

# Scripts1

Tuesday, August 10, 2021 9:43 AM

- export PATH=\${PATH}:/ - add the ./ to the path so that I can execute scripts without writing ./script.sh for example: script.sh (will execute)
- Variables can be deleted using the command: unset

## Shell VARS:

name	value
USER	Name of the user running the shell
HOSTNAME	Name of the host on which the shell runs
PWD	The current working directory
RANDOM	Random value between 0 and 32767
HOME	The user's home directory
PATH	Search path for commands (see ex. 4.19 on page 57)
SHELL	Full path of the shell currently running

Command substitution: VAR=\$(command combination that you want)

Or: VAR=`command combination that you want`

**NOTE:** with \$() you can open subshell in a subshell like: \$(())  
with `` you cannot open another subshell inside. `` is not possible.

## Escaping Strings

- Printing data with echo
- Defining variables

In such a case we need to escape them, i.e. precede them by a \ character, e.

```
blubber=foo
echo $blubber \#;\;
```

Output:

```
$blubber_#;\
```

- Continue- go to loop start without pursuing the iteration.
- Break- ends the loop.

?- place holder

For example: ls aug?st

Output: august august augwst

Can also: ls aug???

Output: august augyyy

**Note:** will only return aug with 3 last letters.

**test** is a very important program that is used all the time in scripting.

Its main purpose is to compare numbers or strings or to check certain properties about files

Most checks the test program can perform follow the syntax

```
test <operator> <argument>
```

Or

```
test <argument1> <operator> <argument2>
```

```
test -z "$VAR" # Test if a string is empty
test "a" == "b" # Test if two strings are equal
test 9 -lt 3 # Test if the first number is less than the second
test -f "file" # Test if a file exists and is a regular file
```

- @ expands the list of arguments that are given in the command prompt.

For example:	./script.sh a b c d
Output:	a b c d

- # gives the total of arguments that are sent.

For example:	./script.sh a b c d
Output:	4

- ? Gives the exit code of the previous command

For example:	Echo \$?
Output:	100

## Test command instead of if in a script:

```
test -z "$VAR" # Test if a string is empty
test "a" == "b" # Test if two strings are equal
test 9 -lt 3 # Test if the first number is less than the second
test -f "file" # Test if a file exists and is a regular file
```

operator	description
-e FILE	True if file exists.
-f FILE	True if file exists and is a regular file.
-d FILE	True if file exists and is a directory.
-x FILE	True if file exists and is executable.
-z STRING	True if string is empty
-n STRING	True if string is not empty
STRING = STRING	True if strings are identical
STRING != STRING	True if strings are different

! EXPR	True if EXPR is false
EXPR1 -o EXPR2	True if EXPR1 or EXPR2 are true
EXPR1 -a EXPR2	True if EXPR1 and EXPR2 are true
( )	grouping expressions
NUM1 -eq NUM2	True if number NUM1 equals NUM2
NUM1 -ne NUM2	True if NUM1 is not equal to NUM2
NUM1 -lt NUM2	True if NUM1 is less than NUM2
NUM1 -le NUM2	True if NUM1 is less or equal NUM2
NUM1 -gt NUM2	True if NUM1 is greater NUM2
NUM1 -ge NUM2	True if NUM1 is greater or equal NUM2

\$?:  
0 is a success!

Tail -f - shows continuously the updates of a content of a file.

# Scripts2

Wednesday, August 11, 2021 9:12 AM

## READ

read: only looks on ONE LINE in the file!!!!

If you want to read all rows, use `for` or something.

Example: `< resources/rr.txt read COM MAT FLAGS`

💡 **NOTE:** Each string will get inside the parameter, if you have less parameters than strings, the last parameter will get all the last strings.

### Two options worth mentioning:

- `-p STRING`: Print `STRING` before waiting for input — like a command prompt.
- `-e`: Enable support for navigation through the input terminal and some other very comfortable things.

`Ctrl+d (EOF)`- ends the read with no data

**Find the length of a string:** `echo ${#string_var}`

## IFS

Internal field separator .

💡 **NOTE:** for loop doesn't use `IFS`, it only uses white space. If you want to change the default delimiter, you need to change the list before hand to the correct delimiter and then send it to `for`.

**LDAP**- tree of levels that a the and sits the "entity"(user) authentication for authorization

LDAP provides the communication language that applications use to communicate with other directory services servers. Directory services store the users, passwords, and computer accounts, and share that information with other entities on the network.

## Scripting conventions:

- Always use a shebang as the first line of your script.
- A block of code doing a task should have a comment explaining what goes in and what the expected result should be. This is especially true for functions
- Whenever funny *bashisms* are used that could make code unclear, explain what happens and why. Think about the future you ;).
- *bashisms* - Chain of special characters which look like magic to someone new to shell scripting
- One script should only do one job and no more.  
Split complicated tasks into many scripts.  
This makes it easier to code and easier to reuse.
- Try to design scripts as filters, i.e. better read from `stdin` and write to `stdout` rather than to/from files. This simplifies code reuse, too.  
Think about the core Unix tools: The utilities you use most often are very likely just some kind of elaborate filter from `stdin` to `stdout`.
  - Use shell functions to structure your script. Have a comment what each function does.
  - Reserve `stdin` for data:
- Do not use the `read` command to ask the user for data or parameters, much rather use argument parsing for this. The reason is that using `read` interferes with reading data from `stdin`
  - Use helpful error messages with as much info as possible. Print them to `stderr`
- Reserve `stderr` for errors, `stdout` for regular output.

If you need to output two separate things, have the more important one printed to `stdout`, the other into a file.

Even better: Allow the user to choose what goes into the file and what to `stdout`.

⇒ Again all of this can be summarised as "design each script as a filter"

- Each script should support the arguments `-h` or `--help`.

If these arguments are provided, explain what the script does and explain at least the most important commandline arguments it supports.

