

Study of Ansible as an automation tool for **Site Reliability**

CSI ZG628T: Dissertation

by

Baskar Balasubramanian

2019HT66015

Dissertation work carried out at: IBM India, Chennai



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)**

April 2021

**Birla Institute of Technology & Science, Pilani
Work-Integrated Learning Programmes Division
Second Semester 2020-2021**

CSI ZG628T : Dissertation Mid-Sem Report

ID No.	: 2019HT66015
NAME OF THE STUDENT	: BASKAR BALASUBRAMANIAN
EMAIL ADDRESS	: 2019HT66015@wilp.bits-pilani.ac.in
STUDENT'S EMPLOYING ORGANIZATION & LOCATION	: IBM India
SUPERVISOR'S NAME	: Swetha J
SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION	: IBM India, Chennai
SUPERVISOR'S EMAIL ADDRESS	: swethaj.iyer@in.ibm.com
DISSERTATION TITLE	: <u>Study of Ansible as an automation tool for Site Reliability</u>

Table of Contents

BACKGROUND.....	4
SITE RELIABILITY.....	4
INTRODUCTION TO THE TECHNOLOGICAL TERMS.....	5
OPERATING SYSTEMS.....	5
COMPUTER NETWORKING.....	6
PROGRAMMING LANGUAGES.....	6
OPEN SOURCE TOOLS.....	7
OBJECTIVE.....	8
WHY CHOOSE ANSIBLE FOR THIS PROJECT?.....	8
SCOPE OF WORK.....	8
USE CASE 1 – AUTOMATION TO REDUCE TOIL WITH ANSIBLE.....	8
USE CASE 2 - CAPTURING THE SYSTEM TCP METRICS WITH PROMETHEUS AND VISUALIZE IT WITH GRAFANA FOR ANALYSIS.....	9
MID-SEM UPDATE.....	10
LAB ENVIRONMENT.....	10
USE CASE 1 – DEPLOYMENT OF TCP ECHO SERVER WITHIN DOCKER CONTAINER.....	10
PROJECT PLAN AND PROGRESS TIMELINE.....	15
LITERATURE REFERENCES.....	16
PARTICULARS OF SUPERVISOR AND ADDITIONAL EXAMINER.....	17
REMARKS OF THE SUPERVISOR.....	17
REMARKS OF THE ADDITIONAL EXAMINER.....	17

BACKGROUND

SITE RELIABILITY

In the current world, there are innumerable number of websites and applications that are hosted in remote computers and accessed by users across the world through Internet that forms a communication medium and backbone of the computer network. A **Site** can be defined as any useful application or software available for use over computer networks which is accessible over the Internet or private interconnected networks.

Users experience in using the sites to access application is based on various parameters, which makes the user to make repeatable use of the websites to realize the benefits of the application. Users have various expectations which include the site to be available whenever they want to access, cater to the need of the user irrespective of the number of concurrent users using the system, irrespective of the geography of the server hosting the application, tolerant response times, etc.

The above parameters are measured in terms of the site's,

- availability (how much time the application is available for use?)
- scalability (how flexible is the system when there is a need to address an increase in the number of users or resource requirements?)
- recoverability (how quickly the system can recover from a failure?)
- maintainability (how effectively application changes can be incorporated?)
- security (What is the level confidentiality and integrity that the system provides to user's data within the systems and the network?).
- elasticity (how robust the system responds to sudden surge or drop in the processing load?).
- economic value (what is the cost savings for the IT service provider?).

From the perspective of the service provider, they would want to ensure they are able to address all the above expectations and many more, to provide the best experience to the end users. In the same way, users of the sites would fall back to the websites that are able to meet their expectations. Such sites are reliable from user experience perspective, which is the primary goal for anyone providing information services. If such reliable services are realized with the hosted websites, then the sites are meant to have an added quality called as **Site Reliability**. The art of practicing the principles to meet the expectations from reliability perspective can be named as **Site Reliability Engineering**.

The following are considered as some of the important principles of Site Reliability engineering

- **Automation** of tasks that are manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows. In IT industry terms this could be called as *eliminating the toil or backlogs*.
- **Measurement and Interpretation of the system data** which is essential in a system that automatically adjusts its resources and configurations, thereby meet the demands of the end users. This could be termed as the *Observability* principle in Site Reliability.
- **Alerting** the support personnel and experts and **effective communication** among them, about the system malfunctioning and take corrective actions for speedy recovery. This functionality is normally categorized as *Event Management or Incident Management* based on the severity of the issue.

