29/04/22

To store a statefile :-

Mkdir state
cd state/
vi main.tf
→ paste the code.

terraform init.
terraform plan.
terraform apply --auto-approve.

Mkdir iam
cd iam
vi main.tf.
→ Paste the code (creating user).

terraform init
teraform plan.
terraform apply --auto-approve.

State files are manged in S3, the part of the code is Backend "S3".

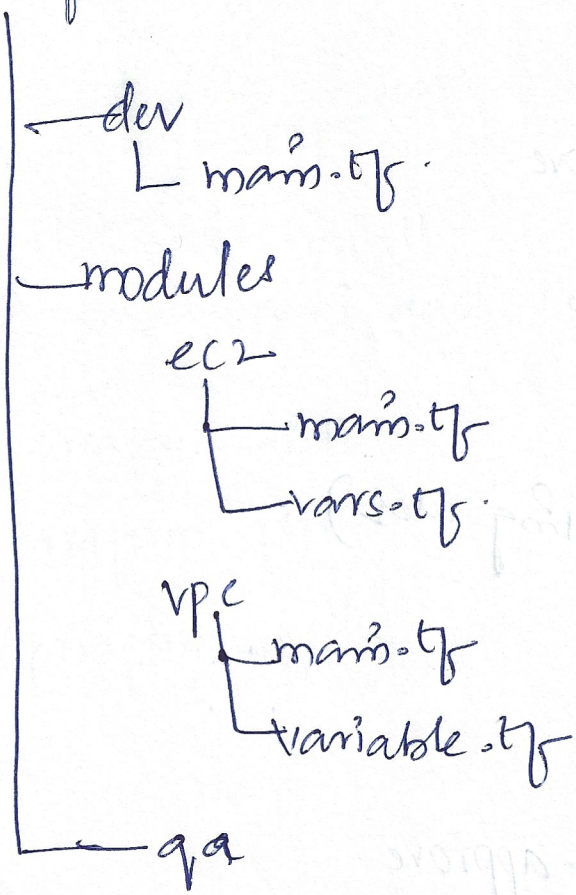
Reusability

Jenkins - shared libraries.

Ansible - Roles.

A.W.S - cloud formation Templates.

Terraform - Modules.



Terraform cloud [Pdf - 40]

→ Server less creation.

02/05/22

Nagios:-

Monitoring Tool for Server

→ Pods which are running inside cluster - prometheus, Grafana.

→ cloud watch - A.W.S.

→ servers - Nagios.

Directory structure:-

/usr/local/nagios/bin - binary files
/usr/local/nagios/sbin - CGI files
/usr/local/nagios/libexec - plugins
/usr/local/nagios/share - PHP
/usr/local/nagios/etc - config files
/usr/local/nagios/var - logs, lock

→ Launch instance on "18"

→ Go with the link in the Pdf

To unlock /

Sudo lsof /var/lib/dpkg/lock.

Sudo lsof /var/lib/apt/lists/lock.

Sudo lsof. /var/cache/apt/archives/lock

⇒ Sudo make install-daemoninit [when we restart the Nagios it will automatically up and run].

04/05/22.

Monitoring Remote Server :-

→ (Pdf 5) link paste in browser.

→ NRPE :- Nagios Remote plugin executor.

→ Launch a server with Ubuntu flavour (18.5 V).

* configure NRPE on the server

* sudo lsof /var/lib/dpkg/lock-frontend → Here we get PID.
* kill -9 2672 → This is PID
* sudo rm /var/lib/dpkg/lock-frontend
* sudo apt update.
* sudo apt-get install nagios-nrpe-server nagios-plugins.
* vim /etc/nagios/nrpe.cfg

→ Go to property "allowed_hosts = -----, Add nagios server Ip

→ For connectivity

* sudo /etc/init.d/nagios-nrpe-server restart

Now. Go to nagios server and check the connectivity.

usr/local/nagios/libexec/check_nrpe -H (remote server Ip)

→ error

cd /usr/local/nagios/libexec

ls

⇒ Since check_nrpe is not present create the plugin.

* Sudo ap-get install nagios-nrpe-plugin

⇒ it is install in plugin location (/usr/lib/nagios/plugins

now, move it

* mv check_nrpe /usr/local/nagios/libexec/

ls

cd

* /usr/local/nagios/libexec/check_nrpe/ -H (server I.P)

⇒ Now, open the ports for the remote server & nagios server

cd

⇒ Now, creating our own local file.

* cd /usr/local/nagios/etc/

* ls.

* touch hosts.cfg service.cfg.

* ls

* vi hosts.cfg.

⇒ Paste pdf pg 23

→ change the server id (Give remote server id).

* cd objects/
* ls
* vi template.cfg.

⇒ add the custom block from pdf 23.

⇒ Now, create rules.

* cd ..
* vi services.cfg.

⇒ Paste the code from pdf.

⇒ Now, add this two files hosts.cfg and services.cfg.
files to Nagios.cfg file
from pdf paste this file path.

cd objects/
ls
vi commands.cfg.

