

WitZeal - DevOps Engineer - Assessment

Fazil Shaik - Devops Engineer - 2.8 Years of Experience

Assessment -1

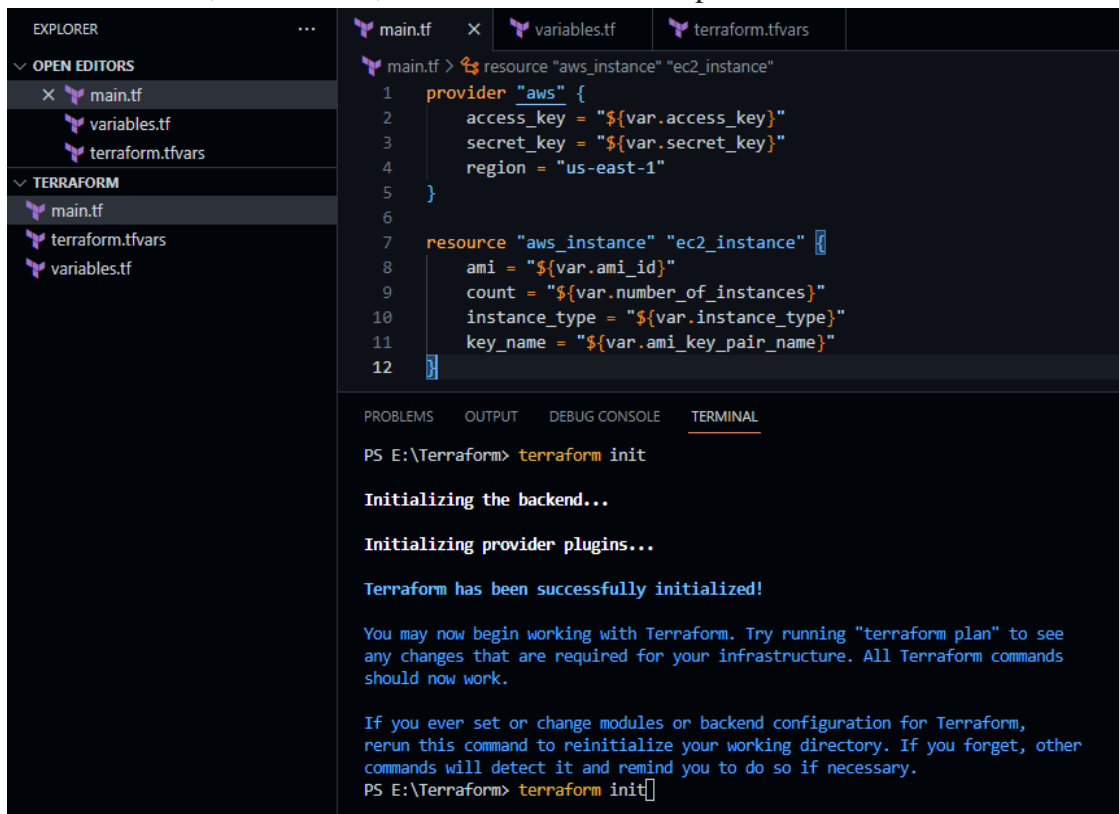
Setup a server with use of terraform and ansible and install and configure MySQL, tomcat on 80 port Memcached Redis and deploy a sample war file

Code Snippets - WorkFlow

1. Terraform - to create the Infrastructure for Ansible configuration
2. Ansible - Will configure the management machines here and Install Tomcat and deploy the sample.war generated from the Maven package manager.
3. Creating an IAM User with Ec2 Full access - This step is to create Management Machines and Configuration Machines for Ansible Configuration.
4. Code has been made available in the GITHUB.

STEPS:

Created Main.tf, Variables.tf, Terraform.tfvars file to provision the infrastructure.



