ $SIDE1 -eq $SIDE2 ] || [$SIDE1 -eq $SIDE3 ]

-a : AND

Exit 1 : יוצא מהCONDITIONAL

[סקריפט שממיין מספרים שלמים בלבד(ומתעלם מעשרוני)](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe6.sh)

For loops

For COLOR in red green blue

Do

Echo “the color is ${COLOR}”

Done

הכנסת פלט של פקודה לתוך משתנה: PICTURES = $(ls \*.jpg)

[סקריפט דוגמא FORLOOP](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.1/assignment1.3.sh)

Positional parameters

Script.sh <param1> <param2> <param3>

$0 : script.sh

$1: <param1>

$2: <param2>

Date: date +”%T %F” מציג שעה ותאריך בלבד ללא יום

חילוץ מתוך טבלאות:

ראה –format בפקודות, לדוגמא:

for image in $(docker images --format "{{.Repository}}" | grep react) : שמראה רק את עמודת REPO ושולף את השם של GREP

או

| grep '^mysql\|arielguez/mysql' : נותן או שם אחד או שם שני

echo ${mysql\_and\_tag} | awk -F: '{print $2}' : מפריד את המחרוזת שאחרי : ופולט אותה

CREATE SCRIPTS THAT USES ARGUMENTS

$0 refers to the name of script

$1 refers to the arg after the script’s name

Example:

See arg.sh script

./arg.sh ariel

[סקריפט שבודק אם פקודה הצליחה(דרך ARGS)](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.2/exe1.sh)

[סקריפט יצר מספרים טבעיים בשורה עם פסיק](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.1/assignment1.1.sh)

[סקריפט מספרים אי זוגיים](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.1/assignment1.2.sh)

[סקריפט PROMPT לסוג של תיקייה](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/assignment1.1/assignment1.4.sh)

# VAGRANT

Vagrant init

Vagrant up

Vagrant halt

Vagrant suspend השהיה/מצב שינה והחזרתו.

Vagrant resume

Vagrant reload מעלה מחדש, אם שיננו קצת בהגדרות

Vagrant ssh default user name for vagrant machine: vagrant

Vagrant destroy משמיד את הכל כולל הקופסא.

Vagrant box list רשימת קופסאות שהורדנו לVAGRANT ואיתם אפשר להרים מכונות

ניתן לקחת קופסאות מתוך האתר VAGRANT CLOUD

Vagrant box remove <name of box> כמו DOCKER RMI - מוחק את הדמות של הקופסא

Vagrant package –output <name we want as box> צריך שנהיה בתיקייה של הVAGRANTFILE ,

ואז נארז הנוכחי לBOX, שניתן להרים בכל עת, כמו הקופסאות שאנחנו מרימים.

Vagrant global-status –prune מציג ID של VAGRANTים שרצים כרגע וסביבות קיימות.

כדאי לשים gitignore על .vagrant כדי לא להעלות לGITHUB קבצים שהם לא טקסט

Vagrant box add <name>  [vagrant box add laravel-ariel file:///K:/DevOps/Vagrant/laravel/laravel-ariel](file:///c:\\\\users\\\\ariel\\\\laravel-ariel)

הוספה לתוך REPO מתוך הנתיב של הBOX בלוקאל, הPATH זה הנתיב של הקובץ והשם שלו-אחרי שהפכנו אותו לBOX

כמו שמוסיפים ב - DOCKER IMAGE לרשימת IMAGES

\*שינויי משאבים במכונה:

בעורך של Vagrantfile ,תחת הבלוק של config.vm.provider נניח זכרון vb.memory = “1024” , או מעבד- vb.cpus = “2” ,

לשים לב שסוגרים את הבלוק עם end

\*הוספת קבצים למכונה מתוך הLOCAL:

בתוך Vagrantfile מוסיפים (נניח תיקיית DATA שבה יש קבצים שנרצה שיהיו במכונת VAGRANT)-

Config.vm.synced\_folder “./data”, “/vagrant\_data”

כאשר DATA זה תיקיה שנייצר ונשים בה קבצים, שיופיעו במכונה.

הרמת סקריפט שירוץ במכונה על VAGRANT

Config.vm.provision “shell”, inline: <<-SHELL

Apt-get update

Apt-get install -y apache2

SHELL

end

או פשוט דרך BASH SCRIPT:

Config.vm.provision “shell”, path: “deployLamp.sh”, privileged: false

end

FLAGS

Privileged: false >>vagrant user

Privileged: false >> root user

הגדרות רשת או הוספת התקן רשת

בVagrantfile

Config.vm.network “forwarded\_port”, guest: 80, host: 8080 כאן יש שרשור של פורט כדי שיוכל להתחבר לLOCAL

Config.vm.network “private\_network”, ip: “192.168.33.10” וכאן יש הגדרה של כתובת IP קבועה מתוך הרשת פנימית של ואגרנט

אם נרצה לקנפג לNAT, נוריד את הprivate\_network –

אין צורך להגדיר IP כי הוא מייצר לעצמו

להפוך לbridge

config.vm.network “public\_network”, bridge:” Realtek PCIe GbE Family Controller”, adapter: 2 , type: “dhcp" השם של הBRIDGE לקוח מה PROPERTIES של הכרטיס רשת. כל זה בשביל שיעבוד דרך BRIDGE ולא NAT.

במידה ויש לנו רק ADAPTER 1 שהוא BRIDGE אז אפשר לשנות לאחד, אם יש לנו גם NAT וגם BRIDGE אז צריך להגדיר סידורי.

נניח NAT-אדאפטר 1, BRIDGE אדפטר 2.

התקנת LARVEL דרך VAGRANT- ראה [סקריפט1](https://github.com/meditator3/devops_scripts/blob/main/vagrant/deployLamp.sh), [סקריפט2](https://github.com/meditator3/devops_scripts/blob/main/vagrant/createUser.sql).

Vagrant destroy – f : : כאשר צריך לאתחל מכונה מחדש(לא מוחק את הקובץ וואגרנט פייל)

לבדוק מפתחות SSH vagrant ssh-config –

לגלות בעיות בVAGRANT- (GITBASH) export VAGRANT\_LOG=debug

# CENTOS

cat /etc/redhat-release - כמו lsb\_release -a מראה את מערכת ההפעלה והגרסא

yum check-updates: כמו apt update, בודק מה יש לעדכן

yum update -y: מעדכן אוטומטית \*להזהר לא לעשות בPROD\*

yum update system: מעדכן רק קבצי מערכת

עדכוני עורכי טקסט ודברים בסיסיים: yum install nano vim wget curl net-tools

Vi /etc/yum/Yum-cron.conf : קובץ הפעלות מתוזמנות

Yum list installed : מציג מה שמותקן כבר

yum search apache: חיפוש חבילה על פי שם(עדיף httpd)

yum info httpd: יציג מידע על החבילה

yum deplist httpd : מציג את כל הDEPENDENCIES של החבילה. לדוגמא במצב שאין אינטרנט או ברשת סגורה, להוריד את כל הdependencies כדי שיהיה אפשר להתקין כמו שצריך.

Yum install –downloadonly : לא מתקין -מוריד, לדוגמא עבר תסריט כמו בפקודה מעל עם הרשת הסגורה. וגם מוריד את כל הdependencies!

Vi /etc/yum.repos.d/CentOS-Media.repo : מאפשר לנו לגשת לREPO כCD ROM, שהCD הוא הREPO. ברגע שזה מקונפג, אפשר להוריד עם yum install

Yum remove <package> : הסרת התקנות

Yum install epel-release איפשור של הורדות מREPO צד שלישי(לא רשמי)

Firewall-cmd פיירוול של CENTOS

Yum install כמו APT INSTALL

Systemctl status firewalld – מראה אם הפיירוול פתוח

Firewall-cmd –state: האם רץ

Firewall-cmd –list-all: פרטים

Firewall-cmd –list-ports: מראה איזה פורטים פתוחים

Firewall-cmd –list-services: מציג את כל הSERVICES שפתוחים לFIREWALL

פתיחת פורטים: sudo firewall-cmd --zone=public --add-port=80/tcp –permanent

Firewall-cmd –permanent –add-service httpd : מאפשר פתיחת פורטים שקשורים לSERVICE HTTPD

לא לשכוח RESTART כדי לאפשר פתיחת פורטים

Rpm : red hat package manager, התקנות לפי קישור/לינק. חבילה חבילה.

Rpm -Uuh <link>: Upgrade HASH מציג

Systemctl list-unit-files: מציג קבצי מערכת שפועלים או משבתים או סטאטיים. לשים לב שזה לא אומר RUNNING.

Systemctl -t service -a –state running: שירותים שרצים, :type -t -a :all

Hostname : שם המכונה

כדי לשנות לSTATIC ip – כותבים

Nmtui (יש להתקין NETWORK MANAGER אם לא קיימת הפקודה)

משנים ipv4 לMANUAL, ומכניסים את הכתובת הרצויה-בהנחה שהיא לא תפוסה-או סדרה של מספרים שלא תפוסה. ולא לשכוח SUBNET MASK

לדוגמא: 192.168.14.180/24 שזה אומר שכל מה שתחת 24ביט שזה 255.255.255.0 נכלל כרשת המקומית. ולכן המספר האחרון מהווה את כל תחום הרשת.

מכניסים לGATEWAY את הכתובת ROUTER המקומית(לא של בחוץ באינטרנט).

מרפרשים על ידי ניתוק והחזרה של החיבור רשת.

Nmcli connection show : יכול לאפשר סדר עדיפות לחיבורי רשת(במצב שבו רוצים לתעדף חיבור ספציפי או שיש קונפליקט)-כאן נותן רק את שמות החיבורים

nmcli connection modify INTERFACE\_NAME ipv4.route-metric METRIC\_VALUE

להחליף את השמות INTERFACE\_NAME ו METRIC\_VALUE לפי שם החיבור והתעדוף שלו

אפשר גם להכנס לRouteMetric ולשנות תחת NETWORK

Tee: פקודה שאפשר להשתמש בה לכתוב לתוך קובץ לדוגמא:

containerd config default | sudo tee /etc/containerd

מביא את כל תוצאות ברירת המחדל ומכניס אותם לקובץ המצוין אחרי הPIPE

HOSTS כמו בWINDOWS: /etc/resolv.conf

: Dig <ip> <port> כמו PING מפורט יותר, דורש bind-utils בהתקנה

Netstat- צריך yum install net-tools

date MMDDhhmm[[CC]YY][.ss] : לשנות תאריך ושעה

alternatives --set python /usr/bin/python3 : allows python to become python3 command

TROUBLESHOOTING:

CURL לא עובד ופינג כן.

Systemctl status NetworkManager -l

sudo ip -6 neigh flush all

לשים לב שכל MAC של הADAPTOR שונה ממכונה אחרת ברשת

# **-GIT**

Source Control versioning options:

Centralized(CVCS): Centralized Version Control System.

central Repo of all versions, in which everyone works and commit/edit/change

It means the code sits on the server, not locally. Can’t work on offline.

Examples of centralized: Azure Devops Server(TFS), SVN(subversion), ClearCase, perForce, CVS(IBM)

Distributed(DVCS): each user can copy the repo and work on it, and not rely on the central repo.

After commits it can be applied later to the central repo

Code can sit locally. And can develop offline.

Examples of distributed: Mecurial (Hg), Git, Bazaar

States of git: working directory, Staging area(pre-Commit), Commit(enters history of repo)

אם מייצרים REPOבGITHUB לשים לב שבוחרים ביצירה .gitignore

כי אחרת בכל משיכה ועדכון של COMMIT הוא עלול למשוך קבצים ענקיים שהם לא תואמי גיט-שזה בעיקר קוד.

Staging-adding files to repo, working dir -no add files. Repository- commits(still local). Remote-to add to remote repo.

To make staging area on console:

Git init: creates staging area for the current folder

rm -rf .git : removes all staging.

Git commit -am “added ..” : this also adds to staging and commits, -a is add.

git branch -m master main : change name of branch from master to main

הוספת ריפוסיטורי לתוך גיטהאב-חייב אח"כ לעשות PULL אם זה חדש git remote add origin [git@github:meditator3/git-flow.git](mailto:git@github:meditator3/git-flow.git)

Git remote -v: to check what is origin url

Git remote set-url <new url>: will update it

git remote remove origin : removes the current remote origin

דחיפה לתוך הREMOTE ריפוסטורי – git push -u origin master

Git branch <name> - הוספת הסתעפות

Git checkout -b “main” - מייצר עוד ענף של main

Git branch – אומר לך באיזה בראנץ' אתה

Git push origin <branch name> - הוספת ההתסעפות לתוך הרמאוט

Git checkout <branch name> - לעבור לענף/הסתעפות

Git log – כל הקומיט שהיו(זה שמופיע אחרון ברשימה הוא הראשון)

Git status- האם יש משהו שצריך לעדכן או לעשות קומיט

Git add . – להוסיף את כל הקבצים לסטייג'

Git commit -m “<comment>”

Git rm –-cached <file> = מחיקת קובץ מגיט

Git tag -a v1.0.0 -m “version 1.0.0”

Git log - check history

Git config –global user.email “ariel G.” : tags the user only with the name(not an identification process)

Git push –tags דוחף תיוג חדש חזרה לריפו של גיטהאב

**שיטות איחוד** (MERGE)

*Fast forward* – לינארית, מאחדים בצורה לינארית נניח 2 בראנצ'ים שבמצב שאחד לא התקדם בCOMMIT.

*Non-fast forward* – לא לינארי. מצב שבו שתי בראנצ'ים או יותר זזו בCOMMIT אחר. ונוצר MERGE COMMIT שמצביע על הCOMMIT הקודמים

ב merge commit הקוד של כל הענפים יתמזג, ולא תהיה עדיפות לאחד מהם.

הפוינטר יצביע על פי GIT CHECKOUT ולפיו יעשה הMERGE, לדוגמא אם הפוינטר על MASTER ועושים git merge secondary, האיחוד יהיה בין הMASTER לSECONDARY. Git branch רק יחליף את הענף שבו יופעל הCOMMIT או ADD.

PULL REQUEST – זה למעשה בקשת MERGE. GATEWAY של הראש צוות או מעליו לאיחוד עם בראנץ' שצריך להישמר איתו.

בוחרים REVIEWER (הבוס בד"כ) והוא זה שמאשר.

CONFLICT: מצב שבו נניח יש קובץ בענף אחד ולא קיים בענף אחר, ומנסים למזג בינהם. אז קודם צריך לפתור.

לדוגמא להחליט על מחיקת הקובץ, או הוספתו ב GIT ADD ואז ב GIT COMMIT, ואז אפשר לעשות PULL

TRACKBILITY: לא כל ענף נתמך ב REMOTE ORIGIN. ניתן להוסיף כך:

Git branch –set-upstream-to=origin/<another branch to track>

וכך כל PULL נתפש גם עבור הBRANCH הנוסף.

Git reset (flags --soft , --mixed , --hard)

Soft: git reset --hard HEAD או >ID of commit>

לדוגמא אם בחרנו קומיט קודם , אז הוא יחזיר אותו מצב אחד אחורה.

לSTAGING

COMMIT ואז נשאר רק לעשות

Mixed: working directory יחזיר אותו ל

GIT ADD ואז צריך לעשות ו-

Hard: מוחק גם את הקובץ וגם את הקומיט

Git rebase -I <id of commit before the problem> change of the history!

בריבייס אנחנו משנים לוקאלית את מה שאנחנו רוצים ואז על ידי

Git commit --amend

Git rebase --continue

השינויים תופסים ומוחקים את ההיסטוריה.

לדוגמא היה לי מצב שבו נחשף מפתח ונאלצתי לשנות כדי שלא יראו אותו בהיסטוריה

Git blame <name of file> מראה מי ערך ונגע בקובץ

Git revert <ID of commit> : reverts back one stage before, for ex. If you wrote “I QUIT!” to your boss,

Then revert the I quit file, it will tell you it will delete it.

But this doesn’t delete the commit – but creates a new commit, and maintain the history.

Git cherry-pick <ID commit> : will place the files/updates of that particular commit into the current branch progress.

See [I quit example](https://github.com/meditator3/git-practice)

SUB-MODULE git

Git submodule add <link of repo> <name of new folder> :

this adds a repo INTO an existing repo, by using “shortcuts” not real files in the parent repo.

If we clone the parent repo, we will clone both repos. See [example](https://github.com/meditator3/sub-module) << that repo has been assimilated into [this repo](https://github.com/meditator3/git-practice)

GIT FLOW CHART

מצב שבו מעדכנים בזמן אמת את הPROD/MAIN (נניח מצבי חירום) אז במקום לעשות FFW, יוצאים לענף HOTFIX.

ומHOTFIX מעדכנים את הMASTER(לא משנה כמה COMMIT עשינו בדרך בענף של הFIX)

וגם חייבים לעדכן את ענפי ה DEV וה STAGING שלא יהיו מאחור אחרי שכבר יש עדכון.

הסנריו הוא: הבעלים מביא גרסא 1.0 לחברה, שנמצא בענף MAIN, החברה מפתחת מהגרסא הזאת בענפים אחרים – DEV STAGING

לגרסא 2.0 ואז דוחפים/ממזגים חזרה לענף PROD, ומשם הבעלים גילה שיש תקלה ובמקום להוציא מיד תיקון לתוך הPROD

הוא יוצא לענף HOTFIX ואז אחרי כמה COMMIT, דוחף חזרה לDEV וSTAGING שגם מתקנים ומשדרגים, ואז משם מגיעה

גרסא 2.0.1 אחרי הHOTFIX.

ראה שרטוט.

[ראה REPO](https://github.com/meditator3/git-flow-example)

נעילת BRANCH: בSETTING >BRANCHES > לסמן do not allow bypassing

git credential fill <<<"url=https://github.com/meditator3/Grad\_proj.git"

protocol=https

host=github.com

username=meditator3

password=

בתוך קובץ CONFIG כדי לעשות SSH

Host github.com

IdentityFile ~/.ssh/<name of rsa ssh key>

User <name of your user>

ONBOARDING:

Git clone [git@github.com:meditator3/google.git](mailto:git@github.com:meditator3/google.git)

משכפל מתוך ה REMOTE אל LOCAL

Fast forwarding: linear commits

Non fast forwarding: more commits that are not present in another branch, must use merge or rebase for histories to match

Git reset –hard head^ מוריד את מה שיש בסמן, אולי הקומיט האחרון

Git reset –soft <commit code>= מחזיר את הקומיט המצוין לסטייג'ינג

Git reset –mixed <commit code> = מחזיר את הקומיט המצוין לאזור העבודה

Git reset –hard <commit code>מוחק את הקומיט וגם את הקובץ =

Git pull – מושך שינויים

Git commit -am “<comment>” – מוסיף ומעדכן קבצים שהשתנו או נוספו וגם מוסיף קומיט

Git tag -a(add) v1.1 -m(comment) “version 1.1” – הוספת תיוג

Git push –tag – מעלה את התיוגים לרמוט

REBASE: git rebase -I <commit id> REBASE changes history!

UNDO REBASE: git rebase –abort

Git log –oneline : מציג לוג מקוצר

Squash – איחוד כמה קומיטים לאחד(לצרכי פישוט), נעשה דרך ריבייס

Git Rebase -i: PICK)זה פותח עורך טקסט שבו נשנה לסקווש במקום פיק

לדוגמא:

pick f12a345 Commit C

squash f12a345 Commit C

SSHבדיקות חיבור :

Git remote -v

Ssh -T [git@github.com](mailto:git@github.com)

התקנת גיט לפי גרסא- (בדיקה: git –version(

curl -LO https://github.com/git/git/archive/v2.37.0.tar.gz

מעדכן את הרפוסיטורי כדי שיהיה אפשר להתקין

* tar -zxf git-2.15.1.tar.gz
* cd git-2.15.1
* make configure
* ./configure --prefix=/usr
* make all doc info
* sudo make install install-doc install-html install-info

**Note:** If the command make configure error you can install autoconf apt-get install autoconf.

if the command make all doc info error you can install zlib with apt-get install zlib1g-dev.

If the command sudo make install install-doc install-html install-info error you can install asciidoc and docbook2x with apt-get install asciidoc and apt-get install docbook2x.

# **APT**

Apt update: רק מרפרש את הרשומות REPO במידה ויש גרסאות חדשות או אפליקציות חדשות. לא מתקין כלום.

לגלות dependencies : apt-rdepends <name>

התקנת PING במע’ הפעלה שאין לה: apt install iputils-ping

גיבויי ספריות עדכונים: sudo cp /etc/apt/sources.list /etc/apt/sources.list.0

לראות את הרשומת סיפריות ללא הCOMMENTS: grep -v “#” sources.list (גורע את השורות עם ה#)

Sudo apt list –upgradable : רק מציג מה ניתן לשדרג

Vi sources.list : לערוך ולהוסיף REPOSITORY

Sudo do-release-upgrade : לשדרג גרסאת UBUNTU (נניח 20 ל22)

Sudo apt full-upgrade: עדכןו מלא

Sudo apt autoremove : מוריד חבילות לא נחוצות או שפג תוקפן

חיפוש:

Apt search <search word>

מידע על חבילה: apt show <name of package>, sudo apt show apache2

מה תלוי בחבילה שנרצה להתקין: apt depends <name of package>, sudo apt depends apache2

# **Cryptography**

מטרת הקריפטוגרפיה:

authentication, Privacy/confidentiality, Integrity,

Non-repudiation(mechanism that prove the sender indeed sent this message)

דוגמת ארכיטקטורה

מפתח של אליס הפרטי+הציבורי. מהווה חותמת דיגיטלית שלה. כך שכולם מכירים אותה.

שאליס שולחת הודעה, ההודעה עוברת HASH, והצפנה וש-בוב ינסה לקרוא אותה מוצפן, בוב צריך לפתוח את ההודעה עם מפתח ההצפנה

שרק לו יהיה-כדי לקרוא את ההודעות של אליס.

המפתח צריך להיות גם אצל בוב וגם אצל אליס.

הגדרה

המפתח הפרטי נועד *לפתוח הצפנה*, והמפתח הציבורי נועד *להצפין*.

