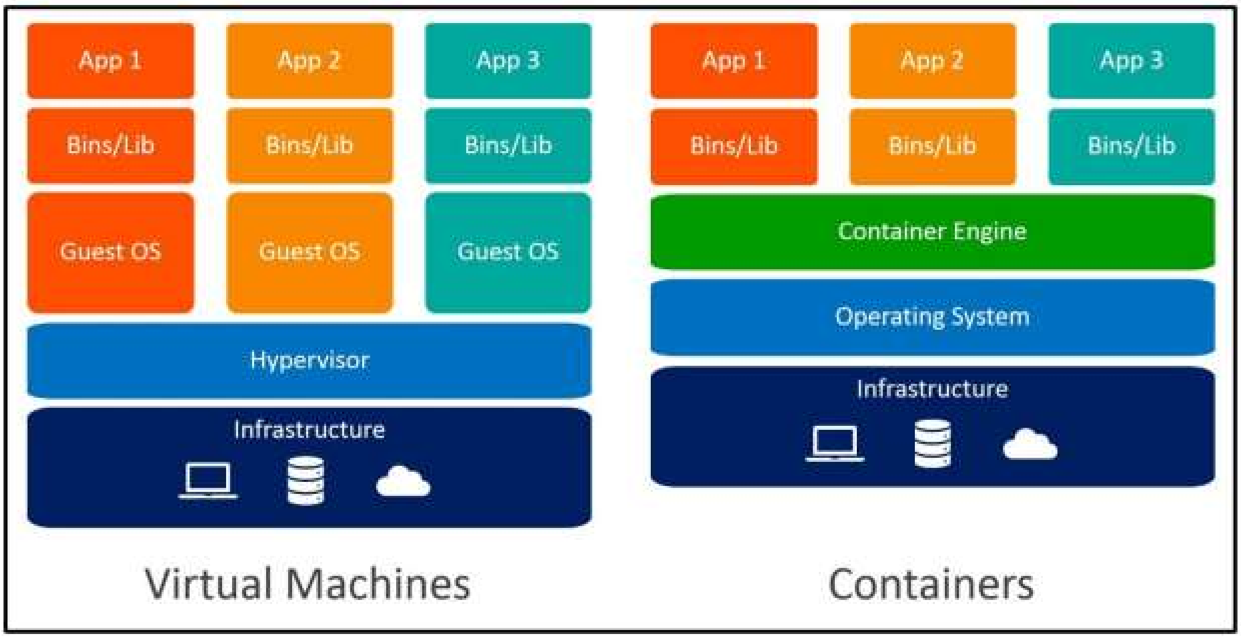
48. Docker

Docker is a platform that helps you package an application and all its dependencies into a lightweight, portable container.



Virtualization

- ➔ Virtualization creates virtual machines (VMs) that simulate entire hardware environments.
- ➔ It is very costly.
- ➔ **Analogy** ⇒ Imagine a hotel. Each VM is like a separate hotel room with its own amenities (like TV, bed, and bathroom). Every room is independent, but duplicating everything increases cost andsize.

Containerization

- ➔ Containerization runs applications in containers that share the host OS but remain isolated.
- ➔ Low cost.
- ➔ **Analogy** ⇒ Think of a shipping container. Each container holds its own cargo (app and dependencies) but shares the ship (host OS) for transport.

Key Difference

- ➔ **Virtualization:** Multiple OS environments on one machine.
- ➔ **Containerization:** Multiple apps share one OS but run independently.

Note

- ➔ Follow instructions in Docker pdf for installation and commands.
- ➔ Just check Docker Desktop is running in window services or not.If not just start the service.

conftest.py

```
import pytest
from selenium import webdriver
@pytest.fixture()
def setup(browser_platform):
  browser,platform = browser_platform
  options = {
     "chrome": webdriver.ChromeOptions,
     "edge": webdriver.EdgeOptions,
        "firefox": webdriver.FirefoxOptions
    }
    if browser not in options:
        raise ValueError(f"Unsupported browser: {browser}")
    platform_mapping = {"windows": "WIN10", "mac": "MAC", "linux": "LINUX"}
    platform_name = platform_mapping.get(platform)
    if not platform_name:
        raise ValueError(f"Unsupported platform: {platform}")
    opt = options[browser]()
    opt.add_experimental_option("detach", True) if browser in ["chrome", "edge"] else None
    opt.platform_name = platform_name
    driver = webdriver.Remote(command_executor="http://localhost:4444/wd/hub", options=opt)
    yield driver
    driver.quit()
```

<u>**Hook to add command-line options for browser and OS**</u>

```
def pytest_addoption(parser):
    parser.addoption("--browser", default="chrome", choices=["chrome", "edge", "firefox"], help =
"Browser to test")
    parser.addoption("--os", default="linux", choices=["windows", "mac", "linux"], help = "Operating
system to test")
```

<u>**Get value from command Line**</u>

```
@pytest.fixture()
def browser_platform(request):
    browser = request.config.getoption("--browser")
    platform = request.config.getoption("--os")
    return browser,platform
```

<u>**test_Parallel.py**</u>

```
class TestTitle:
    def test_title_chrome(self,setup):
        driver = setup
        driver.get("https://www.google.com/")
        assert driver.title == "Google"  # validation
    def test_title_edge(self,setup):
        driver = setup
        driver.get("https://www.google.com/")
        assert driver.title == "Google"  # validation
    def test_title_firefox(self,setup):
        driver = setup
        driver.get("https://www.google.com/")
        assert driver.title == "Google"  # validation
```

<u>**Command to Execute**</u>

- ➔ `pytest -s -v --os=linux --browser=chrome -n 3 test_Parallel.py`