The screenshot displays an IDE interface with three tabs at the top: `main.tf`, `variables.tf`, and `terraform.tfvars`. The `main.tf` tab is active, showing Terraform code for provisioning an AWS EC2 instance. The code includes a `provider "aws"` block with variables for `access_key`, `secret_key`, and `region`, and a `resource "aws_instance" "ec2_instance"` block with variables for `ami`, `count`, `instance_type`, and `key_name`. The left sidebar shows the Explorer view with folders for `OPEN EDITORS` and `TERRAFORM`, listing the files `main.tf`, `variables.tf`, and `terraform.tfvars`. The bottom panel shows the `TERMINAL` view with the output of the `terraform init` command, indicating successful initialization.

```
main.tf > resource "aws_instance" "ec2_instance"
1  provider "aws" {
2      access_key = "${var.access_key}"
3      secret_key = "${var.secret_key}"
4      region = "us-east-1"
5  }
6
7  resource "aws_instance" "ec2_instance" {
8      ami = "${var.ami_id}"
9      count = "${var.number_of_instances}"
10     instance_type = "${var.instance_type}"
11     key_name = "${var.ami_key_pair_name}"
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Terraform> terraform init

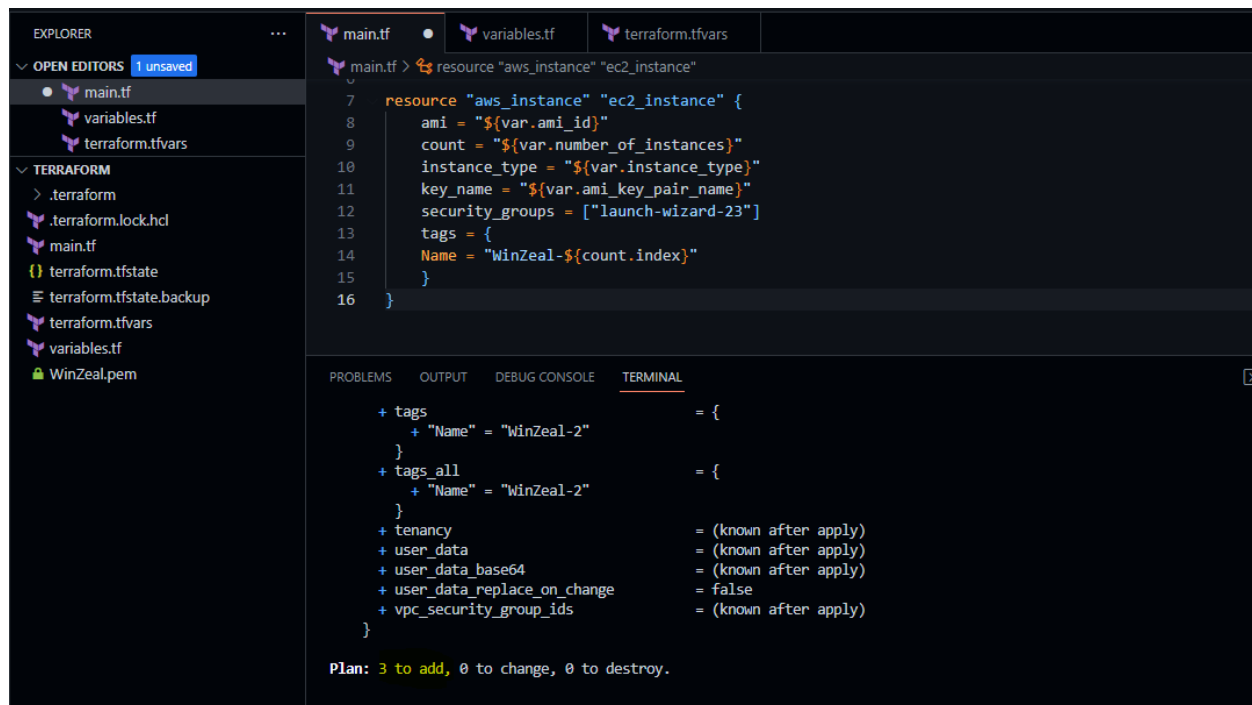
Initializing the backend...

Initializing provider plugins...

Terraform has been successfully initialized!

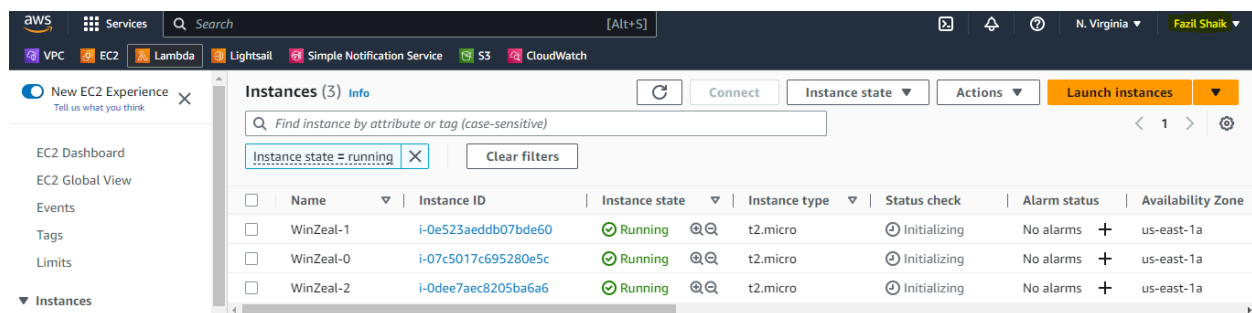
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS E:\Terraform> terraform init
```



