**DevOps SB**  
**Teacher : Saeed Bostan Doost**

**Start Date: 1404/06/13**

فیلد منیجر ؟؟؟

Kubeflow

Spark

جلسه اول

Cloud Native Applications

متودولوژی : یک سری بست پرکتیس و راه و روش برای پیش بردن و پیاده سازی موفق یک پروژه

Cloud Native : یک سری قوانین برای پیاده سازی اپلیکیشن هایی که در محیط های مختلف قابل اجرا باشند.

12-factor : یک متودولوژی برای طراحی و پیاده سازی برنامه های Cloud Native

1. One code base - ما باید برای پروژه یک مخزن کد داشته باشیم که قابلیت ورژن کنترل و بازگردانی تغییرات وجود داشته باشه و بتونیم اون پروژه رو با همون کد تو محیط های مختلف اجرا کنیم، مثلا git
2. Dependencies - پیش نیازهای هر برنامه باید کاملا مشخص باشه و بتونه توی یک محیط ایزوله اجرا بشه.
3. Configuration - برنامه ها حتما و حتما باید کانفیگ هارو از طریق متغیرهای محیطی سیستم (environment variables) بخونه. خوندن از فایل و ... راه مناسبی نیست و جزو اصول 12-factor نیست.
4. Backing Service - سرویس های دیگری که برنامه نیاز داره باید به صورتی باشه که بتونن اضافه یا حذف بشنو عملکرد برنامه مختل نشه. اتصال برنامه به برنامه ها و سرویس های دیگه از طریق لینک اون برنامه ثانویه باشه

جلسه دوم

Bash Script

اسم متغیرها توی bash به هرشکلی(حروف بزرگ،حروف کوچک،\_ و اعداد) میتونن تعریف بشن به جز حالتی که اسم با عدد شروع نشه.

صدا زدن یک متغیر:

1. $var
2. ${var}

برای پاک کردن متغیر از دستور زیر استفاده میکنیم

unset varName

./sc.sh var1 var2

$1 -> ./sc.sh

$2 -> var1

$3 -> var2

با استفاده از دستور زیر میتونیم تعداد ارگومان های ورودی رو بگیریم

echo $# -> number of args

echo $\* -> shows all args -> creates one string include all args

echo $@ -> shows all arg -> each arg is one string, an array

echo $$ -> shows pid

echo $? -> shows exit code

echo $! -> shows a pid of last process that’s running in background

echo $\_ -> shows last arg that user passed

echo !$ -> shows last command in history

نحوه نمایش دیتای یک ارایه:

arrayName=(val1 val2 val3)  
echo ${arrayName[0]} -> value of index 0 in array  
echo ${arrayName[@]} -> all values in array  
arrayName[1]=newValue  
echo ${#arrayName} -> length of array

IPS=$(hostname -i)  
for x in ${IPS[@]}; do  
 echo $x  
done

تو حالت بالا خط دوم خودش میاد خروجی کامند رو به ارایه تبدیل میکنه

IPS=$(hostname -i)  
for x in “${IPS[@]}”; do  
 echo $x  
done

#!/bin/bash  
A=10  
B=20

C=5  
echo `expr $A + $B`  
echo $(($A + $B)

let C=$A+$B

let A -> یکی کم میکنه

((A--)) -> مثل دستور بالا

[ $A == $B ] -> مقایسه برابری A و B

[ $A -eq $B ] -> مقایسه در bash

[ $A -neq $B ] -> not equal

[ $A != $B ]

[ $A -gt $B ] -> greater than

[ $A -lt $B ] -> less than

[ $A -ge $B ] -> great equal

[ $A -lt $B -a $A -le $C ] -> less than ->, -o = or

-a = and = &&

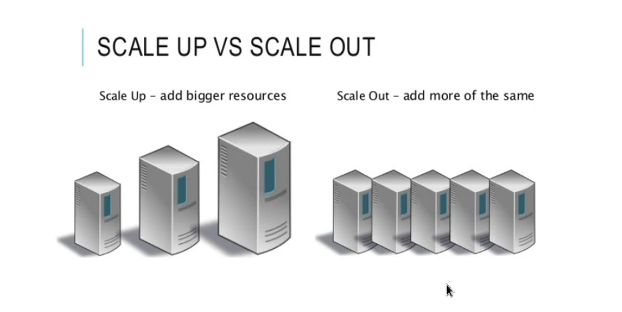
-o = or = ||

زمانی که از حالت اخر استفاده میکنیم باید بزاریم داخل [[ ]] چون سینتکس جدیدیه.