מידע>מוצפן( ע"י מפתח ציבורי)>נפתחת הפצנה(ע"י מפתח פרטי)>מידע נגיש בחזרה

סוגי אלגוריתמים:

SKC: secret key (symmetric) cryptography. שימוש במפתח אחד גם להצפנה וגם פיענוח הצפנה

PKS: Public Key (asymmetric) cryptography שימוש בשתי מפתחות(SSH לדוגמא) -אחד להצפין, ואחד לפיענוח

HF: Hash Function(one way cryptography) חסרות מפתח **ולא** ניתנות להיקרא אחרי ההצפנה שלהן(אין פענוח), לכן נקראות ONE WAY

המטרה היא יותר לאמת מידע מאשר לקרוא את התוכן שלו. סוג של חתימה בלבד.

כי אם יש למידע שנשלח אותו HASH אז כנראה שהוא לא עבר שינויי. או לשימור סיסמאות.

כך שגם במצב שהמאגר סיסמאות נפרץ-עדיין לא ניתן לקרוא אותו. אבל אפשר להשתמש בו. וגם מהווה חותמת דיגיטלית.

התקפות קריפטוגרפיה

Chipher text only: המידע הזמין לתוקף הוא רק CIPHER TEXT, המידע המוצפן בלבד.

Known plaintext attack: מה שזמין לתוקף הוא גם CIPHER TEXT וגם חתיכות/חלקים של CIPHER ו PLAINTEXT אחרים-ע"י ניחוש או שפגומים.

**Examples:**

Pretty good privacy- very strong encryption that is very hard to break

Bitlocker – encryption of a disk(Microsoft).

# **PYTHON**

[סקריפטים](https://github.com/meditator3/devops_scripts/tree/main/Python/project1)

Print(“Hello World”) = print(“Hello” + “ “ + “World”)

Print(‘ “Hello-world”? ‘) : מדפיס עם הגרשיים

Name = input(‘type your name: ‘) יקבל את הערך המוקלד name

Print(“hello\nhello2”) יודפס בשתי שורות נפרדות

Print(“1\t2\t3\t4\t5”) 1 2 3 4 5

ESACPES

Print (‘ the pet shop owner said “No, no, \’e\’s uh, ..he\’s resting”.’) the pet shop owner said “No, no, ‘e’s uh, …he’s resting”.

print(""" eberything ff' ' ' "" '''' "" '''' can be entered here you know""")

מכניס כל דבר לתוך 3 גרשיים

Print(4/3) = 1.25 float

Print(4//3)= zeros the float to integer (unless 4//3.0, where its still float)

Loops:

1. Only with integers.
2. Once declared range- cannot dynamically change inside the loop
   1. For I in range(a, b):
   2. b=b+10 <<< wont change the range

[דוגמאות LOOP סקריפט](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/for-loop.py)

[While loop scripts](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/while%20loop.py)

For i in range(1,13):

For key in <dict> :

For x, y in [(1,2), (2,3), (3,4)]:

חזקה = \*\*

Conditionals

If <condition> :

Else :

Elif <condtion> :

While <condition> :

Continue, break = ממשיך את הלולאה או יוצא

<variable> = <variable1> or <variable2> נותן ערך בהתאם אם התנאי מתקיים.1 אם מתקיים.

2 אם 1 לא מתקיים

[Conditionals scripts](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/conditionals.py)

Functions:

Def <name of function>(<values to pass into it>): הגדרת פונקציה

Return <value we want the function to return>

Def calc(weight, system = ‘metric’) מגדיר ברירת מחדל אם לא הועבר ערך בסיסטם

[Function scripts](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/functions.py)

Operators:

// = חלוקה ללא שארית מעגל למטה תמיד

A % B כמה שארית יש לאיי אחרי חלוקה לבי =

[Operators Scripts1](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/operators2.py), [operators scripts2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/operators.py)

Sequence type [list, string, tuple, range, bytes & bytearray]

[Script list1](https://github.com/meditator3/devops_scripts/blob/main/Python/list.py) , [script list2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/list2.py)

list[-1] = מתחיל מהסוף

list[0:6] = מציג את 0-5

list[:9] = מציג 0-8

list[10:] = מ10 ואילך

list[:] = על הטווח

list[0:6:2] = בטווח של 0-5 מדלג על 1 כל פעם

list[1::4] = מתחיל מאינדקס 1 עד הסוף וקופץ 4 כל פעם

list[25:0:-1] = מסדר אותו ב-סדר הפוך

השמת ערכים במשתנים

<var> += <value> מוסיף את הערך למשתנה/מערך

a, b = b, a הפיכת הערכים במשתנים:

<value> in <array/string/tuple/> בודק אם קיים הערך שם

או not in

Booleans:

“b” > ”a” : אמת. כי משווה לפי סדר הצ'רקטרס

1 and 2 : מחזיר 2 כי שניהם אמת, מחזיר את האחרון(2)

0 and 1 : מחזיר 0 כי הוא נחשב לא אמיתי בפייטון, לכן מחזיר את הראשון(0)

If False ----כל ערך לא נכון יוחזר בתנאי כלא-נכון

[Conditionals scripts](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/conditionals.py)

[Comparisons script](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/comparisons.py)

None = NULסוג של

True/False - None will give false value when prompted.

Tuple

Fixed array values. Constants. But can add cells(not values).

Len(<array/tuple/string>) מחזיר את אורך המחרוזת/מערך/טופל:

Tuple(<array>) : ממיר את המערך ל-טופל

[Tuple scripts1](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/tuple.py), [Tuple scripts2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/tuple2.py)

Methods

<string/array>.join(<string/array>) מחבר בין המחרוזת/מערך :

Int(num) : הופך למספר שלם

Float(“1.1”) : ממיר את המחרוזת למספר עשרוני

<array>.remove(<value>): מסיר את הערך מהמערך

<array>.pop(<value>) : מוריד את הזנב של המערך

<array>.append(<value>) : מוסיף את הערך למערך

[דוגמאת סקריפט למתודות במערך](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/list2.py)

Type(<var/value>) מציג את הסוג של הערך או המשתנה

Len(<array/tuple/string>) מחזיר את אורך המחרוזת/מערך/טופל:

Tuple(<array>) : ממיר את המערך ל-טופל

Str(<value>): ממיר הכל למחרוזת

<string{key} => {value}>.**format**(key=<dict key>, value=<dict val>)

Print( **f”**<string{value we pass}> : {template literals}.” 🡨instead of format

age = 24  
print("my age is {0} years old".format(age))

[דוגמאת סקריפטים](https://github.com/meditator3/devops_scripts/blob/main/Python/replacement_prac.py) להטמעת ערכים במחרוזת

[Template literals script2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/template_literals_format.py)

<string>.startswith(<value>) : אם המחרוזת מתחילה בערך הזה

Min(items) מחזיר את הערך הנמוך ביותר ברשימה

Max(items) highest value of list

Key value pairs/ Dictionary

An object that has a key as a name and a value –

<key> : <value> ex. {‘bob’:64, ‘age’:’old’}

[Example 1 script](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/dictionary1.py)

[Example 2 script](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/dictionary2.py)

Dict([(tuple2D)]) : ממיר מערך שיש בתוכו נסטד טופל של 2 משתנים כמו מילון

Dict(kevin=160, bob=50, jana=30) : מגדיר מילון {‘kevin’:160, ‘bob’:50, ‘jana’:30}

<dict>.values() : פולט ערכי המילון

<dict>.keys(): מפתחות המילון

<dict>.items() : מחזיר את הצמד מפתח:ערך

Strings

“\n” מוריד שורה

“\t” עושה טאב

‘ \” hello\” ‘ : מתייחס לגרשיים בתוך המחרוזת ולא סוגר, אפשר לשים כל סימן

**“””** 342/4erw/34’’’/// \\~````` **“””** הכל נכנס בשלוש גרשיים, כולל ירידת שורות ומרווחים

[סקריפט מחרוזות דוגמא](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/print_escape_chr.py)

**1סיפריות**

Import time

: package from lib, that has reserved functions

אבל כל קריאה לפונקציה זה יהיה דרך השם שלה:

Time.localtime() כל המתודות נצטרך לייחס דרך הסיפריה טיימ

Time.strftime(‘%X’, time.localtime()) מציג בפורמט קריא

Time.mktime(time.localtime()) : ממיר לשניות

אם לא נרצה לייבא את כל הסיפריה:

From time import localtime, mktime, strftime

ואז לא נצטרך לייצג דרך טיימ

Localtime() במקום time.localtime()

[Example script time package1](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/libraries-stuff.py), [example time script2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/libraries-stuff2.py)

משתנה סביבה -לתוך ווינדוז

Import os : ייבוא ספריית מערכת הפעלה

Os.environ[“<שם משתנה>”] מושך את המשתנה מתוך הסביבת משתנים של ווינדוז

Os.getenv(“שם משתנה”) מושך ערך מהמשתנה אם קיים ואם לא- מייצר אותו

[סקריפט משתני סביבה](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/environment-var.py),

[Script env.var.injecting one](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/environment-var2.py)

**FILES**

Type <name of file> : cat <file>מדפיס את התוכן של הקובץ, כמו

<variable for file> = open(“<filename>”, ‘r’) : פותח קובץ –

‘r’: read

‘w’: write

‘r+’: read + write

‘a’ : append to file מוסיף לקובץ

Xman\_file = file(“xmen\_base.txt”, “r”) : “r” means read.

Print(<variable for file>.read()) : מדפיס את הקובץ בעזרת READ אבל צריך לאפס את הסמן שיהיה אפשר לקרוא שוב

Xman\_file.read()

<variable for file>.seek(0) : מאפס את הסמן שאפשר יהיה לקרוא שוב מהקובץ

Xman\_file.seek(0) – READואז להדפיס מחדש עם

<variable for file>.close() : סוגר את הקובץ ללא זמין

[Script\_file/read\_ex1](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/file-interactions.py)

<variable for file>.write(‘<text>’) : כתיבה לתוך הקובץ, צריך להיות פתוח לכתיבה

New\_xmen = open(‘new\_xmen.txt’, ‘w’)

New\_xmen.write(xmen\_base.read()) >> this pours old xmen to new xmen file using write

[Script file/write ex2](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/file-interactions2.py)

<file variable>.name : xmen\_file.name is the same as saying xmen\_base.txt

With <object example:open(‘<filename>, ‘a’) as f:

f.write(‘<something>’) משתמש באובייקט שאחרי ה"עם" ומגדיר אותו כמשתנה שבחרנו, "אף", ואז ניתן להשתמש בו

with open(‘xmen\_base.txt’, ‘a’) as f: <<< f is a declared variable, of that file

f.write(‘Professor Xavier\n’)

שימוש ב"עם" גם סוגר את הקובץ ללא צורך להשתמש ב

f.close()

[script file/with ex3](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/file-interactions3.py)

.strip() – מוריד ספייס, מקצץ, או איזה ערך שנשים בסוגרייים.

Strip(“a"( : a יקצץ כל

Line.strip() : יחזיר אמת אם קיים כל כרכטר.

f.writelines(<array of lines>) : writes lines for each cell of an array

[example writelines script](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/exe5-userInto-file-class.py)

**סיפריות2**

Passing arguments in a file name( like echo $1, $2)

Need lib sys-

Import sys (std library)

Sys.argv[0] – leads to the path of the file in question. From 1 and above it means the args beyond it.

Print(f”first argument : {sys.argv[0]}”) – will print the path of the current file running this.

Print(f”Second argument : {sys.argv[1]}”) – if you execute <script\_name>.py running\_now it will print-

Second argument: running\_now >>> but this can still result in an error if there’s no argument,

or there are more.

Positional args:

Sys.argv[1:] will use any argument(args) we pass. Even if we don’t pass one. But it puts on a list/array on default.

[Example args-passing script](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/argument_passing.py)

Running a script without .py

Depends on : import argparse (std library)

Process: create .py file, use the usr/bin/env/python 3.6 at first.

[Script for CLI-based-reverse-file](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/file-reverse-CLI.py),

[Almost Same script-adapted to linux system](https://github.com/meditator3/devops_scripts/blob/main/all%20scripts%20from%20ubuntu%20server/reverse-file)

תפיסות שגיאות

try:  
 f = open(args.filename)  
 limit = args.limit  
except FileNotFoundError as err:  
 print("Error: this file does not exist! {err}")  
 #to make it as an error, otherwise it won't be marked as an error  
 sys.exit(1)

[דוגמאת קוד לתפיסת שגיאות](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/file-reverse-CLI-error.py)

סיפריות3

import subprocess : מאפשר שימוש בפקודות לינוקס/באש

כמו LSOF

[PROCESSדוגמאת סקריפט שהורג](https://github.com/meditator3/devops_scripts/blob/main/Python/project1/port-kill-process.py)

# DOCKER

בניית או הורדה של דמויות שהן יכולות להיות אפליקציות או מערכות הפעלה, או שתיהן.

הדמות היא התבנית. הקונטיינר זה שהתבנית רצה והיא זמינה לשימוש, אם זה אפליקציה, שרת או מע' הפעלה.

docker run -itd <name of image> אפשר להוסיף –name <container name>

docker logs <name of docker>: לוג

רשימת לוקלי IMAGES: docker images

רשימת הקונטיינרים: docker ps -a

מחיקת IMAGE: קודם לעשות STOP לקונטיינר- docker stop <name of container> <several names>

ואז – docker rm <name of container>

למחוק IMAGE- docker rmi <image name>

הפניית פורטים מהמכונה(צד שמאל) לאינטרנט: docker run -itd –name ariel-nginx -p 8080:80 nginx

-d חשוב DETACHED, זה מה שמאפשר להגיע מהאינטרנט

כניסה לDOCKER עצמו (אם זה UBUNTU): docker exec -it ariel-nginx /bin/bash מכניס אותך לטרמינל של הדוקר

אתחול/התחלת קונטיינר שנעצר: docker start <name of container>

VOLUME: חלוקת קונטיינר בתוך וולויום שהגדרנו, חולקים משאבים ורשת.

Docker run -itd -p 8080:8080 -v /var/www node

-v מייצר את הווליום. אם לא נשים הנדל -v לא יישמרו והכל יימחק!

זיהוי VOLUME: docker inspect <name of volume>

הורדת גרסא הכי אחרונה של IMAGE: <image name>:latest ששמים בסוף פקודת הרצת הDOCKER

Docker top – איזה PROCESS עובדים עכשיו

Docker stats – סטטיסטיקות של הקונטיינר

docker push <username>/<repository>

להכנס לדוקר HUB: docker login <user and password>

Docker volume ls : מציג רשימת ווליומים

Docker system prune –force --all –volumes : לא משאיר זכר לDOCKER שבמכונה

לצאת מתוך SHELL של CONTAINER : CTRL P ואז CTRL Q , אם הרצת docker -it

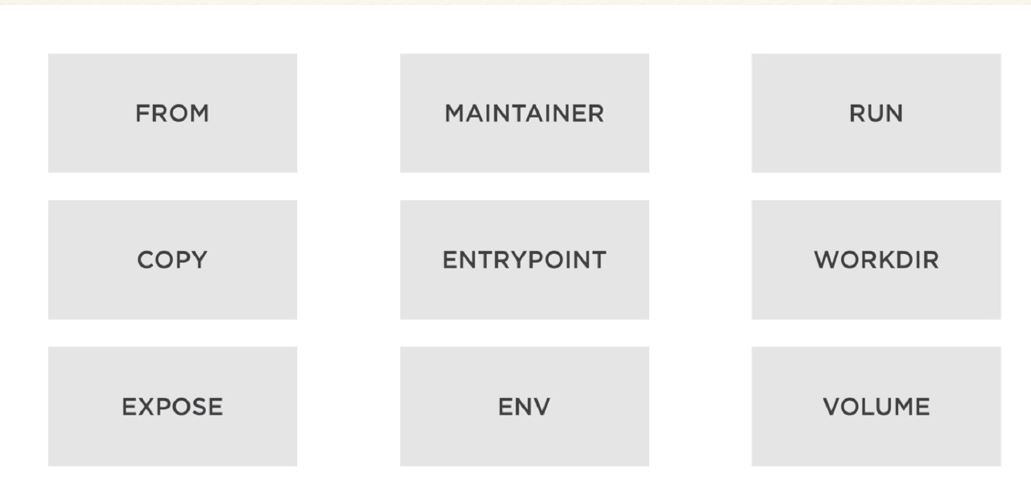
# DOCKER FILE-

מייצרים אותו כהוראות לבניית דמות הקונטיינר, שאותו אח"כ נפרוש בDOCKER BUILD.

כדי שתהיה לנו גישה לדמות מבחוץ אנחנו מעלים לDOCKERHUB דרך DOCKER PUSH

יש להוריד מה REPO כדי שיהיה לנו את כל הנתונים של האפליקציה שעובדת עם הבסיס של הדמות.

לדוגמא אפליקציה של FLASK עובדת על פייתון-ולכן צריך שיהיה את הקבצים של האפליקציה מהריפו- לוקאלית, וגם את הIMAGE של פייתון.



<docker file is inside the app main folder:”Dockerfile” > עם f קטנה!

FROM <python/js/language>

WORKDIR <PATH we want to put temp files>

COPY <what to copy(. If in current folder) where to copy(. If in same WORKDIR folder)>

Ex:

COPY requirements.txt (PYTHON)

RUN <what we run while installing/building,usually dependencies>

Ex:

RUN dotnet restore

RUN dotnet publish -c Release -o /app

RUN apt update && apt install -y maven (JAVA)

RUN mvn clean install (JAVA)

RUN mvn package (MAVEN)

RUN npm install (NODE, EXPRESS JS)

RUN pip install –no-cache-dir -r requirements.txt (PYTHON)

ENTRYPOINT [“<run command(node, python3)>”, “<what to run>”]

Ex:

ENTRYPOINT ["dotnet", "YourApp.dll"]

ENTRYPOINT ["java", "-jar", "your-app.jar"]

ENTRYPOINT ["node", "app.js"]

CMD [ "python", "app.py" ]

Docker build -t <image name>:v1 . : this builds an image according to dockerfile

וודא שיש לך FROM מגרסא שמתאימה לאפליקציה ולא סתם LATEST

בכלליות הDOCKERFILE – משתמש בדמות ממקום אחר כרפרנס, ומריץ DEPENDENCIES, וגם מה שצריך כדי להריץ את האפליקציה.

docker push your-docker-hub-username/my-nginx-image:v1

לפני PUSH לעשות LOGIN

לבדוק שהאפליקציה רצה והקונטיינר:

docker run -p host\_port:container\_port -itd <image name>

* host\_port is the port on your host machine where you want to access the container's application.
* container\_port is the port on which the application inside the container is listening.
* -itd : interactive+ tty, keeps STDIN open+ detached(runs in background)

**DOCKERFILE**

Maven/java: [“java”,”-jar”,”<app file name.jar>”]

Multiphasic: FROM <value> AS build

Next phase:

COPY –from=build <path of target> .

Ex:

(maven) COPY pom.xml /workdir

EXPOSE : לחשוף פורטים

**DOCKERFILE EXAMPLE NODEJS**

# Use the current directory as the build context

# No need for a base image, as you're building from the local files

# You can leave this line out altogether if desired

FROM node:4

# Set the working directory inside the container

WORKDIR /app

# Copy the entire project directory into the container

COPY . .

# Install application dependencies

RUN npm install

# Expose the port your application will run on

EXPOSE 8080

# Define the command to start your Node.js application

CMD ["node", "server.js"]

**NETWORK**:

Docker network ls

Docker network inspect

Docker network create <name> --driver

Docker network connect

Docker network disconnect

Docker uses aliases to avoid using static ip’s , has inherent DNS(domain name system) so it

Can interact with each container instead of using ip’s.

# COMPOSE

Compose מאפשר לנו לבנות כמה קונטיינרים בקובץ YAML אחד, בד"כ שיתקשרו בינהם עם רשת שנקבע(ראה לעיל)

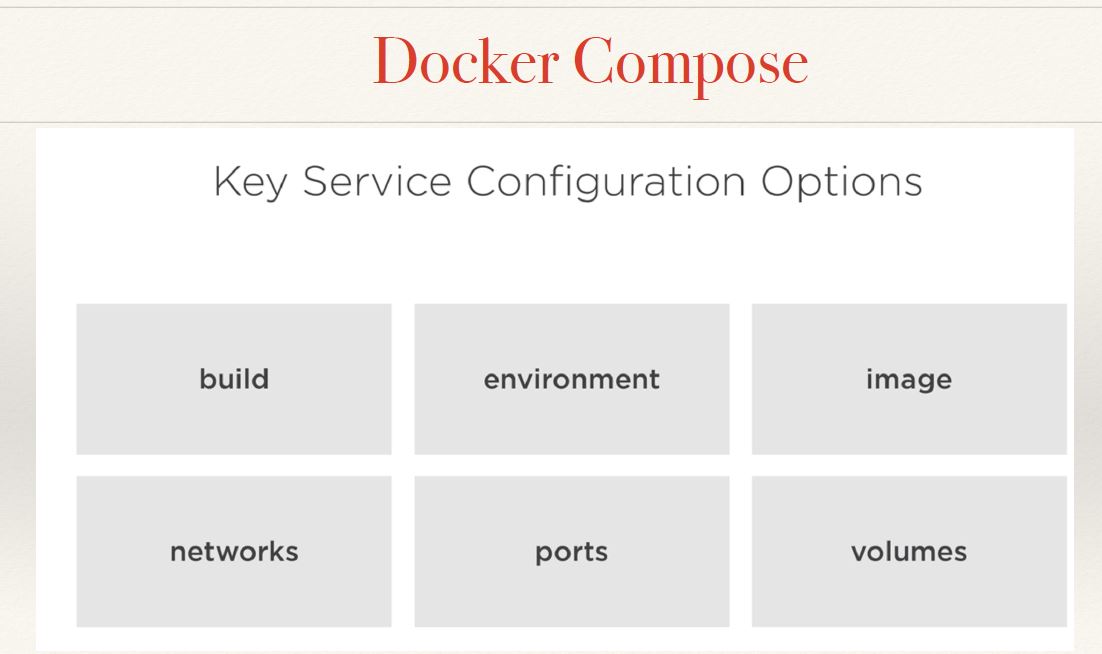
COMPOSE לוקח מהדמות שייצרנו בDOCKERFILE, ומניח SERVICES עליהם. כדי שיתקשרו וגם כדי להרים את כולם.