- For each argument there should be a descriptive "long option" preceded by two `--`. There may be short options (preceded by one `-`).

- Do not worry about the long argument names.
  - You can code tab completion your script.

# Scripts3

Thursday, August 12, 2021 8:59 AM

## Crontab

Definitions:

**cron** is the name of the tool, **crontab** is generally the file that lists the jobs that cron will be executing, and those jobs are, surprise surprise, **cronjobs**.

- Schedules jobs.

Each user, including root, can have a cron file. These files don't exist by default, but can be created in the **/var/spool/cron** directory using the **crontab -e** command

From <<https://opensource.com/article/17/11/how-use-cron-linux>>

**Tail -f - shows continuously the updates of a content of a file.**

/bin	User Binary Files
/boot	Boot Loader Files
/dev	Device Files
/etc	Configuration Files
/home	Home Directories
/lib	System Libraries
/media	Mount point for removable media
/mnt	Mount point for temporary file systems
/opt	Optional Addon Applications
/sbin	System Binaries
/srv	Service Data
/tmp	Temporary Files
/usr	User programs
/var	Variable Files
/root	Root User Directory
/proc	Process Information
/lost + found	

## AWK

Echo hi bye why | cut -d" " -f3

Echo hi bye why | awk -F" " '{print \$3}'

Both will give the same result.

- \$NF- will print the last field  
\$ (NF+1)=\$1 # add a new field equal to field 1

From <<https://www.bing.com/search?q=first%20field%20in%20awk&qsn=&form=QBRE&sp=-1&pq=first%20field%20in%20awk&sc=1-18&sk=&cvid=A75BD48130F74CD29C6D45B0481B5868>>

## TR

tr ':' ' ' --> changes the first field to the second field in a string

Date- shows date

Cal- shows calendar

RunLevel	Mode	Action
0	Shutdown	Shuts down system
1	Single-User Mode	Maintenance mode
2	Multi-User Mode	Does not configure network interfaces.
3	Multi-User Mode with Networking	Starts the system normally with all the services(Text mode Interface)
4	N/A	Not used
5	X11	As runlevel 3 + GUI
6	Reboot	Reboots the system

# Devops

Sunday, August 15, 2021 9:21 AM

Bloom taxonomy (?)  
100, 200, 300 exe.

Dev folder- contains all the stuff about all the hardware.(devices)

**Artifact-** output of a process (package or executable)  
**Matureness-** the compliance level of an artifact, during its development against the business standards and rules. (such as unit test, tests, security, stress, scale)  
**Build tool-** able to perform transition between one programming lang into a target lang.(compiler) the output is an ARTIFACT.  
**Test tool-** able to perform transition between one programming lang into target lang (compiler)  
**Framework-** structure for content or process that can be used as a tool to achieve technical goals.  
**Build frameworks-** for interpreted lang, "package" the code project to a single file or a single zipped folder.

Container = process (single one)

What is DevOps ?

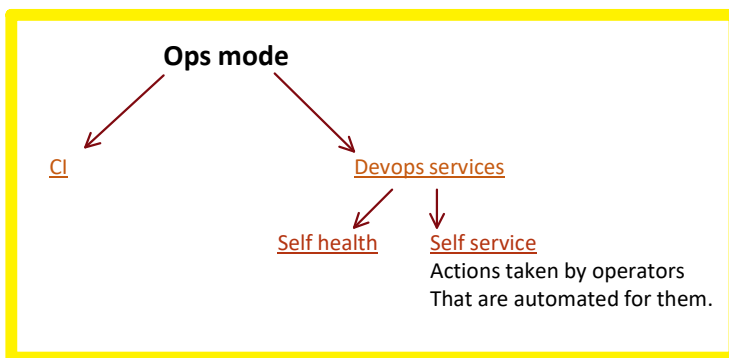
Combination of software & development  
Provides continuous delivery and quality.

I give the organization the agility during development  
And quick changes at the market. And lowering the cost of  
The resources.

## 3 MAIN FIELDS OF DEVOPS:

Production support  
Automation support  
CI support

Logical environment- (later)



- Su- superuser
- Gnu- free Unix-like os
- Authority:
  - The first bit- type of file(file, directory etc.)
  - User- owner(the user that created the file)
  - Group-my user group(the business unit)
  - Other- all the system users
- How to RDP from web? Guacamole
- NEVER SHOW YOUR IP TO THE WORLD!
- Kubernetes unifies secure resources.
- Node is an allocation unit (ec2...)
- Distributed systems- separate a big work to microservices in many nodes and jobs.

## Artifactory essentials

### Only Jfrog

#### INTRO

Source code

|

|

v

Works with symbolic link

File distribution of the same configurations:

save the status of the file that you want to save.

**What is Salt?** *Fast, scalable and flexible software for data center automation.* Salt is a new approach to infrastructure management. Easy enough to get running in minutes, scalable enough to manage tens of thousands of servers, and fast enough to communicate with them in seconds Salt delivers a dynamic communication bus for infrastructures that can be used for orchestration, remote execution, configuration management and much more..

**What is CFEngine?** *IT Automation at WebScale.* It is an IT infrastructure automation and Continuous Operations framework that helps engineers, system administrators and other stakeholders in an IT organization manage IT infrastructure while ensuring service levels and compliance.

From <<https://stackshare.io/stackups/cfengine-vs-salt>>

There are others like ansible, chef and puppet

Open source: infrastructure configuration, app deployment

And use of Infrastructure as a code.

Artifact I upload to binaric repository manager. JFROG is a B.R.M!

It knows how to manage versions of binaric files.

Saves the difs only, the base it saves once and that what gives you the freedom to move between versions and save A LOT of space!

Plan -> what I want to do? (informative file of all the project /high level & low level design)

Code -> the actual code to push to a remote repository

Build -> compile your product, python jumps on this step because of the interpreter

Test -> check your build, is it right?, can be internal code check or security checks.(or anything you want)

Release-> code revues and checks of other teams like QA. (final artifact to use in production)

Deploy -> final working stage to the production or any other place.