In order to practice the above, service providers have to choose the tools wisely, that could be integrated and interfaced with the core systems that serves the user requests. For example there are **automation tools** like *ansible*, *chef*, *puppet*, etc that can automate complex IT system tasks with simple yet feature rich modules. There are open source tools like *Prometheus*, *Logstash*, etc that can **capture the metrics** captured from the functioning system and other open source tools like *Grafana*, *Kibana*, etc that can **interpret the data** captured as useful information for analysis.

INTRODUCTION TO THE TECHNOLOGICAL TERMS

OPERATING SYSTEMS

Linux and it's flavors

Linux is a free software built and ported as the operating system binary for various processor architectures. Initial version of Linux was developed by Linus Torvalds. Linux is one of the largest free software projects actively developed by the Linux community. [5]

Linux System Base (LSB) comprises of kernel and shell utilities that forms the core of linux operating system. There are various flavors of Linux developed on top of the LSB to give added features to the Linux OS. Ubuntu, RedHat, openSUSE, debian are well known Linux flavors to name a few. In this project, redhat flavour of Linux has been used for host machine as well as the virtual machine that hosts the docker container.

Virtualization

Virtualization of operating system means emulating operating systems to share the computing resources like processor, memory, storage, I/O system and networks. Virtualization could be either directly done over the hardware using the drivers named as hypervisors or emulated over a host operating system which manages the system resources.

libvirt

libvirt is a tool kit comprising of set of operating system libraries virtualization management system. There are server side and client side components to it. The server side component manages the virtualized guest operating systems by starting, stopping, pausing, unpausing the virtual OS. The client libraries and utilities that connect to the server side component to issue tasks and collect information about the configuration and resources of the host system and guests. [6][7]

KVM

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc

KVM is open source software. The kernel component of KVM is included in mainline Linux while The userspace component of KVM is included in mainline QEMU. [8]

QEMU

QEMU is a generic and open source machine emulator and virtualizer. When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). When used as a virtualizer, QEMU achieves near native performance by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. [9]

Note: In this project, a RHEL OS is virtualized using qemu-kvm emulator. **qemu-kvm** is an open source virtualizer that provides hardware emulation for the KVM hypervisor. qemu-kvm acts as a virtual machine monitor together with the KVM kernel modules, and emulates the hardware for a full system such as a PC and its associated peripherals. [10]

docker

Docker is written in the Go Programming Language, it takes advantage of several features of the Linux kernel to deliver its functionality. Docker uses a technology called namespaces to provide the isolated workspace, which is called as the container. Docker provides the ability to package and run an application in a loosely isolated container environment. A container is a runnable instance of an image, while image is a read-only template with instructions for creating a Docker container. [20]

COMPUTER NETWORKING

TCP (Transmission Control Protocol)

Internet Engineering Task Force's RFC (RFC793) [11] mentions about TCP as below

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks.

TCP is based on concepts first described by Cerf and Kahn in <https://tools.ietf.org/html/rfc793#ref-1> [11]

The TCP fits into a layered protocol architecture just above a basic Internet Protocol (IP) which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes".

TCP Echo Server

TCP Echo Server is a preliminary application written using C programming language. It demonstrates TCP Server design using socket programming. It uses TCP sockets to listen to the requests from a TCP Echo client.

TCP Echo Client

TCP Echo Client is basically connects to the Echo Server using socket programming with C language.

Note: In this project TCP Echo Server and Echo Client have been developed and compiled using the instructions from the book "The Linux Programming Interface" authored by Michael Kerrisk [12]

PROGRAMMING LANGUAGES

C

C is a popular programming language created by Dennis Ritchie and Ken Thompson of AT&T Bell Laboratories in 1970s. C programs are pre-processed, assembled, compiled and linked to result in an executable, which can be run directly on the processor. Popular operating systems like UNIX and Linux, programming languages like Java and Python uses C programming language. In this project, TCP Echo Server and TCP Echo client have been developed using C programming language and compiled using gcc (GNU Compiler Collection) compiler. [13]

