DEVOPS EXP 1 TO 7

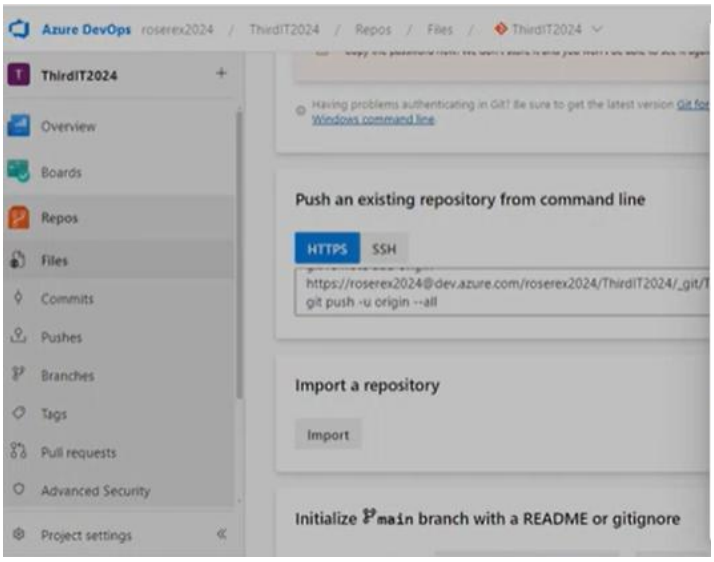
**EXPERIMENT 1**

**AIM :** Create Maven Build pipeline in Azure

**ALGORITHM :**

1. Install Java and Maven and set the system environment variables
2. Create a github account
3. Create an Azure student login and enter into Azure DevOps
4. Install a Java IDE
5. Develop a Java Maven Project and build the application in the local server
6. Push the java maven project into Git Repository
7. Create Azure DevOps organization and import or clone the git repo of the java maven project
8. Create a pipeline in Azure and build the project

**OUTPUT**




**RESULT** : Thus, a maven project was created and build a pipeline in Azure

**EXPERIMENT 2**

**AIM :** To create a maven project and build a pipeline in Azure and create test cases to run regression test

**ALGORITHM :**

1. Create a Maven Project:
2. Azure DevOps Pipeline Configuration:
3. Create a new pipeline using the starter template.
4. Modify the pipeline YAML file to include the necessary tasks.
5. Define the trigger (e.g., on each commit to the master branch).
6. Specify the VM image (e.g., 'Ubuntu-16.04').
7. Add tasks to build, test, and deploy your project. For testing, you'll need to run your
8. JUnit tests.
9. Run Tests Automatically:
10. View Results:
11. Publish Build Artifacts:
12. To publish build artifacts, add the following task to your pipeline YAML:

**Program :**

**pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>demoapp</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

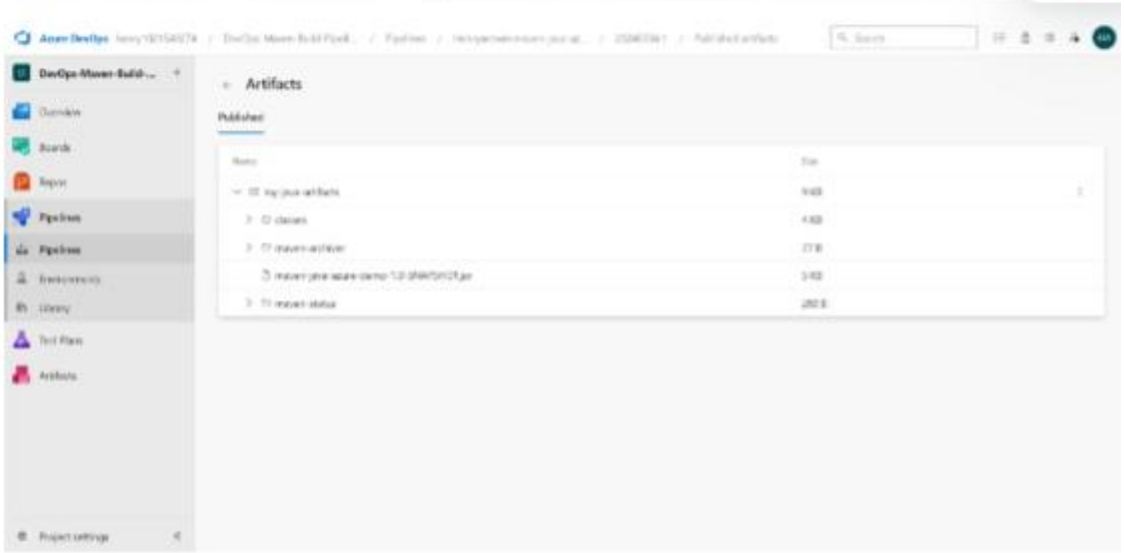
**Java Program**

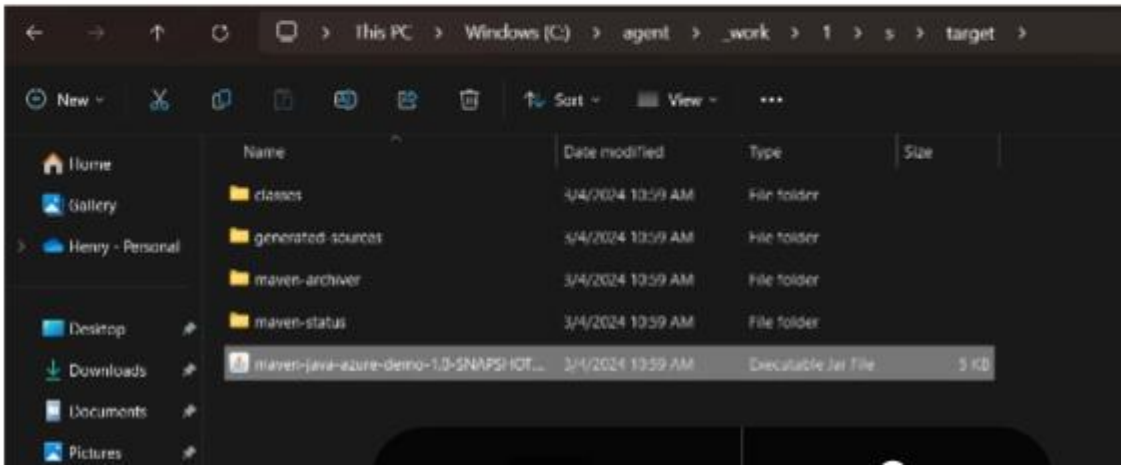
```
package com.example;

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    public int divide(int a, int b) {
        if (b == 0) throw new ArithmeticException("Cannot divide by zero");
        return a / b;
    }
}
```

**CalculatorTest.java**

```
package com.example;
import org.junit.Test;
import static org.junit.Assert.*;

public class CalculatorTest {
    Calculator calc = new Calculator();

    @Test
    public void testAdd() {
        assertEquals(5, calc.add(2, 3));
    }

    @Test
    public void testSubtract() {
        assertEquals(1, calc.subtract(4, 3));
    }

    @Test
    public void testMultiply() {
        assertEquals(6, calc.multiply(2, 3));
    }

    @Test
    public void testDivide() {
        assertEquals(2, calc.divide(6, 3));
    }
    @Test(expected = ArithmeticException.class)
    public void testDivideByZero() {
        calc.divide(10, 0);
    }
}
```

**OUTPUT:**


**RESULT : Thus a regression test cases are executed using Maven Build pipeline in Azure**

**EXPERIMENT 3**

**AIM :** Build a simple application using Gradle

**ALGORITHM :**

1. Install Gradle (if not already done): Binary only
2. Create a Project Directory:
3. Create Java Classes:
4. Create a Gradle Build Script:(optional)
5. Build Your Application:
6. Type the command
7. Display the output