Blue ocean- plugin in jenkins that: (google it you are not done)

# Ports

Sunday, 22 August 2021 9:22

JNLP -Java Network Launch Protocol

בסט פרקטים השתמש בפורטים להשתמש בפורטים מעל 1024  
תחת root אפשר להשתמש בפורטים מתחת ל-1024 אבל זה לא בסט פרקטים

## Well-Known Ports

Service	Port	Function
HTTP	80	Web traffic
HTTPS	443	Secure web traffic
FTP	20, 21	File transfer
DNS	53	Name resolution
SMTP	25	Internet mail
POP3	110	Post Office Protocol (POP) mailbox
IMAP	143	Internet Message Access Protocol (IMAP) Mailbox
Telnet	23	Remote login
SSH	22	Secure remote login

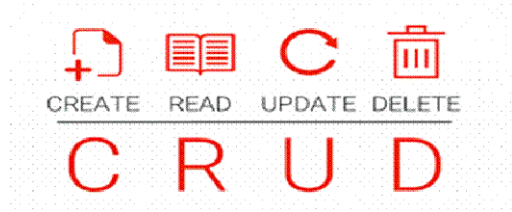
Port Numbers	Description
1-1023	The "well-known" ports
1024-49,151	Registered ports
49,152-65,535	Private ports

# Data bases

Sunday, 22 August 2021 9:23

## Data Base

erase condition נעילה בעולם של דטה זה משהו שהוא לא בהכרח רע, הוא רק כשפעולה מסוימת מתעכבת בגלל זה הוא רע נעילה היא אמצעי להגן על שלמות המידע, נעילות יכולות להתבצע על כל ריסורס.



על פעולות בדטה בייס של create update and delete ינעל בצורה אקסקלוסיבית על פעולת update הוא ינעל בצורה

דוגמא לdead lock: מצב לדוגמא שלי ולכן הוזג שלי יש כרטיס וגישה לאותו החשבון בנק המשותף, שנינו באותו זמן מנסים למשוך כסף מהחשבון שנינו מנסים ליצור פעולת delete, ואז שנינו מגיעים לאותה המשאב לאותו הריסורס שלי שהוא חשבון הבנק שלי ושנינו נועלים את אותו המידע, אנחנו לא רוצים להגיע למצב הזה ונלכד יש לנו 2 תפיסות

**יש 2 תפיסות:**

# תפיסה פסמיסטית-בעת ביצוע פעולות המשנות נתונים cud לא תותר אף פעולה אחרת על אותה אויבייקט (ריסורס)  
# תפיסה אופטימיסטית- בעת ביצוע פעולות המשנות נתונים פעולת הקריאה תותר לפעולת read בלבד (המידע אשר ישלח, יתקבל ללקוח חזרה יהיה המידע טרם ביצוע הפעולה)

בעולם שלנו זה מתקשר לעיקרון immutability, בעצם מה שנרצה זה שתהיה לנו מכונה ובתוכה קונטיינרים שכל אחד מהם הוא פרוסס בנפרד כדי שלא תיווצר לי התנגשות כזו של 2 פרוססים שונים שמנסים להשתמש באותו הריסורס בו זמנית

**קיימות שתי תפיסות עיקריות לניהול נעילות:**

**פסמיסטית-** בעט ביצוע פעולות המשנות את המידע כמו : UPDATE CREATE DELETE, לא תותר אף פעולה אחרת על אותו אויבייקט.  
**אופטימיסטית-** בעט ביצוע פעולות המשנות נתונים, פעולת הקריאה תותר לפעולת READ בלבד. (המידע אשר יישלח, המידע הינו טרם ביצוע הפעולה)

כדי להמנע מנעילות: מבודדים לורקספייס, לפרוסס אחד בלבד על כל נוד לתכנן את התהליך להיות בדיד ואטומי בלי תלות בגורמים אחרים.



# Jenkins practice

Sunday, 22 August 2021 9:26

## class practice

לבדוק כאן שכתובת האיפי שלנו היא כתובת האיפי של המכונה שהיא המאסטר שלי



<b>Jenkins Location</b>
Jenkins URL
<input type="text" value="http://3.121.113.154:8080/"/>
System Admin e-mail address
<input type="text" value="address not configured yet &lt;nobody@nowhere&gt;"/>

### Commands for download jenkins on amazon linux machine and connecting volume :

#### on aws console:

- 1) IAM role with ec2 full access and ssm access from ec2
- 2) security group with out all traffic 0.0.0.0/0 , in all traffic from my ip and all traffic to my sg
- 4) ssh key
- 5) ebs create volume
- 6) ec2

#### inside the machine:

- 1) sudo su
- 2) yum update -y
- 3) yum update -y && yum install -y yum-utils git jq aws-cli docker bind-utils nano apache-maven java java-1.8.0-openjdk-devel wget unzip bash-completion
- 4) sudo wget -O /etc/yum.repos.d/jenkins.repo <https://pkg.jenkins.io/redhat-stable/jenkins.repo>  
sudo rpm --import <https://pkg.jenkins.io/redhat-stable/jenkins.io.key>  
yum install jenkins -y  
systemctl enable jenkins  
systemctl start jenkins
- 5) attach my volume to the ec2 on console
- 6) lsblk
- 7) mkdir -p /mnt/data
- 8) sudo mkfs -t xfs /dev/xvdf
- 9) sudo mount /dev/xvdf /mnt/data
- 10) systemctl stop jenkins
- 11) chown -R jenkins:jenkins /mnt/data/jenkins/
- 12) cp -r /var/lib/jenkins /mnt/data
- 13) mv /var/lib/jenkins /mnt/data/jenkins
- 14) ln -s /mnt/data/jenkins /var/lib/jenkins
- 15) systemctl start jenkins