Python

Python is the most popular programming language in the world where the programs are interpreted before executed by the processor. The language is interactive and object-oriented. Python is a free software and distributed under an Open Source license. Python is bundled with a large number of modules which includes the core package and extensions. This project uses Ansible which is built with Python. [14]

OPEN SOURCE TOOLS

Ansible

Ansible is a free software originally written by Michael DeHaan. It is released under the terms of the GPLv3 license. It uses the python interpreter, providing an automation platform for a wide range of modules to automate applications and computing systems and infrastructure deployment and maintenance. Ansible uses SSH for network connections, with no agents to install on remote systems. [1]

Prometheus

Prometheus is a system and service monitoring system. It collects metrics from configured targets at given intervals, evaluates the rule expressions, displays the results, and can trigger alerts when specified conditions are observed [16]. It is 100% open source, available under Apache 2 License and development is completely a community driven project.

Grafana

Grafana is an analytics and visualization web application. It is used to visualize the data captured with monitoring systems in the form of charts and graphs by creating monitoring dashboards [18]. It is used in combination with time-series databases such as InfluxDB, Prometheus, etc to visualize metrics, logs and traces. Written in Go language, Grafana is an open source project [19].

OBJECTIVE

This project concentrates mainly on the SRE principle **eliminating toil using ansible automation**. [1] [4]

The project also involves studying the **interfacing capabilities of ansible with tools that monitor the service level metrics**. [1][2]

WHY CHOOSE ANSIBLE FOR THIS PROJECT?

Using Python as the interpreter, ansible is basically a python module and classified as configuration management tool, supporting traditional IT and cloud based system automation. This helps to automate tasks on the website hosting systems. The extent of automation possible with Ansible seems to be having a larger scope including Observability and event management, against simply using it for configuration management.

- Ansible is chosen for study as it is a free software (released under the terms of the GPLv3 license) with community based development.
- It has a rich set of modules providing extensive scope for experimentation on IT systems.
- Unlike other open source tools, it is a simple automation tool using SSH (Secure Shell) protocol to perform tasks on the remote hosts, without dependency of agent processes on the hosts.
- Interfacing capabilities with monitoring and analytic tools, is another area of interest.
- Ansible modules not only caters to needs of automating, but also interfacing with various tools that orchestrate together to provide infrastructure solutions at large.

This research work provides an opportunity to

- Study the internals of how ansible modules interact with operating system libraries
- Develop an automation solution with ansible and interface the solution with monitoring and analytic tools.
- Practice the principles that eventually results in Site Reliability.

SCOPE OF WORK

The scope of this project is to

- Study, practice and document how Ansible would help to automate system functions and reduce toil the with its configuration management & provisioning modules
- Understand and implement the potential to interface [2] with Prometheus and Grafana for service metrics observability.

There are two broad use cases that would be worked upon as part of the project listed here.

USE CASE 1 – AUTOMATION TO REDUCE TOIL WITH ANSIBLE

Automation of the following with ansible

- Boot a Linux virtual machine from a pre-built image repository
- Install Docker on top of the VM
- Spin a Docker container within the VM
- Host a TCP Echo Server within the container
- Demonstrate TCP Client-Server communication between the containers

USE CASE 2 - CAPTURING THE SYSTEM TCP METRICS WITH PROMETHEUS AND VISUALIZE IT WITH GRAFANA FOR ANALYSIS

- Push infrastructure operational time-series data (eg. TCP connections, state, etc) to Prometheus
- Visualization of observable data (TCP connections and state) with Grafana

Assumptions for the use cases:

1) Use case assumes the availability of the operating system image in the local system storage. The pre-built OS image is used to create the VM for the experiments. Building an image from the scratch is not within the scope in this project. However automation for building an image is a prospective extension to the project in the future. The automation only involves managing the life-cycle of the operating system image.

2) Network interfaces are well defined and configured on both host and virtual operating systems., so that the development work in the project involves only at the application layer and TCP layers of the Network protocol stack.

