Important DevOps Technologies (3+2+3days)

kubernetes by Google

for Deployment

DevOps is the blending of tasks performed by a company's application development and systems operations teams. The term DevOps is being used in several ways. In its most broad meaning, DevOps is an operational philosophy that promotes better communication between development and operations as more elements of operations become programmable.

DevOps (development and operations) is an enterprise software development phrase used to mean a type of agile relationship between Development and IT Operations. The goal of DevOps is to change and improve the relationship by advocating better communication and collaboration between the two business unit.

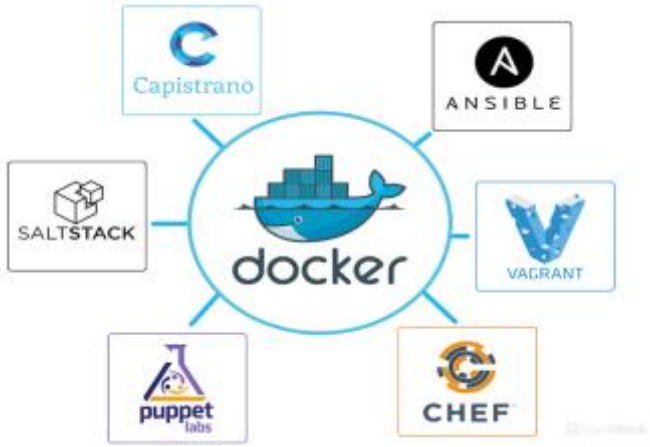
Why Do We Need DevOps in the Enterprise?

In an enterprise there is a need to break down silos, where business units operate as individual entities within the enterprise where management, processes and information are guarded. On the software development side -- and for those working in IT operations -- there needs to be better communication and collaboration to best serve the IT business needs of the organization.

One answer to breaking down enterprise silos is the move towards a DevOps-based culture that partner's developers with operations staff to ensure the organization achieves optimal running of software with minimal problems. This culture is one that supports a willingness to work together and share.

1. DOCKER Training content

Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.



Day 1

1. Introduction to Docker

- LIGHTWEIGHT

  Containers running on a single machine share the same operating system kernel; they start instantly and use less RAM. Images are constructed from layered filesystems and share common files, making disk usage and image downloads much more efficient.
- OPEN

  Docker containers are based on open standards, enabling containers to run on all major Linux distributions and on Microsoft Windows -- and on top of any infrastructure.

- SECURE BY DEFAULT

  Containers isolate applications from one another and the underlying infrastructure, while providing an added layer of protection for the application.

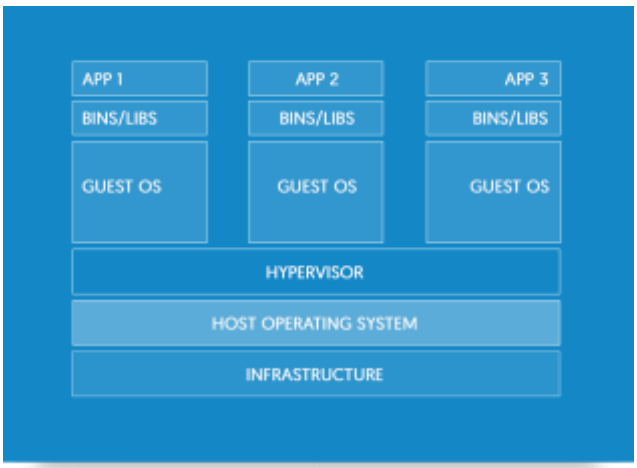
2. Container Vs Virtual Machines

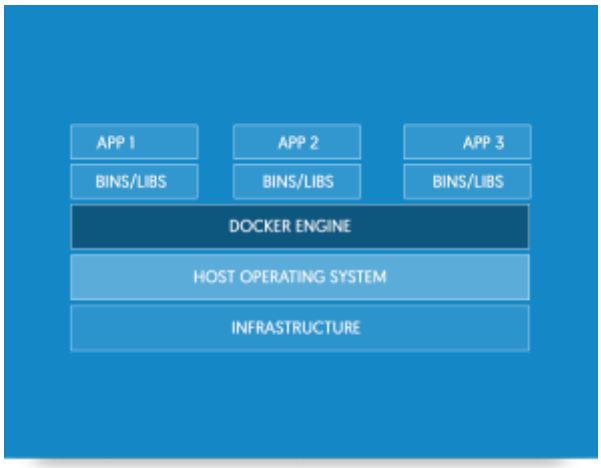
COMPARING CONTAINERS AND VIRTUAL MACHINES

Containers and virtual machines have similar resource isolation and allocation benefits -- but a different architectural approach allows containers to be more portable and efficient.

