RV UNIVERSITY

Go, change the world

an initiative of RV EDUCATIONAL INSTITUTIONS

NEW-AGE GLOBAL UNIVERSITY FOR LIBERAL EDUCATION

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

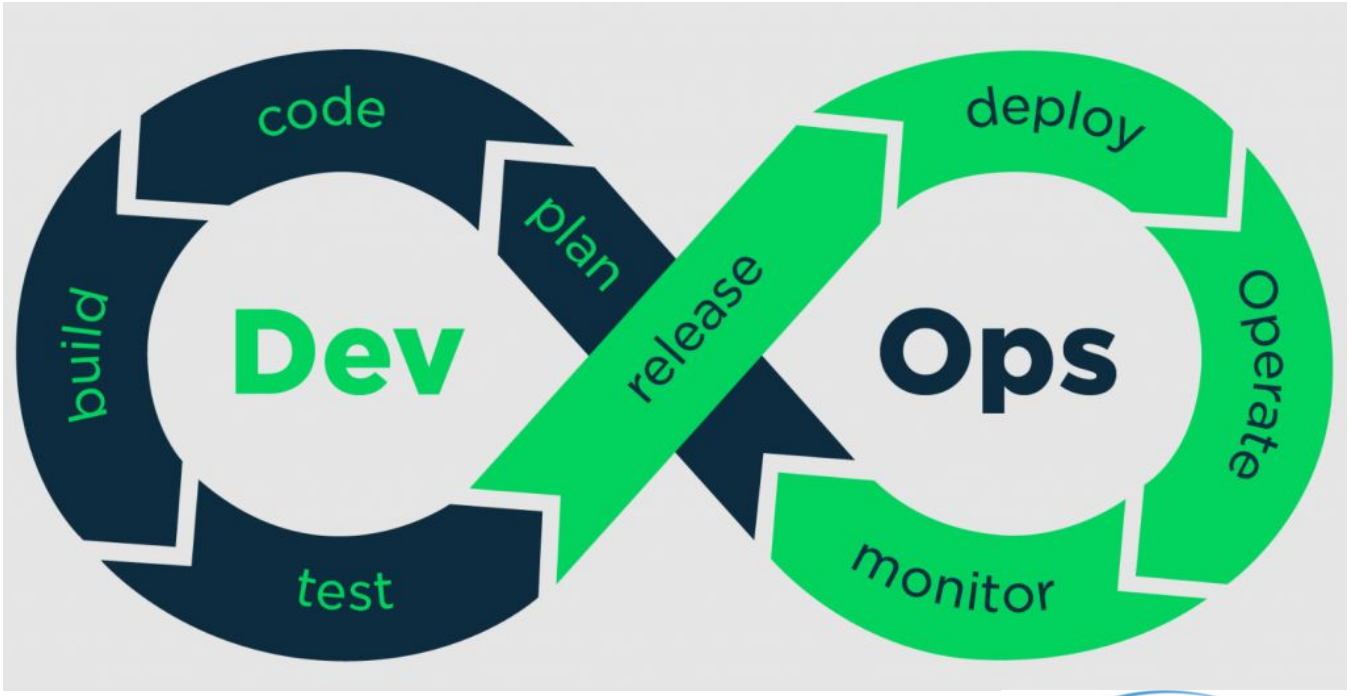
Agile Software Engineering
CS 2004
Unit 4
DevOps Foundations

Agenda

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Overview of Devops and why it is needed;
- Continuous Integration, Build, Testing, Delivery and Deployment(CI and CD): Jenkins, Pylint, PyUnit.
- Overview of version control, branching and release using GIT -Git/Github/GitActions
- Creating pipelines, Setting up runners Containers and container orchestration (Docker, DockerHub and Kubernetes) for application development and deployment;

