Efi Hadar

IBM EXAM Answers

**BASH**

1. Write a script that takes all .log files in the dir “/var/log/app” and if

they are older than X days (the number of days will be received as an

argument), tar.gz them with the current date suffix and move them to

“/var/log/app/archive”.

#!/bin/bash

[[ ! -z $1 ]] && for x in $(find /var/log/app/\*.log -type f -mtime +$1); do tar -cvf "$x.tar.gz" "$x" && rm -f "$x" ; done || echo "please provide number of days"

2. How will you make the previous section a process running in the

background? (No need to implement it, short explanation will do)

**Cronjob  
Run in background (./test.sh &)**

3. Describe how you will create a single file named “test.sh” on 200 Linux

servers. (You can use any method you’re familiar with)

**With Ansible by running the following “.yml” file**

- name: Create file on linux servers

  hosts: Linux

  tasks:

  - name: Copy the logs script

    ansible.builtin.copy:

      src: /scripts/test.sh

      dest: /scripts/test.sh

      owner: efi

      group: efi

      mode: ‘0755’

**PYTHON**

Read **config.json** file located in:

*/data/python/config.json*

make a GET request to the URL under ‘url’ key and add the first 15 characters

to a key name ‘content’ in the json file.

config.json:

{"url": "https://www.google.com"}

config.json after code run:

{"url": "https://www.google.com", "content": "<first15>"}

Where <first15> should be the first 15 characters from the response.

import json

import requests

response = requests.get('https://google.com')

# JSON data:

with open("/data/python/config.json", "r") as f:

     x = f.read()

# x = "{\"url\":\"https://google.com\"}"

# object to be appended

y = {"content":f"{response.text[:15]}"}

# parsing JSON string:

z = json.loads(x)

# appending the data

z.update(y)

# the result is a JSON string:

# print(json.dumps(z))

with open("/data/python/config.json", "w") as f:

    f.write(json.dumps(z))

**DevOps Tools**

1. Open the attached test.tf file

**what is terraform?**

Terraform is an open-source infrastructure as code software tool that allows DevOps engineers to programmatically provision the physical resources an application requires to run  
Terraform allows users to define their entire infrastructure simply by using configuration files and [version control](https://whatis.techtarget.com/definition/version-control). When a command is given to deploy and run a server, database or [load balancer](https://techtarget.com/searchnetworking/definition/load-balancing), Terraform parses the code and translates it into an application programming interface ([API](https://searchapparchitecture.techtarget.com/definition/application-program-interface-API)) call to the resource provider

**and what this file does?**

The file tells Terraform how tomanage a given collection of infrastructure

**In this file (test.tr) does:**

* Open S3 Bucket
* Create VPC and his Network configuration
* Create Security Group for the VPC (only SSH)
* Create Public key
* Launch 10 EC2 instance (update from 4)
* Instance type change from m5.4xlarge to r6g.4xlarge

**modify the file so it will:**

A. Increase the EC2 server count to 10

B. Change the server type to r6g family type

C. Add new resource of elasticache cluster with 6 Nodes of Redis 6

resource "aws\_elasticache\_cluster" "example" {

  cluster\_id           = "cluster-example"

  engine               = "redis"

  node\_type            = "cache.r6g.large"

  num\_cache\_nodes      = 6

  parameter\_group\_name = "default.redis3.2"

  engine\_version       = "3.2.10"

  port                 = 6379

  tags = {

    Environment = "production"

    Project ="Efi"

  }

}

D. Add tags to all resources new tag name “Project” containing

your name

**Update all of the tags:**

  tags = {

    Environment = "production"

    Project ="Efi"