Now Considering the WinZeal-1 as the Configuration machine ---- and remaining 2 are Management machines

Manually installing the Ansible in the Configuration machine and therefore by installing the Tomcat in both MM and deploying the .war generated from Maven package manager



Steps for Ansible Installation

```

apt install python3 ---- Python is the prerequisite for ansible
apt update
sudo apt update software-properties-common
sudo apt-add-repository --yes --update ppa:ansible/ansible
sudo apt install ansible

```

We can achieve this step by using provisioners as well.

After installation established the SSH Connection between the Configuration machine and Management machines

Checked the connection using **ping** module

```
root@ip-172-31-95-93:/etc/ansible# ansible all -m ping -i hosts
172.31.88.98 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
172.31.86.137 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

Developed a winzeal.yml , which include the cloning the GIT repository, Maven Installation, Build the project , and Installing the Tomcat, and copying the .war to the webapps folder to expose the project.

```
root@ip-172-31-95-93:/etc/ansible# ansible-playbook winzeal.yml -i hosts
PLAY [CI/CD deployment Script] *****
TASK [Gathering Facts] *****
ok: [172.31.86.137]
ok: [172.31.88.98]
TASK [Cloning the code from git] *****
ok: [172.31.86.137]
ok: [172.31.88.98]
TASK [Installing the maven] *****
changed: [172.31.88.98]
changed: [172.31.86.137]
TASK [check the version] *****
changed: [172.31.86.137]
changed: [172.31.88.98]
TASK [Build the Project] *****
changed: [172.31.86.137]
changed: [172.31.88.98]
TASK [installing the tomcat] *****
ok: [172.31.86.137]
ok: [172.31.88.98]
TASK [renaming the tomcat folder] *****
changed: [172.31.86.137]
changed: [172.31.88.98]
PLAY RECAP *****
172.31.86.137      : ok=7    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.88.98      : ok=7    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Windows
Go to Settings to activate Windows.
```