What is DevOps



What is DevOps

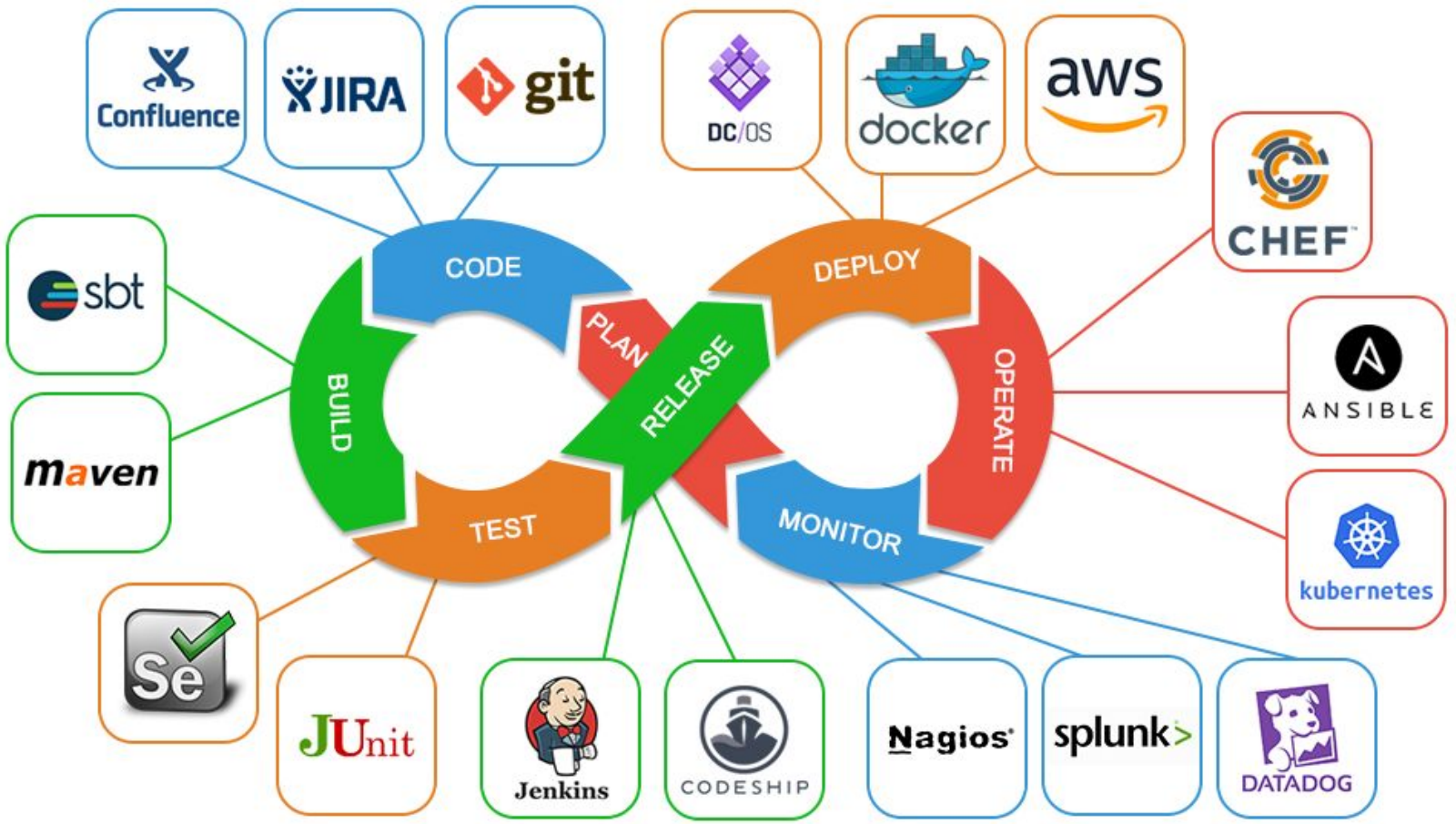
- DevOps is a compound of development (Dev) and operations (Ops).
- It is the union of people, process, and technology to continually provide value to customers.
- DevOps outlines a software development process and an organizational culture shift that speeds the delivery of higher quality software by automating and integrating the efforts of development and IT operations teams – two groups that traditionally practiced separately from each other, or in silos

What is DevOps –contd.

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products.
- By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster

DevOps Tools



RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Puppet is an **open-source configuration management and automation tool** used in DevOps.
- It helps IT operations teams automate the provisioning, configuration, and management of infrastructure and applications.
- Puppet uses a **declarative language** to define the desired state of the system, ensuring that systems are consistently configured and compliant with the desired configurations.

Puppet – key features

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **Platform Support**: Compatible with various platforms like Windows, Linux, and macOS.
- **Scalability**: Suitable for small to large organizations.
- **Idempotency**: Ensures that running the same configuration multiple times does not change the result beyond the initial application.
- **Open-Source**: Free to use and extend.
- **Reporting Compliance**: Provides graphical reporting and real-time visibility into infrastructure changes.

- Chef is a **powerful configuration management and automation tool** used in DevOps.
- Like Puppet, Chef helps automate the configuration, deployment, and management of infrastructure across an organization's IT environment.
- Chef uses a **domain-specific language (DSL)** written in Ruby to define the desired state of infrastructure, enabling teams to manage and scale systems more efficiently.

Chef – key features

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **Infrastructure as Code (IaC)**: Chef allows you to define your infrastructure using code, making it easier to version, share, and replicate configurations.
- **Scalability**: Chef is designed to manage thousands of nodes and is suitable for large-scale environments.
- **Cross-Platform Support**: Chef supports multiple platforms, including Windows, Linux, and macOS.
- **Community and Ecosystem**: Chef has a large community and a rich ecosystem of cookbooks (pre-configured templates) that can be used to automate common tasks.
- **Idempotency**: Chef ensures that running the same configuration multiple times results in the same state, without causing unintended changes.

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Ansible is an **open-source automation tool** used in DevOps for configuration management, application deployment, task automation, and orchestration. It is known for its simplicity and ease of use.
- Unlike other configuration management tools, Ansible operates without requiring a dedicated agent on the target machines, making it agentless.
- It uses **SSH** (Secure Shell) for communication, which adds an extra layer of security.