בד"כ גם מייצרים ווליומים כדי שהאפליקציה יהיה לה PRESISTENT STORAGE, נניח כמו DB, שדורש זכרון שמתעדכן.

פקודות:

Docker compose <up, version, down,run, logs,start, port, event>

sudo docker compose down -v --remove-orphans : מוריד מה שנתקע ולא יורד בDOWN רגיל.



Filename: docker-compose.yml or .yaml

docker-compose down --volumes --rmi all : אם רוצים להוריד את כל מה שהעלנו בקומפוז

docker compose up -d : -d is detached, so it can work in the background

מציג רשימת כל הווליומיםdocker volume ls:

-p זה PUBLISH, לדוגמא 8080:3000 זה אומר שהמכונה המארחת (127.0.0.1) מדברת עם הקונטיינר דרך 8080, והקונטיינר מדבר ב3000

Version: “x.x” – number we choose as a start version for the docker compose file itself

Services:< what images we are pulling and their config>

(two spaces in the indent field)

Drupal: (container name)

(two spaces in the)

Image: <image name>

Ports:

Add minus before port number

8080:8080

Networks:

-drupal-network<a name we choose for the network>

./ <PATH> : <PATH WE INSTALL TO> <the ./ -uses the relative path.where YAML file is >

Ex:

Volumes:

אם מוגדר מחוץ לBUILDS אז צריך לעדכן בתוך הבילד את ה PATH, דוגמא:

NESTED בתוך SERVICES(לא בתוך איזה BUILD):

Volumes:

Frontend\_vol:

בתוך FRONTEND(BUILD):

Volumes:

-<space>frontend:/var/www/

-<SPACE>./modules:/var/www/html/modules

Environment:

<passwords or user names we need for auto login> Ex:

-<space>MYSQL\_DB=mysql

-<space>MYSQL\_USER=user

-<space>MYSQL\_PASSWORD=pass

בשימוש עםDOCKERFILE (שבונים YAML בעזרת דוקרפייל)

Build:

Context: ./<PATH of DOCKERFILE>

\*\*\*\* קונטקסט חייב לקנן בתוך BUILD \*\*\*\*

Networks:

<name of network>:

Ex:

Networks:

Webapp-net: (indented inside Networks)

**לוודא פורטים שנחשפו בYAML COMPOSE שהם גם פתוחים בVM BOX!!!**

**וגם לבדוק איך הBACKEND מתקשר עם הDB(MYSQL HOST:DB) לדוגמא**

Docker compose up -d – מרים את הקומפוז בצורת DETACHED בגלל זה הD(אחרת הוא תוקע את הטרמינל)

Docker compose down – מוריד את הקונטיינרים-ומוחק!

# **JENKINS**

מה זה?

פלטפורמה להרצת CI לבדיקת BUILD שונים וגם ייצור ארטיפקטים לשלוח לקליינטים או למפתחים.

יותר טכני:

כותבים בשפתPIPELINE DECLERATIVE ואפשר להשתמש בGROOVY SYNTAX כדי לקבל את הSYNTAX המורכבים.

בשלב צ'ק אאוט, הג'נקינס פותח סיפריית(תיקיה) WORKSPACE, שאיתה הוא עובד, ומתקין DEPENDENCIES, DOCKER BUILD וכולי.

נעזרים בצד שלישי כדי לקבל יותר אפשרויות כמו SONARQUBE לבדיקות קוד, ARTIFACTURY/JFROG או NEXUS כדי לייצר ארטיפקטים ולשלוח אותם

או BLUE OCEAN שזה גם תצוגה יפה יותר של CI

רכיבי JENKINS:

Agent: מכונה שמריצה JOB או PIPELINE עבור השרת הראשי של JENKINS , כמו SLAVE

Executer: מיוצג בד"כ ע"פ כמות CPU, כמה THREAD יכול לרוץ במקביל, לדוגמא אם יש לנו 2 STEPS הם ירוצו במקביל שיש לנו 2 EXECUTERS.

Menus:

Security->Authentication->LDAP

מאפשר לי להגדיר את כל ההראשות לפי קבוצה, משתמש או אפילו JOBS.

דרך-

-<MATRIX BASED

מאפשר לי להגדיר/להוסיף יוזר, קבוצה וכולי.

-אפשור למשתמשים להירשם עצמאית:

ALLOW USER TO SIGN UP

הרמת PIPELINES 1:

1)פורק לתוך המכונה מגיט

2) ווידוא שיש את הפקודת הפעלה מהדוקר(אם מריצים מריפו) צריך להוריד את הכל לתוך המכונה קודם

אם אין-צריך להתקין או דרך APT או דרך WGET והתקנה דרך סימבוליק לינק:

Ln -s <path of command folder> <path of usually /usr/local/bin/>

אפשר לבדוק דרך WHICH

וגם כדאי לעשות סימבוליק לינק לפקודה שתרוץ על ה LATEST דרך אותו רעיון

3) הרצה של האפליקציה-לוודא שהפורטים הנכונים פתוחים ושהכל פתוח גם בVM וגם בFIREWALL.

וגם שהנתיב נכון.

4)בדיקת אינטגרציה- CURL על הכתובת עם הפורט ואפשר לעשות תנאי החזרות שגיאה

5) כיווץ ארטיפקט-האפליקציה, בצורה אוטומטית(דרך הסקריפט)

6) שחרור לתוך הPIPELINE שיהיה זמין להורדה דרך פקודת ARCHIVE שקיימת דרך SYNTAX

**הרמת SLAVE לJENKINS שיריץ DOCKER**

-לפתוח מכונה חדשה(CENTOS) שעובדת BRIDGE וגם בTOMCAT JENKINS משנים ל BRIDGE

- בודקים ב IP ADDR במכונת SLAVE מה הכתובת שהוגדרה ומתחברים SSH

- הכניסה לJENKINS בדפדפן היא עדיין דרך פורט 8080 אבל הכתובת החדשה (של הMASTER לא של הSLAVE)

- משנים שם בSLAVE כדי לא להתבלבל הין הGIT BASHים. NMTUI בSUDO

- לוודא שאנחנו ROOT ולא משתמש אחר

- אם אין הגדרת רשת בSLAVE יש להגיע ל vi /etc/sysconfig/network-scripts/ifcfg-enp<number>

ולשנות ONBOOT=YES

* להוריד ולהתקין דוקר בSLAVE : curl -fsSL [https://get/docker.com -o get-docker.sh](https://get/docker.com%20-o%20get-docker.sh)
* אתחול DOCKER ולוודא שהוא רץ : systemctl status docker וגם docker info לראות שאין שגיאה
* צריך גם להתקין JAVA כדי שהSLAVE ידע לדבר עם JENKINS שעובד בJAVA
* Yum install java-11-openjdk-nevel
* MKDIR JENKINS בתיקיית ROOT

עוברים לדפדפן ב JENKINS:

* פותחים NODE חדש
* מסמנים PERMENANT AGENT
* 2 בCPU , לשנות REMOTE ROOT DIRECTORY לאיפה שהסיפריה של JENKINS ב SLAVE(ROOT)
* USAGE בוחרים אך ורק ניהול שלנו ONLY BUILDS MATCHING THIS NODE אחרת ג'נקיס יחרבש את כל הפייפליינים של כל החברה
* LAUNCH VIA SSH הMASTER והSLAVE מדברים דרך SSH
* מכניסים את הכתובת של SLAVE.
* כאן מייצרים CREDENTIALS שג'נקינס שומר ומייצר כדי שלא יהיה חשוף לכל עין.
* אז עושים ADD לCREDENTIALS ושמים את הפרטים, שיהיה ROOT
* בוחרים ב CREDENTIALS המרשימה את מה שהזנו עכשיו
* בוחרים NON VERFIABLE ו SAVE

עכשיו הNODE של הSLAVE צריך להיות ONLINE רואים בSTATUS וגם צריך לראות EXECUTERS(CPU) שזמינים.IDLE 1 IDLE 2.

INTEGRATION AND TESTING

* פותחים PIPELINE חדש ונכנסים לSYNTAX
* בוחרים NODE ושמים בלייבל את המכונה שאיתה אנחנו עובדים(לדוגמא DOCKER). מדביקים ל DSL (declarative syntax)
* CHECKOUT בוחרים REPO ומדביקים הכל לDSL,
* אם לקחנו מGIT וזה לא מותקן בSLAVE נקבל שגיאה ב BUILD NOW, אז להתקין GIT(YUM INSTALL GIT)
* בונים את הCONTAINER עם BUILD -T אבל צריך שיהיה מוכן REPO FOLDER אחרת זה יהיה חסר שם ולא משוייך.
* נעשה REPO חדש לCONTAINER שאנחנו בונים עם השם שנבחר
* השם של הREPO שבחרנו זה השם של הIMAGE. ולכן נשים אותו בפקודת BUILD בDSL
* הגענו ל sh”docker build -t arielguez/hello-from-jenkins”:
* נוסיף גרסא. שצריך להשיג מתוך (לא מתוך SYS INFO >>ENV) כי זה עלול לדרוס קודמים. אנחנו רוצים את הגרסאת BUILD הנוכחית
* כדי להגיע לזה נפתח JOB חדש. נבחר EXECUTE SHELL, ושם יש רשימה של כל AVAILABLE ENVIRONMENT
* ואנחנו צריכים את BUILD\_NUMBER שזה מספר ההרצה-המספר שהפייפליין רץ בBUILD (1-2-3 וכולי)
* אז נוסיף אחרי : ${env.BUILD\_NUMBER} וזה ייתן את המספר לעיל.
* Sh”docker build -t arielguez/hello-from-jenkins:${env.BUILD\_NUMBER} .”
* תזכורת- הנקודה בסוף הסקריפט הזה היא המיקום של הDOCKERFILE. כלומר אם היה בPATH אחר יש לציין
* ERRORS:: לוודא שהSLAVE אכן רץ ברשימת NODES. וגם אם זה WAR לוודא מה שם האפליקציה בתוך הקונטיינר(EXEC) ואז לשים כROUTE בLOCAL HOST לדוגמא: localhost:8888/hello-world-war-1.0.0
* נוסיף לSTAGE TEST: sh “docker run -itd –name <new name> -p 3000:3000 arielguez/hello-from-jenkins:${env.BUILD\_NUMBER}” יש לשים לב ששמנו את הנתיב והשם המלא של הIMAGE.
* יש להוסיף SLEEP לפני ואחרי CURL כדי שהוא יספיק לקרוא אותו, וגם צריך לסגור אותו בהרצה הבאה
* Sh “sleep 4”
* Sh “curl localhost:3000” זה הבדיקה שהאתר רץ
* נשרשר שתי פקודות באמצעות &&. אחת לעצור את האתר ואחת למחוק.
* Sh “docker stop hello-from-jenkins && docker rm hello-from-jenkins”
* עבור STAGE PUSH TO DOCKER:
* Docker login בשימוש עם WITH CREDENTIALS: מוסיפים בהגדרות עוד+ קרדנציאלז. ומזינים לפי ההגדרות של DOCKER HUB. ואז נותנים משהו פיקטיבי לJENKINS שיראה בDSL
* withCredentials([usernamePassword(credentialsId: ‘docker login', passwordVariable: 'pass', usernameVariable: 'user')]) {
* sh "docker login -u $user -p $pass"
* sh "docker push arielguez/hello-from-jenkins:${env.BUILD\_NUMBER}"

הרמת PIPELINE 2

- **NODE**,.

Node(‘docker’) {

STAGES

באינדנטציה, NESTED בתוך-

**Checkout**

לוקחים את הקוד העדכני כדי לעדכן את הפייפליין

Stage(‘Checkout’) {

קוד עבור כל שלב(מGENERATE PIPELINE SYNTAX)

אם לוקחי מGIT לבחור הורדת CODE>> HTTPS. כדי להדביק לCHECKOUT

**BUILD**

Docker build -t <name WITH repo name for ex:> Arielguez/hello-world: ${env.BUILD\_NUMBER}

**TEST**

Docker run

Curl

שלבים אחרים

Archive, artifact, garbage disposal

# **BLUE OCEAN**

-blue ocean from Jenkins>new pipeline

- select github(or other repo)>access token:

- to get access token: click to get from link>authenticate>(inside github):”new personal access token”, type ref.name we want+expire time+generate

-copypaste to blue ocean>choose your main repo of github>select sub-repo to work on the pipeline

- ניתן לראות את כל הפייפליינים והג'ובים

*pipeline layout*:

- click start>select agent (node) and label for it (docker or other)

- put all stages > checkout, build, test, artifact, push to dockerhub

-for checkout select GIT from list> paste URL and branch name

-for build>shell script from list

- test> same as DSL

- artifact> zip or tar(usually in the folder itself) tar -czvf nodejs.tar.gz \* <<the \* is because all files in the folder

- push to dockerhub> need to use bind credentials with variabls>docker login -u $user -p $password (shell)

+ docker push arielguez/reponame:$BUILD\_NUMBER (DSLבשונה מ )

# **STATIC CODE ANALYSIS**

# SONARQube

מאפשר STATIC CODE ANALYSIS – הערכה של קוד ללא הרצה שלו, מספק צפיה בVULNERABILITES, SECURITY, מאפשר שערים GATES בהתאם לבעיות שנצפו.

שזה בדיקת קוד כמו בוויזואל סטודיו – אבל עם אפשרות לחסום קודים ספציפיים על ידי RULES.

ואז לחסום או להוריד PIPELINES

SonarQube = לוקאלי. Sonar Cloud = בענן, פשוט יותר

יש לו TOKEN שהשרת JENKINS נותן וצריך לשמור

אפשר לקחת מGIT דרך SONAR CLOUD, ומראה לך בעיות על כל קוד שיש בכל REPO

כיצד רואים?

נכנסים לאחד הREPO שיש בו בעיות – ולוחצים על המספר של הבעיות , וזה מציג.

בSONAR SCANNER CLI, נצטרך לעשות קובץ בGIT sonar-project-properties

לא נשים STATIC CODE ANALYSIS על COMMIT, כי זה יאיט והמפתחים יתקעו. אבל כן נשים על MERGE בין BRANCH

# JIT Security

# Snyk

# **GIT ACTION**

מעבר לCI כמו בג'נקינס, Git-Action מספק לך HOSTING לTESTING. מה שנקרא RUNNERS.

Supercharge workflow model הוא מספק הטמעה של כל תהליך CI CD בכל שלב של הפיתוח. לדוגמא, בתוך FEATURE אחרי גרסא ראשונית.

הגדרות של ההטמעה יכולות להיות לפי מיקום בבראנץ, או לפי EVENT טיפוסי או כמה במקביל.

Runner זה כמו NODE בג'נקינס, וזה מה שגיט אקשן משתמש בDEPLOY של COMPOSE או DOCKERFILE.

לדוגמא, מציינים זאת דרך RUNS ON וכל מה שנכתוב עם LATEST זה יבחר RUNNER של GIT.

אם נרצה שלנו (SELF HOST) -צריך לציין.

אפשר לקחת תבנית מוכנה עם פקודות מוכנות בהתאם לפרויקט.

אפשרויות עבור ON – release, pull, check\_suite, deploy

או הפעלה על פי שעון/טיימר - on: schedule:

אם אנחנו מפעילים כמו פעולות על הקונטיינרים שרצים בRUNNER של ACTION, אז משתמשים בשמות של הקונטיינרים כמו IP.

לדוגמא\_COMPOSE)

services:

frontend:

image: your-frontend-image

ports:

- "3000:3000"

בדוק שם הקונטיינר:

DOCKER PS -A

ואז בACTION:

- name: test\_frontend\_container

run: sleep 5 && curl <container name>:3000

X not working

To get own runner (self-hosted:

Settings>actions>runners

Choose OS והתקן את הפקודות בRUNNER שלך.(במכונה)

On:

Push: <<<< only on push command

Branches:

-master <<<< only for master branch

Tags:

-v1 <<<< only applies over v1 commits

Paths:

-‘test/\*’ <<<< only files inside test folder

SCRIPT EXAMPLE- AND MORE ON SCRIPTS:

name: trigger workflow

# Controls when the workflow will run

on:

# Triggers the workflow on push or pull request events but only for the "main" branch

push:

# pull\_request:

# branches: [ "main" ]

# Allows you to run this workflow manually from the Actions tab

# workflow\_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel

jobs:

# This workflow contains a single job called "build"

trigger:

# The type of runner that the job will run on

runs-on: ubuntu-latest

# Steps represent a sequence of tasks that will be executed as part of the job

steps:

- name: trigger another workflow

run: |

curl -X POST https://api.github.com/repos/${{ github.repository }}/actions/workflows/issue.yml/dispatches \

-H 'Accept: application/vnd.github.v3+json' \

--header 'Authorization: Bearer ${{ secrets.REPO\_ACCESS\_TOKEN }}' \

--data '{"ref": "${{ github.ref }}"}'

בפתיחת WORKFLOW של REPO, יש לשים לב בהגדרות הACTIONS>WORKFLOW PERMISSIONS שאפשר לעשות READ WRITE PERMISSIONS

TRIGGER כהפניה לWORKFLOW אחר, יש לשים לב שאנחנו מציינים ב CURL את ה PATH ושם הקובץ YML של הWORKFLOW המופנה

בתוך DATA יש גם לציין את הBRANCH אבל זה נגיע דרך

--data '{"ref": "${{github.ref}}"}'

יצירת WEBHOOK ל SLACK:

פותחים SECRET חדש ב: CURRENT REPO>SETTINGS>SECRETS>ACTIONS>NEW REPO SECRET

ב SLACK יש להכנס ל WORKSPACE שנמצא מצד שמאל למעלה>תפריט>ADMINISTRATION>MANAGE APPS

CREATE NEW APP>SELECT WORKSPACE>לבחור שם>לסיים

בWORKSPACE שאנחנו בחרנו-MANAGE APPS> שם של האפליקציה שעשינו שלב קודם>BASIC INFORMATION>ALLOW WEBHOOKS>למטה:GENERATE TOKEN>COPY של הTOKEN

בGITHUB בריפו שאנחנו עובדים עליו>SETTINGS>SECRETS> להכניס שם הSECRET שיהיה בYML(SLACK\_WEBHOOK)>להדביק מהקודם.

ראה SCRIPT לפירוט

# **JFrog – Artifactory**

אחרי התקנה (טריאל זה 14 יום עם כל הדפנדנסיז), OSS גרסא חינמית אבל כמעט בלי כלום. רק מייבן.

יצירת חוק-

תפריט > XRAY > WATCHER & RULES > להכניס חוקים. נניח רק לעצור התקנות עם שגיאה קריטית.

יצירת WATCHER

- והשמה של החוק שלעיל. ואז כל התקנה עם שגיאה קריטית-לא תותקן או תמחק.

יצירת REPORT

בוחרים REPO, ועושים GENERATE(אם לא צריך עוד הגדרות)- ואז זה מייצר דוח עם כל הVULNERABILITIES. שאפשר לשלוח למפתחים.

(הם האחראים על פגיעויות של הארטיפקטים)

# **NEXUS & JFROG– ARTIFACTORY**

נקסוס וJfrog הם REPOSITORY MANAGER. מאחסן ARTIFACTS וגרסאות BUILDS.

NEXUS

תחת גלגל השיניים, ניתן לראות REPOS קיימים, BLOB STORE זה סוג של ווליום האחסון של נקסוס-שם נשמרים כל הARTIFACTS, אם נגמר המקום מייצרים BLOB נוסף.

SYSTEM- יש מה שנקרא SWAGGER. API של נקסוס. והוא מאפשר עבודה מולו דרך סקריפטים. (טוב לאוטמציה?), עבור כל תפריט שקיים DASHBOARD שמציג SCRIPT דרך הCURL פורמט JSON

הוספת REPO

בGROUP יש לציין HTTP עם PORT,(לסמן V) בהנחה שאנחנו מאחורי Secure Proxy .

צריך שיהיה גם HOST ו PROXY

הסרה של NEXUS חייבים להוריד יוזר NEXUS אחרת לא יעבוד

-חייבים להוסיף REALM TOKEN BEARER לכל REPO שנרצה להוסיף.security>realm

- עובד רק עם PRO LICENSE ב DOCKER NPM

-יש לצרף ריפו – HOSTED וPROXY כGROUP, ואז להסב פורט שתהיה גישה לריפו כי זה לא מאפשר HTTPS רק HTTP, אז מאפשרים דרך HTTP בGROUP

-יצירת חיבור בין JENKINS לNEXUS:

יש ליצור/לערוך במחשב המארח-WINDOWS- קובץ שנמצא בתור users/ariel/.docker בשם daemon.json:

להוסיף –,”insecure-registeries”:[<ip of nexus machine>”192.168.14.13: 8088”]

ואז יש לעשות docker login לIP של מכונת הNEXUS docker login 192.168.14.13:8088 עם הסיסמא של NEXUS!!

# **Kubernetes**

KUBERNETES ייחודי בגלל Scaling, ניצוח וניהול קונטיינרים עבור CLUSTER של מכונות. יכולת להרים כמה מופעים של אותה אפליקציה (horizontal(

יכולת ניטור מצב בריאות של שרתים וקונטיינרים וגם אתחול שלהם לרציפות בDEPLOYMENT.

\*\*\*\*סכנה: MASTER של קוברנטיס אסור שיפול!

(זה בקטנה על קוברנטיס, פירוט מלא בנושא הבא KUBERNETES CLUSTER)

MINIKUBE

---------------------

יש התקנה בPOWERSHELL לווינדוז:

Winget install minikube

Minikube version בדיקת הגרסא

פתרון שגיאות: אם יש בעיה FAIL-

Minikube delete –all

אם יש בעיה בהתקנה גם אפשר להוסיף לפקודות

Minikube start --no-vtx-check

POD: תאור האפליקציה, יכול להיות קונטיינר או שניים.