Edited inbound rules for the port value, We can configure the port value by modifying the port value service.xml in the conf folder of tomcat.

Inbound rules Info						
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-021fc41cdc3f7f9aa	All traffic ▼	All	All	Custom ▼ Q 0.0.0.0/0 ✕		Delete
sgr-0a750a5c7f792ece4	SSH ▼	TCP	22	Custom ▼ Q 0.0.0.0/0 ✕		Delete
sgr-02aa25caa6133f705	HTTP ▼	TCP	80	Custom ▼ Q 27.5.173.18/32 ✕		Delete
Add rule						

As per given I have modified the port value in the /conf/server.xml as the value 80

O/P



This is Testing assessment by winzeal

I haven't got an opportunity to work on the MongoDB and SQL ----- Will learn that configurations as well

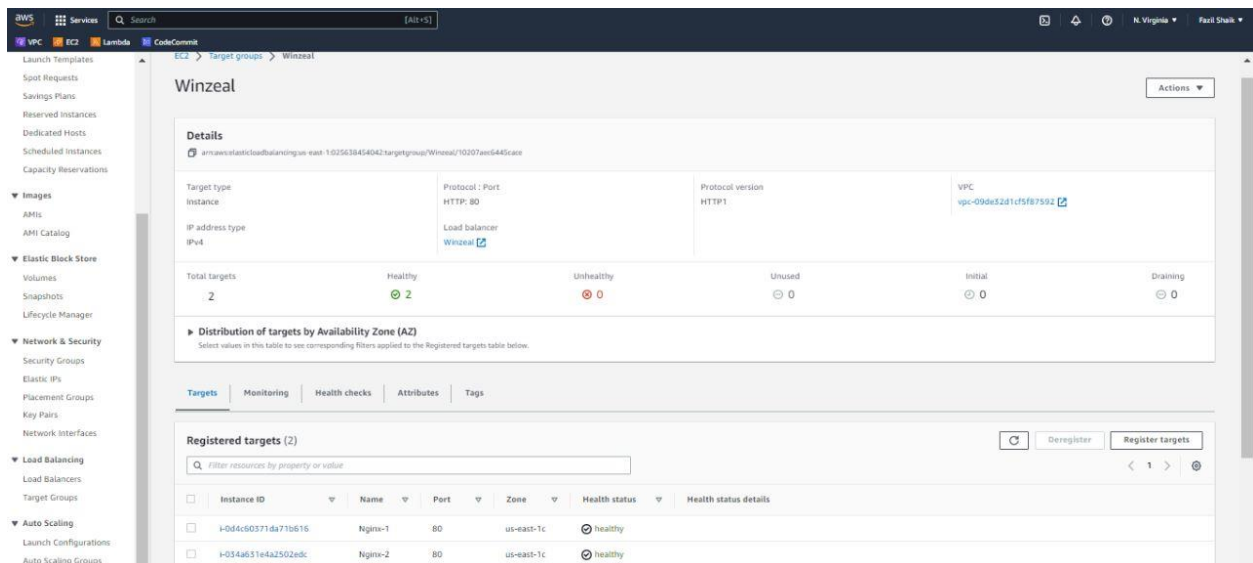
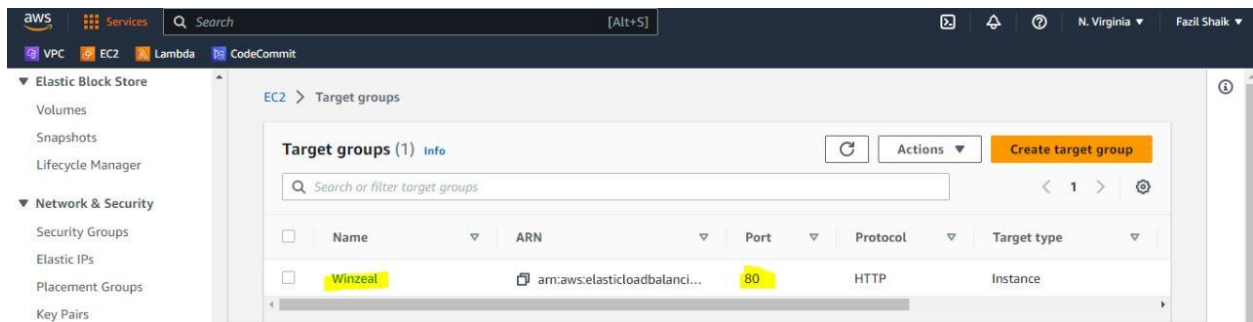
2. Create ALB and add machine and configure health check on 80 port.

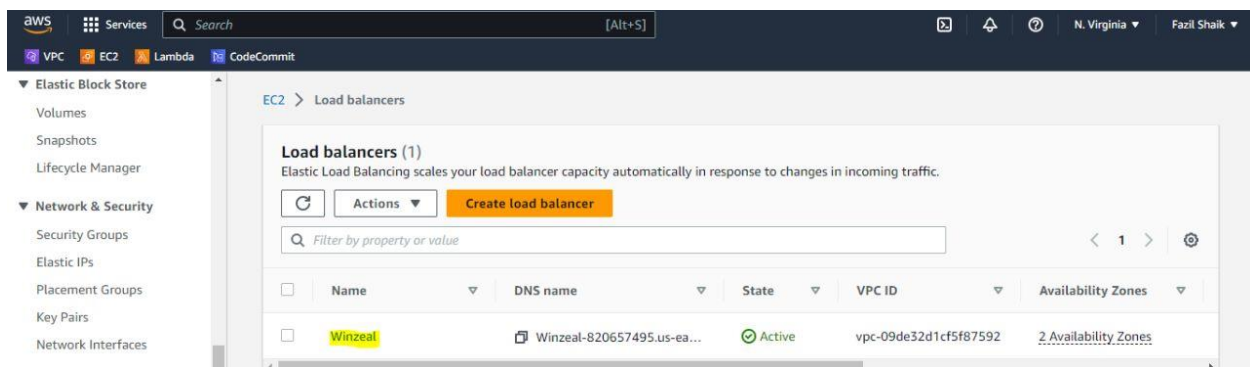
Created 2 instances and installed nginx on those machines- which are configured at the port value of 80

Created a Target group with name “Winzeal” and added these instances in that target group.

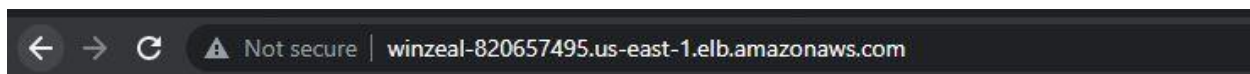
Configured the load balancer with name WinZeal and attached that target group and the Security group configurations to the load balancer.

Checked the status of the machines by health parameter

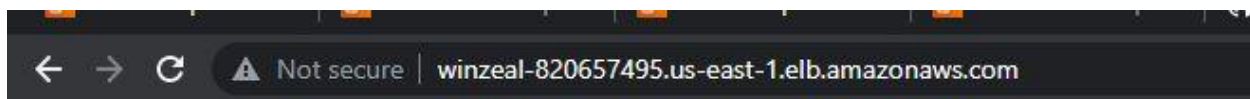




Now tried by connecting the servers by using the DNS name of the ALB – Connected Successfully for the both machines



New Server 1



New Server 2

If same port values is assigned to the different machines or services, We can achieve by this using the listeners and the modification of rules in the Load balancer options.

3. How to check process up and running using ansible.

We can achieve this by creating a playbook with **Service** module, **Shell** module and the O/P can be register through the **Register** and **Debug** modules

Using Service Module – File Name : Winzeal-Service.yml(file has been placed in the GITHUB)

Using Shell Module – File Module : Winzeal-Shell.yml

By Service module, I have installed Nginx server and checked the status of the service

O/P:

```
}
ok: [172.31.91.251] => {
  "data": {
    "changed": false,
    "failed": false,
    "name": "nginx",
    "state": "started",
    "status": {
      "ActiveEnterTimestamp": "Wed 2023-04-19 11:25:49 UTC",
      "ActiveEnterTimestampMonotonic": "5935768558",
      "ActiveExitTimestamp": "n/a",
      "ActiveExitTimestampMonotonic": "0",
      "ActiveState": "active",
```

Using the Shell Module – If the service is up and running, we get the process details like this

```
<172.31.91.251> (0, b'', b'')
ok: [172.31.91.251] => {
  "changed": false,
  "cmd": "ps -ef | grep nginx | grep -v grep",
  "delta": "0:00:00.011387",
  "end": "2023-04-19 11:32:07.610837",
  "invocation": {
    "module_args": {
      "_raw_params": "ps -ef | grep nginx | grep -v grep",
      "_uses_shell": true,
      "argv": null,
      "chdir": null,
      "creates": null,
      "executable": null,
      "removes": null,
      "stdin": null,
      "stdin_add_newline": true,
      "strip_empty_ends": true
    }
  },
  "msg": "",
  "rc": 0,
  "start": "2023-04-19 11:32:07.599450",
  "stderr": "",
  "stderr_lines": [],
  "stdout": "root        6153      1  0 11:25 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;\nwww-data  6154      1  0 11:25 ?        00:00:00 nginx: worker process",
  "stdout_lines": [
    "root        6153      1  0 11:25 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;",
    "www-data    6154      1  0 11:25 ?        00:00:00 nginx: worker process"
  ]
}
<172.31.94.17> (0, b'', b'')
ok: [172.31.94.17] => {
  "changed": false,
  "cmd": "ps -ef | grep nginx | grep -v grep",
  "delta": "0:00:00.011706",
  "end": "2023-04-19 11:32:07.619063",
  "invocation": {
    "module_args": {
      "_raw_params": "ps -ef | grep nginx | grep -v grep",
      "_uses_shell": true,
      "argv": null,
      "chdir": null,
```

If the process is stopped then the module couldn't find the process and gets error and then we have to start the process by either service or shell module again

```
fatal: [172.31.91.251]: FAILED! => {
  "changed": false,
  "cmd": "ps -ef | grep nginx | grep -v grep",
  "delta": "0:00:00.011064",
  "end": "2023-04-19 11:33:24.456817",
  "invocation": {
    "module_args": {
      "_raw_params": "ps -ef | grep nginx | grep -v grep",
      "_uses_shell": true,
      "argv": null,
      "chdir": null,
      "creates": null,
      "executable": null,
      "removes": null,
      "stdin": null,
      "stdin_add_newline": true,
      "strip_empty_ends": true
    }
  },
  "msg": "non-zero return code",
  "rc": 1,
  "start": "2023-04-19 11:33:24.445753",
  "stderr": "",
  "stderr_lines": [],
  "stdout": "",
  "stdout_lines": []
}
<172.31.94.17> (0, b'', b'')
fatal: [172.31.94.17]: FAILED! => {
  "changed": false,
  "cmd": "ps -ef | grep nginx | grep -v grep",
  "delta": "0:00:00.011850",
  "end": "2023-04-19 11:33:24.463540",
  "invocation": {
    "module_args": {
      "_raw_params": "ps -ef | grep nginx | grep -v grep",
      "_uses_shell": true,
      "argv": null,
      "chdir": null,
      "creates": null,
```

References:

GITHUB: <https://github.com/FazilShaik1707/WinZeal---Assesment>