#### בעצם מה אנחנו רוצים לעשות?

- 1) אנחנו לא רוצים שכל מה שנעשה ישמר על דיסק על המכונה שלנו אלא אנחנו רוצים לשמור אותו על ווליום חיצוני כדי שבמקרה והמכונה תיפול , הכל יישמר על דיסק חיצוני
- 2) אז אנחנו רוצים בעצם להעביר את תיקיית `var/lib/jenkins` שבה בעצם יש את כל גנטינס אנחנו מעבירים לתיקית `/mnt/data` שהיא מקושרת לווליום שלנו השלבים שלנו הם :
- (1) יוצרים תיקיה
- (2) מאתחלים את הווליום (נעשה את זה רק אם הווליום ריק) `mkfs -t xfs /dev/xvdf`
- (3) נעשה `mount` שייכתב לתוך הדיסק `/dev/xvdf` לתיקית `/mnt/data`
- (4) אכבה את השירות של גנטינס
- (5) אשנה הרשאות לתיקיה שיצרתי כי יצרתי אותה לפני כן עם יודור רוט ואני צריכה לתת ליודור גנטינס גישה לשם
- (6) מעתיקה את התיקיה של גנטינס לתיקיה שמקושרת לווליום
- (7) מעבירה את מה שהיה בתוך `var/lib/jenkins` לתיקיה עם `cp -r /var/lib/jenkins /mnt/data` ולזכור ששם יש לי את הגיבוי שלי
- (8) יוצרים סימבוליק לינק סימבוליק לינק מהמקום האמיתי (איפה שבאמת הקבצים נמצאים) לאיפה אני רוצה שזה יצביע (המקום המזויף)
- (9) יצרתי עוד מכונה אותו הדבר שהיא `slave` עם אותו `awscli` key

### connecting slave to the controller(master) with SSH:

#### THE OLD WAY:

הגדרות אקסקיוטר נמצאות בסיסטם קונפיגוריישן מחברת את `slaves` ואוטומטית כשאני מחברת אותם הוא מנסה לתקשר איתו מאחורי הקלעים הוא עושה פקודה `env` כדי לעזור לנו לדבג

#### in jenkins:

- 1) go to : dashboard > manage jenkins > manage nodes and clouds
- exe: 2
- labels: master
- usage : only build jobs with label expressions matching this node
- 2) go to : dashboard > manage jenkins > manage nodes and clouds new node
- Node name : slave1
- permanently > ok
- remote root directory : /var/jenkins
- Usage : use this node as much as possible
- launch method: SSH > host (private ip of the slave machine)
- create credentials with my SSH key : ID = SLAVE-KEY , USERNAME = EC2-USER
- COPY THE PRIVATE KEY
- Host Key Verification Strategy : NON VERIFYING VERIFICATION STRATEGY
- 3) relaunch

#### into the slave terminal :

- 1) mkdir -p /var/jenkins
- 2) chown ec2-user:ec2-user /var/jenkins
- 3) yum update -y && yum install -y yum-utils git jq aws-cli docker bind-utils nano apache-maven java java-1.8.0-openjdk-devel wget unzip bash-completion

#### THE AUTOMATIC WAY:

#### download plugins:

עכשיו אנחנו נעשה את זה בצורה אוטומטית אנחנו ניתן לגנטינס הוראות אם הוא רוצה להרים עוד סלייב הוא ילך ל `aws` וירידים לעצמו מכונה ויפעיל אותה כסלייב וישמיד אותה לאחר מכן גם

```

1)docker
2)kubernetees
3)amazon ec2
configure cloud :
1)choose ec2
2)
Name: auto-slave
V >> Use EC2 instance profile to obtain credentials
Region : eu-central-1
EC2 Key Pair's Private Key >> connect the key we made before
AMI'S:
Description : amazon-linux
AMI ID : the id ( find in ec2 in aws console )
instance type : T2Micro
Availability Zone : eu-central-1a
Security group names : jenkins-nicole-sg
Remote FS root : /home/ec2-user/jenkins
Remote user : ec2-user
AMI type : unix
Remote ssh port :22
Labels : cloud
Usage: ONLY BUILD ....
host key verification strategy :off
user data:
#!/bin/bash

```

```

yum update -y && yum install -y yum-utils git jq aws-cli docker bind-utils nano apache-maven java java-1.8.0-openjdk-devel wget unzip bash-completion
Number of Executors:2
tags: owner , type , app

```

### connecting the volume automatically the volume to the machine after downtime:

בגלל שבכל פעם נמחקות לי המכונות מה שאנחנו נעשה אנחנו נערוך את הקובץ `/etc/fstab` בכל פעם שהשרת יעלה הוא יעשה מאונט אוטומטי בכל עלייה לווליום שלי במאסטר שלי  
(fstab = file system table) שום דבר אפשר לשחזר כלום ואי אפשר לשחזר שום דבר

```

nano /etc/fstab
/dev/xvdf /mnt/data xfs defaults,nofail 0 0
mount -a >> אוטומטית מתיקית זאת אומרת טעינה אוטומטית זאת אומרת efc/fstab
mount >> בדיקה של השרה האחרונה
init 6 >> ריסטרט

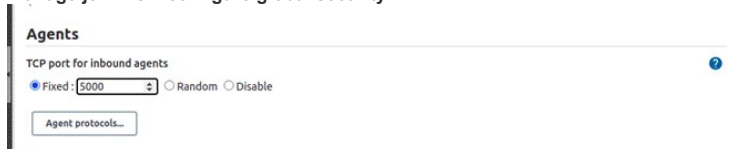
```

### connecting slave to the controller(master) with JNLP:

```

dashbord > manage jenkins > manage nodes and clouds
CREATE NEW NODE
name:JNLP
V:Permanent Agent
Remote root directory: /home/ec2-user/jenkins
manage jenkins > configure global security

```



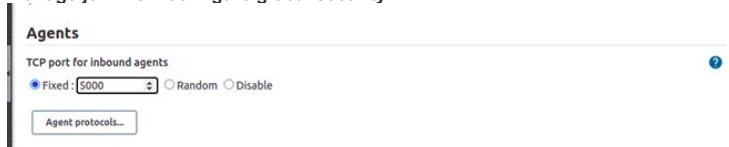
go to the agent in jenkins console:  
download the agent.jar with wget (right click on agent.jar > copy link adress)  
copy the lines (echo...java -jar ...)