Minikube kubctl – get pods -A

אפשור של KUBECTL מכל מקום(גם ללכת לENV ולשנות ב PATH)

NODE: מכונות או מכונה וירטואלית או פיסית שמריצה services וגם את הPODS

בדיקת

צריך גם KUBECTL

סגירת MINIKUBE:

קודם SAVESTATE ואז minikube pause

וכדי להפעיל מחדש קודם מעלים בVM ואז minikube unpause

Minikube addon list

Minikibe enable addon

פקודות KUBECTL:

Kubectl get nodes: איזה מכונות רצות עכשיו

Kubectl get pods -A : איזה פודים רצים

Kubectl get all -A: מציג את כל ה RESOURCES

Kubectl describe node minikube : מתאר את פרטי הנוד של מיני קיוב-אפשר לבחור גם שם אחר עבור NODE אחר.

הרצת אפליקציה דרך POD:

קובץ YAML

Kubectl create -f <path of yaml file, and its file name> יצירת האפליקציה בפוד

גילויי של הדומיין : kubectl get pods -A

Kubectl logs <(domain name) >

Kubectl port-forward <name of app/pod> 8081:3000

מאפשר הרצה של האפליקציה בדפדפן ע"י הפניית פורטים

שיטות SERVICE:

Service נותן לנו גישה לאפליקציה דרך הדפדפן לדוגמא.

LoadBalancer-

-Node Port

NODEPORT

מאפשר כניסה לכל POD מאיזה פורט שנבחר כשירות, וגם מכל NODE

הרצת SERVICE:

Kubectl apply -f <name of yaml file>

בדיקת השירותים שרצים SERVICES:

Kubectl get services – או svc גם עובד

בדיקת הIP שרץ :

Cat c:\users\ariel\.kube\config

לחפש איי פי פנימי

ERROR אם יש להריץ דרך ADMIN את הPOWERSHELL

הרצת קונטיינר חיצוני מדוקר האב:

kubectl run -i --ty --image=busybox --restart=Never – sh

ומאחר ואנחנו מריצים מהמכונה הפנימית שלנו אפשר ממש מתוך הCONTAINER הזה שהבאנו

מDOCKER HUB לעשות PING ל SERVICE :

Ping helloworld -service

Minikube service helloworld-service: מציג את השירות וגם פותח אותו בדפדפן

REPLICATION CONTROLLER

מאפשר שיכפול HORIZONATLLY, שזה אומר שהוא לא מבזבז עוד משאבים ואנרגיה-בניגוד לוורטיקלי.

האפליקציות המשוכפלות חייבות להיות STATELESS – כלומר STATIC, שלא צריכות הזנה של נתונים מהCLIENT.

ראה SCRIPTS

Kubectl delete <name of container> ימחוק את הקונטיינר אבל בגלל שזה תחת REPLICATION CONTROLLER, זה ימחוק ויבנה אחד **חדש**

כדי למחוק את הקונטיינרים תחת RC:

Kubectl delete rc helloworld-controller

Kubectl get rc : יציג מה קיים בREPLICATIN CONTROLLER

REPLICATION SET(advanced version of controller)

חלק מ DEPLOYMENT(בCONTROLLER אין DEPLOYMENT)

לא מבוסס רק על SELECT על פי EQUAITY, ניתן לבחור בצורה אחרת גם

עובד יותר טוב עם עדכוני גרסאות

DEPLOYMENT

מאפשר קוסטמיזציה של עדכונים, הגדרת מצב עדיף של האפליקציה, יש ACTIONS:

Create - יצור DEPLOY וגם יאתחל REPLICASET בהתאם למספר השכפולים שהגדרנו

Update – ישלוט בSCALING וייצור REPLICASET חדש בהתאם לעדכון.

Rollback – לחזור אחורה גרסאות

PAUSE/RESUME

Kubectl get deploy מציג DEPLOYMENTS פעילים

Kubectl get pods –show-

Kubectl rollout status deploy/helloworld-deployment – הסטטוס בודק אם הDEPLOY הצליח

Kubectl set image deploy/helloworld-deployment k8s-demo=webdevian/k8s-demo:2 - מעדכן לגרסא 2 על ידי SET IMAGE

כדי לבדוק מריצים את הפקודה עם ה ROLLOUT

Kubectl rollout history deploy/helloworld-deployment : בודק את כל ההיסטוריה של הגרסאות

UNDO

Kubectl rollout undo deploy/helloworld-deployment

>>>>>לכן גם חשוב מאוד לא למחוק IMAGE אן לעשות kubectl delete –all -A כי זה יכול להשמיד את ה UNDO

LABEL

הדבקת תווית על אובייקט בדרך כלל בYAML(ראה SCRIPTS)

:kubectl get pods –show-labels מציג לייבלים של הפודז

Kubectl get rs –show-labels : יציג את הלייבלים של REPLICASET הנוכחיים

kubectl get deploy --show-labels: יציג לייבלים של ה DEPLOYMENT

node labeling:

kubectl get nodes : כדי לדעת את שם הNODE

kubectl label nodes minikube(node name) <label>hardware=low

kubectl get nodes –show-labels יציג אלנו את הלייבלים שעשינו- HARDWARE=LOW

TROUBLESHOOTING LABLES

נניח שהYAML כתוב hardware=high-spec

צריך לגלות איפה השגיאה כי הוא רק יעצור ולא יתן לעשות

APPLY

בגלל הקונפליקט בלייבלים -low<>high-spec

nodeSelector:

קובע באיזה NODE שבחרנו יהיה מחוייב הלייבל (ראה SCRIPT)

פתרון TROUBLESHOOT:

Kubectl describe pod helloworld-deploy: שם נמצא בתוך nodeSelector את הלייבל הלא מתאים

אם נרצה לעדכן ל HIGH SPEC:

Kubectl label nodes minikube hardware=high-spec –overwrite: OVERWRITE מאפשר כפיה

READINESS CHECK + HEALTH CHECK

ראה SCRIPT

HEALTH CHECK – עושה בדיקה אם האפליקציה מצליחה לעשות DEPLOY לדפדפן

Kubctl describe pod <name of pod> : יציג אם הHEALTH CHECK הצליח או נכשל

READINESS CHECK : בודק אם האפליקציה במצב RUNNING ראב SCRIPT

POD STATE

Pending: יש בקשה אבל לא רץ עדיין-או שמוריד לאט בגלל מרחק, או בעיית RESOURCES)

Succeeded: מחק והשמיד את כל הקונטיינרים ולא אתחל אותם

Failed: כל הקונטינרים הושמדו ואחד לפחות החזיר שגיאה

Unknown: כנראה בעיית רשת או חיבור שלא מאפשרת לבדוק את המצב של הPOD

הכל נמצא ב DESCRIBE

**SECRETS**

שיטת ENV VARIABLES, ודרך FILE:

שימוש בFILES:

Echo -n “root” > ./username.txt

Echo -n “password” > ./password.txt

זה שופך לתוך הקבצים האלו את השם משתמש והסיסמא-ללא שורה רווח.

Kubectl create secret generic db-user-pass –from-file=./username.txt –from-file=./password.txt

מייצר SECRET ג'נרי תחת השם user-db-pass מהקבצים שבנינו קודם

כדי לראות הסבר:

Kubctl describe secret db-user-pass

SECRETS דרך YAML:

מעבירים לENCODING של בסיס 64,

Kubectl get secret db-user-pass -o yaml

יציג לנו את הסיסמא והשם משתמש בENCODING של בסיס 64

Echo -n “root” | base64

Echo -n “password” | base64

יציג את אותו משתמש וסיסמא בENCODING של בסיס 64

דרך YAML כENV. VARIABLES. ראה ENV בSCRIPTS

לעשות APPLY לקובץ כדי להפעיל

Kubectl apply -f deploy-helloworld.yaml

בדיקה שהסיסמא והמשתמש אכן במשתני סביבה:

Kubectl exec -it <name of pod> /bin/bash

בתוך הקונטיינר:

Env

ונראה שם את SECRET\_PASSWORD= וכולי

SECRETS WITH YAML in FILES

אפשר ליצור SECRET דרך קובץ YAML ראה SCRIPT db-secrets.yaml

+ראה החלפה של הSCRIPT של ENV מהDEPLOY הקודם. תחת deploy-helloworld-with-secrets.yaml

שזה מייצר VOLUME וSECRET שנשמר בתוך הVOLUME.

DASHBOARD MINKUBE

להריץ DASHBOARD של מיניקיוב בדפדפן:

Minikube dashboard

כל הדברים שהרצנו קיימים שם: SECRET, PODS- מה רץ LIVENESS CHECK..

אם נרצה להוסיף RESOURCES – לוחצים על + גדול מצד ימין למעלה והכל נכתב ב YAML

או נרצה לראות LOGS DESCRIBE של POD דרך ה-3 נקודות ליד השם, גם EXEC יש

Config map

מה זה?

Kubectl get cm -A : מביא CONFIGMAP של כל הCLUSTER

CHARTS

הגדרה:תבנית קבועה(TEMPLATE) של אוסף של סקריפטים:שירותים-DEPLOY-PV-PVC כTEMPLATE אחד שניתן לקנפג, במקום להריץ לחוד כל אחד.

מנוהל ע"י:

HELM

שלוקח את הTEMPLATE מהCHARTS (שבקוד GO) ומייצר קובץ YAML לTEMPLATE.

# KUBERNETES CLUSTER

# **הגדרות של KUBERNETS CLUSTER**

CLUSTER משמע לפחות 3 מאסטרים- מספר אי זוגי של מאסטר. ו-2 WORKER

הסיבה בגלל שרק 2 משאירים אותך עם סכנה שאם אחד קורס לא יהיה לך חלוקת ניהול כמו שצריך.

שומר על HIGH AVAILABILTY

**COMPONENTS של קוברנטיס:**

5 עיקריים:etcd, controller manager, api server, scheduler, kubelet(on worker node only)

MASTER

-ETCD : בעל KEY PAIR , שומר CONFIGURATION DATA

-API SERVER : כל מה שמתקשר עם הכלים של הקוברנטיס KUBECTL

- CONTROLLER MANAGER: מה שאחראי לניהול כשלים והפעלה מחדש של פודים, כמו REPLICASET

- SCHEDULER: מאפשר השמה של POD חדש שנוצר (בעקבות כשל או מחיקה) לNODE קיים.

- kubelet : סוג של AGENT שרץ על כל NODE ומתקשר עם הCONTROL PLANE

WORKER

* KUBELET : AGENT שמתקשר עם הCONTROL PLANE , אחראי על אתחול עצירה וניטור הCONTAINER בתוך הPOD.
* KUBE PROXY: מאפשר CONNECTIVITY בתוך הCLUSTER , תקשור אחד עם השני.גם תומך ב LOAD BALANCER לSVC שחושף SERVICE
* CONTAINER RUNTIME : אחראי על הרצת קונטיינרים ומשיכת דמויות, זו הSOFTWARE שמריץ את הCONTAINER ים בתוך הPOD

CNI – container network interface מה שמאפשר לקונטיינרים ולפודים לתקשר בינהם.

Kube flannel:פלאג CNI פופלרי

NFS: Network FileSystem ,פרוטוקול שמאפשר לclient גישה לקבצים ברשת כמו local storage, וגם מאפשר את PERSISTENT VOLUMES

Services in k8s:

NODEPORT: נותן גישה מבחוץ אבל רק דרך PORT. עובד בפורט סטאטי, מפנה TRAFFIC לא משנה מאיזה NODE. אין דינמיות וללא LOAD BALANCING CAPABILITIES

LOADBALANCER: מספק תמיכת LOAD BALANCE חיצוני דרך CLOUD ומפנה את התנועה בהתאם. בא גם עם בדיקות HEALTH, SSL וAUTOMATIC SCALING

ClusterIP: עובד רק עבור הקלאסטר-לא יוצא החוצה. אין גישה מבחוץ. הקצאה של IP קבוע ויציב לPOD. בד"כ גם משוייך לזה DNS

Headless: ללא הקצאת IP , מקובל בשימוש בDB כדי להפוך את הגישה לחסויה יותר-שאין גישה מבחוץ(HEADLESS קיים בSTATEFUL SET)

**PERSISTENT VOLUMES**

יצירת ווליומים באמצעות SERVER של NFS(ראה טקסט התקנה) שמאפשרים חלוקת VOLUME בתוך הCLUSTER מכל מקום(LOCAL או CLOUD)

PVC- persistent volume claim מה שאחראי על שחזור ווליומים, אם אחד נופל או נכשל או נמחק, הCLAIM משחזר. אבל אם נמחוק את הCLAIM

הVOLUME יעלם

NFS SERVER – service שמאפשר MOUNT של וויליומים בצורת PERSISTENT. מורכב כservice בקוברנטיס דרך YAML ברכיבים הבאים:

RBAC: Role BasedAccessControl. מקצה תפקידי הרשאות.

Provisioner: מה שמקצה את הנתיב של הMOUNT ואיזה SERVER NFS לעבוד(לדוגמא)

SC: STORAGE CLASS ההגדרה ויצירתו.

PV נשלט על ידי הCONTROLER שפונה לSTORAGE CLASS כדי להקצות VOLUMEים. אם לא מצוין SC הוא פונה לDEFAULT, לכן יש לוודא שהSC שלך הוא אכן DEFAULT.

Volumes טובים שרוצים לחלוק שטחי אחסון בCLUSTER בין כל הפודים והקונטיינרים, מה שלא מתאפשר ברק פודים שזה סביבה סגורה.

**REPLICASET**

מה שמוודא שמספר ספציפי של POD או שכפול שלהם (REPLICA) תמיד רץ בכל זמן בקוברנטיס.

גם מחליף אוטומטית פודים שנפלו/הושמדו/נמחקו

לא אחראי על SCALING

מומלץ להפעיל DEPLOYMNET ולא להשתמש בצורה ישירה בREPLICASET, כי DEPLOYMENT נותן יותר יכולות ושימוש טוב יותר בREPLICASET.

**STATEFULSETS** (STS)

הקצאות משאבים על פי שם, למרות שהשם נשאר יש הפרדה לוגית בPODS. שומר על סדר(טכנולוגיות METALLB וINGRESS)

-מה ההבדל בין DEPLOYMENT? השם ג'נרי ואקראי.

STATEFUL – אומר שהוא שומר את הSESSION DATA, לעומת STATELESS שזה ללא. המספור מאפשר להנגיש על ידי REPLICA SET במידה ונניח מספר 0 נופל הוא הולך ל-1 וכולי.

נניח במצבים שבהם הDB קורס. עובד עם HEADLESS SERVICE.

**HPA**

Horizontal Pod Autoscaler- יתמרן משאבים בהתאם לדרישות QUOTA כולל להוריד פוד או להרים, בצורה אוטמטית. לדוגמא אם CPU USAGE נהיה עומס-הוא יוריד משאבים.

**פקודות**

Kubectl get pods -o wide : יציג לנו פירוט יותר, באיזה NODE מותקן איזה POD

Kubectl get nodes -o wide: יותר פירוט- IP , מע' הפעלה ועוד.

פריסה של APP בWORKER גם תפרס בMASTER וגם בWORKER אחרים.

Kubectl describe service hello-service.yaml

Kubectl get deploy: מציג את האפליקציות שפרשנו

Kubectl delete deploy <app name>: הסרה של APP ממצב DEPLOY- לא משחזר!

יצירת STORAGE CLASS: ראה SCRIPT

Kubectl get sc : מציג את הSTORAGE CLASS הקיים שלי

Kubectl get pv : אחרי שייצרנו PVC זה מציג מידע עליו(PERSISTENT VOLUME)

Kubectl get pvc : מציג עבור שימוש/בקשות לSTORAGE מהאפליקציה PERSISTENT VOLUME CLAIM

Kubectl delete -f <svc or pod or deploy> מוחק כאלה שיש הרבה SVC בבת אחת במקום אחד אחד

Kubectl scale replica:3 : משנים את הסקלזיציה של הרפליקות

Kubectl get rs : מביא replicaset

Kubectl get pods -n myspace : מציג פודים של namespace המצוין

Kubectl get sc- מציג storage class כשיש.

Kubectl get roles – מציג את התפקידים, חלקם של ההרשאות של RBAC

Kubectl get sts – מציג מצב STATEFULSETS

Kubectl config get-contexts : אומר מול מי אני עובד(נניח בתוך WINDOWS רואים שזה MINKIKUBE)

Kubectl config view: מציג את הSETUP של הCLUSTER

Kubectl top nodes : נותן כמו TOP מצב המעבדים והעומס בNODES >>> דורש METRICS SERVER עבור הNODES

איחוד של CONFIG של WINDOWS כדי שיהיה אפשר לעבוד מרחוק על הCLUSTER, ללא צורך להתחבר למכונות.

KUBECONFIG=~/.kube/config:~/.kube/kubeadm.conf kubectl config view –merge –flatten : פקודת איחוד של כל הCONFIG גם של הWINDOWS וגם של מה שיש לנו בCLUSTER של VM BOX

kubectl config use-context kubernetes-admin@kubernetes: לבחור קלאסטר ספציפי מאלו שמותקנים

kubectl get events -w: עושה TAIL לEVENTS

kubectl api-resources: מראה את המשאבים והאופרטורים של הCLUSTER

kubectl get all : מציג את כל הדברים שהואתקנו בCLUSTER, PVC, HPA, SVC, PODS, DEPLOY וכולי

kubectl top : צריך להתקין METRICS – מראה בריאות ומה רץ, כמו בלינוקס

kubectl get nodes -w: עושה TAIL לNODES, מעדכן שיש שינוי.

Kubectl api-resources : מציג את כל המשאבים עם התפקידים והתקציר שלהם בקלאסטר.

kubectl scale deployments/my-deployment --replicas=4 : מוסיף או מחסיר REPLICA כמה שנרצה -אפשר גם לPOD או כל דבר

kubectl delete pod <pod name> --force --grace-period=0

kubectl get all -n argocd: נותן את כל הRESOURCES תחת ה NS הזה

קיצורי דרך:

in your .bashrc file:

# Set an alias for the production context

alias kc='kubectl config use-context $1'

in cli run:

kc <namespace name>

TROUBLE SHOOTINg stuff in Kubernetes:

Kubectl describe pod <name of pod>

Kubectl logs pod <pod name>

Systemctl status kubelet: מראה גם על שגיאות שיכולות להיות קשורות בפריסת קונטיינר

Journalctl -u kubelete

Kubectl get clusterroles : מציג את כל ההשמות תפקידים בCLUSTER

Kubectl logs -n <namespace> <name of pod> - מציג לוג של הפוד ביחס לNAMESPACE שלו

ALWAYS השמד פודים או כל דבר אחר שנתקע. לעיתים זה רק בעיות משאבים או סתירות אחרות.

Kubectl describe nodes : מציג מצב בריאות כללי של כל הקומפננטות של בNODES

Kubectl get events : מציג LOG של הצלחות כשלונות/פעולות שנעשו בDASHBOARD

Kubectl get pods -n kube-system: מציג את רשימת הרכיבים של הקוברננטיס

kubectl logs -n kube-system <name of ku8s component from list above>

לבדוק אם לא מצליח לעשות BIND או ליצור PV, לוודא שהSC אכן נחשב כDEFAULT שצריך לפנות אליו כדי ליצר ולהשתמש בו.אם לא:

kubectl patch storageclass nfs-client -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

הפקודה לעיל מסדרת שהSC הספציפי NFS-CLIENT יהיה הDEFAULT.

kubectl get nodes -v=10: מציג פירוט אדיר – מצוין לINSTALLATIONS VERBOUSE scaling

journalctl -u kubelet -o cat : מציג בצורה מסודרת יותר את הלוג

curl -vv <ip address or domain> : מציג פירוט נרחב של השגיאות אם יש

# טכנולוגיות תומכות קוברנטיס

**Flannel**

Required to allow communication between pods and nodes in the cluster.

**Calico** = flannelאבל עם יתרונות חסרונות אחרים

**MetalLB( מטאל LB)**

טוב למצבים ללא CLOUD. מנתב בקשות IP של ARP(address resolution control) ומפרסם את כתובת הMAC של אחד הWORKER NODE.

TRAFFIC מנותב, והקוברנטיס מנתב אותו לPOD הנכון.

עובד ב2LAYER (של מודל הOSI)

בעל 2 COMPONENTS:

controller pod-

speaker pod-

בקצרה – מנפיק כתובת -עובד בשכבה 4.

. אפשר גם ליצור LB סרביס לכל אפליקציה שתאפשר כניסה מבחוץ. הMETALLB ינפיק לנו כתובת IP -שונה- לכל LB שנקבע, מתוך הPOOL שהגדרנו לו בCONFIG-MAP

**INGRESS**

מאפשר כניסה מהOUTSIDE WORLD, עובד בשכבה 7, ברמת האפליקציה. ניתן להוסיף אותו כRULE לתוך SPECS ולהגדיר גם DNS ZONE שיעבוד לפי הINGRESS.

דרוש IP חיצוני שMETALLB מנפיק כדי לתקשר. .

על כל אפליקציה נריץ INGRESS שיתפקד לפי הIP שMETALLB הנפיק לנו

**METRICS SERVER**

נותן סטטיסטיקות במיוחד דרך:

kubectl top nodes

**HAPROXY**

loadBalancer that supplies external IP for the whole cluster. And can also provide stats using browser access.

OpenEBS, GlusterFS

Thin provisioning עם אופציות נרחבות – SNAPSHOTS, DATA REPLICATION, הגנות, סקליזציה. FILE BASED STORAGE(GLUSTERFS(

CHARTS: מאפשר פרישה של כמה DEPLOYMENTS בבת אחת בקובץ אחד שינוהל על ידי HELM

CRD

Costume resource definition

**ISTIO**

מספק ניטור ושליטה במיקרוסרבסים באמצעות ENVOY או SIDECAR שמנפיקים TLS שאיתם האפליקציות יכולולת לדבר עם הפודים ללא שינויי באפליקציה עצמה.

יש לו 3 רכיבי(שבד"כ פורשים בHELM) BASE, ISTIOD, GATEWAY.