Ansible – key features

- **Agentless Architecture**: No need for agents on managed nodes, simplifying setup and reducing overhead.
- **Declarative Language**: Uses YAML (Yet Another Markup Language) for its playbooks, which are easy to read and write.
- **Idempotency**: Ensures that running the same configuration multiple times does not change the result beyond the initial application.
- **Cross-Platform Support**: Can manage systems running on various operating systems, including Linux, Windows, and macOS.
- **Modular and Extensible**: Ansible has a wide array of modules for various tasks, and you can write custom modules as

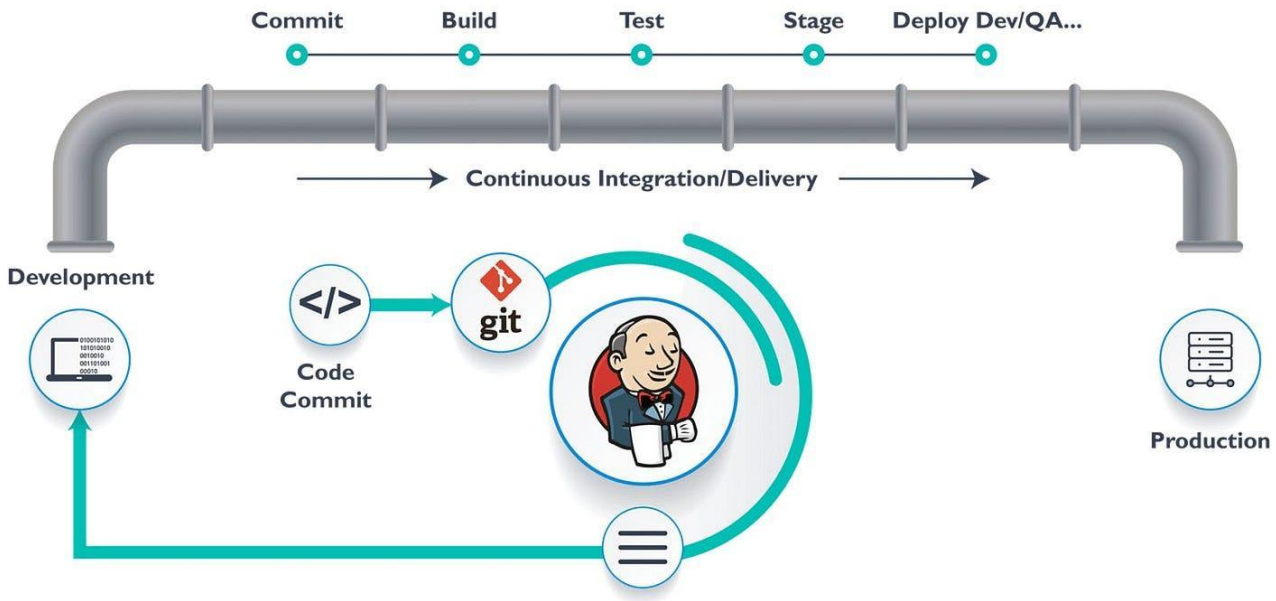
Puppet Vs Chef Vs Ansible

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Feature	Puppet	Chef	Ansible
Architecture	Master-Agent	Master-Agent	Agentless
Language	Custom DSL	Ruby DSL	YAML
Ease of Use	Steeper Learning Curve	Requires Ruby Knowledge	Easiest to Learn
Scalability	Highly Scalable	Highly Scalable	Scalable
Community and Ecosystem	Strong Community	Active Community	Large Community
Configuration Language Model	Declarative	Mix of Declarative and Imperative	Declarative with Procedural Elements
Idempotency	Yes	Yes	Yes
Use Cases	Large Enterprises	Flexible Configuration Needs	Small to Medium Organizations

Jenkins

- An open-source automation server used to automate parts of the software development process, including building, testing, and deploying code.
- It is widely used in DevOps and continuous integration/continuous delivery (CI/CD) pipelines to streamline and speed up the development lifecycle.



Medium

Pylint

- A static code analysis tool for Python.
- It helps developers identify issues such as errors, code quality issues, and coding standard violations in their Python code.
- Pylint analyzes the code without running it and provides a detailed report that includes warnings, errors, and suggestions for improvement.

Pylint – key features

- **Error Detection**: Identifies potential errors in the code, such as undefined variables or improper function calls.
- **Code Quality Checks**: Analyzes code for adherence to PEP 8, the Python style guide, and other coding standards.
- **Refactoring Suggestions**: Offers suggestions for refactoring code to improve readability and maintainability.
- **Configurable**: Allows customization of the checks and reports based on user preferences.

PyUnit

- PyUnit, also known as the unittest module, is a unit testing framework for Python. It is a port of the JUnit framework used for Java, and it provides a way to write, organize, and run tests for your Python code.