VIRTUAL MACHINES

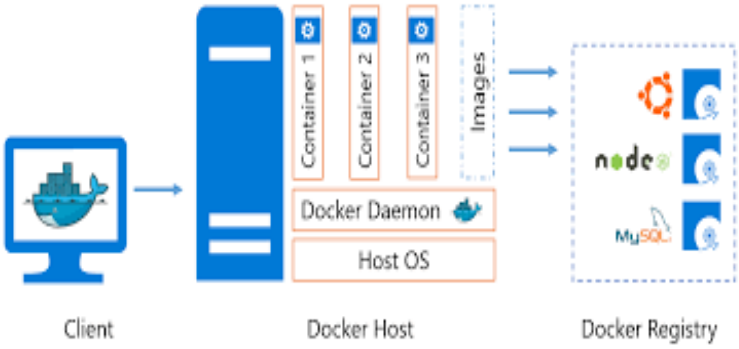
Virtual machines include the application, the necessary binaries and libraries, and an entire guest operating system -- all of which can amount to tens of GBs.



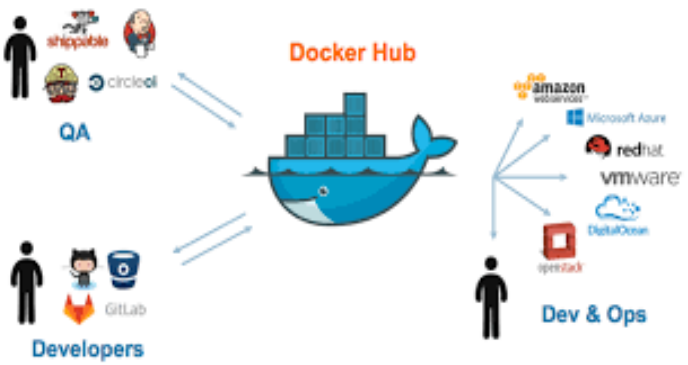


CONTAINERS - Containers include the application and all of its dependencies --but share the kernel with other containers, running as isolated processes in user space on the host operating system. Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.