הISTIO משתמש בINJECTOR שנותן את הניטור, ובעזרתו מייצר את הSIDECAR שזה בעצם קונטיינר נלווה לכל SVC שנבחר.

עושים זאת על ידי שינויי בDEPLOYMENT עם LABEL על פי NAMESPACE. כלומרISTIO יזריק SIDECAR לכל המשאבים של הNS שעשינו לו LABEL.

וכך שנראה פריסה של פודים נראה כפילויות כאילו יש לו 2 REPLICA, אבל זה בעצם התוספת של הSIDECAR.

**KIALI**

בשיתוף עם METRICS מספק המון מידע בתצורת DASHBOARD על הדפדפן. על בריאות הCLUSTER, התנועה, מצב כל הCOMPONENTS של הקלאסטר ועוד הרבה.

CONFIGMAP

מאפשר הפרדת הקונפיגורציות של האפליקציות שבתוך קוברנטיס, ולזמינות שלהן בכל הקלאסטר.

ההגדרות של האפליקציות שמורות תחת CONFIGMAP ואפשר לשנות אותן בכל עת, מבלי להכנס לאפליקציה עצמה, וזה מכל מקום בקלאסטר

ROLLING DEPLOYMENT

BLUE GREEN STRATEGY

מצב שבו יש לנו 2 ישויות של פיתוח/הפקה/הפצה שלמה אחת שהיא ONLINE בחוץ והכחולה נמצאת בשלבי פיתוח תמידיים, וכל פעם מחליפים ביניהם

יתרונות:

אין DOWNTIME, TEST WHILE PROD, FAST UNDO, RELEASE צפוי

חסרונות: יקר פי 2, DATABASE דורש המון עבודה ותזמון, זמן פיתוח

CANARY STRATEGY

מצב שבו חושפים עדכון, ומרימים יישות חדשה למעט משתמשים בקנה מידה קטנה יחסית.

בד"כ ידני.

**שלבי הרמת אפליקציה:**

לוקחים IMAGE

מרימים DEPLOYMENT על פי הפורט והIMAGE

מרימים SERVICE שמייצג את הDEPLOY, כדי שתהיה גישה לאפליקציה

ואז INGRESS שמקשר בין הSERVICE ל LOAD BALANCER-

מוסיפים בHOSTS את הIP של הBALANCER. עם שם האתר/אפליקציה שקבענו. כדי שיהיה אפשר לפנות מהדפדפן.

הקוברנטיס יכול להפנות לאותה כתובת ואותו פורט, ובזכות הBALANCER -INGRESS הוא מנתב לשם שהגדרנו בSERVICE וב INGRESS.

אפילו אם יש לנו כמה אפליקציות על אותו הIP ואותו PORT

כל PORTFORWARD שעושים זה הLOCALHOST 127.0.0.1, שמדבר עם האפליקציה. לדוגמא אם האפליקציה עובדת ב20001, אז:

Kubectl port-forward svc/kiali 80:20001

80 מייצג את הפורט שאליו ננתב את האפליקציה כדי לראות אותה בדפדפן/LOCALHOST

# **HELM**

מאפשר בניית YAML בבת אחת של הכל – DEPLOY, SERVICE, או כל RESOURCE אחר, באמצעות TEMPLATEים שכתובים בGO.

סוג של TERRAFORM עבור קוברנטיס

לכן אי אפשר לעשות APPLY לצ'ארט של HELM.

Helm create <name of folder/app>: מייצר TEMPLATES ג'נרים של צ'ארט. ללא כל ערכים. זו התחלה של בניית צ'ארט לאפליקציה מוכנה

Helm repo list : מציג רשימת REPO של HELM

helm pull bitnami/redis –untar: מושך ופותח REPO לתוך HELM .

helm repo update: מעדכן אחרי שמשכנו(בהנחה שמשמכנו)

Helm ls : מראה רשימת אפליקציות שיוצרו בHELM

helm install my-redis -f my-values.yaml bitnami/redis : מתקין אפליקציה בשם my-redis עם הערכים בקובץ my-values.yaml מתוף הREPO שמצוין בסוף

Helm lint <yaml deployment FOLDER> : עושה בדיקת SYNTAX

Helm template <yaml>: עושה בדיקת TEMPLATE-מראה אם המשאבים שאמורים להפרש -אכן נפרסים.

helm upgrade –install <app name> -f <folder path>/app-values.yaml -n <namespace> <path of deployment>

Example:

helm upgrade –install my-app -f application-demo/app-values.yaml -n staging application-demo/.

אני חושב שהחלק האחרון- זה שם הREPO לא PATH OF DEPLOYMENT

<< הפקודה הזאת עובדת טוב באוטמציה, לכן לא עושים INSTALL, כי אפשר תמיד להריץ אותה.

kubectl describe ingress my-app-application-demo -n staging : מציג את הINGRESS של האפליקציה ולא את הפוד(?)

helm ls : מראה התקנות של HELM

**TROUBLESHOOTING HELM/K8s**

Problem of getting connection refused from pulling images from docker, including busybox.

Attempts to resolve DNS:

Changing the /etc/NetworkManager/NetworkManager.conf

[main] section:

dns=none

rc-manager=unmanaged

which will not update /etc/resolv.conf and let me manually resolve dns’s

# Generated by NetworkManager

#nameserver 2a06:c701:9ee4:d00::1

nameserver 2001:4860:4860::8888

nameserver 2001:4860:4860::8844

kubectl get events --sort-by .metadata.creationTimestamp

perhaps a containerd problem:

crictl pull busybox

showed that it expects docker.

I’ve changed the end points socket via:

export CRIO\_ENDPOINT=unix:///run/containerd/containerd.sock

and also on the

/etc/systemd/system/multi-user.target.wants/kubelet.service

Add the --container-runtime=containerd flag to the ExecStart line. The line might look something like this:

javascript

ExecStart=/usr/bin/kubelet --container-runtime=containerd

WE see that the endpoint for crictl images or crictl ps -a is always dockershim.sock, so we change the endpoint of crictl:

export CONTAINER\_RUNTIME\_ENDPOINT=unix:///run/containerd/containerd.sock

export IMAGE\_SERVICE\_ENDPOINT=unix:///run/containerd/containerd.sock

more problems:

client for DHCPv6 was down, so it can’t confirm address via ipv6 protocols.

--probably not the problem

Unable to pull images via k8s

Docker hub prevents pulling more than 100 images per six hours.

Check if docker hub pull limits exceeded:

TOKEN=$(curl "https://auth.docker.io/token?service=registry.docker.io&scope=repository:ratelimitpreview/test:pull" | jq -r .token)

curl --head -H "Authorization: Bearer $TOKEN" <https://registry-1.docker.io/v2/ratelimitpreview/test/manifests/latest>

**solution**

The coreDNS were deployed on worker nodes- which didn’t have proper resolv.conf ip’s

After updating them to 8.8.8.8 and 8.8.4.4 they appear to be working, after restarting all problematic pods

To make networkmanager NOT reset these settings of resolv.conf:

Vi /etc/NetworkManager/NetworkManager.conf

Add to [main]

Dns=none

*How to know cluster is talking with services/apps from the inside?*

Kubectl run -it busybox –image=busybox – bin/sh

And then inside

Wget <name of the service> :<port number>

Check the file downloaded

TROUBLESHOOTING ISTIO-KIALI

Curl -v localhost or curl -v <ingress ext.ip>

DELETING RESOURCES WITHOUT NAMESPACING

kubectl delete all -n default --selector=app.kubernetes.io/part-of=argocd

### **Scenario-How Ingress/Ingress-controller deploys a public app into the LB**

Suppose you have a Kubernetes cluster, and you've deployed two web applications: app1 and app2. You want to make these applications accessible from the internet. app1 should be accessible at http://www.example.com/app1 and app2 at http://www.example.com/app2.

### Kubernetes Setup

1. **Deploy Applications**: You deploy app1 and app2 in your cluster, each running in its own set of pods.
2. **Create Services**: You create a Kubernetes Service for each application. These Services are of type ClusterIP (default type), which means they are only reachable within the cluster.

### Introducing Ingress

1. **Deploy Ingress Controller**: You deploy an Ingress Controller in your cluster (e.g., NGINX Ingress Controller). The Ingress Controller will listen for incoming HTTP requests.
2. **Create an Ingress Resource**: You define an Ingress resource with rules for how traffic should be routed to your services.

yaml

1. apiVersion: networking.k8s.io/v1
2. kind: Ingress
3. metadata:
4. name: example-ingress
5. spec:
6. rules:
7. - host: www.example.com
8. http:
9. paths:
10. - path: /app1
11. pathType: Prefix
12. backend:
13. service:
14. name: app1-service
15. port:
16. number: 80
17. - path: /app2
18. pathType: Prefix
19. backend:
20. service:
21. name: app2-service
22. port:
23. number: 80
24. This Ingress resource tells the Ingress Controller to route traffic based on the URL path: requests to /app1 go to app1-service, and requests to /app2 go to app2-service.

### Public Exposure

1. **Load Balancer**: Your cloud provider assigns a public IP address to the Ingress Controller through a Load Balancer service. This Load Balancer is what makes the Ingress Controller accessible from the internet.
2. **DNS Configuration**: You configure a DNS record for www.example.com to point to the public IP address of the Load Balancer.

### Accessing the Applications

* When a user navigates to http://www.example.com/app1, the DNS resolves www.example.com to the Load Balancer's IP, which forwards the request to the Ingress Controller. The Ingress Controller then routes the request to app1 based on the defined Ingress rules.
* Similarly, a request to http://www.example.com/app2 is routed to app2.

### Conclusion

In this example, the Ingress Controller and its rules define how traffic to www.example.com/app1 and www.example.com/app2 is managed. The actual public exposure and IP address come from the Load Balancer provided by the cloud platform. The Ingress itself doesn’t provide the public IP but orchestrates the traffic flow based on the URL paths.

# **CLOUD**

Properties: on-demand, broad network access, rapid elasticity, measured service, resource pooling

Without these properties- they can’t be called cloud, but “cloud carriers”/ex. Nimbus

Cloud brokers aren’t cloud. Ex. Cloud zone automatic.

3 kinds of cloud like: cloud carrier, cloud consumer(us), cloud broker

service models:

**IaaS**: Infra stracture as service, allows to create machines and all the modern cloud computing. AWS, AZURE, ORACLE, GOOGLE CLOUD COMPUTING

**PaaS**: platform as service, google as ex. Allowing to run codes on their servers

**SaaS**: software as service.ex:pay per seat. Ex2:salesforce:service is closed env. Can’t be customized.

Differences:

Paas you bring the code, you pay what you run

SaaS: the provider is responsible, for the health of cloud+applications+data

IaaS: 90% is on the client

Deployment Model:

Private cloud(real one): only the client has it, VMware, private completely, costs millions. Only I can manage it.sometimes banks consume private cloud.

Public Cloud(most clouds are): azure, google, amazon, IBM, VMware. That the public can go in.

Maintaining private cloud is not just IT, mostly software and more than devops.

Community Cloud: NIMBUS, github uses this cloud.

Hybrid cloud: Azure, you have active directory, disaster recorvery. Public+ private

But very expensive.

Aws=used by devops and developer

Azure=used by IT

Simple reference:

Public= credit card number

Private= 6-10 million NIS

Elastic computing

can automatically create servers, 500 for example, without real person involved. All automatic

Regions and Availability Zones:

Regions: כמו US-west או euro-frankfurt

AZ(availability zones): Data Centers in the region. Which have DR(disaster recovery), and to allow HA for servers

***AWS(Amazon-Web-Servers)***

COMPONENTS:

IAM:identity access management כל מה שקשור להרשאות, USERS, POLICES, ROLES.

3S: STORAGE, בחירה של GP(GENERAL PURPOSE) או מהירים יותר

2EC: ELASTIC CLOUD COMPUTING, יצירת הINSTANCE להקמת מכונה. כל הוירטואליזציה.

SECURITY GROUPS: סוג של FIREWALL

VPC: virtual private cloud, מה שמאפשר IP חיצוני. וגם רשתות פנימיות. מאפשר NAT(network address translation). בזכות הROUTE TABLES.

לא עולה כסף, VPN עולה . למעשה ענן פרטי שבתוך הענן הציבורי בו אני שומר את המשאבים שלי.

Elastic Block Storage(EBS)

Not only online.

Gives persistent file system for ec2

No need to be plugged to instance

Can backup while connected or disconnected from machine

Can transfer to different AZ or regions

Snapshots

Glacier: very cheap storage like bucket(0.01 per giga), like backuptapes, only for emergencies, takes 4-6 hours to use, costs a lot of money to restore

ECS: Elastic container service לא כמו קוברנטיס, יותר כמו דוקר.

ECR: Elastic Container Regitery בדומה לDOCKERHUB, REPO של IMAGE/CONTAINERS

ROUTE53: מאפשר ניתוב DOMAIN NAME מאחד קיים(PUBLIC) ולהעביר אותו לשרתי AWS(HOSTED ZONES).

LAMBDA: סקריפט שיכול לרוץ בכל פעם שנפעיל שירות של AWS, אם זה INSTANCE או אחר

CODE DEPLOY: מאפשר CONTINUOUS DEPLOYMENT ע"י עבודה אוטמטית עם GITHUB או אחר(אוטמציה של DEPLOYMENT)

CODE PIPELINE: בדומה לג'נקינס, וגם מאפשר חיבור לג'נקינס

ELASTIC BEANSTALK: הרמת סביבת עבודה בקלות ובמהירות למפתחים או QA, כאפשרות במצב של עומס עבודה עבור הרמת סביבת עבודה שמרים DEVOPS.

RDS: relational database service מאפשר מגוון סוגים של DB בשילוב עם שירות אמאזון.

LOAD BALANCER

סוגים:

ALB: APPLICATION LB. עובד רק HTTP, לא מתאים לSTREAMING, רק WEB REQUESTS.

NLB: NETWORK LOAD BALANCER, עובד גם על TCP UDP ןTLS (TCP מאובטח) . HIGH PERFORMANCE, מתאים לSTREAMING.

GWLB: GATEWAY LB, ?חדש

CLASSIC LB: גרסא ישנה של הLB, כולל בתוכו גם HTTP וגם TCP, אמזון פחות מעדיפה שנתשמש.

**IAM**

יצירת ROLE : IAM>ROLES>

אפשרות לייצור AGENT של SSM שמאפשר גישת ADMIN למכונה ללא SSH וללא SG. רק באמצעות הROLE

**2EC**

פתיחת LB:

עבור LISTENER פותחים 80 או 8080 TCP, אם אנחנו בNETWORK LB.

80 זה HTTP, אבל TCP 8080 זה לא.

המכונות חייבות לרשום

בדיקת LB: לוחצים על DNS NAME RECORD COPY

TARGET GROUPS

חייבים לרשום את הINSTANCES לTARGET GROUP אחרת הוא לא יפעיל עליהן LOAD BALANCER

TARGET GROUP יכול לקבל רק פרוטוקול אחד- במקרה של PORT -רק פורט יחיד. כך שצריך עוד TG כדי להביא עוד פורטים.

ELASTIC IP

נותן כתובת חיצונית(באינטרנט) לINSTANCE שלי. מגיע לי אחד חינם על כל מכונה, אם רוצים יותר אז זה עולה כסף.עד 5-ומעבר לזה צריך לבקש מהתמיכת אמאזון. זה נקרא ELASTIC כי זה פר שעה ולא לחודש. משתמשים בALLOCATE כדי לשייך לINSTANCE

PLACEMENT GROUP

אם נרצה שהמכונות ישבו קרוב אחד לשני. כלומר 2 מכונות שיושבות תחת אותו PLACEMENT GROUP מאפשר שיהיו קרובות פיזית. זו רמת פיזור ברמת CLUSTER.

CLUSTER: שלא יהיה מצב שהם עוברים ל SWITCH אחר או מעבירים מידע דרך נתב.

ואם נרצה הפוך-לפזר, נבחר :SPREAD. וגם באיזה רמה -RACK או HOST.

רמת פיזור: PARTITION, אפשר להגיד שכולם ישבו באותן מחיצות.

**כדי לייצר מכונות שפתוחות לאינטרנט** יש לעשות את השלבים הבאים:

מייצרים אינסטנס, פותחים LB שרוצים, פותחים security group שמותאם לפורט של השירות, נניח NGINX אז נוסיף INBOUND ל8080.

מקשרים בין הSECURITY GROUP לLB, פותחים גם TARGET GROUP שבו מיוצגים האינסטנסים שלנו, ואז מקשרים בLB עם הTG הרצוי.

כעת אמורים לראות IP למכונות. משם אפשר לפתוח את המכונות דרך הכתובת של הLB

פתיחת SECURITY GROUP

INBOUND RULES:

HTTP אם צריך

או TCP מה שפתחנו.

PORT לפי האפליקציה/שימוש

ALLOW – אם נרצה שכולם ישתמשו מבחוץ

OUTBOUND לא נוגעים

-חוזרים לLB

EDIT SECURITY GROUP:

Load Balancer חייב security group – אפשר לצרף אותו אח"כ לINSTANCE

יש לציין ENABLE PUBHLIC IP ביצירת הINSTANCE

BUCKET זה מייצר לך קונטיינר או מע, הפעלה שאנחנו בוחרים.

עבור כל INSTANCE מומלץ לייצר SECURITY GROUP.

**AUTOSCALING**

יצירת AMI:amazon machine image, מייצרים קודם כל רפליקה של אינסטנס שלו נרצה לעשות AUTOSCALING-שם תיוג GOLDEN

AUTOSCALING GROUP, בוחרים שם ומוסיפים גם ASG, שנדע מה זה.

מפעילים LAUNCH CONFIGURATION שזה ההגדרות של (ממליצים על AUNCH TEMPLATE מבחינת אמזון) ובוחרים את הAMI שלי, בוחרים איזה מכונה שעבדנו איתה(לא לשנות מהקודם עדיף)

משתמשים באותו SECURITY GROUP שכבר אנחנו עובדים עם האינסטנס ששכפלנו.

אותו KEY- מסיימים LAUNCH CONFIGURATION.

חוזרים לAUTOSCALING-

בוחרים LB קיים. ובוחרים את מה שהשתמשנו למכונה

מאפשרים V ELASTIC LOADBALANING HEALTH CHECKS כדי שהLB יגיד לנו שעבר הבדיקת בריאות.

בוחרים DYNAMIC SCALING (PREDICTIVE לא הצלחנו) ולפי רמת העומס נניח 60% CPU.

***VPC***

VPC is Like a local private network

Subnets are the range of ip addresses, and if private or public

We mark private ip’s with single or 2 digits, while public have 3 digits. For example: 10.0.1.0 =private, 10.0.101.0=public

This is to better differentiate when selecting subnet groups or other settings that don’t show public/private

Components:

*GATEWAY*

-IGW-internet Gateway, via ingress & egress: מאפשר אינטרנט לכל המכונות

- VPG: Virtual Private Gateway- מאפשר חיבור לטכנולוגיות פיירוול ונתבים כמו סיסקו וצ'קפויננט.

-:CGW costumer GateWay – איך הראוטר שלי מיוחצן בAWS

*VPN*

Direct-Connect: יקר. מהווה חיבור ישיר לאמזון מהמשרד. מבודד מהאינטרנט לחלוטין.

Hardware-Based VPN: גם יקר מאוד.

Region של VPC מורכב מPUBLIC subnet, וגם PRIVATE subnet

VPC SECURITY

Security groups: פיירוול, ingress and egress, stateful: מאפשר תנועה TRAFFIC בחזרה.

Access Control Groups: מורכב, כל דבר צריך להגדיר.

VPC PEERING: קישור בין שני VPC שונים

ישנה הכנה אוטמטית של VPC דרך VPC & MORE

יצירת VPC ידנית

פותחים VPC בלבד, עם נניח רשת פנימית שלנו 192.168.0.0/16 כטווח בCIDR

פותחים subnet public , עם כתובת 192.168.101.0 עבור אזור 1a וכל השאר בהתאמה (אם יש 3 אזורים)

וגם private subnet עם כתובת 192.168.1.0/24 וכל השאר בהתאמה לפי הREGION

פותחים IGW)internet gateway( , ומשייכים אותו דרך ACTIONS לVPC , במקרה זה הידני-אין לשייך את הIGW לPRIVATE SUBNETS!! רואים ייצוג של PUBLIC בSUBNETS כאשר זה לא PRIVATE.

ROUTE TABLES פותחים ומתחילים לשייך subnet ע"י EDIT ROUTE. WITH EXPLICIT ASSOCIATION>>PUBLIC שייצרנו קודם לכן.

לייצר NAT GATEWAY: בוחרים SUBNET איפה שיישב הNAT GATEWAY.. לכן זה רק אחד מה-+PUBLIC ALLOCATE ELASTIC IP

DB זה דוגמא שאנחנו **לא** רוצים שיהיה IP חיצוני, אלא רק PRIVATE

המשמעות של: NAT GATEWAY נותן לנו את הכתובות דרכו, ולא דרך הLB. כדי שמה שPRIVATE יהיה לו אפשרות להעביר כתובת חיצונית ללקוח. נניח אפליקציה בPRIVATE שתהיה זמינה ללקוח מבלי לחשוף את הכתובת הפרטית שלו לעולם.

Internet gateway הוא התשתית ו NAT הוא הספק.

איך לחבר שרתי PRIVATE( מסאבנט PRIVATE) לVPC?

יש ליצור:

Aws service> filter type = gateway > assign route table(the private)>create

Make sure the private route tables are connected to vpce

התחברות לשרתים פרטיים- לא ציבוריים!

אפשרויות:

* NAT GATEWAY: עדיף כשהלקוח צריך להכנס לPRIVATE דרך האפליקציה, ולא שום דרך אחרת.
* OPEN VPN: אפשרות שADMIN יכנס לPRIVATE, דבר שלא אפשרי דרך NAT. עולה כסף
* TUNNLING WITH BASTION: כמו OpenVPN, לא עולה כסף(כמו INSTANCE) אבל פחות מאובטח, ראה SCRIPT

**BASTION TUTORIAL**

https://aws.amazon.com/solutions/implementations/linux-bastion/

command for tunnling bastion.

