

Experiment – 01

AIM: To implement a simple Remote Method Invocation (RMI) system in Java for cloud computing, enabling communication between a client and a server over a network.

Theory: This mechanism is used for communication between a client and a server over a network. RMI provides a powerful and seamless way to perform distributed computing in Java applications.

Remote Method Invocation (RMI) in Java is a mechanism that enables objects in one Java Virtual Machine (JVM) to invoke methods on objects in another JVM, typically across a network. RMI abstracts the complexities of network communication, allowing the client to call remote methods on the server as if they were local. It involves key components like the remote interface (which defines methods accessible remotely), the remote object (which implements the interface), and the stub (client-side proxy) that forwards requests to the server. The RMI registry allows clients to discover and access remote objects. Communication between the client and server is achieved by serializing method calls and transferring them over the network. RMI simplifies distributed computing in Java, but its performance can be impacted by serialization overhead and network issues, and it is mainly Java-specific, limiting cross-platform use.

Client-Server Communication Model:

Client: The client makes requests to a remote object hosted on the server, essentially invoking methods on objects located remotely, as if they were local.

Server: The server hosts remote objects, which contain the methods that can be invoked by clients. The server listens for incoming requests from clients and processes these requests accordingly.

Code for Server:

```
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Server is waiting for a connection...");

            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected!");
```

```
BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

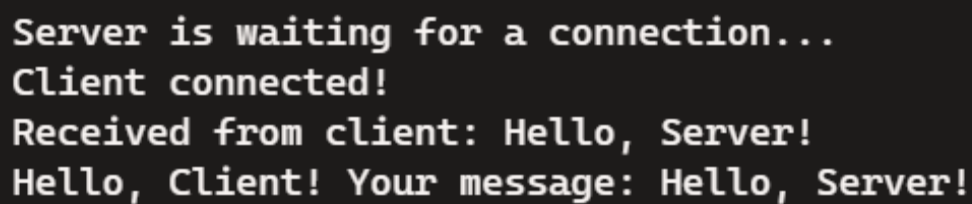
String message = in.readLine();

System.out.println("Received from client: " + "Hello Svvv");

System.out.println("Hello, Client! Your message: " + "Hello Svvv");

in.close();
out.close();
clientSocket.close();
serverSocket.close();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

Output:

A terminal window with a black background and white text. The text shows the sequence of events: the server waiting for a connection, a client connecting, the server receiving a message from the client, and the server responding to the client.

```
Server is waiting for a connection...
Client connected!
Received from client: Hello, Server!
Hello, Client! Your message: Hello, Server!
```

Code for Client:

```
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        try {
```

```
Socket socket = new Socket("192.168.150.153", 12345); // Connect to server

System.out.println("Connected to the server!");

BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println("HELLO, Server!");

String response = in.readLine(); // Fixed method name
System.out.println("Server says: " + response);

in.close();
out.close();
socket.close();
} catch (IOException e) {
    e.printStackTrace(); // Handle IO exceptions
}
}
```

Output:

```
Connected to the server!
Server says: Hello, Client! Your message: HELLO, Server!
```

Conclusion:

Implementing a simple Remote Method Invocation (RMI) system in Java facilitates seamless communication between a client and a server over a network, a key feature for distributed systems and cloud computing. By utilizing RMI, developers can invoke methods on remote objects transparently, as if they were local. This reduces complexity and enhances efficiency in distributed application development. RMI supports object serialization, security, and networking, making it a practical choice for enabling remote interactions in cloud environments. Overall, RMI simplifies distributed computing while maintaining robust functionality.

Experiment – 02

AIM: Insatallation of VMware Workstation on windows.

Theory: VMware Workstation is a virtualization software that enables users to run multiple operating systems on a single physical machine. Installing it on Windows involves downloading the installer from the official VMware website, ensuring your system meets the requirements (e.g., enabling virtualization in BIOS), and following a simple installation wizard to complete the setup. After installation, you can create and manage virtual machines for testing, development, or learning purposes.

Types of Virtualization:

Virtualization can be categorized into several types based on the level of abstraction and the purpose.

- **Hardware Virtualization:** Virtual machines on physical hardware.
- **OS Virtualization:** Isolated containers on one OS.
- **Application Virtualization:** Apps running independently of the OS.
- **Desktop Virtualization:** Remote virtual desktops.
- **Network Virtualization:** Logical networks over physical ones.
- **Storage Virtualization:** Unified management of multiple storage devices.

Hypervisor:

A **hypervisor** is software or firmware that creates and runs virtual machines (VMs) by managing hardware resources. Its types are:

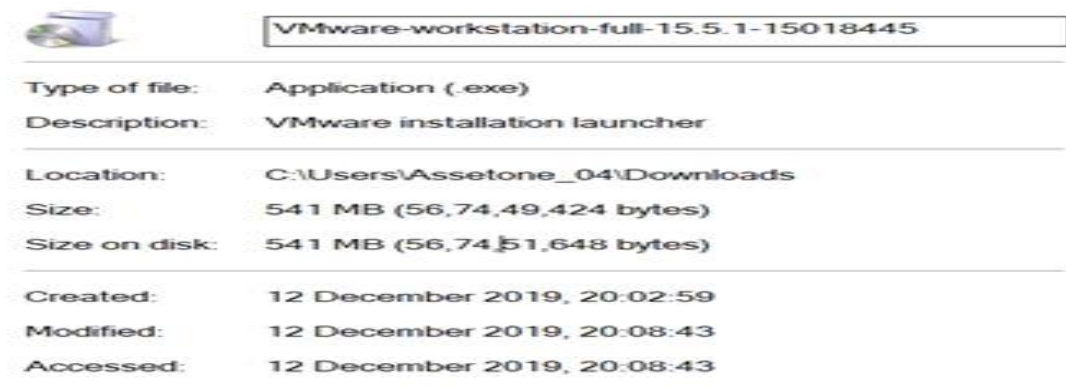
- **Type 1 (Bare-metal):** Runs directly on physical hardware (e.g., VMware ESXi, Microsoft Hyper-V).
- **Type 2 (Hosted):** Runs on top of an operating system (e.g., VMware Workstation, Oracle VirtualBox).

Both types enable virtualization for efficient resource utilization.

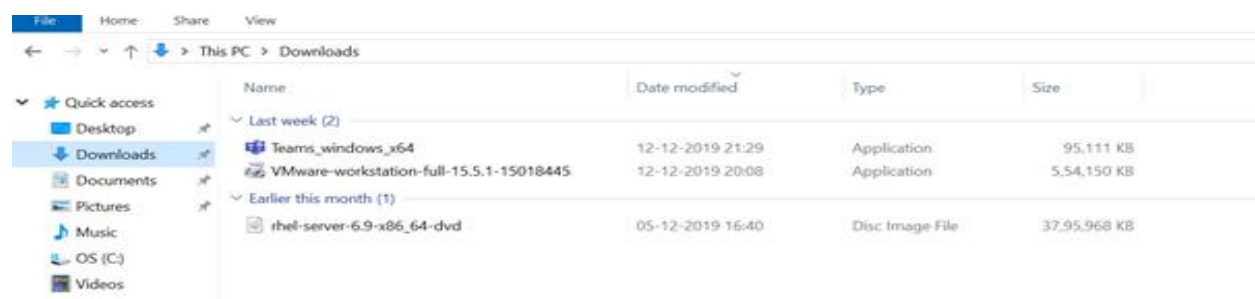
1. Installing VMware Workstation from given below link. There are two options for downloading one is Windows and other for Linux. My Base Operating System is Windows8, So I choose for VMware for Windows. If Your Base OS is Linux go and choose VMware for Linux Link.



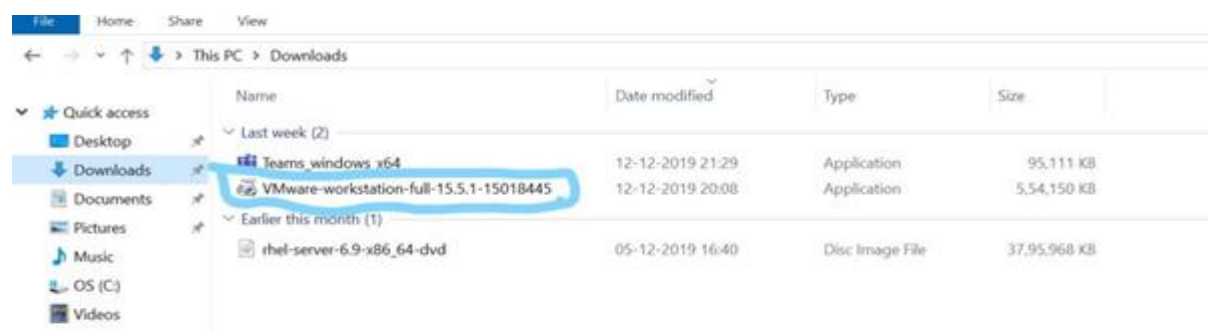
2. Check your VMware Properties.



3. Go to Download Folder.



4. Click the VMware downloaded File and Install it.



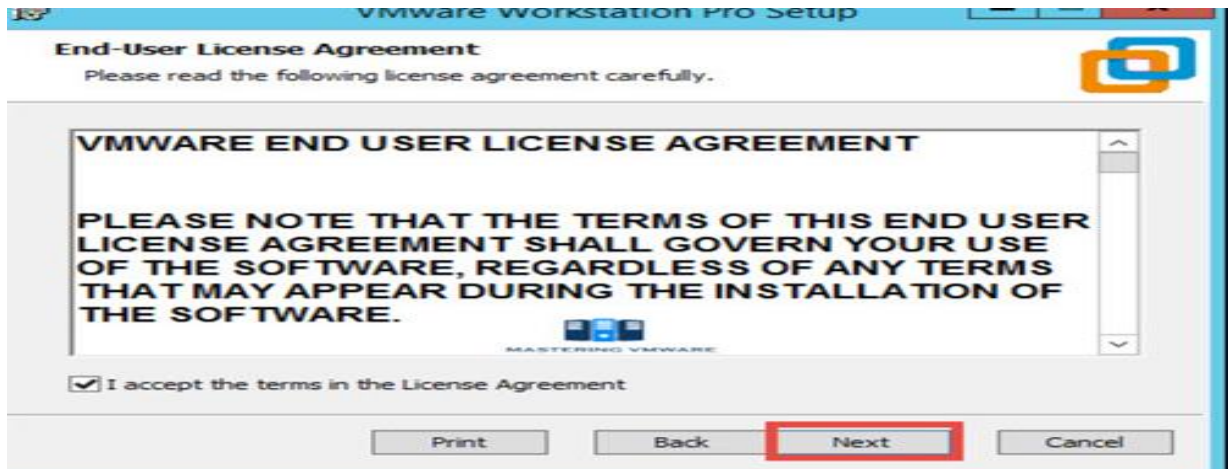