⇒ Add the command block here (since we created check_nrp) we have to configure it

⇒ Now validation to check an corruption of files.

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

* Service nagios restart.

→ creating group

→ vi hosts.cfg

→ Paste the code.

* service nagios restart.

⇒ Other monitoring Tools Data Dog.

05/05/22

Shell Script

Bunch of command placed in a file and executing the file is shell script. {extention ⇒ .sh}.

#!/bin/bash → shebang line.

create any shell script with (.sh) extention and provide the information inside the file then provide the executable permission to the file and execute the file.

We have an alternate way too ⇒ sh (file name).

→ $( ) ⇒ to treate as a command

ex:- $(date) (or) `date` (Back comits)

Arrays:- These variables are scalar variables.

→ Grouping of item is an 'Array'.

When dealing with arrays we use { }

ex ${Name[0]}

To get all the details ${Name[*]} (or) {Name[@]}.

To cleanup values of variable

echo "cleanup------"

Quoting:-

always use (" ")

IFS:- Internal field separator. (its space).

For Arithmatic calculations.

i) $(( )) (or)

ii) EXPR expression (' ') (back comas)

'expr $num1 + num2'

Path expansion:-

ex:- * echo file{1...10}.txt

⇒ to get alternate number.

* echo file{1..:10..2}

→ file1.txt, file3.txt file5.txt.

06/05/22.

Conditions :-

```
if condition
then
command 1
command N
fi
```

```
if condition
then
command 1
else
command
fi
```

Nested if

if in if (C+)

ex:-
```
if condition
then
    if condition
    then
        ----
        do this
    fi
    else
        ---
        do this
    fi
```

if in if (shell)

```
if [ ]; then
  echo
elif [ ]; then
  echo
else [ ]; then
  echo
fi
```

Exit status :-

* echo $?
  - if 1 it is not successful
  - if 0 it is successful

1 Command line Arguements :-

→ $* holds all command line parameters (or) arguments accessed via $1, $2, $3 --- $9.

→ $# holds the number of positional parameters. (No. of. args).

→ $? holds the return value set by previously executed command.

→ $$ process number of shell (id)

→ $0 script name.

→ $@ command line parameters

Case statement :- Alternative of 'if statement'. → Nested.

Loop :- Real Real-time

Para extentions {1 --- 10}.

i) For loop, ii) While loop, iii) until loop

< read
> write.

```
For i
do
echo " $i "
done.
```

init
cond
incr/decr.
[Number of records]

if its false
it will execute
its reverse of
of while loop.

for (init ; cond ; incr/decr)

→ break to stop the loop.

→ continue to continue the loop

Shell script and Python :- 09/05/22

Functions :- create the code and place in function and when ever we want the code we will call the 'Function'.

ex:-
```
hello( )
{
  echo "Hello guys"
}
echo "calling function 1st time"
hello.
```

Saving the time & minimizing the code is 'Function'.

Nested function :- it means calling one function from another function.

variable, loops, condition & functions

→ conditions and loops. Major 'Real time'

Modules is one extra in Devops

connecting Work station to Remote

connecting through ssh mechanism.

cd .ssh/.

ls.

ssh-keygen

On Remote server

vi /etc/ssh/sshd_config

→ #Permit Root login Yes.

Permit Root login Yes

restart sshd.service

cd .ssh/.

ls.

vi authorized_keys.

work station

Cat id_rsa.pub

→ copy this id and paste it in Remote server

authorized_keys

and now connect.

ssh root@ ip address

*cd shellscript/

*ls

*Vi servers

→ Paste Remote server ipaddress

*Vi apache.sh

→ #!/bin/bash

Yum install httpd -y

service httpd start

echo "this is demo of shell scripting" > /var/www/html/index.html

* Vi main.sh → TO Read the file.

→ #!/bin/bash

while read ip

do

echo "======= installing apache on $ip ======="

ssh root@$ip 'bash -s' < apache.sh

echo "===== apache installation completed on $ip ==="

done < servers

* sh main.sh → [for shell execution]

[corey schafer :-] Youtube.

python.

ls

cd python/

ls

python → To Go to python string.

>>> print (variable).

method type :-

→ { } set

→ ( ) tuple

→ [ ] list

Common commands used in GIT

Mkdir → Create a directory.

cd → Change directory.

K. Vamsi

Remote desktop → Programs → ptu

BE - Supplier store - server.war

FE - ROOT.war

(1) Tomcat logs
↓ right click
properties (size)

(2) Task manager → cpu & memory utilization

(3) Services → Tomcat
mongo DB

→ Screen Shot ←

c-drive

TCP

d-drive Back ups
Builds back

2.47 QA

2.51 PRD

Tomcat 8 console

Tomcat8w Service level

workflowsdev.catchemical.com/smartportal-server";

TCL - Tomcat 8

ROOT

B.E

Branch :- (release/tcl-updated)

workspace/~~tcl~~build/libs/0 file

Name in webapps/smartportal-server.war

WEB-INF/classes/conf/app.yml

R.D.P :-

TCL - PRD.

Backups < Mongo DB. ← 1.
Webapps.