-use on \*local dir\* not bastion machine console:

-make sure same key pair is assigned for BOTH the private and the bastion instance!

ssh -o ProxyCommand="ssh -W %h:%p -i ariel-key2.pem ec2-user@13.48.73.84" -i ariel-key2.pem ubuntu@10.0.2.127

what it does is jumping.

connects to the bastion, and then jumps to the private subnet/machine with the same key.(the key that is in the local client)

configuring bastion inside CloudFormation-Stack:

-after using link- make sure on correct region.

- the yaml is the S3 url, that should be automatically pasted from the link. if not- download it, and upload it.

->(next page)apply right VPC, and the right subnet publics, that are related to that vpn.

- 0.0.0.0/0 or your own IP that you connect in "allowed bastion external access CIDR)

- key pair name: just the name of the file of the key in your client machine(local). ariel-key (no file extension)

- bastion banner, TCP forwarding(to allow tunnling), X11 forwarding, all true.

-next page>perserve resources. > apply/mark the box> complete

---

after installing the instance: go to EC2

- goto linuxBastion instance(this is th ebastion instance we created now with CloudFormations),

check the security groups.

- allow in the ec2 instance to connect+update

- connect with ssh -i ariel-key.pem ec2-user@<bastion instance address>

bastion address:51.20.244.25

**OPEN VPN TUTORIAL**

ec2 instance>launch

select AMI marketplace>marketplace tab>openvpn> free tier only(o fees on desc.)> select>check pricing tabs

edit network:

public IP

enable ip public

Launch

login using ssh but user= openvpnas

in the setting :

yes>eula

yes>stand alone

listen on all network interface>1

rsa

size of cert> 2048

default>rsa, 2048, 943, 443

should client traffic be routed by default through VPN>NO! (if yes-clients can abuse traffic for illegal activity!!)

no-for dns traffic of client

access to private subnet: yes! this is why we put the vpn-to access private servers

user: openvpn

password:

>didnt use activation key> leave blank

DONE

access via the browser on the logs after fininsh installing: example:

Admin UI: https://13.48.127.63:943/admin

because it is risk via browser/shows “risky website warning”, use advanced on the browser> agree to continue

login

in the UI:

can add users for other to use

we will use client ip granted from after the installation.

after login(can be same as admin), we download openvpn access server for windows

download and install

use same user + password

should be running a window of openVPN with network bar/graph

now to connect to private servers

------------------------

make simple windows instance. create password: connect>rdp client>get password>upload pem key>copy password

for windows,

user: administrator

password

now open RDP on your local machine, enter private IP as the host.

----

to connect to mysql checkout MYSQL init

**RDS-relational Database Service**

RDS זה DB לכל דבר שמנפיקים מאמזון ועולה כסף. הם מנהלים את הכל ודואגים לעמידות שלו.

עדיף להתחבר לRDS דרך OPENVPN מאשר BASTION

יצירה: COSTUME>free tier>save password/db name/user> NO ec2 connection> connectivity:choose VPC>

Before creating DB, create subnet group(under subnet group)>name+desc>all AZ+ip of privates>fiunish

>go back to RDS creation>refresh connectivity>create DB

View credentials for db password/user

\*\*\*BEFORE all that must install OPEN VPN.must be running on machine while connecting to private server \*\*\*

save password, amster username, db name:

ariel-db

admin

qwerty1234

use mysql workbench(windows) to connect via UI.

click + for new connection setup:

type any name

paste host from amazon endpoint

paste user+password>>store in vault

open connection-use mysql

**LAMBDA**

Serveless application, like a function. That can manage scalability, use event cases, integration with aws services and have its own script

In javascript or python

Pricing is per usage of the function-which is milliseconds, but if lots of requests can be more costy.

Layers: adding virtual env.

Running/uploading your own code: it includes the dependencies, zipping it all and uploading to S3/upload from computer.

Then lambda can use it.

+add trigger: can use aws services as triggers to execute the lambda function

Test: running it shows the output or log of the function

Example1:

לידור סיפר שעבד בחברת מדפסות של תרשימי בניינים. שרצו להמיר את המדפסות למדפסות חכמות-שניתן להדפיס בכל מקום.דרך CLOUD

LAMBDA סיפקה פונקציה שמנטרת אם יש עומס מתח על המדפסת אז שתעצור את הפעולה שלה כדי שלא תהרוס את התוף.

Example2:

אם נריץ מכונה שעולה כסף, נראה רק אחרי 7 שעות בברירת מחדל שלך באמזון את הדו"ח.

אז אפש/ר לבנות LAMBDA שתתן BILLING מיידי, נניח אחרי כל שעה. או כמה זה עלה עד עכשיו.

Example 3:

כיבויי מכונות בערב או בסופ"ש.

**BEANSTALK**

אפשרות פשוטה למפתחים להריץ סביבות עבודה ושפות מבלי ידע מקדים בתשתיות. יושב בין PaaS ל IaaS . נניח שDEVOPS מאוד עסוקים,

אבל לQA אין סביבת עבודה בנתיים והם מבזבזים זמן בשיחות מסדרון>>BEANSTALK נותן את המענה הזה.

עובד בצורה שוטפת עם שאר השירותים של אמזון.

בדומה לNETLIFY

Application:

tutorial

can get Domain name on its own or one that we have

platform is automatically installed

high availability-depends on your resources to allow for HA

can use DB to connect via VPC-enable a database-it creates a NEW one.erasing beanstalk will erase the DB

completely run via amazon and everything we do to the servers/instances is via beanstalk.

Environment:

ניתן לשייך סביבות לתוך האפליקציה. כשמייצרים אפליקציה גם מייצרים סביבה(ENV). אז לכן אם כבר יש סביבה עדיף לשייך אפליקציות נוספות לאותה סביבה. או שאפשר לעשות כמה סביבות שאותה אפליקציה עובדת על כל אחת מהסביבות.

***ROUTE 53***

מאפשר ניתוב, כמו שרת DNS, כל פעם שניגש לDOMAIN השירות הזה ינתב לאן שנרצה.

PUBLIC: אומר שהדומיין צריך להיות כבר קיים.

PRIVATE: דומיין שאנחנו בונים

Tutorial

מייצרים HOSTED ZONES

RECORD: זה לדוגמא WWW, או A RECORD

נבחר LB. שיפנה רק מהDOMAIN שהגדרנו ב ROUTE 53.

ALIAS: שירותי אמאזון שיכולים להפנות אליהם. לדוגמא למכונה, או לDB או כל דבר של אמאזון.

עולה כסף אבל מעט.

לוקליזציה:ניתן להפנות גם מאיזורים שונים בעולם- כדי לייצר לוקליזציה לאפליקציות ולPROXY ללקוחות רחוקים.

או אנטי לוקליזציה-שנתאים שפות לפי האזורים שמהם מגיע הלקוח/מדינה. נפנה אותו לRECORD של אתר אחר.

**CDN – content delivery network/ CloudFront**

מאפשר CACHE לאזורים מרוחקים, במקרה של בקשות מרוחקות, השירות יוריד את התכנים קרוב יותר ללקוח, ואז יהיה מהיר ונגיש יותר בעתיד.

INVALIDATION- מאפשר REFRSH לשרתים/אתרים שעוברים עדכון, ויעדכן את הCACHE במקרה הצורך.

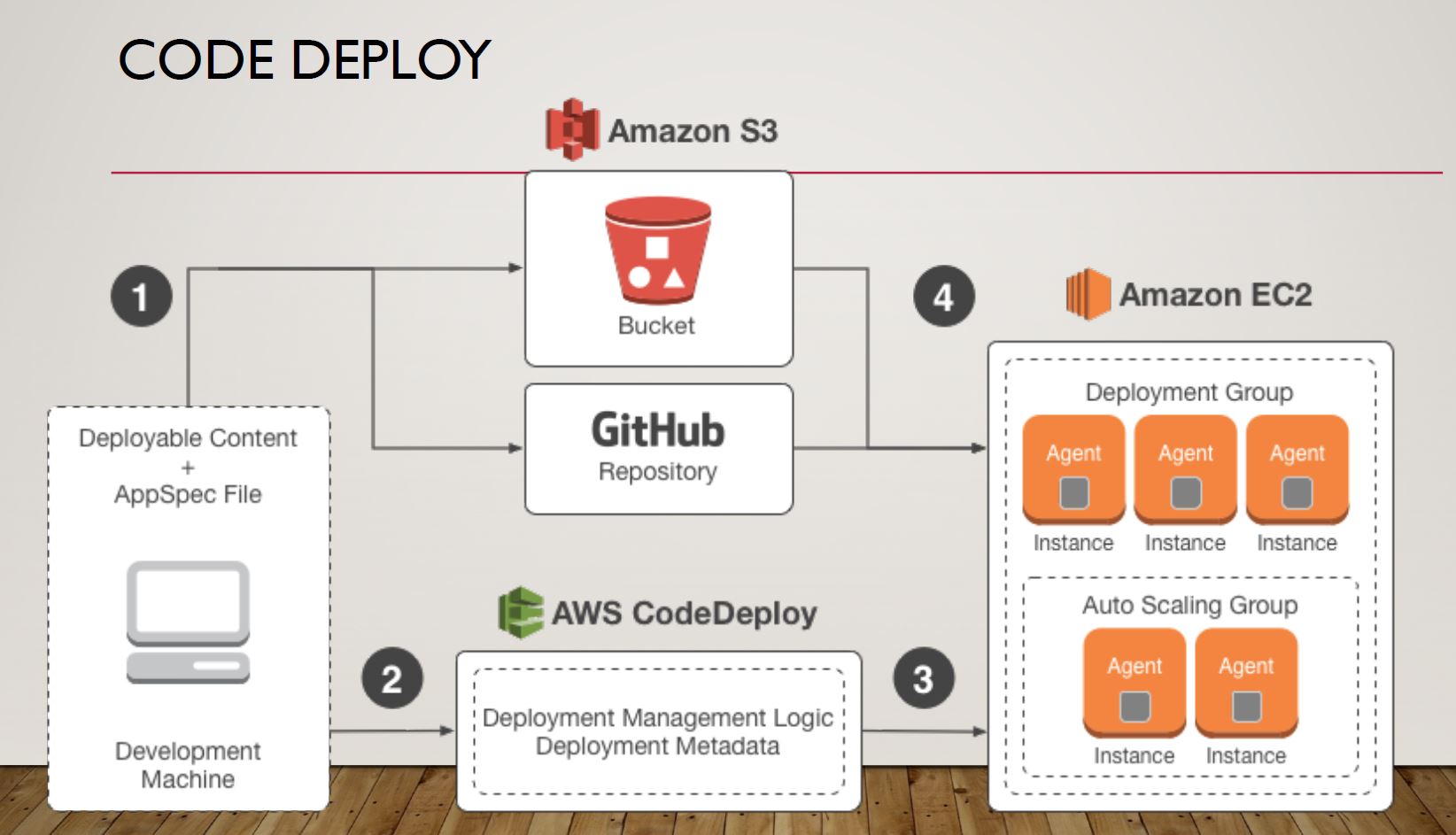
עובד בד"כ עם נתוני אחסון-תמונות סרטונים. לדוגמא טלגרם או וואצפ.

CLOUDFLARE, שירותי CDN לא של אמאזון אבל אמאזון יודעת לעבוד איתם ולהיפך.

AKAMAI אותו דבר.

ואז צריך לדעת לפצל את התשלומים שלי.

**CODE DEPLOY**

בקצרה מאפשר אוטומציה מוחלטת לCD.

( לעומת continuos delivery שבה יש MANUAL בשלב שלפני הPRODUCTION. שעוברים על הכל ידנית.)

CD elements-

Unti test>platform test>deliver to staging > application acceptance tests> deploy to production >post deploy

Code deploy creates agent on an instance that will allow interaction with other services. And can update the app inside too.

הAGENT שמשתילים בINSTANCE מתקשר עם APPSPEC ומנתב לאוטומציה עם הCODE DEPLOY

Tutorial

Todo:

Create CodeDeploy application.

create roles(EC2+codeDeploy).

Create agent in EC2+attach roles.

Create deployment group

1. ייצורfor IAM ROLE של 2EC:
   1. >Aws service>ec2>next>codedploy>aws tool for code deploy>name>finish
2. ייצור ROLE עבור ה CODE DEPLOY
   1. IAM>Role>codedeploy>next…>next>name>finish
3. מייצרים CODE DEPLOY
   1. >code deploy>application> create app>ec2/on premise>create app
4. Building instance for code deploy
   1. 4 instances named web-ariel->aws **linux 2**> +tag : ariel-deply, value:true
   2. Vpc on public subnet+enable security group+ ssh+http
   3. Advanced details>Select IAM role of ec2 we create on phase 1!
   4. Install agent -one time run, advanced details>user data
      1. Via guide on internet : install codedeploy agent on user data:take only formal script of aws
      2. יש לשנות REGION שיותאם לשלנו בתוך הסקריפט-ראה סקריפט.
      3. בודקים בSSH שהמכונה אכן מריצה CODEDEPLOY: sudo systemctl status codedeploy-agent.service
5. מייצרים DEPLOYMENT GROUP בתוך CODE DEPLOY שיצרנו,
   * 1. Name>role of code deploy(NOT ec2!)>in place>EC2 instance>tag: ariel-deploy:true(what we put in tag)
     2. מוודאים שיש את המכונות של הINSTANCE שיצרנו מקודם כאופציה בתוך הDEPLOYMENT GROUP
     3. Install AWS agent>NEVER(because we installed via that script)>all at once> X no load balancer>create
   1. אפשר לבדוק בתוך הDEPLOYMENT GROUP, אם הTAG אכן נמצא כENVIRONMENT CONFIGURATION
6. נפרוש קוד לאפליקציה דרך create deployment(in the application)-GITHUB
   1. קובץ שחייב להיות בשם ובפורמט:appspec.yml שאומר איך CODEDEPLOY יפרוס את הקוד בשרתים
   2. Create deployment>stored in GitHub>token:give name(ariel-github-token)>connect to github>account+repo name(meditator3/codedeploy)
      1. Last commit ID, get through github>under Sync Fork, click on clock icon and commits>get last commit ID
      2. Paste into AWS commit ID>description>nothing else>create deploy
      3. Check your app is deployed by copying public ip to http(not https!)

Code Deploy משתמש בBUCKET של 3S כדי להעביר או למשוך גרסאות ישנות/חדשות או ארטיפקטים. באופן כללי לCI CD

## **CODE-PIPELINE**

מפעיל אוטומטית על ידי טריגר של COMMIT את הDEPLOYMENT.

מחבר את כל הPRE REQUISITS, ובסוף מנקה את כל השלבים.

עולה מעט כסף יחסית- לכל PIPELINE 1$ לחודש.

הכנת PIPELINE

V2>new role>allow awes to create role>NEXT

Source:github v2>connecting to github-allow everything to give aws access to all commits!>repo name we want to allow>

Trigger>push(whenever we want)>branch name(master)>default>NEXT

No build>SKIP>deploy:app deployment name+deployment group>finish

ועכשיו ברגע שנעשה נניח שינוי באפליקציה, נניח בריפו של meditator3/codedeploy נשנה לצבע רקע אחר, זה **אוטומטית** יריץ את הפייפליין

ECR**: elastic container registery**

סוג של DOCKERHUB, יש REPOSITORIES. אבל משלב עם SERIVCES של AWS.

זה לא קוברנטיס, רק CLUSTER של DOCKER.

**ECS- elastic container service**

בניית CLUSTER: name>EC2>linux2/win>machine size>scale>no need for ssh because everything is run via amazon>root volume(just root!)>VPC>all private subnets, because LB can lead you to them. Can’t be public for production level>sec.grp: allow from anywhere>finish

בדיקה: We see autoscaling group of ECS. We see instances run by ECS as cluster

*הרמת אפליקציה בCLUSTER*

בניית NEW TASK:

Task definition: זה כמו DOCKER COMPOSE או DOCKER RUN בתוך שירות של ECS

Create task definition for the app:

Bridge means-the containers can talk to themselves via their names

Host: they can only talk to themselves without outside link

None: can’t talk with anyone

Awsvpc: can talk with subnets of the vpc

Task definitions>new>EC2>change from awsvpc🡪bridge>task role:none

Container -1> name + image name+version

IMAGE URI is the image name from the pull request(not URL)

Deploy/service

Name of cluster>tasks, create new task> launch type:EC2> service/task, which will resuscitate if down>

Task placement>one task per host

יצירת security group: עם INBOUND שיקשיב לPORT של האפליקציה, מעבר לSG הרגיל

יצירת target group עבור ה NLB(network load balancer)

יצירת LB: NLB, לבחור את כל הPUBLIC כי מתחברים מבחוץ לCLUSTER.

בדיקה: לקחת מתוך ה LB את הDNS שלוף להוסיף את מספר הפורט בדפדפן, ולבדוק שזה רץ.

הרמת אפליקציית WORDPRESS + MYSQL DB של RDS

אותו תהליך עם הtask definition לוקחים IMAGE מתוך DOCKER HUB.

ייצור קבוצת SUBNET:RDS>subnet groups>create> choose only private

מרימים DB של אמזון:50gb storage RDS>standard create>mysql>free tier>name:wordpress-db>engine version: (according to yaml of image)5.7>

>disable autoscaling for storage>

התחברות לDB:

אפשרות 1:

להתחבר דרך MYSQL WORKBENCH ועם OPENVPN. יש להעתיק ENDPOINT מתוך הפרטים של הDB-ל MYSQL WORKBENCH

מחיקת INSTANCE ששייך לCLUSTER שלו: אפשר להוריד את ASG ל0, או למחוק דרך CLOUD FORMATION

אפשרות 2:

להרים INSTANCE רגיל של אובונטו, עם כתובת חיצונית.

לוודא שאנחנו גם בSECURITY GROUP שעבדנו עם הRDS, אחרת אנחנו לא באותו VPC כביכול. וגם שיש חיבור לSSH

ככה מוודאים שהINSTANCE מתקשר עם הRDS (כי הם אמורים להיות באותו SEC של אותוהVPC): telnet <endpoint of mysql RDS> 3306

sudo apt search mysql

sudo apt install mysql-client-8.0

mysql -h [dblidor.cxwjnmc8j0ya.eu-central-1.rds.amazonaws.com](http://dblidor.cxwjnmc8j0ya.eu-central-1.rds.amazonaws.com) -u admin -p

create database wp2; כדוגמא לזה שזה עובד

CREATE USER 'wordpress\_user'@'%' IDENTIFIED BY 'a1a1a1'; האחוז מאפשר להתחבר מתוך המסד נתונים עצמו, אחרת לא ניתן כלל להתחבר

לבדיקה: SELECT \* FROM mysql.user WHERE user = 'ariel' AND host = '%';

GRANT ALL PRIVILEGES ON wp2.\* TO 'wordpress\_user'@'%' IDENTIFIED BY 'a1a1a1';

FLUSH PRIVILEGES;

ייצור הWORDPRESS:

Task definition: wordpress>image:wordpress>80:80>add ENV variables, according to

<https://hub.docker.com/_/wordpress>

using the word press ENV. (wrodpress user, wordpress DB etc)

create Volume:

השתמש ה SOURCE כשם שנבחר בYAML של הIMAGE שבDOCKERHUB, ואז MOUNT POINTS העתק את הPATH מתוך הYAML כ בCONTAINER PATH

ייצור SERVICE:

רגיל, לשים TARGET GROUP מתאים עם הפורט המתאים, ולבדוק שאכן המכונות בHEALTHY תחת הTG.

חסרון גדול לVOLUME ב ECS: הVOLUME נעלם אם המכונה מושמדת. זה לא כמו קוברנטיס שזה נשמר בצד.

TIPS

Migration of instance between regions:

You can copy instances to OTHER regions: EC2>create AMI from instance>AMI> right click+name of instance>copy>destination

On new region>AMI> launch instance from AMI

No public IP?

Attach elastic ip to an instance

בעיות בAWS CLI

להכניס תחת ~/.aws/config:

[default]

region = Global

output = json

[profile aws4]

region = eu-west-2

output = json

aws\_access\_key\_id = AKIAU77SCMU3KD52NRNP

aws\_secret\_access\_key = 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

כל פעם שנשנה את הCONFIGURE -סיסמה וSECRET, נעבור בהתאם למשתמש הCORRESPONDING(ש-לו משוייכת המפתח+סוד)

אם לא נכנס השם לפרופיל:

$env:AWS\_PROFILE=”aws4” עבור POWERSHELL

## EKS- Elastic Kubernetes Service

קוברנטיס כשירות של אמזון.

משמש כ PaaS יותר מאשר IaaS כי אמאזון מנהלת את הMASTER NODES, וגם אי אפשר להתקין כל דבר כי המערכת נעולה יחסית, לדוגמא אי אפשר להתקין openEBS

יצירת NODE: compute>node group>add node group>IAM ROLE

Create IAM role>EC2!

אחרי היצירה יש להצמיד עוד POLICIES

CNI, amazon work NODE,EC2 containerRegistery read only, EBSSCSI policy

עבודה עם AWS CLI

לוודא שאני זה אני: aws sts get-caller-identity

Troubleshooting : למחוק ACCESS ישן ולעשות חדש.

aws eks --region eu-west-2 update-kubeconfig --name ariel-eks-cluster

^^ כדי להחיל את הריגן העדכני בהקשר של הקלאסטר הזה

Kubectl config get-contexts כדי לבדוק אם אנחנו בקלאסטר הנכון

Kubectl config use-context <name of cluster> כדי לבחור עבודה מול הקלאסטר הספציפי

AWS gp2 sc problem:

Doesn’t provision volume, and thus pods won’t be created.

For that we need EBS storage class. That will provision volumes, it is a newer SC implanted in the last few years(2022-2023?)

See below.

\*\* must enable addons :CSI Snapshot Controler + Amazon EBS CSI Driver – to allow volume managments \*\*\*

Kubectl get sc – לא מראה SC, אבל כן מראה פודים חדשים של התוספים האלה ^^

See how-to EBS-CSI script/guide

EKS-volume-Az-Problem + svc + volume-conflict-problem.txt

Kubectl get sc : יראה לנו gp2 שזה אחסון שאמזון מאפשר ומקצה לנו כל פעם.