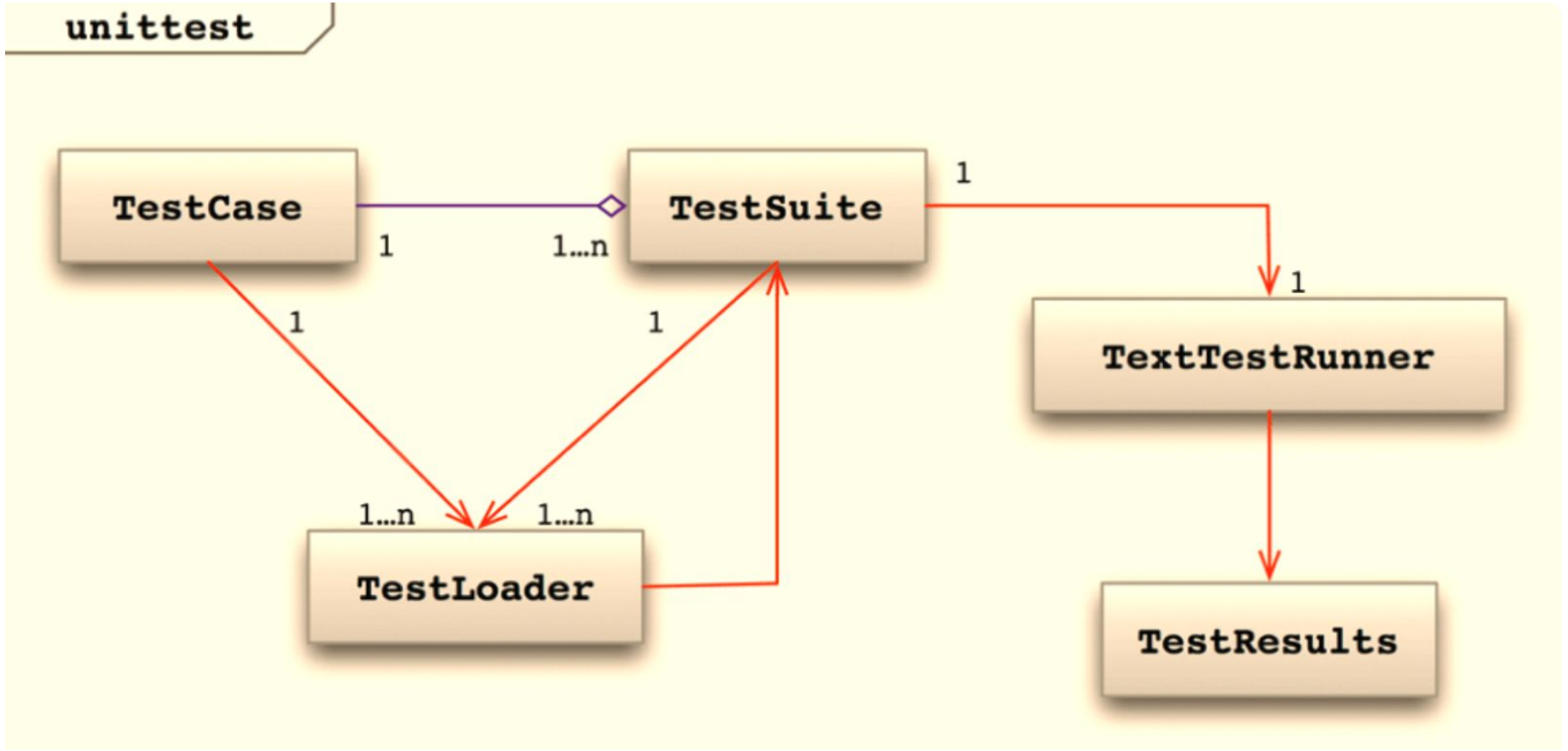
- PyUnit is part of the standard Python library, making it readily available for all Python developers.

PyUnit – key features

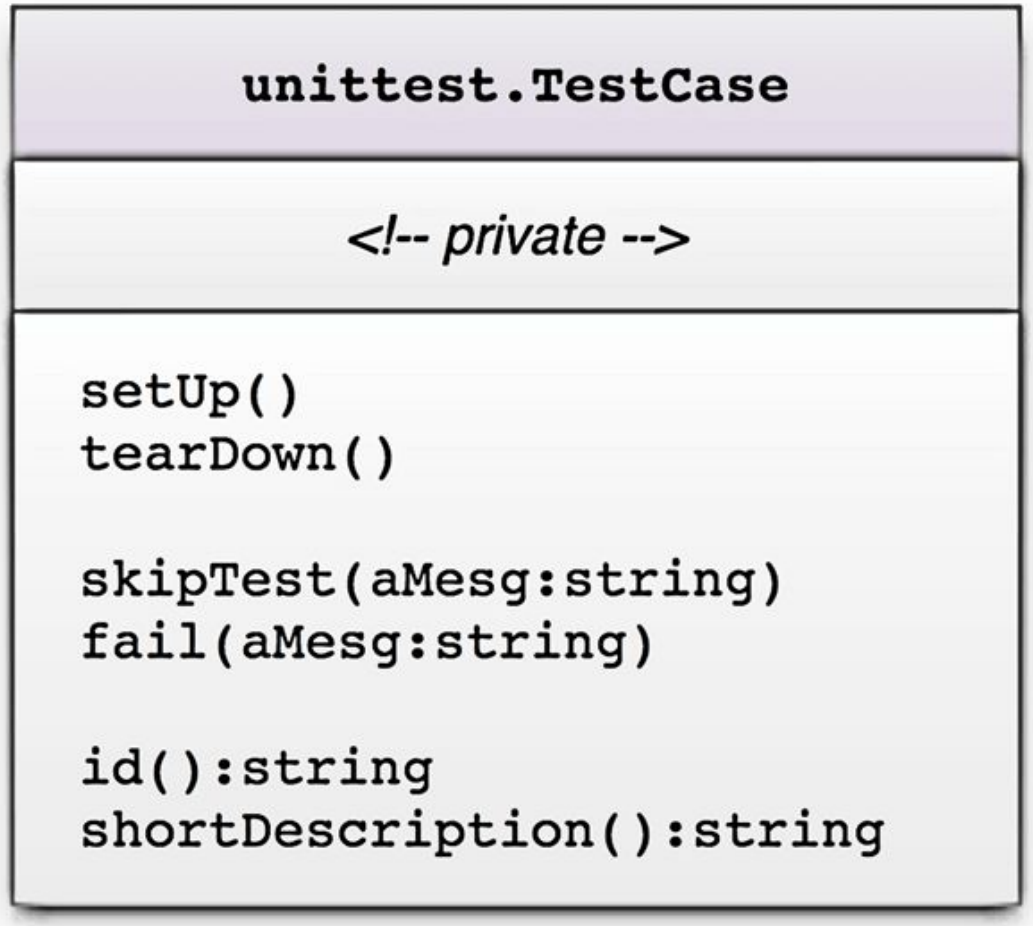
RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **Test Discovery**: Automatically finds and runs test cases in your project.
- **Test Fixtures**: Allows you to set up and tear down resources needed for your tests using setup and teardown methods.
- **Test Suites**: Enables you to group multiple test cases into test suites for easier management and organization.
- **Command Line Interface**: Supports running tests from the command line, with options for verbosity and test filtering.