3. Docker Architecture



4. The Docker Hub

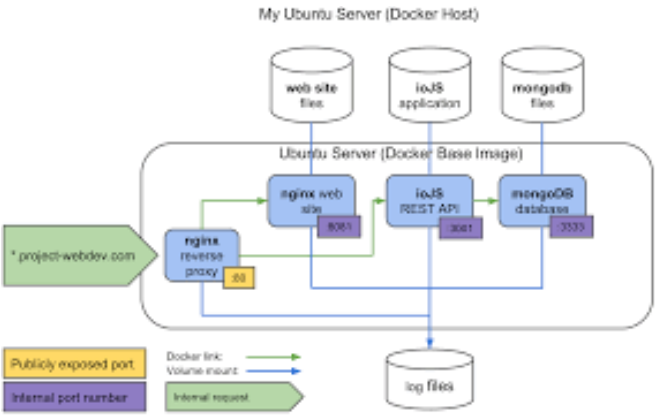


ask@eact-tech.com

5. Docker Installation

6. Creating First Image

7. Working with Multiple Images

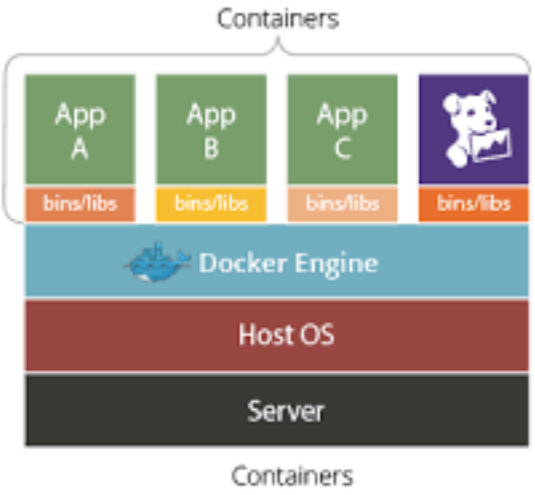


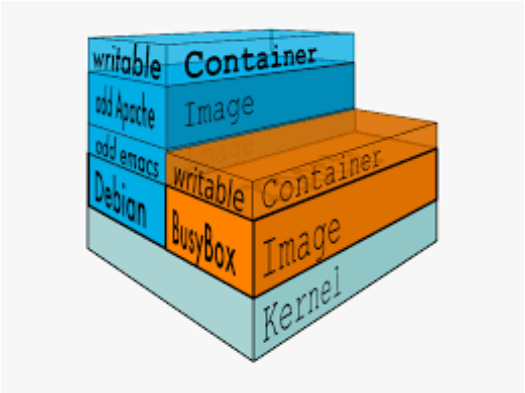
8. Packaging a customized container

9. Running container commands

10.Container Port Redirect