**EXECUTION :**

1. Install Gradle (if not already done):Binary only
2. Create a Project Directory:

**C:\Windows\System32>gradle --version**

**Gradle 8.14**

**C:\Windows\System32>mkdir my-java-app**

**C:\Windows\System32>cd my-java-app**

**C:\Windows\System32\my-java-app>gradle init**

**Starting a Gradle Daemon (subsequent builds will be faster)**

**Select type of build to generate:**

**1: Application**

**2: Library**

**3: Gradle plugin**

**4: Basic (build structure only)**

**Enter selection (default: Application) [1..4] 1**

**Select implementation language:**

**1: Java**

**2: Kotlin**

**3: Groovy**

**4: Scala**

**5: C++**

**6: Swift**

**Enter selection (default: Java) [1..6] 1**

**Enter target Java version (min: 7, default: 21): 21**

**Project name (default: my-java-app): javaapp**

**Select application structure:**

**1: Single application project**

**2: Application and library project**

**Enter selection (default: Single application project) [1..2] 1**

**Select build script DSL:**

**1: Kotlin**

**2: Groovy**

**Enter selection (default: Kotlin) [1..2] 1**

**Select test framework:**

**1: JUnit 4**

**2: TestNG**

**3: Spock**

**4: JUnit Jupiter**

**Enter selection (default: JUnit Jupiter) [1..4] 1**

**Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no] no**

**C:\Windows\System32\my-java-app>javac LeapYearCheck.java**

**C:\Windows\System32\my-java-app>java LeapYearCheck**

**Enter a year: 2004**

**2004 is a Leap Year.**

**gradle run**

**Program :**

```
import java.util.Scanner;

public class LeapYearCheck {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter a year: ");

      int year = scanner.nextInt();

      // Leap year logic

      if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

         System.out.println(year + " is a Leap Year.");

      } else {

         System.out.println(year + " is NOT a Leap Year.");  }

      scanner.close();

   }}
```

**OUTPUT:**

```
C:\Windows\System32\my-java-app>javac LeapYearCheck.java

C:\Windows\System32\my-java-app>java LeapYearCheck
Enter a year: 2004
2004 is a Leap Year.
```

```
Select test framework:
  1: JUnit 4
  2: TestNG
  3: Spock
  4: JUnit Jupiter
Enter selection (default: JUnit Jupiter) [1..4] 1

Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no] no


> Task :init
Learn more about Gradle by exploring our Samples at https://docs.gradle.org/8.14/samples/sample_building_java_applications.html

BUILD SUCCESSFUL in 1m 40s
1 actionable task: 1 executed
```