### connecting slave to the controller(master) with WEBSOCKET:

```

dashbord > manage jenkins > manage nodes and clouds
CREATE NEW NODE
name:JNLP
V:Permanent Agent
Remote root directory: /home/ec2-user/jenkins
V: use websocket
manage jenkins > configure global security

```



go to the agent in jenkins console:  
download the agent.jar with wget (right click on agent.jar > copy link adress)  
copy the lines (echo...java -jar ...)

# Jenkins

Sunday, 22 August 2021 9:26

בעצם ההבדל בין החיבורים של הסליבים למאסטר הוא בגישה:

- SSH: נקינס מזריק את ההגדרות לתוך הסליב
- JNLP: נרשם מול המאסטר ולוקח ממנו את ההגדרות
- websocket: נרשם מול המאסטר ולוקח ממנו את ההגדרות

## important commands:

מראה לי את המצב הנוכחי של הסרוויס >> `systemctl status name_of_service`

מפעיל סרוויס מסוים >> `systemctl start name_of_service`

להפסיק את הסרוויס >> `systemctl stop name_of_service`

יעלה ישר כשמערכת ההפעלה עולה >> `systemctl enable name_of_service`

יראה לי את כל התהליכים שקרו לי במחשב >> `ps tree -a`

עושה ריפיש בלי קאש >> `ctrl + shift + r`

slave הוא האינסטנס שלי המערכת מיושבת שלי

agent הוא רכיב שיועד להריץ אקסקיוטרים, הוא קובץ `jar`

בנקינס כל הגוים נמצאים בתוך תיקיית `jobs` שכוללת:

`config.xml`

תיקיית `builds` כוללת מידע על הבילדים שלנו בפרוייקט שכוללת בתוכה:

`legacy` ייצוג של הפיינטרים

`permalinks` מידע על מספר בילד וכמות בילדים שהצליחו וכמות שלר הצליחה לדוגמא

יש לנו 2 מקומות שבו נראה מספר בילדים יש ביניהם הבדל

`$JOB$/builds/permalinks` - links for previos builds

`$JOB$/nextBuildNumber` - config for next build

## העתקה של גוב מסויים הכפלה שלו:

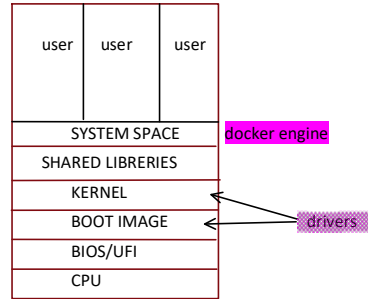
`sudo su`

`cd /mnt/data/jenkins/jobs/name_of_project`

`cp -R -p install_plugin new`

`systemctl restart jenkins.service`

**send box**: סנד בוקס הוא סכיבה בדידה לבהרצה בתוך הסכיבה של המאסטר שאחזק הוא יודע לנקות אותה



איך לראות אינסטנס פרופיל?

instance profile ARNs (amazon unique resource ID) -> iam role

**פונקצית האש:** פונקציית ייצוג - על כל אינפוס שלהת היא מוציאה פלט באורך מוסכם ומוגדר מראש.

**ההבדל בין ריסורס ID וארן:**

amazon resource name

ריסור ID נותן האש קצר מאוד והסבירות ליצירת תוצאה כפולה גבוהה מאוד (colidion)

והארן הוא מזהה יחודי גלובלי. יותר ארוך מהריסורס.

ובתוצאה מכך התאריך הריסורס ID.

הבלס במכונות מסמלים לא רק את השם אלא גם את היכולות שלה. (כמו מכונות של אפל)

`.git` - saves all our changes

Opens in gui only the state that I'm currently at in a commit.

We push to remote the diff between the last commit that was PUSHED and the current PUSH.

`JENKINS_HOME` - is the root dir in the slaves

And in the master it will be the work dir.

**Framework (can be referred to sdk, file of instructions)** - conceptual and technological support structure defined, generally, with specific software artifacts or modules, which can serve as a basis for the organization and development of software.

**Config.xml** - definition of data type

Can be backed up in other places and re-rund later.

**Controller\Master** - The central, coordinating process which stores configuration, loads plugins, and renders the various user interfaces for Jenkins.

**Build** == מופע של ג'וב

**Node\worker\slave** - A machine which is part of the Jenkins environment and capable of executing Pipelines or Projects. Both the Controller and Agents are considered to be Nodes.

**Agent** - An agent is component which typically executes within a machine, or container, which connects to a Jenkins controller and executes tasks when directed by the controller.

**Label** - User-defined text for grouping Agents, typically by similar functionality or capability. For example linux for Linux-based agents or docker for Docker-capable agents.

**Executor** - A slot for execution of work defined by a Pipeline or Project on a Node. A Node may have zero or more Executors configured which corresponds to how many concurrent Projects or Pipelines are able to execute on that Node.

**Cloud** - A System Configuration which provides dynamic Agent provisioning and allocation, such as that provided by the Azure VM Agents or Amazon EC2 plugins.

**Project\Job\pipeline** - A user-configured description of work which Jenkins should perform, such as building a piece of software, etc.

**Step** - A single task; fundamentally steps tell Jenkins what to do inside of a Pipeline or Project.

**Build** - Result of a single execution of a Project

**Core** - The primary Jenkins application (jenkins.war) which provides the basic web UI, configuration, and foundation upon which Plugins can be built.