הבעיה באמאזון היא בהקצאת VOLUMES בAZ אחרים.

הVOLUME לא יעבור לAZ אחר אם יתרומם פוד חדש בAZ אחר. ואז כל הDATA ימחק

פתרון 1: מוסיפים NODEGROUP

Kubectl get nodes -o wide –show-labels : מציג את כל הלייבלים שיש. אפשר גם פקודה יותר ברורה שמראה את הלייבלים: kubectl get nodes -o wide --show-labels | awk 'NR>1{gsub(/,/, "\n\t")}1’

שם נראה איזה AZ יש לכל NODE, נניח 1a או 2b, עבור EU WEST וכולי.

Kubectl label node <name of node> zone=<wanted Availability Zone, for example=1a>: שזה ניסיון לפתרון לשמור על AZ אחד כדי שיקום הVOLUME באותו AZ שלא ימחק לנו שעוברים AZ אחר.

אבל הפתרון לא טוב, כי אם נופל הDATA CENTER של הAZ הזה, הלך כל הCLUSTER. כי עבדנו רק עם AZ אחד.

**הוספת ADMIN ACCESS ל EKS**

to add admin user to EKS(in case more devops need to work on it)

check who's authorized:

kubectl -n kube-system get configmap aws-auth -o yaml

should display like this, and rolearn is the user/s that are registered as admin.

apiVersion: v1

data:

mapRoles: |

- groups:

- system:bootstrappers

- system:nodes

rolearn: arn:aws:iam::343568180534:role/ariel-EKS-nodeGroup

username: system:node:{{EC2PrivateDNSName}}

kind: ConfigMap

metadata:

creationTimestamp: "2023-11-28T09:12:59Z"

name: aws-auth

namespace: kube-system

resourceVersion: "1128575"

uid: 462f5626-136f-4e8e-92f4-b8400b935fa8

to edit the configmap :

kubectl -n kube-system edit configmap aws-auth

<will open a notepad edit, can copy paste to VS code>

to get the user ARN:

go to users>ARN(copy)

and add under mapRoles: | <<<<<<<<<<<<<DANGEROUS CAN DESTROY CLUSTER!!!!!

mapRoles: |

- groups:

:

:

:

mapUsers: |

- userarn: <copy here the ARN of the added user>

username: <username of added>

groups: <<<<<<<<<<<<<<<<<<THIS IS MOST CRUCIAL, because it gives the role of admin! >>>>

- system: masters

\*\*now copy back to the notepad if worked on VS code

no need to reboot anything, it should be applied immedietly.

**EBS – Volumes persistence**

ebs storage class is newer storage class that creates new volumes

the default gp2 doesn't create new volumes automatically by amazon

git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git

cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/

can be used for gp3(newer HD):

echo "parameters:

type: gp3" >>manifests/storageclass.yaml

kubectl apply -f /manifests/storageclass.yaml

test:

kubectl get sc

kubectl get pods

kubectl get pvc

the pvc should be BOUND

another command ??

kubectl patch storageclass ebs-sc -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

SEE: scripts for deploy and PV/EKS DEMO

**The Persistance Volume Bug/Problem**

the problem is that volumes on different AZ of the pod will not revive. because there's no volume on that AZ.

Possible solutions:

* openEBS: having access to disk from anywhere(since openEBS gives you more storage on the SCSi level)
* back up of volume/disk over another AZ(availability zone) and raise a snapshot or AMI of the most recent.
* Newer sort of volume (EFS?) which operates on any AZ.

As long as you have the pvc(pv claim) you can revive/rescue the volume with the new data.

**EKS- AutoScaler**

Serves in general for scaling NODES number/availability, rather than the HPA(horizontal Pod Autoscaler), which serves pods inside the cluster.

Installation:

AUTO-DISCOVERY INSTALL OF ASG:

# Method 1 - Using Autodiscovery

$ helm install my-release autoscaler/cluster-autoscaler \

--set 'autoDiscovery.clusterName'=<CLUSTER NAME>

new values of helm:

cat ariel-values

awsRegion: eu-west-2

autoDiscovery:

tags:

- k8s.io/cluster-autoscaler/enabled

- k8s.io/cluster-autoscaler/ariel-eks-cluster

extraArgs:

skip-nodes-with-local-storage: false

balance-similar-node-groups: false

skip-nodes-with-system-pods: false

--

helm install my-release2 autoscaler/cluster-autoscaler \

--set autoDiscovery.clusterName=ariel-eks-cluster \

--set awsRegion=eu-west-2

**EKS-CTL**

A CLI TOOL of EKS service, closer to kubectl. Working on the cluster with eksctl sometimes, instead of kubectl, to apply various services.

Warning: Eks-ctl does not! Work with the portal of AWS, meaning any changes of EKS portal services will be OVERWRITTEN by eksctl, later on.

So only work over EKSCTL and not parallel to eks portal/aws dashboard.

Installation:

aws cli installed and verified. check access keys or new, and delete old ones if troubleshoot.

download eksctl and add to ENV PATH.

install this yaml called config.yaml

via this command:

> eksctl create cluster -f config.yaml

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

name: cluster-in-existing-vpc

region: eu-west-2

vpc:

subnets:

private:

eu-west-2a: { id: subnet-03fc0359221d6359e }

eu-west-2b: { id: subnet-07f8f2e18d600b9f8 }

eu-west-2c: { id: subnet-07cdef86020d2af0b }

nodeGroups:

- name: ng-1-workers

labels: { role: workers }

instanceType: t3.xlarge

desiredCapacity: 1

privateNetworking: true

- name: ng-2-builders

labels: { role: builders }

instanceType: t3.2xlarge

desiredCapacity: 1

privateNetworking: true

iam:

withAddonPolicies:

imageBuilder: true

to delete:

check you're in right cluster-

kubectl config get-contexts

switch if needed:

kubectl config use-context aws4@cluster-in-existing-vpc.eu-west-2.eksctl.io

eksctl delete nodegroup --config-file=dev-cluster.yaml --include=ng-2-builders --approve

eksctl upgrade cluster -f config.yaml --approve

eksctl delete cluster --name=<name> [--region=<region>]

make sure the name is like the one appearing on amazon portal.

**IAM (identity & Access Management)**

Example: use of IAM role to attach to EC2 instance for allowing access to aws services, without aws identity/configure.

***ArgoCD***

Automation and visualization of continuous deployment. It allows automatic update of either cluster/pod/container via commit push from github changes to the code. Version re-rolling etc.

Installation:

1)argocd via github >create ns argocd

install via:

1.5) install.yaml: change line 21711 under containers :

args:

- --insecure

because this is local, no certificate.

2)kubectl apply -n argocd -f install.yaml <file in this folder

3)ingress rule:

kubectl get secret -n argocd

4) kubectl get ingress -n argocd

5) get ip of LB to put into the local: ping LB domain, copy IP.

paste into etc/hosts of machine>> kubectl exec -it <pod> /bin/bash

ingress installation: change to host accordingly

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: argo-cd-ui

namespace: argocd

annotations:

ingress.kubernetes.io/proxy-body-size: 100M

ingress.kubernetes.io/app-root: "/"

spec:

ingressClassName: nginx

tls:

- hosts:

- argocd.local

secretName: argocd-secret

rules:

- host: argocd.local

http:

paths:

- path: /

pathType: ImplementationSpecific

backend:

service:

name: argocd-server

port:

name: http

get admin and pasword

user: admin

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

password: dfkhcO7xdhfABi5t

check application works and see automated CD in action:

repo: https://github.com/argoproj/argocd-example-apps.git

>sync policy-auto

>git repo path as above(for repo)

> path: guestbook (according to PRIMARY/guestbook)

> self heal

> auto create namespace!

> default k8s cluster

## CERT-MANAGER(UNFINISHED)

cert-manager is free, and automates purchasing of new TLS certs, (newer version of SSL certs) and also renewing them.

versus for example, wild card certificate which you need to pay 300$ ~ for every subdomain name under the host-zone.

as in either pay per service for a cert. or pay for all of them(wild card certs are valid for 5 years).

so cert manager automates everything regarding certs. and for free.

https://cert-manager.io/docs/getting-started/

need a domain name(hostinger for me)

get LB name from k8s:

kubectl get svc -n ingress-nginx

check the LB has 443 open, because we need https for the certificate

-open new hosted zone with my domain name(route53)

-copy the NS to the registar of my domain(hostinger)

- use the LB address as an A record in route 53

CHANGE to ALIAS! that why you can protocol it from aws resources

choose LoadBalancer + region+ the k8s LB from ingress

that way the domain name will route to the LB and whatever we have(which is k8s!) will show.

used this guide:

https://cert-manager.io/docs/tutorials/acme/nginx-ingress/

stage 4

install ingress deployment

edit the ingress service:

name: arieldevops.tech

api.arieldevops.tech

apply the links on stage 4

check the kuard app is working on the domain api.arieldevops.tech

tehn go to helm guide install link

update repos and install the cert-manager

+issuer/yaml

## ArgoCD

ArgoCD is k8s deployment, that can deploy apps into k8s, in helm or other, via github or other, using ssh connection. It’s a GitOps tool.

# TERRAFORM

Terraform is an Infrastructure as Code (IaC) tool that enables users to define and provision infrastructure resources in a declarative configuration language. While it's commonly used to quickly establish and manage environments, its capabilities extend beyond just creating resources.

Key aspects of Terraform include:

1. **Declarative Configuration:** Terraform uses a declarative syntax to define the desired state of your infrastructure. Users specify what resources they want, and Terraform figures out how to create and manage them.
2. **Multi-Cloud Support:** Terraform is cloud-agnostic, meaning it can be used to manage infrastructure across various cloud providers (AWS, Azure, Google Cloud, etc.) as well as on-premises environments.
3. **Resource Provisioning:** Terraform can provision and manage a wide range of infrastructure resources, such as virtual machines, storage accounts, networks, databases, and more.
4. **Dependency Management:** Terraform understands the dependencies between resources and ensures that resources are created and configured in the correct order.
5. **State Management:** Terraform keeps track of the current state of your infrastructure in a state file. This file is used to plan and apply changes, helping Terraform to understand the differences between the desired state and the current state.
6. **Modularity and Reusability:** Terraform configurations are modular, allowing you to reuse code and share infrastructure patterns across projects.
7. **Versioning and Collaboration:** Terraform configurations can be versioned, allowing teams to collaborate on infrastructure changes and roll back to previous versions if needed.

While Terraform is often used for quickly establishing and managing environments, it's also a tool for maintaining and evolving infrastructure over time. This includes making changes, scaling resources, and managing the entire lifecycle of infrastructure components.

In summary, Terraform is a versatile tool that simplifies infrastructure management, providing a consistent and reproducible way to define, create, and modify infrastructure across different environments.

Components:

* Main.tf : what resources to create
* Provider.tf : data for which user/region/keys
* Provisioner.tf

Other .tf files are:

* + - * Variables.tf : define what variables to be used, for example- provider.tf will use var.region, which was defined in variables, either as a default or another.
      * Terraform.tfstate : gives all the info about the planned terraform
      * Terraform.tfvars: keeps the creds, like main.tf
      * Ami.tf : inside selector of the ami ID for the kind of OS/year/version/type(hyper VM-hvm)/architecture:x86\_64

Terraform windows

* Need special script, and also take the ami ID from amazon, change every ferw months on amazon.
* See scripts on terraform
* Open ports for RDP, on winRM of the default security group of the default vpc (if we didn’t declared a vpc variable!)in amazon: winRM + from anywhere ipv4(port is 5985), and winRM https

Working with AWS as terraform launch machine instance:

* Open a new instance, but first create role for terraform separately
* Attach in advanced>IAM role the - terraform role.

The common practice is to work on an instance, without ami credentials. But with the terraform role that is being used.

That means deleting AWS CREDS inside the .tf files

Ingress(inbound rules)/egress(outbound rules)

Expected values: From\_port, to\_port, protocol(tcp/udp/https), cidr\_blocks = [“0.0.0.0/0”] << expects list of strings

Syntax

[Script for Basics](file:///K:\DevOps\TerraForm\terraform-syntax\intro.txt)

[Script for data\_types](file:///K:\DevOps\TerraForm\terraform-syntax\data_types.txt)

[Script for functions](file:///K:\DevOps\TerraForm\terraform-syntax\built-in_functions.txt)

Structure/formatting

Resource “<type of resource>” “<name in terraform>” { ex. “aws\_key\_pair” “mykey”

<attribute/variable name> = “<value>” , ex. : key\_name = “mykey” is the key pair value of key(mykey).

Giving terraform permissions to access aws resources

Bad practice: Insert access/secret key into variables which later be exposed in terraform.tfvars

Better/secure practice:

-either use aws cli and terraform will recognize it, or-

Attach in advanced>IAM role the - terraform role.

Otherwise we would need to specify vulnerable access/secret keys in the .tf provider file. Which is bad practice.

Provider.tf is the one responsible for allowing access to terraform on the cloud.

The role is to the remote instance which runs terraforms init/apply

The credentials after you have configured aws cli, are in: > cat ~/.aws/credentials

Generating keys for instances to be used by either local/terraform

Locally:

To generate public and private keys via .pem:

ssh-keygen -y -f /path/to/your/private\_key.pem > /path/to/projectDir/private\_key.pub

\*same key pair name in aws\_key\_pair block, and in aws\_instance block, will ensure the instance will be using the key I tell him.

Example: resource “aws\_key\_pair” “mykey” {

Key\_name = “key”

}

Resource “aws\_instance” “ec2\_instance" {

Key\_name = aws\_key\_pair.mykey.key\_name # this uses the key from the key-pair block

}

Creating an instance resource

Count = <number of instances>

Ami = <the ami ID, which is the OS[ubuntu/centos/windows etc]> example: ami = “${var.AMIS[var.region]}” >> which calls from the variable AMIS, and then calls to

The ID from the map of AMIS via the region, on var.region (default)

\*\*make sure the ami is current. use the ami finder

Type = <t2.micro or other>

SSH/creating key-pairs for later use to connect to an Instance

We need private key in local , and the public key (from our local) to be in the remote.

Using file(“<path to pub key>”) we upload to aws the public key to the instance for later ssh.

To just create key pair, terraform and aws will extract it from a regular ssh.

Ssh-keygen -t rsa >>> then type name of key

Calling values of resources/variables

Resource “aws\_instance” , or resource “aws\_key\_pair”, is calling for these resources in aws.

You can call the values of resources from inside the aws.

Public\_id, id are values from inside the aws. So by Using output block, you call these values.

For example- resource “aws\_instance” “example” {

#some configuration for instance

}

Output “instance\_id” {

Value = aws\_instance.example.id

}

Output “public\_id” {

Value = aws\_instance.example.public\_id

}

You can’t call something that hasn’t been established.

For example: aws\_instance.example.count , won’t call for count value, but –

output “count” {

value = length(aws\_instance.example)

}

These will be displayed in terraform outputs, after *terraform apply*, or using *terraform outputs* only.

Every reference/reserved name is shown in the attributes inside of terraform.tfstate

Establishing connection to instance

To run any script on the instance, you would need to establish connection first.

Connection block- Inside the block of the resource.

Self.XXXX is a reference to the instance block you are in,

Self.public\_ip and self.private\_ip are aws references, using coalesce will determine which one is not null, and return the value of the non-null:

Connection {

Host = coalesce(self.public\_ip, self.private\_ip)

Type = ssh

User = “${var.INSTANCE\_USERNAME}”

Private\_key = file(“${var.PATH\_TO\_PRIVATE \_KEY}”)

}

Files in terraform

Can name files according to resources, for example key.tf, default\_vpc.tf, ec2.tf, and TF will know to call upon these resources by name

Even though they are not in the files(but are in the folder)

Modules

A way to encapsulate resources and other TF components, for more arranged access.

For example- vpc, instances, s3 etc to be raised and configured separately. also it allows modular changes to be made

, example- in customized cidr blocks, different AZ.

TF calls the module using source = “<path/to/module>” and treats it like any other folder of TF.

Any Variables can be passed from parent to child. But the passed variables from parent, should be defined also in child’s.

Defaults are the goto for modular usage of modules. The value that can be put in variables.tf will override the defaults, allowing modularity.

OUTPUTS in module: write outputs as usual on the module, and pass the defined output name to the outputs in root. To view them on TF apply.

Terraforming from CLOUD

Attach IAM role for the instance that use terraform. This will allow aws access without aws configure.

**VERSIONING in TerraForm**

Why?

As an example, When multiple teams make changes to a terraform template, because of deployment changes or development necessities, they plan and apply, but versioning should be common practice, to prevent loss of backups

The backup/state/version of template is stored in an S3 bucket of aws, and also via a table in DynamoDB(aws DB).

Also it allows for locking of a state while the plan is applied- to prevent conflicts. So another team cannot apply while the other is.

Also, very important- it doesn’t save the state locally(!), since remote is what the current version should be, in case someone updated-non-locally.

Remote backup/ BACKEND

The backend(in the script of demo-8 provider.tf) block, is writing configuration to top level configuration(using terraform block)

In a file backend.tf and knows to apply it to the tfstate, and save the state remotely.

See scripts of demo-8

We use DynamoDB to organize the states of TF, and S3 to save the states. Backend is a block in TF and in TerraGrunt

TAINT

Case where you’d want to “destroy” or rebuild a resource, without ruining everything.

Terraform taint <resource> will “defect” that resource, while terraform apply will correct it.

Terraform Untaint <resource> will restore the resource to be in normal/healthy shape.

העברת PUBLIC KEY לREMOTE INSTANCE

resource "null\_resource" "add\_ssh\_key" {

depends\_on = [aws\_instance.example]

# Provisioner to copy SSH key

provisioner "remote-exec" {

connection {

type = "ssh"

user = "ubuntu"

private\_key = file("${path.module}/private\_key")

host = aws\_instance.example.public\_ip

}

inline = [

"echo '${file("${path.module}/public\_key.pub")}' >> ~/.ssh/authorized\_keys"

]

}

}

בפקודת הINLINE ניתן לשפוך את הPUBLIC KEY ישירות לAUTHORIZED\_KEYS כדי לאפשר אוטומציה בהרשאות מכונות אחת עם השניה.

# **TerraGrunt**

TERRAGRUNT מאפשר מעטפת WRAPPER, על TERRAFORM שדרכה מריצים-

בדיקה על יבש DRY איכותית יותר מPLAN,

בTERRAGRUNT הWORKSPACE רץ הכל דרך הGIT ,לעומת הWROKSPACE שהוא לוקאלי בTERRAFORM – ואז אין צורך שבעבודת צוות

כולם יהיו שותפים לWORKSPACE, אם עובדים בREMOTE.

אפשר שיהיה REMOTE STATE עבור כל סביבה.

ניתן ממקור אחד להרים את כל הסביבה במקום שיהיה חזרתיות של קבצים לכל סביבה. זה השמירה על הDRY, כי TG עוטף את TF.

כתיבת TG: עובדים בBLOCKS-

סוגי הבלוקים: terraform, include, locals, dependencies, remote, generate

Terraform block:

איפה נמצא המודול, ומה אנחנו מרשים לעשות מבחינת terraform, HOOK שלפני ואחרי.



Remote blocks:

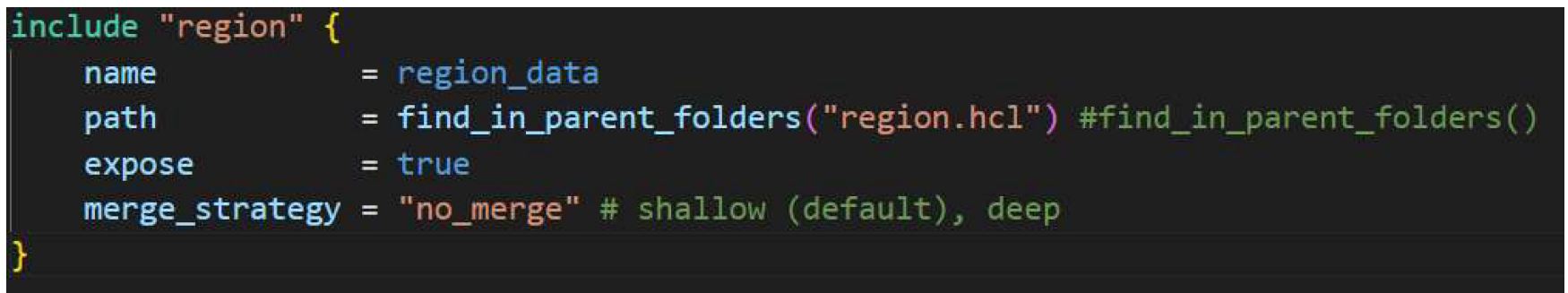
מאיפה לוקחים את הREMOTE STATE , שבתוך DynamoDB+s3, ואת הקונפיגורציה שלו. ומושכים את הקונפיג, חשבונות וכולי ע"י find\_in\_parent\_folders()



\*\*\*VARIABLE של מפתח בBACKEND של TERRAFORM לא יעבוד. TF לא מרשה לקרוא VARIABLE מתוך BACKEND BLOCK.

INCLUDE BLOCK:

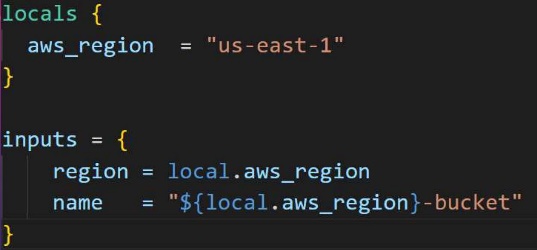
הTG קורא מתוך קבצי HCL ומשם מפעיל TERRAFORM. בבלוק הזה אנחנו יכולים "לרשת" מHCL אחרים ולציין מאיפה לקחת את "הירושה".



מחפש בעזרת find\_in.. כדי למצוא עוד הגדרות שקשורות לREGION במקום שנגדיר אותו שוב מחדש.