5. Click on VMware Software and click and choose “Pin to Taskbar”.

6. Click on VMware Software and Click on Next to the Installation wizard.



7. Read and Accept the VMware End User license agreement.

Click Next to Continue.



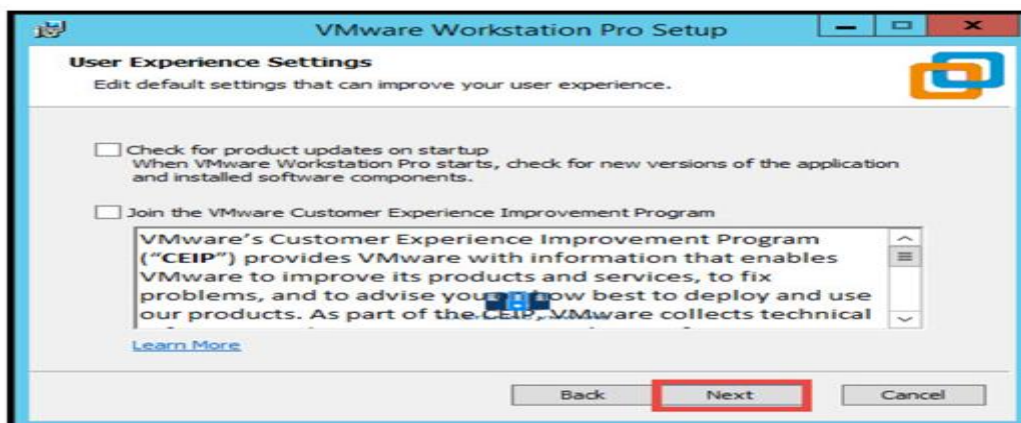
8. Specify the Installation directory. You can also enable Enhance keyboard driver here.

Click Next to continue.



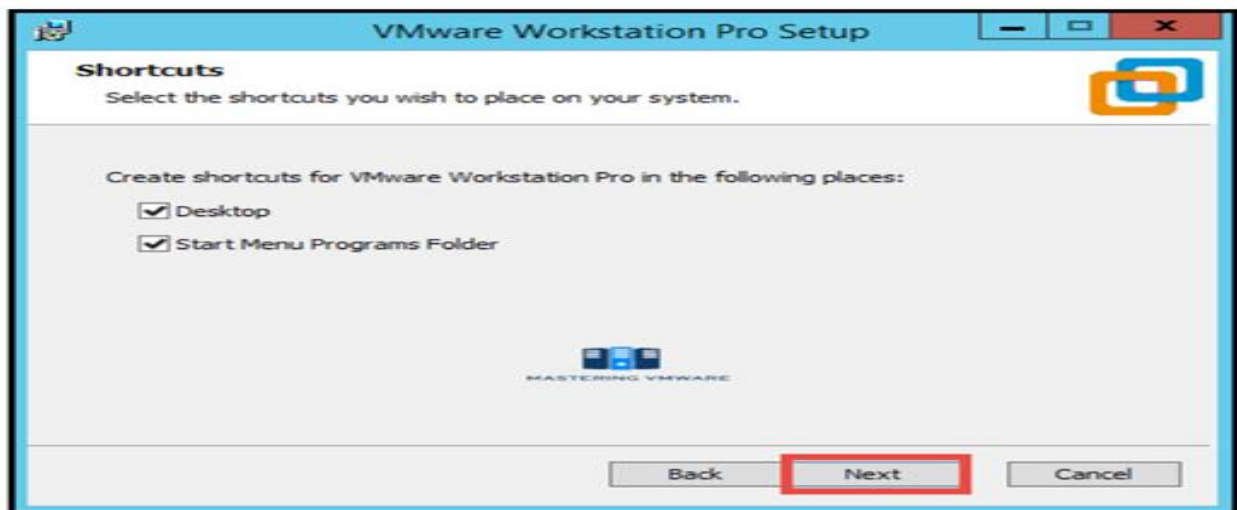
9. You can enable product startup and join the VMware Customer experience Improvement program here.

Click Next to Continue.

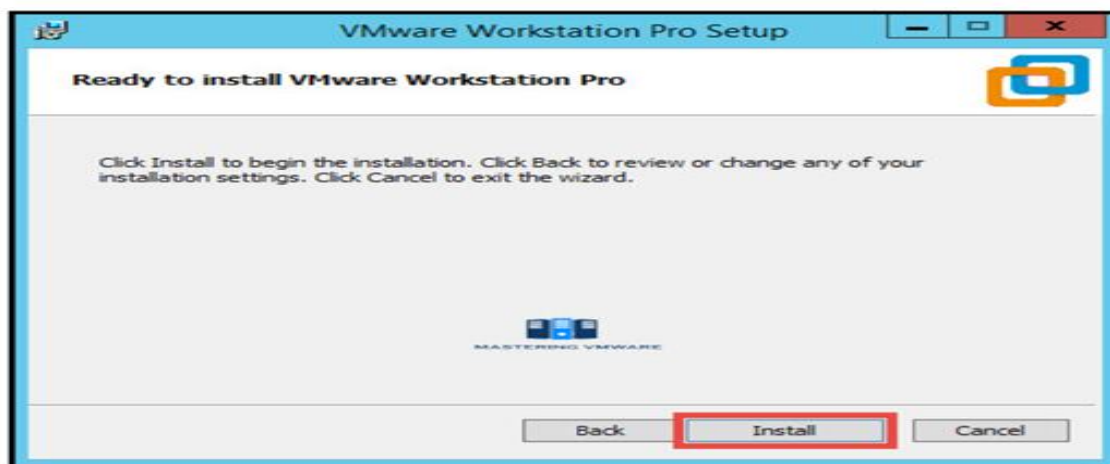


10. Select the shortcuts you want to create for easy access to VMware Workstation.

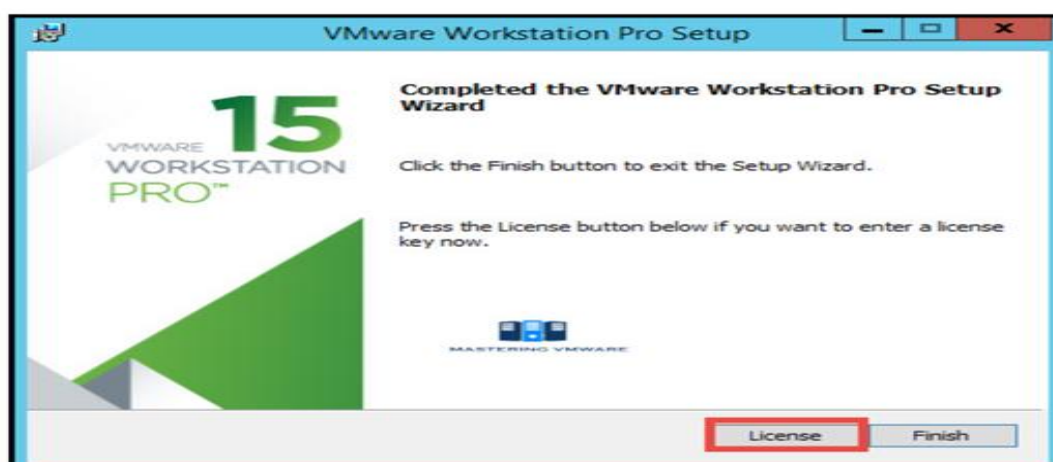
Click Next to Continue.



11. Click Install button to start the installation.



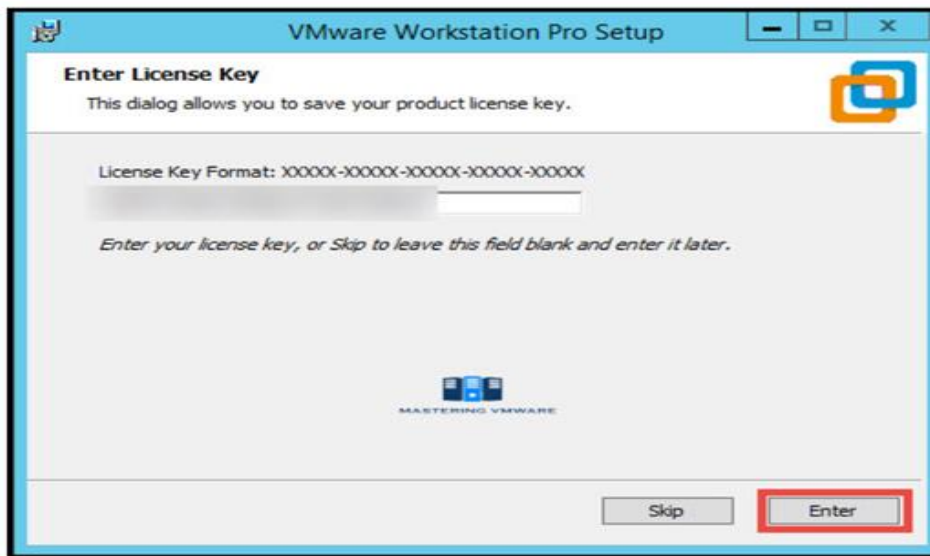
12. Installation will take just few seconds to complete.