**Artifact** - An immutable file generated during a Build or Pipeline run which is archived onto the Jenkins Controller for later retrieval by users.

**Downstream** - A configured Pipeline or Project which is triggered as part of the execution of a separate Pipeline or Project.

**Fingerprint** - A hash considered globally unique to track the usage of an Artifact or other entity across multiple Pipelines or Projects.

**Folder** - An organizational container for Pipelines and/or Projects, similar to folders on a file system.

**Item** - An entity in the web UI corresponding to either a: Folder, Pipeline, or Project.

**Jenkins URL** - The main url for the Jenkins application, as visited by a user. e.g. <https://ci.jenkins.io/>

**LTS**

A long-term support Release line of Jenkins products, becoming available for downloads every 12 weeks. See this page for more info.

**Pipeline**

A user-defined model of a continuous delivery pipeline, for more read the Pipeline chapter in this handbook.

**Plugin**

An extension to Jenkins functionality provided separately from Jenkins Core.

**Publisher** - Part of a Build after the completion of all configured Steps which publishes reports, sends notifications, etc. A publisher may report Stable or Unstable result depending on the result of its processing and its configuration. For example, if a JUnit test fails, then the whole JUnit publisher may report the build result as Unstable.

**Resource Root URL** - A secondary url used to serve potentially untrusted content (especially build artifacts). This url is distinct from the Jenkins URL.

**Release** - An event, indicating availability of Jenkins distribution products or one of Jenkins plugins. Jenkins products belong either to LTS or weekly Release lines.

**Stage** - stage is part of Pipeline, and used for defining a conceptually distinct subset of the entire Pipeline, for example: "Build", "Test", and "Deploy", which is used by many plugins to visualize or present Jenkins Pipeline status/progress.

**Trigger** - A criteria for triggering a new Pipeline run or Build.

**Update Center** - Hosted inventory of plugins and plugin metadata to enable plugin installation from within Jenkins.

**Upstream** - A configured Pipeline or Project which triggers a separate Pipeline or Project as part of its execution.

**Workspace** - A disposable directory on the file system of a Node where work can be done by a Pipeline or Project. Workspaces are typically left in place after a Build or Pipeline run completes unless specific Workspace cleanup policies have been put in place on the Jenkins Controller.

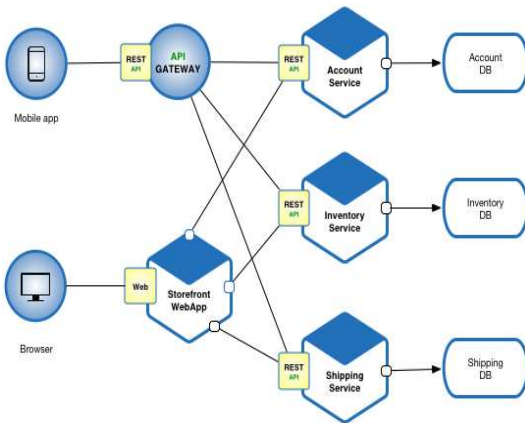
**Create many dirs**: `mkdir -p /etc/{one,two}`

# Micro services

9:17

## Micro services

כל הזמן מפתח וכל הזמן מתמיע - CICD



פירוק מהיר של תקלות

כל חלק קטן שעובר שם, אני יודעת לזהות מתי היתה התקלה ואיפה צריך לשנות בניגוד לגישת המונוליט שאתה לא יודע איפה התקלה וצריך לפרק הכל מההתחלה.  
(הכל מתבצע בתוך אפליקציה אחת) הסייסידי הוא פתרון בגישה של מיקרו סרוויסים.

Faster time to market

החיסרון הגדול במונוליט הוא הזמן של ההוצאה של גרסאות חדשות היה נמוך והיה צורך לזרז תהליכים והגישה של המיקרו סרוויסים קיצרה את זה גם.

**Waterfall**- plan... launch, plan... launch. (looks like a waterfall)- cant launch an update until everyone is ready.  
(monolith approach)

**Agile**- (agility) launch every to weeks with small features (still the monolith approach)

In micro services every service is connected by REST-API (application programming interface)(based on HTTP protocol).  
API- I connect to someone by specific and predictable phrase.

With micro services you can easily horizontally scale. The disadvantage is the high costs.

One more disadvantage of auto scaling is: if you get inside a bug, you can scale the bug and it's a real problem.

# Kubernetes

Sunday, 22 August 2021 13:39

## Kubernetes

<http://devopstrain.pro/k8s-basics/#1> >> מצגת קוברנטים

Desired state- is the final state I always scale to.

Current state- is the state im in now

KUBERNETES give you the outcome of:

**Desired == Current**

## A few common resource types are:

- node (a machine — physical or virtual — in our cluster)
- pod (group of containers running together on a node)
- service (stable network endpoint to connect to one or multiple containers)

## cluster:

### 2 uses for cluster:

fail over cluster >> יהיה לי קלאסטר לגיבוי

distributed processing system >> קבוצת שרתים שאני מרימה שעובדות למען מטרה אחת

### cluster solves 2 problems:

high availability > יכולת לתת מענה בעומס

high scalability > היכולת לגדול כדי לתת מענה לגידול בכמות העבודה

### 2 types of clusters:

-failover cluster: the one you use as backup

-main cluster: group of resources together that depends on the

## הבדל בין קוברנטים לקומפוז:

קוברנטים יודע לעשות אורקסטריציה על קלאסטר (קבוצה של שרתים) וקומפוז יודע להריץ על מחשב אחד בלבד.

המקבילה של קוברנטים היא דוקר סווארם.

אחד היתרונות של קוברנטים זה: automatic bean packing ניהול משאבים משמעותי.