LOCALS BLOCK

משתנים או קבועים שנשתמש בהם לאורך כל ההרמה, כמה פעמים, כדי שלא נצטרך להגדיר אותם שוב מחדש.



הINPUTS הם BLOCK של TERRAGRUNT שהוא כמו vars.tf

לעומת זאת בTF:

locals {

service\_name = "my-service"

owner = "team-name"

}

resource "aws\_instance" "example" {

tags = {

Name = local.service\_name

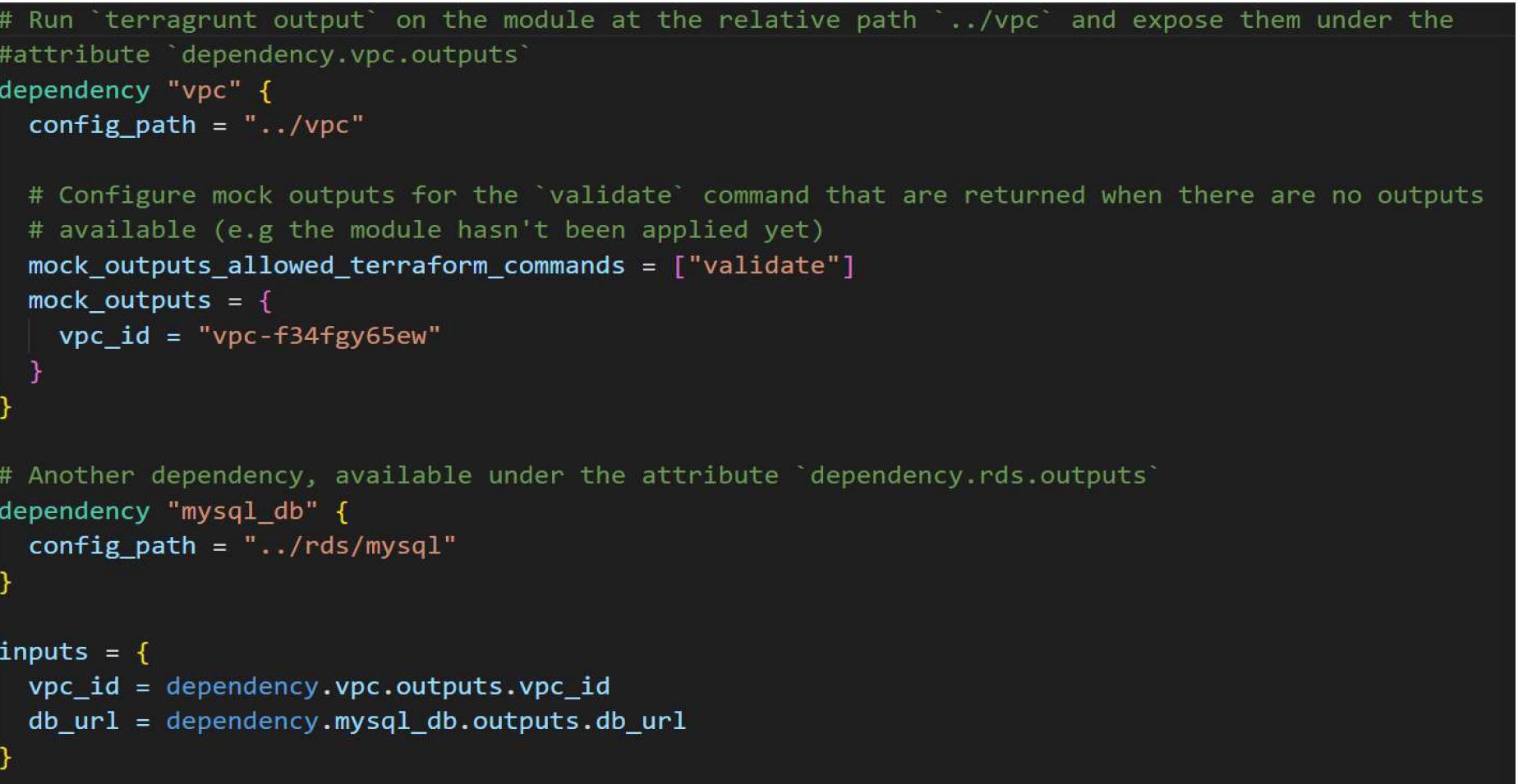
Owner = local.owner

}

}

DEPENDNCIES BLOCK

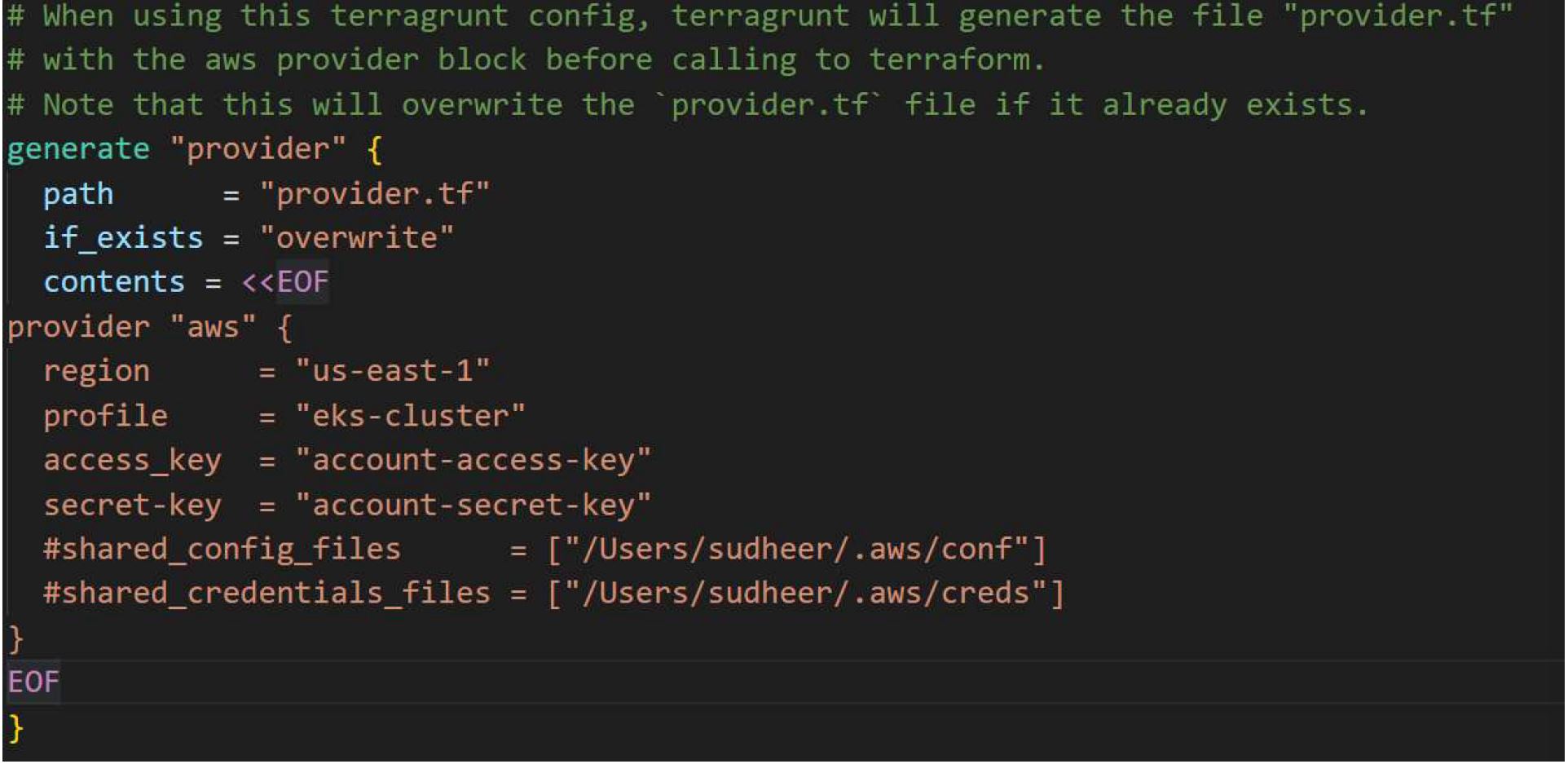
לדוגמא אם צריך להרים VPC,אז קודם להרים SUBNET GROUPS ואז VPC ועליו לפרוש עוד משאב, אז נצטרך לתזמן ולסנכרן אותם לפי סדר.



\*\*אחד אחרי השני הוא יריץ/ירים, ובפקודת run-all יהיה אפשר להרים את זה. יש שימוש בPATH בד"כ.

GENERATE BLOCK

מייצר קבצי TF איפה שHCL נמצאים, עבור כל קוד של TF או להשתמש בנתון, לדוגמא עבור PROVIDER:



אם נרצה להשתמש בPROVIDER קיים: if\_exists= “ignore” ואז טרהגרנט לא ידרוס את הקיים.

[דוגמא לMULTI ENV של terraGrunt ש-מרים גם PROD, DEV וגם STAGING](https://github.com/lidorg-dev/terragrunt-multi-env-management/tree/main) (תחת ENV)

Name of file(hcl) in terragrunt are always terragrunt.hcl

# **ANSIBLE**

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-ubuntu-22-04>

what is ansible?

ansible is sort of script wrapper after we have resources.

using modules, which are declarative blocks over yaml files-the playbooks, or roles, which are basically meant to replace scripts.

These can be wrapped over the instances.

check:

ansible --version

raising 2 ubuntu(web) + 1 aws linux(db) on same vpc and sec.group+ sec make sure SSH from anywhere

the ssh from gitbash is to the remote-controller machine.(ubuntu user)

on remote ansible machine mkdir ansible>inventory>

>vi hosts

----------

[remote-controller]

ansible ansible\_host=127.0.0.1 ansible\_connection=local

[web]

web1 ansible\_host=10.0.101.250 anisble\_connection=ssh ansible\_user=ubuntu

web2 ansible\_host=10.0.101.156 anisble\_connection=ssh ansible\_user=ubuntu

[db]

db ansible\_host=10.0.101.37 anisble\_connection=ssh ansible\_user=ec2-user

----------

but to ssh from the remote-controller to the other servers we need the private key hash:

we copy it from the local machine (windows) where i use it to authenticate with ariel-key.pem

, we copy this, into the remote controller.

copy ariel-key.pem from your local windows machine to the remote-controller, and then change the name to id\_rsa inside

~/.ssh

check ping for remote with ansible:

ansible -i inventory/hosts -m ping remote-controller

ansible -i inventory/hosts -m ping web

ansible -i inventory/hosts -m ping db

ansible -i inventory/hosts -m ping all (checks all machines)

----

but we need ssh from of aws key... of the remote machine?

create a config for ansible to know where is inventory to load from

>ls

inventory (pwd=ubuntu/home/ansible)

>vi ansible.cfg

[defaults]

inventory=./inventory/hosts

now, we can use the command without -i

anisble -m ping web (and other checks

-----

create file to all servers:

>ansible all -m file -a "path=/tmp/adhoc1 state=touch mode=700"

---

if we want telnet on db

change in hosts for sudo permission to get telnet permission/elvated/.escalate permission:

after... ansible\_user=ec2-user ansible\_become=true

>ansible db -m yum -a "name=telnet state=present"

if not present it will install a telnet (yum install)

----

to install apache onto ubuntu web machines(both)

>ansible web -m apt -a "name=apach2 present=true"

------

playbook

------

creating a new playbook on visual studio called playbook-1:

---

- name: playbook1

host: web

tasks:

- name: install apache web Package

apt:

name: apache2

state: present

check:

>ansible-playbook playbook-1

added to start service and enable on boot:

also validatation of status

---

- name: playbook1

host: web

tasks:

- name: install apache web Package

apt:

name: apache2

state: present

- name: start Apache service

service:

name: apache2

state: started

enabled: yes

- name: validate status of apache service

shell:

cmd: "systemctl status apache2"

----

chatgpt fix:

---

- name: Install and start Apache web server

hosts: web # Corrected from 'host' to 'hosts'

become: yes # This is often necessary for tasks that require root privileges

tasks:

- name: Install apache web package

apt:

name: apache2

state: present

- name: Start and enable Apache service

service:

name: apache2

state: started

enabled: yes

- name: Validate status of Apache service

command: systemctl status apache2 # Changed from 'shell' to 'command'

register: result # Registers the output of the command

- debug: var=result.stdout\_lines # Optionally, to print the status in the playbook output

----------

we want to install mysql db via ansible's playbook.

we check in guide how to install mysql on centos7 (the machine of the db)

XXXXXXXXXXXXXXXXXXX not working! XXXXXXXXXXXXXXXXXXXXXXXXXX

we need to first add repo(took script from guide on ansible add repo module)

- name: add mysql db

host: db-1

tasks:

- name: download rpm file

shell:

cmd: curl -sSLO https://dev.mysql.com/get/mysql80-community-release-el7-5.noarch.rpm

- name: install rpm file

yum:

name: mysql80-community-release-el7-5/noarch.rpm

state: present

- name: install mysql package

yum:

name: mysql-server

state: present

update\_cache: true

XXXXXXXXXXXXXXX because its not centos7 but a linux aws machine XXXXXXXXXXXXXXXXXXX

<check remove package if there is installed>

working playbook:

---

- name: play1

hosts: web

become: yes

tasks:

- name: install Apache web Package

apt:

name: apache2

state: present

- name: start Apache service

service:

name: apache2

state: started

enabled: yes

- name: validate status of apache service

shell:

cmd: "systemctl status apache2"

- name: play2

hosts: db

become: yes

tasks:

- name: download rpm file

shell:

cmd: "curl -sSLO https://dev.mysql.com/get/mysql80-community-release-el9-3.noarch.rpm"

- name: install rpm file

yum:

name: mysql80-community-release-el9-3.noarch.rpm

state: present

disable\_gpg\_check: true

- name: install mysql package

yum:

name: mysql-community-server

state: present

update\_cache: true

- name: start Mysql Server service

service:

name: mysqld

state: started

enabled: yes

- name: validate status of Mysql Server service

shell:

cmd: "systemctl status mysqld"

---

to execute it from the ansible main folder

ansible-playbook playbook/playbook-1

add http port 80 to the security group so that you can browse

we want to copy a file to change the index.html of the mysql db opening page

we look for docs of it in ansible under :copy

handlers:

we use if something has change, and for example apache needs to be restrated,

like changing the index.html

file/ariel-index.html can be mkdir files >> ariel-index.html

handler must have notify before it

---

- name: play1

hosts: web

become: yes

tasks:

- name: install Apache web Package

apt:

name: apache2

state: present

- name: start Apache service

service:

name: apache2

state: started

enabled: yes

- name: validate status of apache service

shell:

cmd: "systemctl status apache2"

- name: copy website

copy:

src: ../files/ariel-index.html

dest: /var/www/html/index.html

owner: www-data

group: www-data

mode: '0644'

backup : true

notify:

- restart Apache

handlers:

- name: restart Apache

service:

name: apache2

state: restarted

- name: play2

hosts: db

become: yes

tasks:

- name: download rpm file

shell:

cmd: "curl -sSLO https://dev.mysql.com/get/mysql80-community-release-el9-3.noarch.rpm"

- name: install rpm file

yum:

name: mysql80-community-release-el9-3.noarch.rpm

state: present

disable\_gpg\_check: true

- name: install mysql package

yum:

name: mysql-community-server

state: present

update\_cache: true

- name: start Mysql Server service

service:

name: mysqld

state: started

enabled: yes

- name: validate status of Mysql Server service

shell:

cmd: "systemctl status mysqld"

----

line in file

simply add a line into any file we want. in this case we changed max\_connection in my.cfg to mysql configuration

---

- name: play1

hosts: web

tasks:

- name: install Apache web Package

apt:

name: apache2

state: present

- name: start Apache service

service:

name: apache2

state: started

enabled: yes

- name: validate status of apache service

shell:

cmd: "systemctl status apache2"

- name: copy website

copy:

src: ../files/index-lidor.html

dest: /var/www/html/index.html

owner: www-data

group: www-data

mode: '0644'

backup : true

notify:

- restart Apache

handlers:

- name: restart Apache

service:

name: apache2

state: restarted

- name: play2

hosts: db

tasks:

- name: download rpm file

shell:

cmd: "curl -sSLO https://dev.mysql.com/get/mysql80-community-release-el9-3.noarch.rpm"

- name: install rpm file

yum:

name: mysql80-community-release-el9-3.noarch.rpm

state: present

disable\_gpg\_check: true

- name: install mysql package

yum:

name: mysql-community-server

state: present

update\_cache: true

- name: start Mysql Server service

service:

name: mysqld

state: started

enabled: yes

- name: validate status of Mysql Server service

shell:

cmd: "systemctl status mysqld"

- name: add or change config file line

lineinfile:

path: /etc/my.cnf

line: max\_connections = 100

create: yes

notify:

- restart Mysql

handlers:

- name: restart Mysql

service:

name: mysqld

state: restarted

-----

hosts updated: with become true, that allow root access

-==========

[remote-controller]

ansible ansible\_host=127.0.0.1 ansible\_connection=local ansible\_become=true

[web]

web-1 ansible\_host=10.0.101.250 anisble\_connection=ssh ansible\_user=ubuntu ansib

le\_become=true

web-2 ansible\_host=10.0.101.156 anisble\_connection=ssh ansible\_user=ubuntu ansib

le\_become=true

[db]

db-1 ansible\_host=10.0.101.37 anisble\_connection=ssh ansible\_user=ec2-user ansib

le\_become=true

======

but we can also create variables for all of them which repeat:

[remote-controller]

ansible ansible\_host=127.0.0.1 ansible\_connection=local ansible\_become=true

[windows]

iis-server ansible\_host=18.170.97.30 ansible\_connection=winrm ansible\_port=5985 ansible\_user=ansible ansible\_password=(what we put in windows user)

[web]

web-1 ansible\_host=10.0.101.250

web-2 ansible\_host=10.0.101.156

[db]

db-1 ansible\_host=10.0.101.37

le\_become=true

[servers:children]

remote-controller

web

db

[web:vars]

anisble\_connection=ssh

ansible\_user=ubuntu

ansible\_become=true

[db:vars]

anisble\_connection=ssh

ansible\_user=ec2-user

ansible\_become=true

---------------

to enable windows we create instance of windows t2.xlarge, using winRM-http and winRM-https open to all or to specific ip.

and we also need to open ports of firewall

firwall settings(inside the windows machine-check how to get in in windows\_pass.txt) >

add ports 5985, 443 5986

but even though we tell it to use port 5985 and give him the user password (we created on windwos) :

windows rdp will reject it, because its still from txt(unsecure).

so we need to use

VAULT

----

ansible-vault encrypt inventory/hosts (inside ansible folder)

will encrypt ALL the hosts file, which isn't good for us.

so we have:

ansible-vault encrypt\_string <password we want to encrypt>

then we get:

!vault |

dkjslfkjdlvncxm,vnx

and some encrypted stuff, we copy into the hosts part of : ansible\_password= <copied encrypted password>

then we use the password we know, into the ping that connects to the windows:

anisble windows -m ping --ask-vault-pass

<feed password(normal one)>

but it didn't work because the hosts isnt a formatted yaml file

so we need to convert it to hosts.yaml and change the syntaxs:

preparing IIS windows server:(web-server and common http)

playbook-windows

---

- name: IIS playbook

hosts: windows

tasks:

- name: Install WEB SERVER IIS

ansible.windows.win\_feature:

name: Web-Server

state: present

include\_sub\_features: true

include\_management\_tools: true

register: win\_feature

register is an ansible feature-read on it. something about tasking outputs.

need to add --ask-vault-pass to ansible command, because it has the access there:

ansible windows -m win\_pink --ask-vault-pass

also add inbound rule port 80/http to everywhere in sec.group of instance, to see the sindows server via browser.

according to the inventory- hosts has to run on something that is already in the hosts/inventory. so windows runs on the windows hosts/server we defined on hosts/inventory.

ROLES

--------

roles are NOT like hosts. that has to be pre-defined.

roles are tasks of group related-like something only for hosts:windows, or just for db.

we have roles in :

galaxy.ansible.com

which can raise wordpress/windows etc anything.

but role ISN'T a playbook. we use the role to have it on the playbook.

to install a role, we take the link from the "repo" in galaxy.ansible.com

for example docker, should NOTE filter VIA download count/popularity.

<\*\*DIDNT WORK\*\*>

there's a command also to create role for yourself:

anisble-galaxy init ariel-role

this will create all the structures of the roles/inv+files, a generic/defaultive.

# **PROMETHEUS-GRAFANA**

After installation, there should be Prometheus, alertmanager and Grafana.

To get more graphs from Grafana, use explore>query> cpu<node in seconds>>namespaces>name of name space duplicate query change ns> add to dashboard

Can get dashboards from Grafana dashboard website.

## DEBUGGING

/var/log/auth.log

SSH: key pair are always identical. When transferring them, they must be exactly the same. As long as the public key on the hosting machine won’t match

The private key of the ssh’ing machine, you’d be denied.

To check “pairness” use your private key name here:

ssh-keygen -y -f <path+keyname > ~/.ssh/id\_rsa.pub

LOGS: any /var/log

לדעת את הכתובת של כל NODE: kubectl get nodes -o wide

בדיקת קישוריות/תקשורת בין פודים בCLUSTER:

נכנסים לPOD ועושים CURL לפי הSVC שלו. לדוגמא

Kubectl exec -it <frontend pod> -n react -- /bin/bash

Curl <http://backend:80>

מאחר ורואים שהכתובת הפנימיתצ של הCLUSTER לפי הSVC: backend.react.svc.cluster.local:80

לפי זה: <name of svc>.<namespace>.cluster.local:80

To see how ingress interacts with application routes/paths use : kubectl logs <ingress pod name>

גילויי קבצי קונפיגורציה : cat /etc/systemd/system/kubelet.service

או systemctl cat kubelet

Kubectl config view : מציג מידע על הקלאסטר

הצגת לוג מאתמול: journalctl -u <service name> --since “yesterday”

Journalctl -p err -b -o verbose

בכל אוטומציה אפשר לשים set -e שיעצור את ההתקנה כל פעם שיש ERROR