**RESULT** : Thus, a simple gradle application was successfully created and build

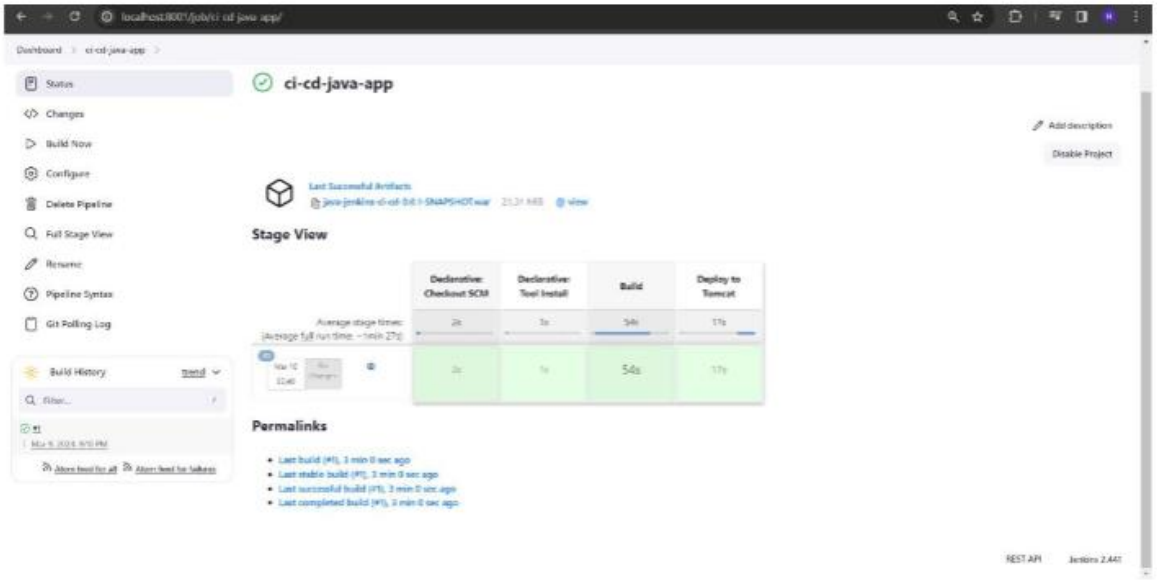
**EXPERIMENT 4**

**AIM :** Create CI pipeline using Jenkins

**ALGORITHM :**

1. Log in to Jenkins:
2. Create a New Jenkins Project:
3. ▯ Once logged in, you'll be redirected to the Jenkins console.
4. ▯ Click on "New Item" in the dashboard.
5. ▯ Choose a suitable name for your pipeline project.
6. ▯ Select the "Pipeline" option.
7. Configure Your Pipeline:
8. Save and Run:
9. ▯ Save your pipeline configuration.
10. ▯ Click "Build Now" to trigger the pipeline execution.
11. ▯ Jenkins will execute each stage sequentially.