MID-SEM UPDATE

LAB ENVIRONMENT

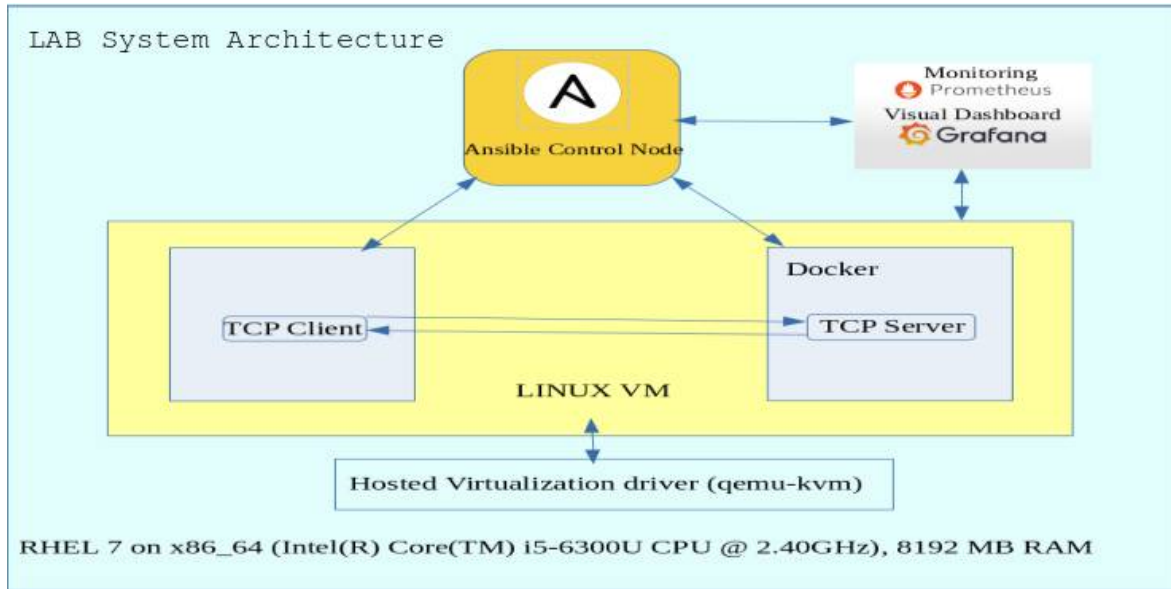


Fig 1. LAB SYSTEM

The figure above shows the system architecture of the lab environment which has been created for project activities. The lab host is a Linux platform (RedHat flavour) on x86_64 processor. The virtualization driver used for the lab environment is qemu-kvm. In the lab environment the virtual machine is also a Linux VM with RedHat flavour. The TCP Echo Server would be started when the Docker container using a centos container image is started by Ansible control node. A TCP Echo Client would be communicating with Echo Server using TCP sockets. Ansible Control node would use SSH to connect to the VM to deploy the TCP applications and operate the docker container. Prometheus and Grafana will be installed to measure and monitor the TCP statistics.

USE CASE 1 – DEPLOYMENT OF TCP ECHO SERVER WITHIN DOCKER CONTAINER

As mentioned in the scope of work, **step 1 of the use case 1** in the project work involves starting a VM using ansible. Ansible's virt module can make use of the libvirt library and use qemu-kvm driver to start virtual machines.

Step 2 of the use case 1 involves installing a docker container which hosts the TCP Echo server. The docker images used in the project includes

- centos container image and
- Container application for TCP Echo server which hosts within the centos container image

Step 3 of the use case 1 starts the TCP Echo server within the docker container and use a TCP echo client to communicate with the TCP Echo server.

Ansible would be used to

- install docker and control the docker operations
- Start the TCP Echo Server within the docker container
- Stop and remove the container and the images as needed

The following screenshots demonstrates the Use Case 1 where a TCP Echo server is hosted and started within a docker container within the RHEL virtual machine. A shell script named as *vm_play.sh* is run, which internally calls a series of ansible command line tasks and playbooks to complete the execution.

A TCP client process is started within the VM connects to the TCP server to the IP address of the container and a specific port (4305). It is shown from the demonstration that the TCP Echo Client is able to send messages to the Echo Server and the Echo Server is able to echo back the message to the Echo client. With this Use Case 1 forms the foundation for the Use Case 2 where TCP statistics would be measured and the data collected and used for interpretation.

```
[baskar@oc6761813003 ansible]$ ./vm_play.sh
*****
Use Case - hosting a TCP Echo server within docker container using Ansible.
*****
Starting the virtual machine from the pre-built image
Hosted Virtualization driver - qemu-kvm
VM OS - RHEL 7
*****
IP4 Address of the RHEL VM is 192.168.122.142

*****
ping_pong check for RHEL VM
*****
pinging the 192.168.122.142 rhel vm. please provide your password for SSH login
Using /etc/ansible/ansible.cfg as config file
SSH password:
192.168.122.142 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

Installing Docker image, starting a docker container with TCP Echo Server hosted and started within the docker container is demonstrated below.