After Backup - open 'QA' system.

copy root file and paste on desk top

Add to archive.
↓
select zip
↓
ok.

and then copy this zip file

paste it in PRD system.

then extract it and
open the root file and open index.xml.
and remove dev.

the copy this root folder and paste it in
the tomcat/webapps/root

before ~~then~~ stop tomcat paste it and start tomcat.

and run the url.

Q/A

properties of FE build :-

var baseurl - workflowsdev.tatachemical.com/smartportal-server

TC1 :- → TC1 → microservice.

RS

Smart Store

BE - Smartportal-server

FE - ROOT

Rest - sapapi

ROOT properties :-

var base url -

var rfc base url -

var microserviceurl -

TC1-GA

BE - Smartportal-server

WEB-INF

Class

conf

- app.yml - url - base url / smartstore :- company, ldap.
- app.dev.yml

ldap :- mongo url :- 30123

host : localhost

data base : Smartportal.

base url : workflowdev.tatachemical.

PRD

Smart store - workflows. txt & chemical.

FE Smart store - Web-INF - index.html.

SAPUI5 :- Web-INF

Classes

app.yml

ROOT - FE -

FE - Smartportal server (micro services).

SAPUI5 - Rest

Smart store -

Site 24x7

Monitor tool

com
coramondel
Q/A

FE = ROOT url :- sscportaltest.coromandel.biz/payable-server

BE = payable-server
↓
WEB-INF
↓
class
↓
conf

url: sscportaltest.coromandel.biz/smartstore-server/api/smartdocs.

dev :- mongodb url.
Data base :- spay-prod
port :-
host :-
Base url :- sscportal test.coromandel.biz
mongodb: //35.190.152.165:27017

Sapapi :-
mongodb url :- mongodb:
host :-
port :-
Data base :- spay-prod.

Smart store - server

web app / smart-store-server / web-INF / conf /
↓
app prop / app.yml
app-dev.yml

sscportaltest.comandel.biz/smartstore-server/api

coromandel :- PRD :- payable-server/web-inf/classes/
conf

Url :- sscportal.coromandel.biz/smartstore-server/api/

smartdocs.

apple-dev.yml.

mongodb :-

url :- mongodb://localhost:37123.

host :-

database :- spay.

Root :-

Url :- sscportal.coromandel.biz/payable-server.

Sapapi :- web/say/web-inf/classes.

↓

appli.properties

appli.yml

database :- spay.

Smartstore :- webapp/folder

sscportal.coromandel.biz/smartstore-server

Smartstore :- web-inf/classes/conf

↓

app.yml app-dev.yml

app.port

/smartportal -

smartstore-v5

.appspot.com

Mongo DB :- devops (DB)
Devops data (collection).

use db(name). → To create a DB (or) Enter execute into a already existing DB.

> db.(DB name).insertOne([{" "}])

to insert collections

> db.(DBname).insertMany([{" "},{" "}])

to insert many collections

> db.(DBname).find().pretty()

To see the data in the DB.

CRUD :- Create Read Update Delete

Get mongoDB as output :- (particular DB)

db.(collection name).find({name:" "}).pretty().

Get only MongoDB data as a output with only name field.

db.(CName).find({name:" "}, {name:1}).pretty()

db.(CName).find({name:" "}).pretty.limit()

↓

To find the particular data function.

AZ 104

Core services:-

* Virtual Machines.
* Virtual Networking
* Storage

Virtual Machine:-

* Windows (or) Linux.
* Can be remotely connected using Remote Desktop. (or) ssh.
* Can be placed on virtual network and placed behind "Load balancers"

Abstractions:-

* Azure batch.
* VM scale sets
* AKS.
* Service fabric

App Servics:- (Platform as a Service) (Paas).

* Web apps
* Fully Managed Servers.
* .Net, .Net core, Java, Ruby. Node.js
* No 'RDP'
* Inputs in scaling, continous integration (Github)

Azure Storage :-

* Create storage account up to 5 Peta bytes each.
* Blobs, queues, tables, files
  ↓
  containers.
* Various level of replication included from local to global.
* storage tiers (hot, cool, archive).
* Manage (vms) or unmanaged.

Data services

* Azure SQL Database
* Azure SQL managed instance.
* SQL server on V.M.
* Synapse Analytics.

Microservice :-

* Service Fabric.
* Azure function
* Azure logic apps.
* API Management
* Azure kubernetes service (AKS)

Networking :-

* connectivity
* Security
* Delivery
* Monitoring

Networking connectivity :-

* Virtual Network (vNet).
* Virtual WAN
* ExpressRoute
* VPN Gateway
* Azure DNS
* Peering
* Bastion (more secure version of RDP)

Network Security :-

Network security Groups
Azure private link
DDOS Protection.
Azure firewall
Web application firewall
Virtual Network Endpoints.

Network - Delivery

* CDN
* Azure Front Door
* Traffic Manager
* Application Gateway
* Load Balancer

Monitoring:-

Network watch

express Route monitor

Azure monitor

VNet Terminal Access point (TAP)