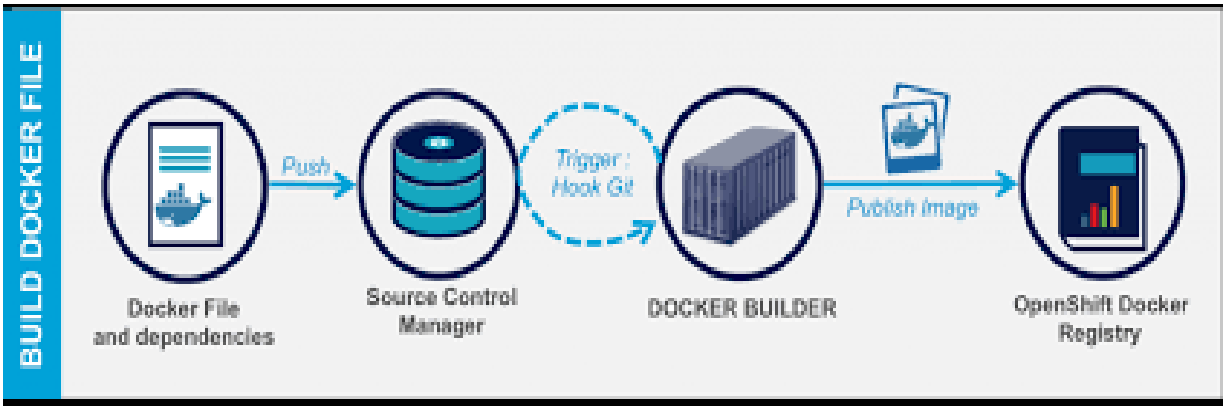
11.Container Snapshots





12.Attach to a Running Container

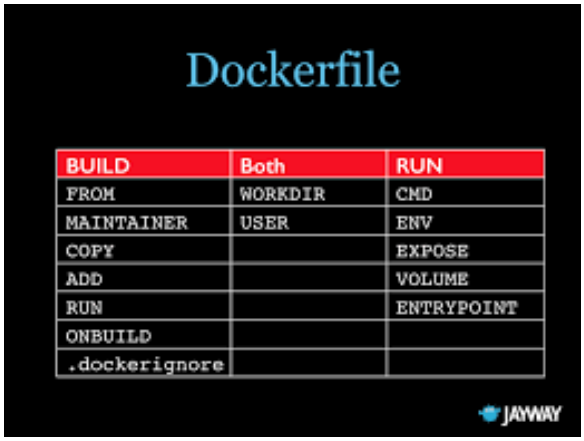
13.Removing Images



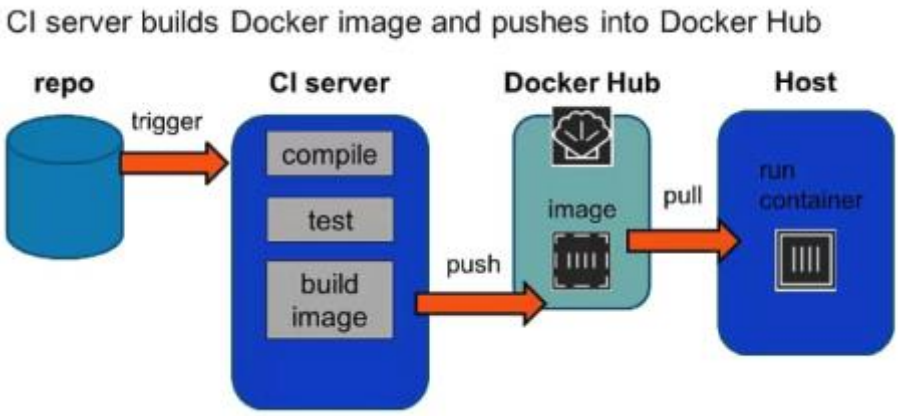
14.Understand Directory Structure

15.Services - on startup

16. Dockerfile

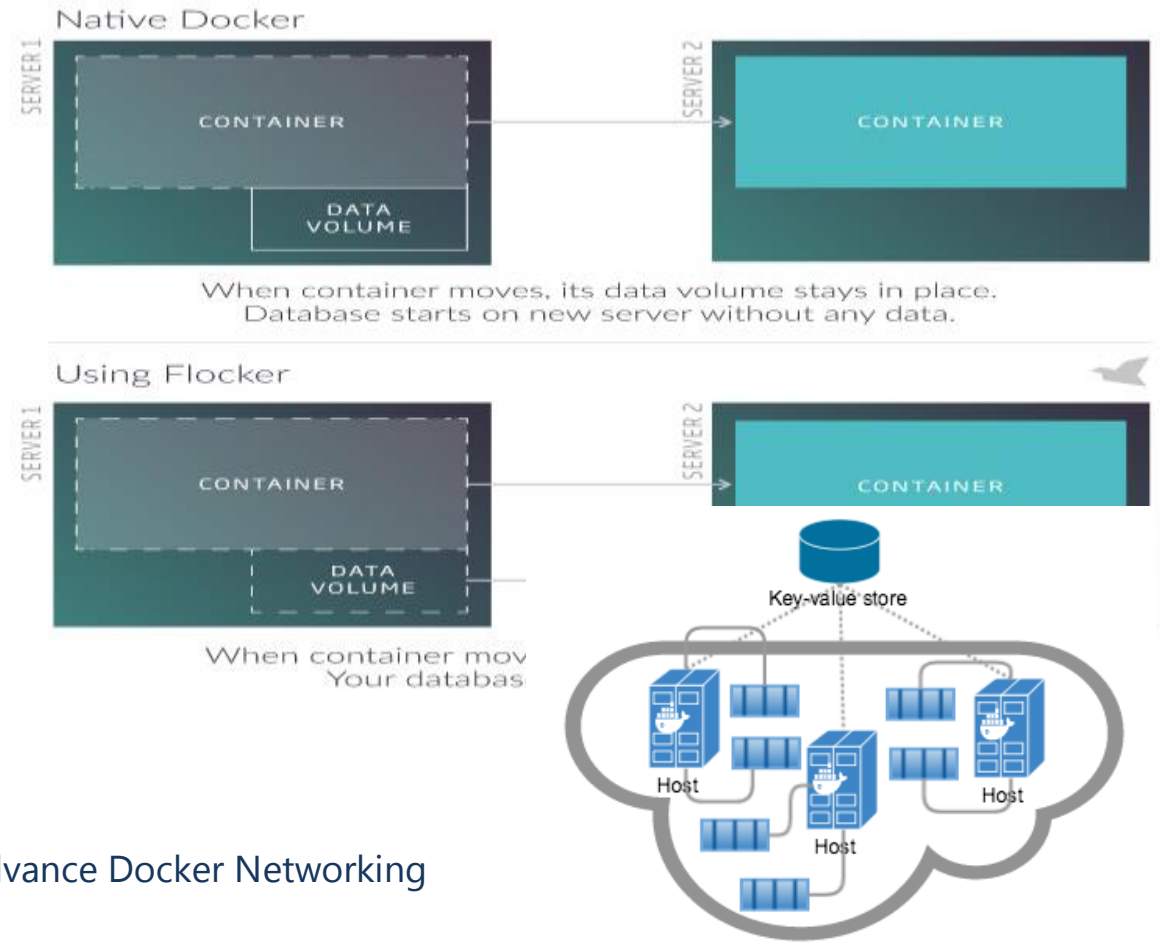


17.Pushing Images to Docker Hub



18.Adding External Content