PyUnit – structure



Designing a test case for Python Testing using PyUnit



Advantages of using Python Unit testing

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- It helps you to detect bugs early in the development cycle
- It helps you to write better programs
- It syncs easily with other testing methods and tools
- It will have many fewer bugs
- It is easier to modify in future with very less consequence

Git and Version Control

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **Version Control**: Git is a distributed version control system used to track changes in source code during software development. It helps developers collaborate, maintain code history, and manage different versions of their projects.
- **Commits**: A commit in Git is a snapshot of your project's files at a specific point in time. Each commit has a unique identifier (hash) and a message describing the changes made.
- **Repositories**: A Git repository (repo) is a storage space where your project files and their version history are stored.

Branching in Git

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **Branches**: Branches allow developers to work on different features, bug fixes, or experiments in parallel without affecting the main codebase. The default branch is usually called main or master.
- **Creating Branches**: You can create a new branch using the command: git branch <branch-name>. To switch to a branch, use: git checkout <branch-name>.
- **Merging**: When your work on a branch is complete, you can merge it back into the main branch. This can be done using the command: git merge <branch-name>.
- **Pull Requests**: In GitHub, a pull request (PR) is a way to propose changes from one branch to another. It allows team members to review and discuss the changes before merging.

Release Management

- **Releases**: A release is a specific point in your project's history that you can share with users. It typically includes compiled binaries, release notes, and other important information. In GitHub, you can create a release from the repository's Releases tab.
- **Tags**: Tags are used to mark specific commits as significant, such as version releases. You can create a tag with the command: git tag <tag-name>. Tags are often used to label releases.

GitHub Actions for CI/CD

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- **GitHub Actions**: GitHub Actions is a CI/CD (Continuous Integration/Continuous Deployment) platform that allows you to automate workflows for your GitHub repositories. You can create custom workflows to build, test, and deploy your code.
- **Workflow Files**: Workflows are defined in YAML files located in the .github/workflows directory of your repository. Each workflow specifies the events that trigger it, the jobs to be run, and the steps within each job.

GitHub Actions for CI/CD –contd.

- **Triggers**: Common triggers include push, pull_request, and schedule. For example, you can trigger a workflow to run tests every time code is pushed to the repository.
- **Jobs and Steps**: Workflows consist of jobs, which are sets of steps executed on a virtual machine. Each step can run commands, execute scripts, or use pre-built actions from the GitHub Marketplace.

Creating Pipelines with GitHub Actions -1

- Define a Workflow File:
- Workflows are defined using YAML files located in the .github/workflows directory of your GitHub repository. Each workflow file describes a pipeline.

Creating Pipelines with GitHub Actions -2

- Specify Workflow Triggers:

- Triggers determine when the workflow should run. Common triggers include push, pull_request, and schedule. For example, you can run the pipeline whenever code is pushed to the repository.

Creating Pipelines with GitHub Actions -3

Define Jobs and Steps:

- A workflow consists of one or more jobs, each containing multiple steps. Jobs run in parallel by default but can be configured to run sequentially.
- Steps are individual tasks that make up a job. Steps can include checking out code, setting up the environment, installing dependencies, running tests, and deploying code.

Creating Pipelines with GitHub Actions -4

- Use Actions:

- GitHub Actions provides a marketplace with pre-built actions that you can use in your workflows. These actions perform common tasks like setting up environments, running scripts, and deploying applications.
- You can also create custom actions to perform specific tasks unique to your project.