**OUTPUT:**




**RESULT : Thus, CI pipeline was successfully created using Jenkins**

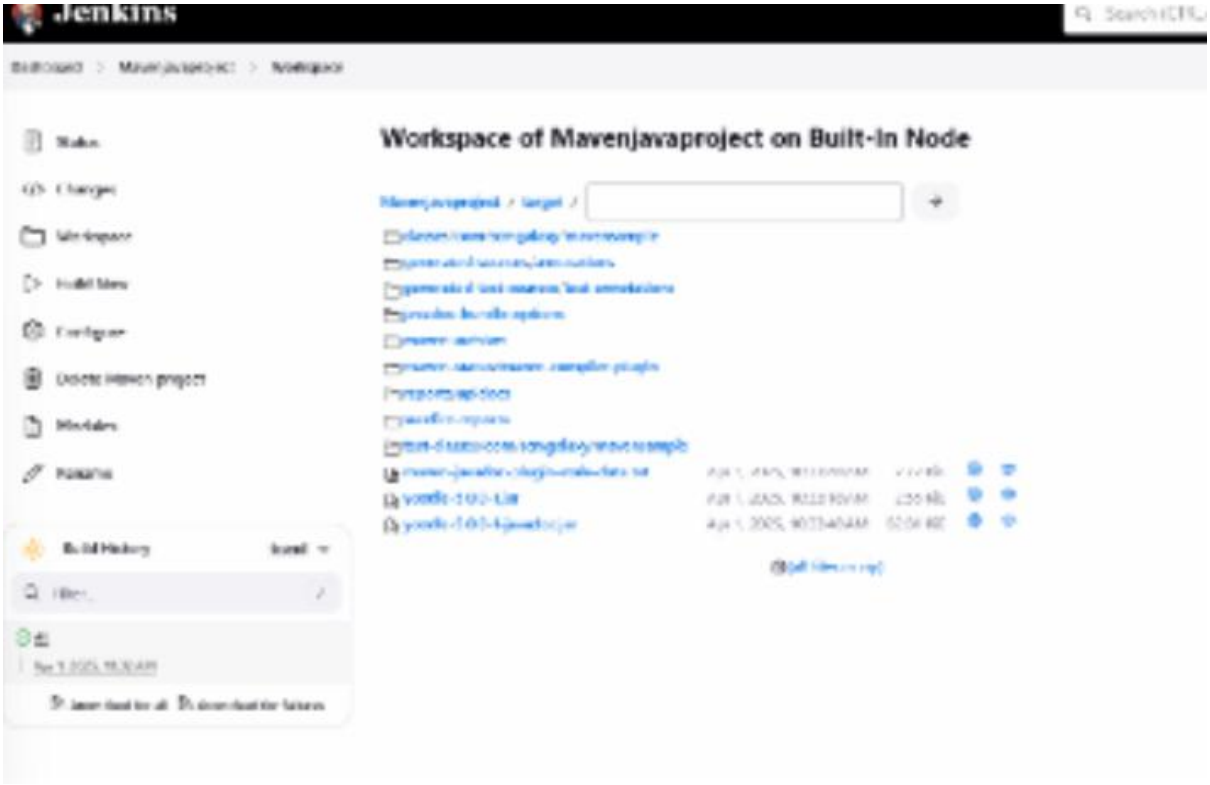
**EXPERIMENT 5**

**AIM :** Create a CD pipeline in Jenkins and deploy in Cloud

**ALGORITHM :**

1. **Install Jenkins: Launch the Jenkins console**
2. **Log in to Jenkins. Navigate to 'Manage Jenkins' →'Global Tool Configuration'**
3. **Click on 'Add Maven'**
4. **Enter the Name as 'Maven'**
5. **Navigate to 'Manage Jenkins' → click on 'Manage Plugins'**
6. **Look for Maven from the filter option and select 'Maven Integration' . Click on Install without restart Deploy to Cloud:**
7. **Once successfully installed , click on 'Go back to the top page'**
8. **Create a new Maven Project**
9. **Source code: https://github.com/mgsgoms/helloworld-java-maven.git**
10. **Click on 'New Item'**
11. **Enter the name as 'Maven-Prj' , select 'Maven Project' and click on Ok**
12. **Enter the Description as 'My Maven Project'**
13. **From Source Code Management , select Git . Use this github repository https://github.com/mgsgoms/helloworld-java-maven**
14. **Mention the Goals and Options as ' Clean install' and click on 'Save**
15. **Click on 'Build Now' → Below 'Build History' , click on #1 to view the console output**
16. **Look for the status**
17. **To check the outcome , click on Project name from the bread crump . Click on 'Workspace'**
18. **Click on 'target'**
19. **Find the jar files which has been build by Maven**

**OUTPUT:**




**RESULT :** Created a CD pipeline in Jenkins and deploy in Cloud

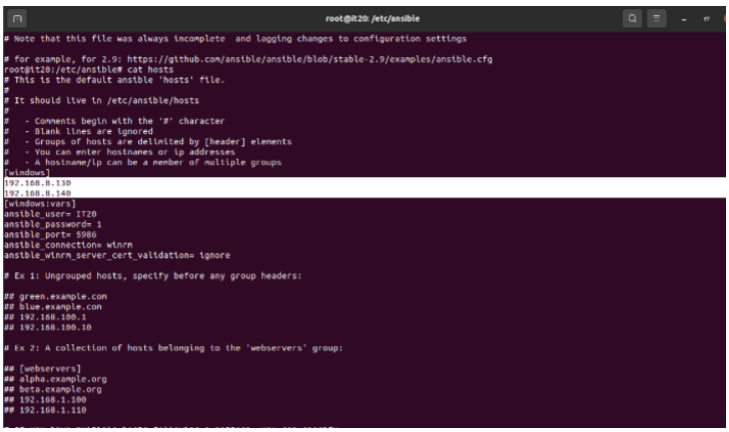
**EXPERIMENT 6**

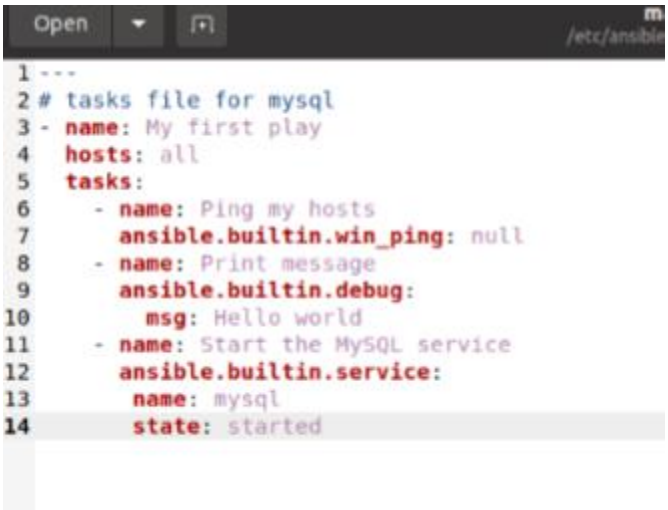
**AIM :** Create an Ansible playbook for a simple web application infrastructure