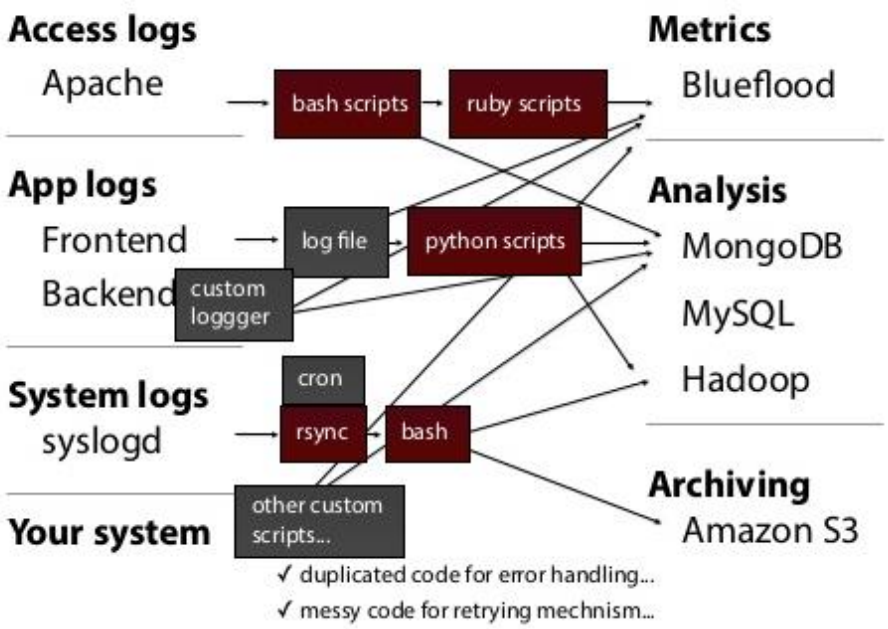
19.Image Volume Management



Day 2

1. Advance Docker Networking

2. Docker Logging

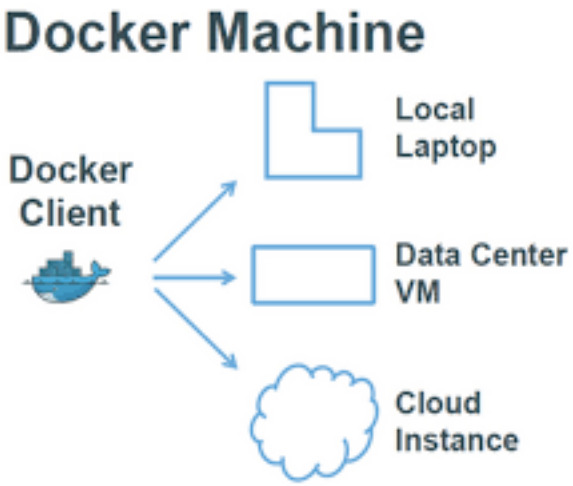


3. DOCKER Machine

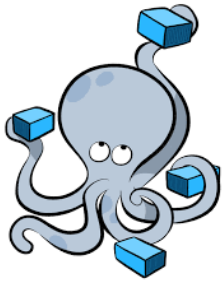
a. Purpose

b. Installation

c. Commands



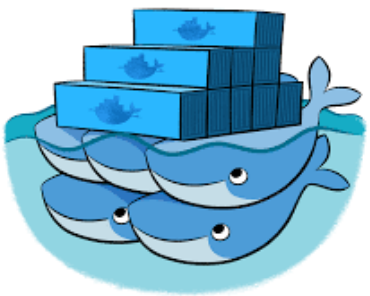
4. DOCKER Compose



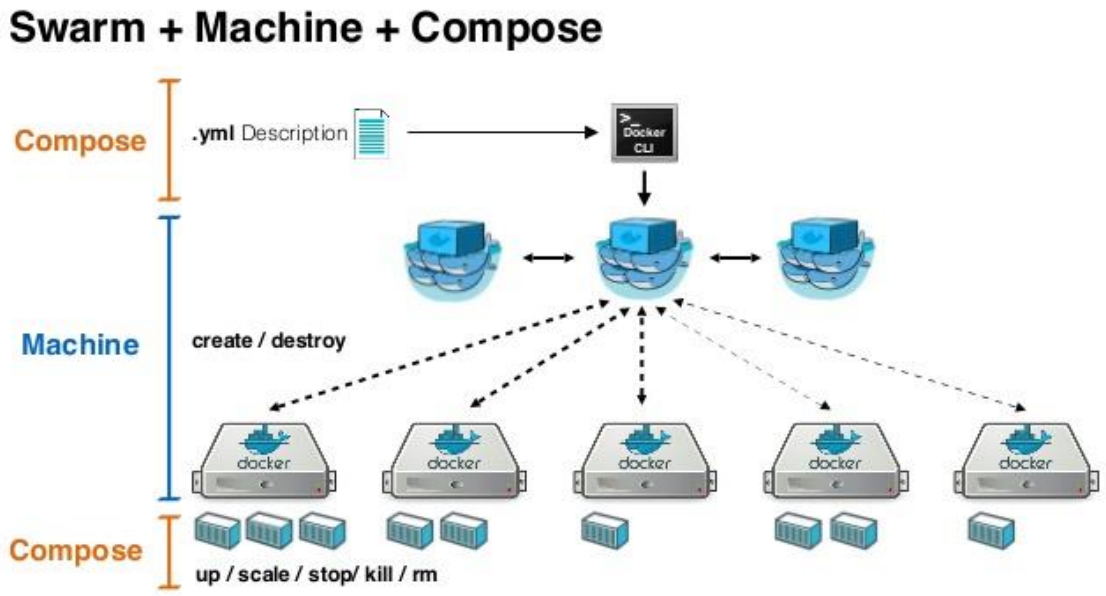
a. Configuring compose