GitHub Pipeline Example
Actions Workflow (YAML)

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

```yaml
name: CI Workflow

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: 3.x

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt

    - name: Run tests
      run: |
        pytest
```

- This workflow
- runs on push and pull_request events,
- checks out the code,
- sets up Python,
- installs dependencies, and
- runs tests using pytest.

Explanation of the Workflow File:

- **name**: The name of the workflow.
- **on**: Specifies the events that trigger the workflow. In this case, it runs on push and pull_request events.
- jobs: Defines the jobs in the workflow. This example has a single job named build.
- runs-on: Specifies the type of machine to run the job on. ubuntu-latest is a virtual machine with the latest Ubuntu version.
- steps: Lists the steps in the job:
- Checkout code: Uses the actions/checkout action to check out the code from the repository.
- Set up Python: Uses the actions/setup-python action to set up Python 3.9.
- Install dependencies: Runs a command to install the project's dependencies.
- Run tests: Runs the pytest command to execute the tests.

DevOps - Docker

What is a container?

What is docker's role in this context?

What is Containerization?

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Containerization refers to the use of containers to package an application and all its dependencies together, to make sure that the application works seamlessly in any environment, whether it be development, testing or production
- Containerization is a form of OS-based virtualization that creates multiple virtual units in the user space, known as containers
- Containers share the same host kernel but are isolated from each other through private namespaces and resource control mechanisms at the OS level

Containers

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another

Containers are executable units of software in which application code is packaged along with its libraries and dependencies, in common ways so that the code can be run anywhere—whether it be on desktop, traditional IT, or the cloud

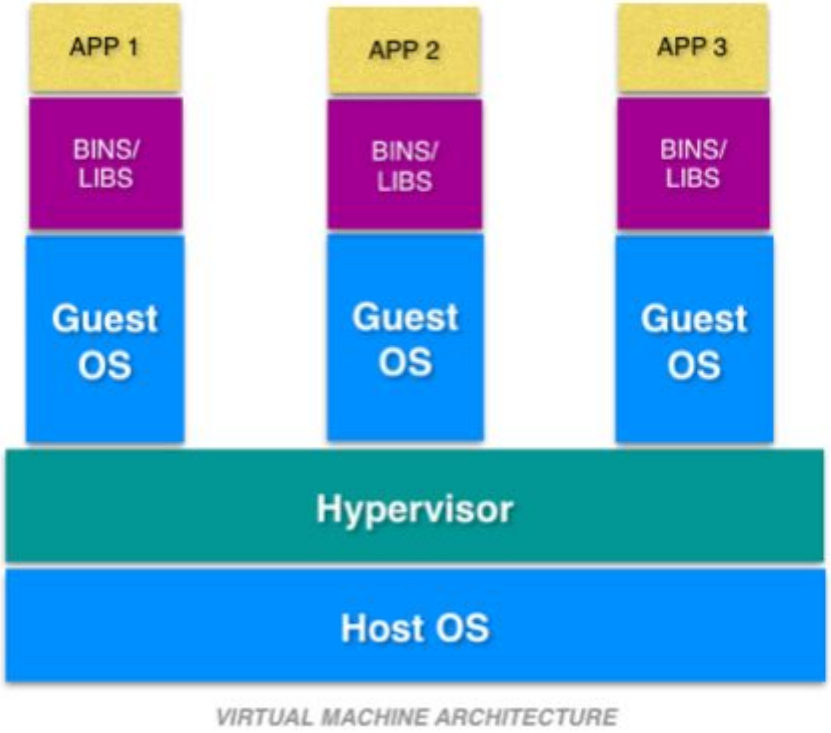
Containers are basically a fully functional and portable cloud or non-cloud computing environment surrounding the application and keeping it independent of other environments running in parallel

Docker’s role in containerization

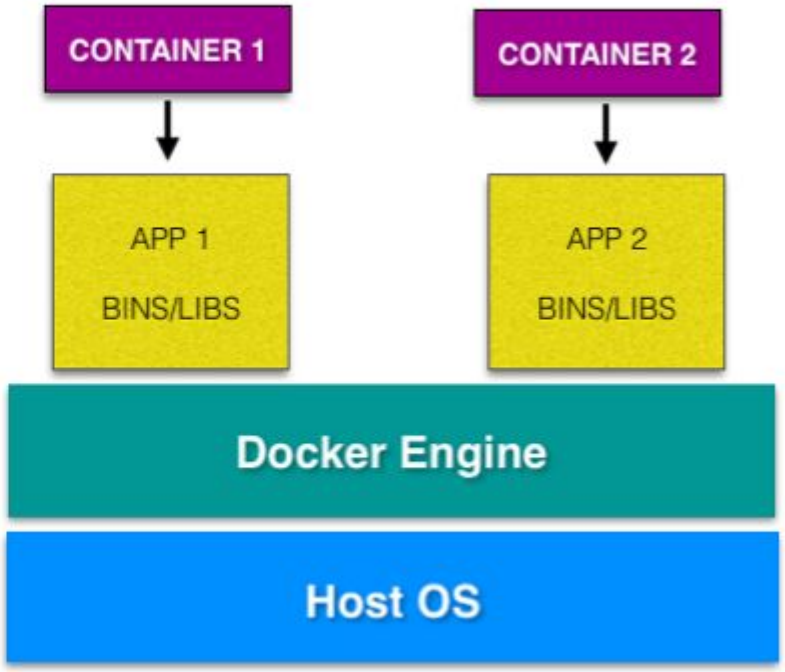
Docker plays a crucial role in the world of containerization. Docker is the containerization platform that is used to package your application and all its dependencies together in the form of containers to make sure that your application works seamlessly in any environment which can be developed or tested or in production