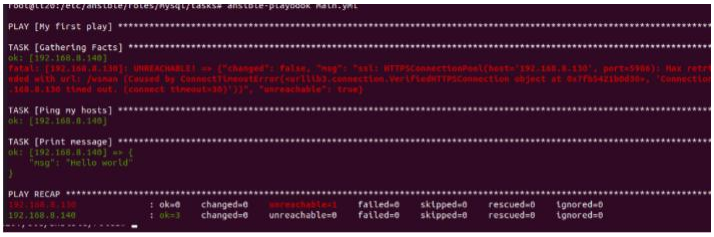
**ALGORITHM :**

- Install Ansible using `sudo apt install ansible -y`.
- Ensure the control node has SSH access to managed nodes.
- Generate SSH keys using `ssh-keygen` if not done already.
- Copy SSH key to managed host using `ssh-copy-id user@host`.
- Open the Ansible hosts file at `/etc/ansible/hosts`.
- Add managed host IPs under a group like `[webservers]`.
- Create a new YAML file named `webapp_setup.yml`.
- Define the playbook with a name and specify `hosts: webservers`.
- Enable privilege escalation with `become: yes`.
- Create a task to install Apache using the `apt` module.
- Add `update_cache: yes` to refresh package index.
- Add another task to start and enable the Apache service.
- Use the `service` module with `state: started` and `enabled: yes`.
- Create a task to deploy a sample HTML file using the `copy` module.
- Set `content: "<h1>Welcome</h1>"` and destination path.
- Save and close the playbook file.
- Run the playbook with `ansible-playbook webapp_setup.yml`.
- Wait for the tasks to complete on all managed hosts.
- Open a browser and enter the host IP to check the web page.
- Confirm that Apache is running and the homepage is displayed.

**OUTPUT:**


**Conclusion**

**Thus, an Ansible playbook for a simple application infrastructure was created and tested the code on managed hosts**

**EXPERIMENT 7**

**AIM : To install Ansible and configure ansible roles and to write playbooks**

**ALGORITHM :**

**1. Install Ansible**

**2. Update the host file to add the managed hostIPs**

**3. Create new roles**

**4. Create a playbook under the roles**

**5. Run the playbook using $ansible-playbook playbookname.yml**

**Note : Write in detail for 20 marks**

**EXECUTION**

**Host file updation**

**$gedit /etc/ansible/hosts**

**To view the files in ansible**

**cd /etc/ansible**

**ls**

**hosts ansible.cfg roles**

**roles**

**ansible-galaxy init apache**

**ls**

**tree apache**

**$ansible main.yml**

**Run the playbook**

**$ansible-playbook wintest.yml**

**OUTPUT:**

```
root@it20: /etc/ansible
# Note that this file was always incomplete  and lagging changes to configuration settings

# for example, for 2.9: https://github.com/ansible/ansible/blob/stable-2.9/examples/ansible.cfg
root@it20:/etc/ansible# cat hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups
[windows]
192.168.8.130
192.168.8.140
[windows:vars]
ansible_user= IT20
ansible_password= 1
ansible_port= 5986
ansible_connection= winrm
ansible_winrm_server_cert_validation= ignore

# Ex 1: Ungrouped hosts, specify before any group headers:

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
```

```
3  - name: My first play
4    hosts: all
5    tasks:
6      - name: Ping my hosts
7        ansible.builtin.win_ping: null
8      - name: Print message
9        ansible.builtin.debug:
10         msg: Hello world
11     - name: Start the MySQL service
12       ansible.builtin.service:
13        name: mysql
14        state: started
YAML  Tab Width: 8  Ln 14, Col 22  INS
```

**Run the playbook**
**$ansible-playbook wintest.yml**

```
root@it20:/home/student# ansible-playbook wintest.yml

PLAY [My first play] ***********************************************************

TASK [Gathering Facts] *********************************************************
ok: [192.168.8.140]

TASK [Ping my hosts] ***********************************************************
ok: [192.168.8.140]

TASK [Print message] ***********************************************************
ok: [192.168.8.140] => {
    "msg": "Hello world"
}

PLAY RECAP *********************************************************************
192.168.8.140              : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

**RESULT : Thus, an ansible roles are configured and new playbooks are created and executed successfully**

DEVOPS EXP 1 TO 7