## Commands:

kubectl: the component that runs on all of the machines in your cluster and does things like starting PODs and containers.

kubectl: the command line , talk to your cluster.

Kubectl --kubecongif >> להריץ קוברנטים עם קובץ קונפיגורציה אחר

Kubctl logs -n "name-space name" "name of container" --tail=100

Kubctl scale --current-replicas=2 --replicas=3 deployment/"name" -n kube-system

Kubctl api-resources - כדי לראות ריסורסים של קלאסטר

--context - דגל לראות קלאסטר מרוחק בלי לעבור אליו

## ~/kube/config

קובץ שמגדיר את כל הקלאסטרים שלי קובץ שאני יכולה לערוך ידנית

יש בתוכו 3 חלקים עיקריים:

clusters(1) : רפרנסים של כל הקלאסטרים שלי

users(2) : משתמשים שלי עם האונטיקציה של כל יוזר

contexts(3) :אומר לי לאיזה קלאסטר אני רוצה ללכת ובאיזה יוזר אני רוצה להשתמש ובאיזה נייספייס ועם איזה אוטנטיקציה אני משתמשת כדי להיכנס עם יוזר לתוך הקלאסטר

Kubectl config get-contexts

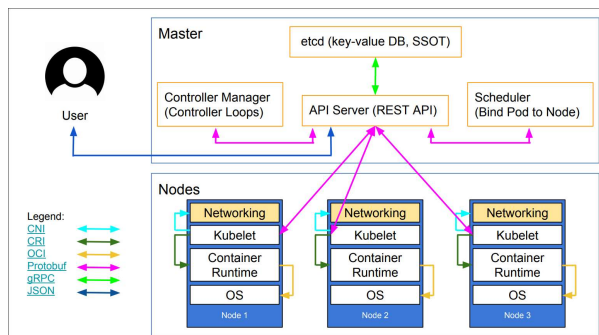
Kubectl config current-context

**K3s**- lite version of kubernetes in local (minikube/ docker-desktop)

**K8s**- 8 segments in the word kubernetes

**K9s**- gui for k8s (like rancher)

**Service mesh** (ISTIO)- היכולת ליצור תקשורת בין סרוויסים בתוך הקלאסטר בעיקר בקוברנטים- (ISTIO)

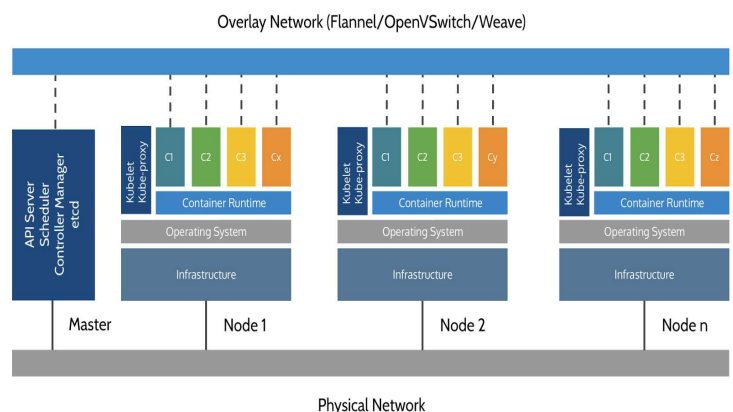


## MASTER:

**Api server** - listening to all requests. ,connect between user and nodes the api server checks that the request came from valid resource and send a request to other processes very secure because we have one endpoint

**Scheduler**- process that checks what node is the most available ,but the kubelet is the one who create the pod

**Controller manager**- detect changes > scheduler > kubelet



**Etcd**-key value store it's the cluster brain all the data stores here , cluster state information information that shares between master and the nodes , all processes rely on this information

#### NODES:

**Kube-proxy** (מגדיר חוקים) כל הנוגדים בין כל הנוגדים ללאד בלנסר בין כל הנוגדים ללאד בלנסר

#### Un-join node from the master:

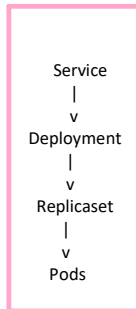
If I joined a node to master and I want to undo my actions: `sudo kubeadm reset` This will un-join the node from the master. Then : `kubectl delete node <name>` It will delete all the pods that are running still.

#### Volumes:

**Volumes** are GLOBAL They don't talk to a Specific namespace

#### Fault tolerance by odd replicas

במערכות מבוזרות שהם סטייטפול אנחנו רוצים לשכפל במספר אי זוגי בשביל שלא יהיה שיוויון כדי שאם תיפול הרשת נוכל להגיע לרוב (משלוח יוצא הרוב)



#### Pod

1. Can have one or multiple containers
2. Runs on a single node  
(Pod cannot "straddle" multiple nodes)
3. Pods cannot be moved  
(e.g. in case of node outage)
4. Pods cannot be scaled  
(except by manually creating more Pods)
5. A Pod is not a process; it's an environment for containers
6. it cannot be "restarted"
7. it cannot "crash"
8. The containers in a Pod can crash
9. They may or may not get restarted  
(depending on Pod's restart policy)
10. If all containers exit successfully, the Pod ends in "Succeeded" phase
11. If some containers fail and don't get restarted, the Pod ends in "Failed" phase

#### Config + Explain commands

# explains what every attribute the kind that I give does.

`kubectl explain (name of kind)`

# add a new user to your kubeconf that supports basic auth

`kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --password=kubepassword`

# permanently save the namespace for all subsequent kubectl commands in that context.

`kubectl config set-context --current --namespace=ggckad-s2`

**NAT**- יודע לקחת את הבקשה ולנתב אותה למקום אחר

הסבר מפורט על נט <https://shushan.co.il/%D7%94%D7%A1%D7%91%D7%A8-%D7%A2%D7%9C-nat-%D7%95%D7%94%D7%92%D7%93%D7%A8%D7%AA-nat-%D7%91cisco-router>