```
*****
Install docker image hosting a TCP Echo server
*****
Installing docker on the rhel vm 192.168.122.142. Provide your password for SSH login
SSH password:
BECOME password[defaults to SSH password]:
192.168.122.142 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "changes": {
    "installed": [],
    "updated": []
  },
  "msg": "",
  "obsoletes": {
    "iwl7265-firmware": {
      "dist": "noarch",
      "repo": "@el76",
      "version": "22.0.7.0-69.el7"
    },
    "urw-fonts": {
      "dist": "noarch",
      "repo": "@base/$releasever",
      "version": "2.4-16.el7"
    }
  },
  "rc": 0,
  "results": [
    "All packages providing docker are up to date",
    ""
  ]
}
```

```

Starting docker.service on 192.168.122.142. Provide SSH password for authentication
SSH password:
BECOME password[defaults to SSH password]:

PLAY [test systemd services] *****

TASK [Gathering Facts] *****
ok: [192.168.122.142]

TASK [Populate service facts] *****
ok: [192.168.122.142]

TASK [Print service facts] *****
ok: [192.168.122.142] => {
  "ansible_facts.services[ \"docker.service\" ]": {
    "name": "docker.service",
    "source": "systemd",
    "state": "running",
    "status": "disabled"
  },
  "changed": false
}

TASK [Check service status of "docker.service" before "start"] *****
[WARNING]: conditional statements should not include Jinja2 templating delimiters such as {{ }} or {% %}. Found: ansible_facts.services[ "{{ service_name }}" ].name
== "{{ service_name }}"
changed: [192.168.122.142]

TASK [Perform the command "start" on the service with systemctl] *****
[WARNING]: conditional statements should not include Jinja2 templating delimiters such as {{ }} or {% %}. Found: ansible_facts.services[ "{{ service_name }}" ].name
== "{{ service_name }}"
changed: [192.168.122.142]

TASK [Check service status of "docker.service" once again after "start"] *****
[WARNING]: conditional statements should not include Jinja2 templating delimiters such as {{ }} or {% %}. Found: ansible_facts.services[ "{{ service_name }}" ].name
== "{{ service_name }}"
changed: [192.168.122.142]

PLAY RECAP *****
192.168.122.142: 1 ok, 0 changed, 0 unreachable, 0 failed, 0 skipped, 0 rescued, 0 ignore=0

```

Copying docker files for TCP Echo Server and Client and performing docker operations.

```

Copying the docker files for TCP server to rhel vm. Provide SSH password for authentication
SSH password:
192.168.122.142 | CHANGED => {
  "changed": true,
  "dest": "/home/baskar/ansible/",
  "src": "/home/baskar/ansible/TCP"
}

Cleaning existing docker containers and images
SSH password:
BECOME password[defaults to SSH password]:
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (command). Using last
defined value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (args). Using last defined
value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (when). Using last defined
value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (ignore_errors). Using
last defined value only.

PLAY [docker operations playbook] *****

TASK [Gathering Facts] *****
ok: [192.168.122.142]

TASK [build docker image for centos] *****
skipping: [192.168.122.142]

TASK [build docker image for tcpserver] *****
skipping: [192.168.122.142]

TASK [Run docker container for tcpserver] *****
skipping: [192.168.122.142]

TASK [stop docker container for tcpserver] *****
changed: [192.168.122.142]

TASK [remove docker container for tcpserver] *****
changed: [192.168.122.142]

```



```

Performing docker operations to pull 'centos' container image, build 'tcpserver' container image and run 'tcpserver_1' container'. Provide SSH password for authentication
SSH password:
BECOME password[defaults to SSH password]:
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (command). Using last defined value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (args). Using last defined value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (when). Using last defined value only.
[WARNING]: While constructing a mapping from /home/baskar/ansible/job_docker_operations.yml, line 11, column 13, found a duplicate dict key (ignore_errors). Using last defined value only.