If you have license-key then click on License to enter the license or you can also click Finish to exit the Installer.

13. Provide the License Key for VMware Workstation Pro.

Press Enter to continue.

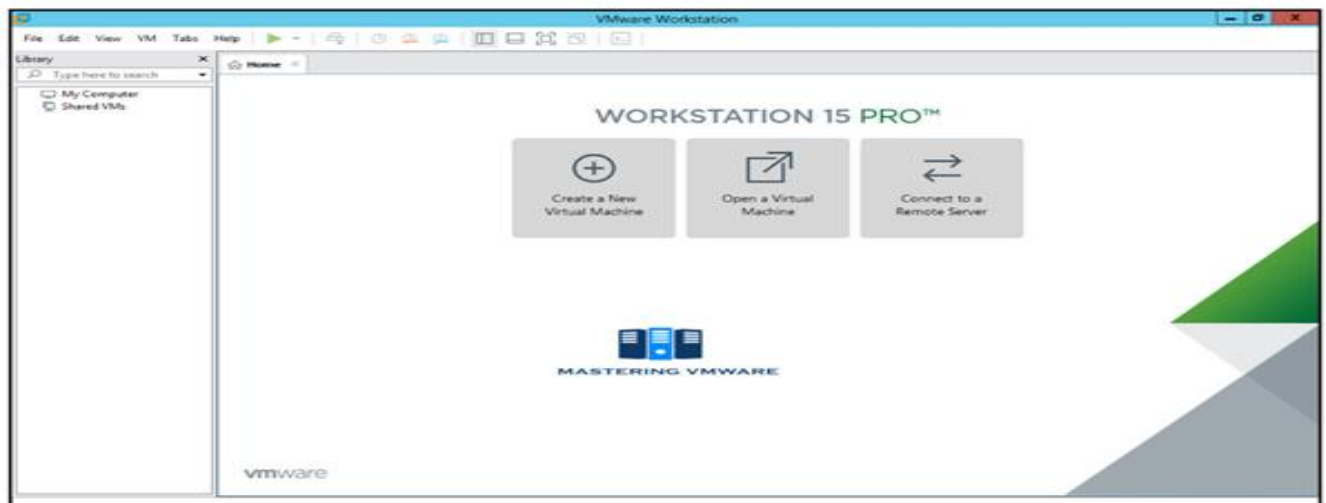


(Suggestion : If you have Don't License key search for internet or ask for who have already installed in their system. They surely have License Key.)

14. Click Finish to exit the wizard.**15. That's it we have successfully installed VMware Workstation Pro.**

Now you can start the VMware Workstation Pro by clicking on the shortcut on Desktop.

Below is the Home screen of the VMware Workstation pro which you will see every time when you start Workstation.



VMware successfully setup and installed.

Conclusion:

VMware empowers you to utilize the potential of virtualization on your Windows 10 machine. VMware offers a versatile solution to run multiple operating systems, test software, and enhance your overall computing experience.