Docker provides the fundamental building block necessary for distributed container deployments. By packaging application components in their own containers, horizontal scaling becomes a simple process of spinning up or shutting down multiple instances of each component

What is VM, Container & Docker



VIRTUAL MACHINE ARCHITECTURE



DOCKER ARCHITECTURE

What is docker?
What is docker engine and docker desktop?
When I install docker desktop, command line version also get installed?

What is Docker ?

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers
- Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels
- The Docker Engine is the main service which runs containers. It’s an open-source containerization technology for building and containerizing your applications
- Docker Engine acts as a client-server application with a server that has a long-running daemon process, APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon, and a command-line interface (CLI) client

Docker File & Docker Images

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Dockerfile
  - Dockerfile is a file, it will describe the steps in order to create an image quickly
  - The Docker daemon runs the instructions in the Dockerfile are processed from the top down, so you should order them accordingly
  - The Dockerfile is the source code of the Docker Image

- Docker Image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files

Docker Desktop

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Docker Desktop, on the other hand, is an application for Mac, Linux, or Windows environments that lets you build, share, and run containerized applications and microservices
- It provides a straightforward GUI (Graphical User Interface) that lets you manage your containers, applications, and images directly from your machine
- Docker Desktop includes Docker Engine, Docker CLI client, Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper
- When you install Docker Desktop, the command-line version also gets installed

What is Kubernetes in simple words how it is different than docker?

What is Kubernetes (K8S)

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Kubernetes, in simple words, is a robust open-source orchestration tool developed by Google for operating microservices or containerized applications across a distributed cluster of nodes
- It provides very flexible infrastructure with zero downtime deployment capacities, automated rollback, scaling, and self-healing of containers
- The principal purpose of Kubernetes is to mask the complexity of maintaining a fleet of containers by implementing REST APIs for the functionalities needed

Docker Vs Kubernetes

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Now, let's understand how it's different from Docker. Docker is a platform that uses OS-level virtualization to deliver software in packages called containers. It allows you to easily build applications, package them with all required dependencies, and ship them to run on other machines
- On the other hand, Kubernetes is an ecosystem for managing a cluster of Docker containers. It turns a set of servers into a complete, private cloud
- Kubernetes comes into the picture when you need to work with a large number of containers across multiple machines
- In summary, while Docker focuses on automating the deployment, scaling, and management of applications within containers, Kubernetes is used to manage clusters of containers across multiple machines for robustness and scalability.

What is Orchestration?

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Orchestration refers to the automation of the deployment, management, scaling, networking, and availability of container-based applications

- Kubernetes provides an abstraction layer over a cluster of virtual or physical machines, accessed through the Kubernetes API, which allows for the automation of many tasks needed to run containers at scale.

Kubernetes vs Docker

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

- Docker is a platform for containerization, while Kubernetes is a platform for managing and orchestrating containers from many container runtimes, including Docker

- Docker allows you to build and run containers on any development machine, while Kubernetes lets you orchestrate a cluster of virtual machines and schedule containers to run on those virtual machines based on their available compute resources and the resource requirements of each container

Can I package more than 1 docker images in single container?

- In Docker, each container is designed to run a single application or process. This is known as the “one process per container” rule. It’s one of the core principles of containers and is what allows containers to be lightweight and easily scalable
- If you have multiple applications or services that you want to run together, the recommended approach is to use separate containers for each one and then use a tool like Docker Compose or Kubernetes to manage them together. These tools allow you to define multi-container applications, handle networking between containers, and manage shared resources
- So, while it’s technically possible to include multiple applications in a single Docker container, it’s not recommended as it goes against the design principles of containers and can lead to issues with port conflicts, logging, scaling, and resource isolation

What is a Pod in Kubernetes?

Pod

- A Pod in Kubernetes is the smallest deployable unit of computing that you can create and manage
- It’s a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers
- A Pod’s contents are always co-located and co-scheduled, and run in a shared context
- Pods are ephemeral by nature, if a pod (or the node it executes on) fails, Kubernetes can automatically create a new replica of that pod to continue operations
- Pods include one or more containers (such as Docker containers)

Pod – contd.