PLAY [docker operations playbook] *****

TASK [Gathering Facts] *****
ok: [192.168.122.142]

TASK [build docker image for centos] *****
fatal: [192.168.122.142]: FAILED! => {"changed": true, "cmd": ["/usr/bin/docker", "pull", "centos"], "delta": "0:00:32.321822", "end": "2021-02-24 02:35:44.907677", "msg": "non-zero return code", "rc": 1, "start": "2021-02-24 02:35:12.585855", "stderr": "error pulling image configuration: Get https://registry-1.docker.io/v2/library/centos/blobs/sha256:300e315adb2f96afe5f0b2780b87f28ae95231fe3bdd1e16b9ba606307728f55: net/http: TLS handshake timeout", "stderr_lines": ["error pulling image configuration: Get https://registry-1.docker.io/v2/library/centos/blobs/sha256:300e315adb2f96afe5f0b2780b87f28ae95231fe3bdd1e16b9ba606307728f55: net/http: TLS handshake timeout"], "stdout": "Using default tag: latest\nTrying to pull repository docker.io/library/centos ... \nlatest: Pulling from docker.io/library/centos\n7a0437f04f83: Already exists", "stdout_lines": ["Using default tag: latest", "Trying to pull repository docker.io/library/centos ... ", "latest: Pulling from docker.io/library/centos", "7a0437f04f83: Already exists"]}
...ignoring

TASK [build docker image for tcpserver] *****
changed: [192.168.122.142]

TASK [Run docker container for tcpserver] *****
changed: [192.168.122.142]

TASK [stop docker container for tcpserver] *****
skipping: [192.168.122.142]

```

```

TASK [stop docker container for tcpserver] *****
skipping: [192.168.122.142]

TASK [remove docker container for tcpserver] *****
skipping: [192.168.122.142]

TASK [sleep for 5 seconds for stop docker container for tcpserver to complete] *****
skipping: [192.168.122.142]

TASK [remove docker image tcpserver] *****
skipping: [192.168.122.142]

TASK [sleep for 5 seconds for remove docker image tcpserver to complete] *****
skipping: [192.168.122.142]

TASK [remove docker image centos] *****
skipping: [192.168.122.142]

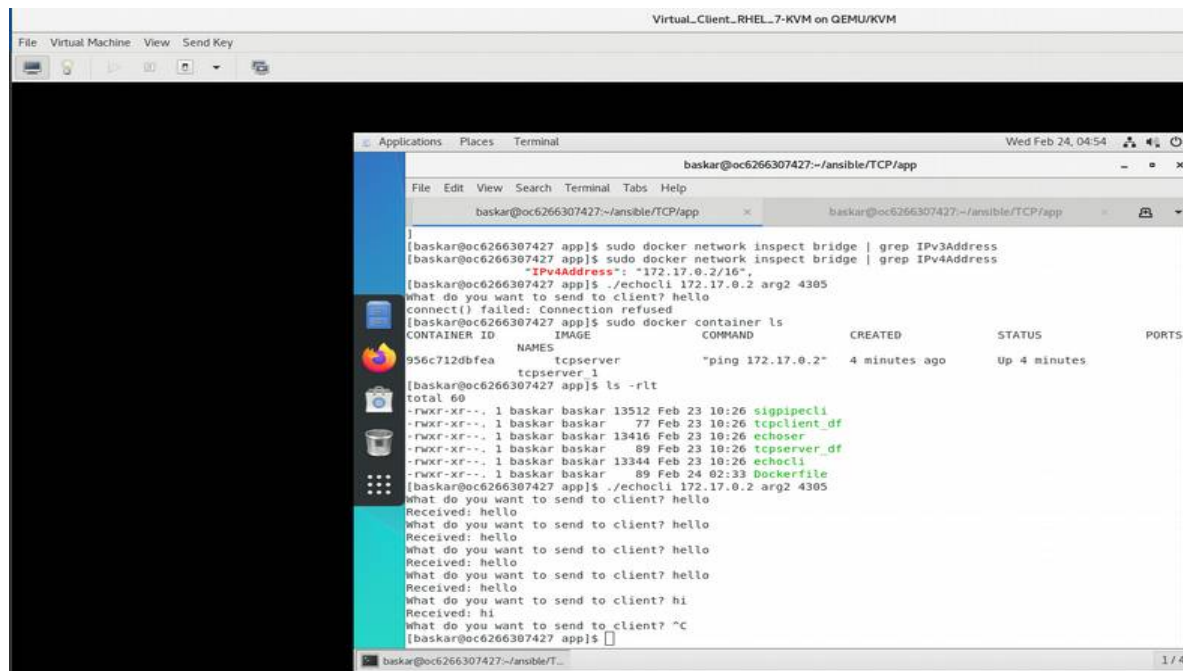
TASK [sleep for 5 seconds for remove docker image centos to complete] *****
skipping: [192.168.122.142]