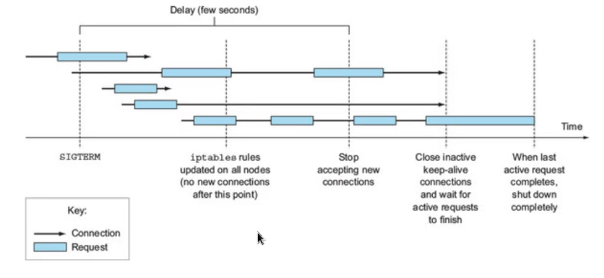
در bash 0 برابر با true و درست اجرا شدن اون برنامس و هر عددی غیر 0 برابر false

ادامه Cloude Native

1. Build, Release, Run - این فازها باید از هم جدا باشن . Build + Config = Release
2. Processes - برنامه باید به صورت یک یا چند برنامه Stateless اجرا بشه. برنامه Stateless در دو لایه میتونه باشه 1. Data management : خودش نباید سرخود دیتا نگهداری بکنه باید با Backing Service ها کار کنه 2.Communication management : اطلاعات ریکوئست ها نباید به هم ربط داشته باشه.ریکوئست اول و دوم ربطی بهم داشته باشند.
3. Port Binding - برنامه ها باید از طریق یک پورت expose بشن.
4. Concurrency - باید به صورت scale out یا horizontal scale بشه.یعنی تعداد رو زیاد میکنیم. پس حتما باید مورد 6 رعایت شده باشه.



1. Disposability - یعنی برنامه سریع استارت بشه و ارتباطی به بقیه سرویس ها نداشته باشه.کار خودش رو باید درست انجام بده.یکی از قابلیت های این اپ ها graceful shutdown یعنی نباید ریکوئست جدیدی بگیره، کارهایی که داشت انجام میداد رو یا تموم کنه یا برگردونه به صف برای اجرای دوباره.موقع اجرا هم باید کارهای قبلیش رو شروع کنه ادامه بده.



1. Dev/Prod Parity - محیط توسعه، استیج، تست و پروداکشن تا حد امکان باید شبیه به هم باشه.
2. Logs - ما سه نوع File descriptor داریم. stdin, stdout, stderr به ترتیب از چپ 0 و 1 و 2 نشانه هاشونه.لاگ های ما حتما باید روی stdout و stderr به صورت استریم ریخته بشن
3. Admin Processes - نیازمند مورد 6 و 8 ، کار مدیریتی یا کنترلی یه صورت جدا اجرا بشه و عملکرد کاربر رو مختل نکنه



جلسه سوم

#!/bin/bash

[ -b /dev/sda ]

* از -b چک کردن اینکه این فایل یه بلاک دیوایس هست یا نه.
* از -c برای کاراکتر دیوایس ها،
* -p برای پایپ لاین فایل ها
* -hو-L برای چک کردن symlink بودن،
* از -S برای چک کردن سوکت بودن فایل.
* -r و-w و -xبرای چک کردن قابلیت خواندن،نوشتن و اجرا.
* -n برای چک کردن تغییر یافتن (modify) فایل.
* از -g و -u برای چک کردن داشتن دسترسی suid و sguid .
* -kبرای چک کردن sticky beat داشتن.
* -Oو -G برای چک کردن owner و group owner .
* چک کردن مقایسه دو فایل از نظر زمان تغییر از -ot و -nt استفاده میشه.
* چک کردن موجود بودن فایل از -e.
* از -s برای چک کردن اینکه یه فایلی حجم داره یا حجمش صفره .
* از -d برای چک کردن داریکتوری بودن. از -f برای چک کردن اینکه ورودی یه فایل معمولیه یا نه.

mkfifo ؟؟؟