- In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host

- Pods in a Kubernetes cluster are used in two main ways
  - Pods that run a single container. The “one-container-per-Pod” model is the most common Kubernetes use case; in this case, you can think of a Pod as a wrapper around a single container

  - Pods that run multiple containers that need to work together. A Pod can encapsulate an application composed of multiple co-located containers that are tightly coupled and need to share resources

References

(1) What is Kubernetes – Basic Concepts in simple words
https://apachebooster.com/blog/what-is-kubernetes-basic-concepts-in-simple-words/
(2) Docker vs Kubernetes – javatpoint
https://www.javatpoint.com/docker-vs-kubernetes
(3) Kubernetes and Docker: What's the difference? - Tutorial Works
https://www.tutorialworks.com/kubernetes-vs-docker/
(4) Difference Between Kubernetes and Docker
http://www.differencebetween.net/technology/difference-between-kubernetes-and-docker/
(5) Introduction to Kubernetes (K8S) – GeeksforGeeks
https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/
(6) Kubernetes in simple words: explained by Eric Swildens
https://dev.to/nimbella/kubernetes-in-simple-words-explained-by-eric-swildens-52j
(7) Kubernetes vs Docker – GeeksforGeeks
https://www.geeksforgeeks.org/kubernetes-vs-docker/
(8) en.wikipedia.org. https://en.wikipedia.org/wiki/Kubernetes

Docker Lab - 2

App Code

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run()
```

In this code:

Flask(__name__) creates a new Flask web server.

@app.route('/') is a decorator that tells Flask what URL should trigger the function hello().

The hello() function returns the string ‘Hello, World!’ which is displayed in the user’s web browser when they navigate to the root URL of the server.

The if __name__ == '__main__': app.run() part allows the server to run if the script is executed directly.

To run this application, save it as a .py file and run it with Python. Then, open a web browser and navigate to http://localhost:5000/ to see the message ‘Hello, World!’.

Instructions (Part 1)

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

1. Create a new directory (docker-yourname)
   Copy the 3 files (app, .dockerignore, Dockerfile)
2. Run the Docker
3. Building and Running the Docker Image, run the following command
   docker build . -t sample-flask-app:v1
4. Run the following command
   docker container run -d -p 5000:5000 sample-flask-app:v1
   The -d flag is short for --detach and runs the container in the background.
   The -p flag is short for --publish and maps port 5000 of the host to port 5000 of the Docker container
   docker container ls
5. Our Flask application is now running inside the container
6. To test it, open a web browser and go to
   http://localhost:5000. You should see the message Hello, World!

How to push docker image to dockerhub

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

1. Create account in docker hub
2. Create repository in docker hub (manjuldocker)
3. docker tag sample-flask-app:v1 manjulkrishnag/sample-flask-app:v1
4. docker push manjulkrishnag/sample-flask-app:v1

Pulling an image from Docker Hub

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Here are the steps:

1. Open your terminal or command prompt.
2. If you haven't already, log in to Docker Hub using the command
3. docker login
4. You'll be prompted to enter your Docker Hub username and password.
3. Use the docker pull command followed by the name of the Docker image you want to pull.
4. The format is docker pull <username>/<image-name>:<tag>.
5. If you don't specify a tag, Docker will pull the latest version of the image.

Here's an example:

docker pull coderkan/docker-demo:latest

This command pulls the latest version of the docker-demo image from the coderkan repository

After pulling an image, you can run it using the docker run command

For example:

docker run -t -p 8080:3000 coderkan/docker-demo:latest

This command runs the docker-demo image, mapping port 3000 in the container to port 8080 on your host machine.

Remember to replace <username>, <image-name>, and <tag> with the actual values for your specific use case.

Docker Quiz

What is a Container?

A. A standalone executable package that includes everything needed to run a piece of software

B. A type of virtual machine

C. A programming language

D. None of the above

What is a Container?

A. A standalone executable package that includes everything needed to run a piece of software

B. A type of virtual machine

C. A programming language

D. None of the above

What is Docker?

A type of virtual machine

B. A platform to automate the deployment, scaling, and management of containerized applications

C. A platform that uses OS-level virtualization to deliver software in packages called containers

D. A programming language

What is Docker?

A type of virtual machine

B. A platform to automate the deployment, scaling, and management of containerized applications

C. A platform that uses OS-level virtualization to deliver software in packages called containers

D. A programming language

What is Kubernetes?

A. A type of virtual machine

B. A platform to automate the deployment, scaling, and management of containerized applications

C. A platform that uses OS-level virtualization to deliver software in packages called containers

D. A programming language

What is Kubernetes?

A. A type of virtual machine

B. A platform to automate the deployment, scaling, and management of containerized applications

C. A platform that uses OS-level virtualization to deliver software in packages called containers

D. A programming language

Which command is used to run a Docker container?

A. docker start

B. docker run

C. docker exec

D. docker build

Which command is used to run a Docker container?

A. docker start

B. docker run

C. docker exec

D. docker build

What is Dockerfile used for?

A. To define an environment variable in a Docker container

B. To build Docker images from a script

C. To deploy applications on a Kubernetes cluster

D. None of the above

What is Dockerfile used for?

A. To define an environment variable in a Docker container

B. To build Docker images from a script

C. To deploy applications on a Kubernetes cluster

D. None of the above

What is a Pod in Kubernetes?

A. The smallest and simplest unit in the Kubernetes object model that you create or deploy.

B. A collection of Docker containers with shared storage/network.

C. Both A and B.

D. None of the above.

What is a Pod in Kubernetes?

A. The smallest and simplest unit in the Kubernetes object model that you create or deploy.

B. A collection of Docker containers with shared storage/network.

C. Both A and B.

D. None of the above.