PLAY RECAP *****
192.168.122.142      : ok=4    changed=3    unreachable=0    failed=0    skipped=7    rescued=0    ignored=1

*****
Use Case completed for hosting a TCP Echo server within docker container.
TCP Echo server is ready to listen on port 4305
Use echocli (TCP Echo client) to communicate with TCP Echo server.
*****
[baskar@oc6761813003 ansible]$ █

```

TCP Echo Client program execution is demonstrated below.



The screenshot shows a Virtual Machine window titled "Virtual_Client_RHEL_7-KVM on QEMU/KVM". Inside the VM, a terminal window is open with the prompt "baskar@oc6266307427:~/ansible/TCP/app". The terminal shows the following commands and output:

```
[baskar@oc6266307427 app]$ sudo docker network inspect bridge | grep IPv4Address
[baskar@oc6266307427 app]$ sudo docker network inspect bridge | grep IPv4Address
"IPv4Address": "172.17.0.2/16",
[baskar@oc6266307427 app]$ ./echocli 172.17.0.2 arg2 4305
What do you want to send to client? hello
connect() failed: Connection refused
[baskar@oc6266307427 app]$ sudo docker container ls
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS              PORTS
956c712dbfea       tcpserver          "ping 172.17.0.2"      4 minutes ago      Up 4 minutes
[baskar@oc6266307427 app]$ ls -rlt
total 60
-rwxr-xr--. 1 baskar baskar 13512 Feb 23 10:26 sigpipecli
-rwxr-xr--. 1 baskar baskar 77 Feb 23 10:26 tcpclient_df
-rwxr-xr--. 1 baskar baskar 13416 Feb 23 10:26 echoser
-rwxr-xr--. 1 baskar baskar 89 Feb 23 10:26 tcpserver_df
-rwxr-xr--. 1 baskar baskar 13344 Feb 23 10:26 echocli
-rwxr-xr--. 1 baskar baskar 89 Feb 24 02:33 Dockerfile
[baskar@oc6266307427 app]$ ./echocli 172.17.0.2 arg2 4305
What do you want to send to client? hello
Received: hello
What do you want to send to client? hello
Received: hello
What do you want to send to client? hello
Received: hello
What do you want to send to client? hi
Received: hi
What do you want to send to client? ^C
[baskar@oc6266307427 app]$
```

BENEFITS OF AUTOMATING THE VM AND DOCKER OPERATIONS USING ANSIBLE

Automation of deploying the TCP Echo Server in the docker container within a VM is a significant improvement over the manual installation of VM followed by installing the docker container and then deploying the TCP Echo server within the container.

This effort would help to spend more time on the actual project work related to monitoring, data gathering and analysis, drastically reducing the time to deploy the TCP Echo Server application compared to manual deployment.

PROJECT PLAN AND PROGRESS TIMELINE

(STATUS AS ON 24th Feb, 2021)

Legend:

Completed Work (Marked in Green)

In progress (Marked in Orange)

Nearing Deadline (Marked in Red)

Removed (and) Changed (Striked out))

Planned (Default)

Setting up a lab environment	14 th Feb, 2021
Kick-off meeting with BITS Faculty	17 th Feb, 2021
Implementing Infrastructure as Code with Ansible	22 nd Feb, 2021
- Create/define, start, stop Linux VM with ansible	16 th Feb, 2021
- Install and start Docker containers with ansible	18 th Feb, 2021
- Install and start second Docker container with ansible	18th Feb, 2021
- Install a TCP Client in Docker container 1 in Linux VM	20 th Feb, 2021
- Install a TCP Server in Docker container 2	21 st Feb, 2021
- View TCP connection status for TCP Client and Server	22 nd Feb, 2021
Documentation work for Mid-semester progress report	27 th Feb, 2021
Review with Supervisor	24 th Feb, 2021
Review with Additional Examiner	24 th Feb, 2021 → 01 st Mar, 2021