#### Types of services

**ClusterIP**: Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster. This is the default ServiceType

**Node Port**: Exposes the service on each Node's IP at a static port (the NodePort). A ClusterIP service, to which the NodePort service will route, is automatically created. You'll be able to contact the NodePort service, from outside the cluster, by requesting <NodeIP>:<NodePort>

**Load Balancer**: Exposes the service externally using a cloud provider's load balancer. NodePort and ClusterIP services, to which the external load balancer will route, are automatically created

#### Best practices:

עבודה עם לייבלים בשביל בידוד לוגי - Labels

```
git clone https://github.com/otomato-gh/maskshop.git
```

```
Cd maskshop
```

**build 2 image : api & front**

```
Docker-compose build
```

```
docker pull mongo
```

**creating registry deployment**

```
kubectl create deployment registry --image=registry:2
```

```
kubectl expose deploy/registry --port=5000 --type=NodePort >> expose deployment
```

```
kubectl describe svc/registry >> check tat it build
```

```
NODEPORT=$(kubectl get svc/registry -o json | jq .spec.ports[0].nodePort)
```

```
REGISTRY=127.0.0.1:$NODEPORT
```

**tag images and push to registry:**

```
docker tag otomato/maskshop-api $REGISTRY/api >> tag the image
```

```
docker push $REGISTRY/api >> push the image to repository
```

```
docker tag otomato/maskshop-front $REGISTRY/front
```

```
docker push $REGISTRY/front
```

```
docker tag mongo $REGISTRY/mongo
```

```
docker push $REGISTRY/mongo
```

```
curl $REGISTRY/v2/_catalog >> בדיקה שכולם נמצאים
```

**creating deployments from the images**

```
kubectl create deployment mongo --image=$REGISTRY/mongo
```

```
kubectl create deployment api --image=$REGISTRY/api
```

```
kubectl create deployment front --image=$REGISTRY/front
```

```
kubectl expose deployment api --port 80
```

```
kubectl expose deployment mongo --port 27017
```

```
kubectl expose deploy/front --port=80 --type=NodePort
```



# הגדרות נוספות

Sunday, 22 August 2021 9:50

## Auto scale :

in vertical autoscaling the capacity or size of the instance is increased as per demand whereas.

in horizontal autoscaling the number of instances (not the size)

# Helm

Tuesday, 31 August 2021 6:43

Helm is the package manager & repository manager

The package of helm is helm chart

**Difference between package manager and repository package:**

Both statis pages - web servers

**Repository manager:**

index directory that holds pointers (the package that u want is in this url , he points on where the binary of this package)

When we do "helm add" we tell the repository to add another entry to the index directory

הפאקג הזהה בגרסה הזו יושב כאן

**Package manager:**

A package manager or package-management system is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs

# Ops port

Tuesday, 31 August 2021 7:38

Ops port - operation port

- 1) healthiness
- 2) readiness
- 3) monitoring

Metrics- operative=logs of errors, etc. Is there overload? Is there a bottle neck?  
Business efficient= metrics of user usage.

Ready - checks all the dependencies

Healthiness - is it up?, the first to get up

פאז קונטיינר נותן את המערכת האופרטיבית לפוד הוא אחראי למאונטים לקונטיינרים השונים, הוא היישות שמזדהה מול הקרנל, אם נהרג את הפוד הזה אז ייווצר מצב זומבי שהקונטנטים לא מזדהה יותר את הפוד.

## How to change a process?

Config (global (2), user (1), local (0))

Environment variables

Flags (the first that executes)

- עדיפות 2
- עדיפות 1
- עדיפות 0

The process knows how to deliver log files (stdout + stderr)

- Local backups- (תומך אופרטיבי - מאפשר לנהל את הסביבה בצורה נוחה)-
- Log forwarder- listens to all logs from the node and sends them to a central place for aggregation.(daemonset)
- Operation support environment- all of our jobs.
- What Is logical environment? Namespace for example is a logical env that will sit in the cluster. Nodes for instance, are the physical env.
- What is node group? Kubeadm join, labels shows capability. So group allows to manage the env.
- Cluster autoscaler- scales by the latency of the resources.
- Aws stack = ingress controller.
- Inspect in k8s == describe (kubect! describe does NOT exist).
- Afinity otaint = you tell what to run and where.
- Service is some virtual endpoint inside the cluster (lb-external of cluster, clusterIP-shared cluster adress, nodePort- same external port to all nodes in a cluster.
- We want to run as daemonset all the things that are תשתיתיים
- What is init.d? the place that all the servicer are managed at.
- Exeml & json = data structure
- INSIDE THE KUBECONFIG WE HAVE THE CONTEXTS!!
- EE- enterprise edition , CE- commercial edition

# Terraform

Tuesday, 31 August 2021 7:55

Aws sts get-caller-identity : aws מזהה מי אני מבחינת aws

## Why infrastructure as code?

1. Reusable
2. Easy to reproduce
3. Easy to follow changes

## Workspaces

Folder that contains the code  
File ends with: \*.tf \*.tfvars  
Terraform init - boots the workspace

The full slide: <https://hashicorp.github.io/field-workshops-terraform/slides/aws/terraform-oss/index.html#42>

## Anatomy of a Resource

Every terraform resource is structured exactly the same way.

```
resource type "name" {  
  parameter = "foo"  
  parameter2 = "bar"  
  list = ["one", "two", "three"]  
}
```

resource = Top level keyword

type = Type of resource. Example: aws\_instance.

name = Arbitrary name to refer to this resource. Used internally by terraform. This field cannot be a variable.

## Terraform console

Play with commands of terraform in the command prompt

## Concat string to var:

"\${var.my\_var} all my string"

## terraform fmt

Simply run it in a directory containing \*.tf files and it will tidy up your code for you.

-diff = shows what was changed

## Terraform validate

Shows where the syntax is wrong but do not change the file itself.