Meeting with BITS faculty	24 th Feb, 2021 → 3 rd Mar, 2021
Submit mid-sem report	28 th Feb, 2021
Implementing Observability with Prometheus, Grafana and Ansible	28 th Mar, 2021
- Install and configure Prometheus	28 th Feb, 2021
- Install and configure Grafana	28 th Feb, 2021
- Create Ansible Interface to Prometheus / Grafana	14 th Mar, 2021
- Demonstrate TCP connections status with Ansible/Prometheus/Grafana integration	21 st Mar, 2021
Implementing ChatOps with Slack (Check for the feasibility)	28 th Mar, 2021
Review with Supervisor and Additional Examiner	22 nd - 26 th Mar, 2021
Documentation work for Dissertation	11 th Apr, 2021
Review with Supervisor and Additional Examiner	12 th - 15 th Apr, 2021
Final Documentation work for Dissertation and Upload of softcopy	18 th Apr, 2021
Final Presentation	23 rd Apr, 2021

LITERATURE REFERENCES

- [1] Ansible references – <https://github.com/ansible/ansible>, <https://docs.ansible.com/>
- [2] Integration with DevOps tools: <https://www.ansible.com/integrations/devops-tools>
- [3] Site Reliability Engineering principles and practises – <https://sre.google/sre-book/part-II-principles/>,
<https://sre.google/sre-book/part-III-practices/>
- [4] Eliminating Toil - <https://sre.google/sre-book/eliminating-toil>
- [5] GitHub page for Linux - <https://github.com/torvalds/linux>
- [6] Linux manual pages - <https://www.kernel.org/doc/man-pages/>
- [7] libvirt reference - <https://libvirt.org/>
- [8] Linux KVM website - https://www.linux-kvm.org/page/Main_Page
- [9] QEMU wiki - https://wiki.qemu.org/Main_Page
- [10] QEMU website - <https://www.qemu.org/>
- [11] TCP RFC 793 - <https://tools.ietf.org/html/rfc793>
- [12] The Linux Programming Interface book by Michael Kerrisk - <https://www.man7.org/tlpi/>
- [13] GNU Compiler Collection (GCC) webpage - <http://gcc.gnu.org/onlinedocs/gcc/>
- [14] Python website - <https://www.python.org/>
- [15] Visualize Prometheus data with Grafana - <https://www.openlogic.com/blog/how-visualize-prometheus-data-grafana>
- [16] GitHub page for Prometheus - <https://github.com/prometheus/prometheus>
- [17] Prometheus web site - <https://prometheus.io/>
- [18] Wikipedia page for Grafana - <https://en.wikipedia.org/wiki/Grafana>
- [19] GitHub page for Grafana - <https://github.com/grafana/grafana>
- [20] Overview of Docker - <https://docs.docker.com/get-started/overview>

PARTICULARS OF SUPERVISOR AND ADDITIONAL EXAMINER

Supervisor's Name: Swetha J
Supervisor's Email: swethaj.iyer@in.ibm.com
Supervisor's Qualification: M.S in Systems Engineering from BITS, Pilani
Supervisor's Designation & Address: SAP Technical Architect at IBM with 13 years of experience in SAP and Infrastructure.

Additional Examiner's Name: Raghu Srinivasan
Additional Examiner's Email: sriniv@us.ibm.com
Additional Examiner's Qualification: B.E. Instrumentation and Technology
15 years of experience as IT Architect
6 years of experience as Senior Technical Staff Member
Additional Examiner's Designation & Address: Senior Technical Staff Member (STSM)- Technology Architect
Lead Client Transformation SRE (Site Reliability Engineer)
Progressive Hybrid Services Integration, IBM Services

REMARKS OF THE SUPERVISOR

Completed initial review of the project work till mid-sem was completed. Mid-Sem report document review is still pending.

REMARKS OF THE ADDITIONAL EXAMINER

Review meeting scheduled with Additional examiner on 1st Mar, 2021

Signature of Student

Signature of Supervisor

Signature of Additional Examiner