b. Create a php application with mysql

5. DOCKER Swarm

a. Building a cluster
b. Launch Strategy



6. Docker Compose & Swarm Together



7. Docker Cloud

a. Advantage

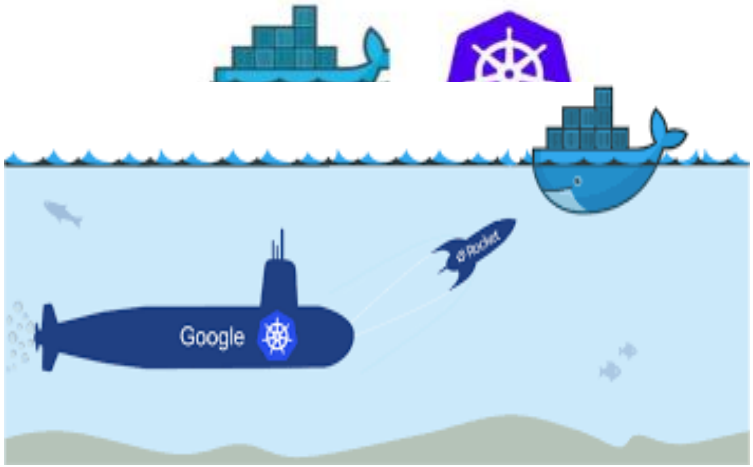
b. Integration

c. Demo

8. Kubernetes - Google

a. Introduction

b. Image Build



ask@eact-tech.com

c. Creating POD

d. Scaling

DAY - 3

9. Docker & Amazon Web Services ECS

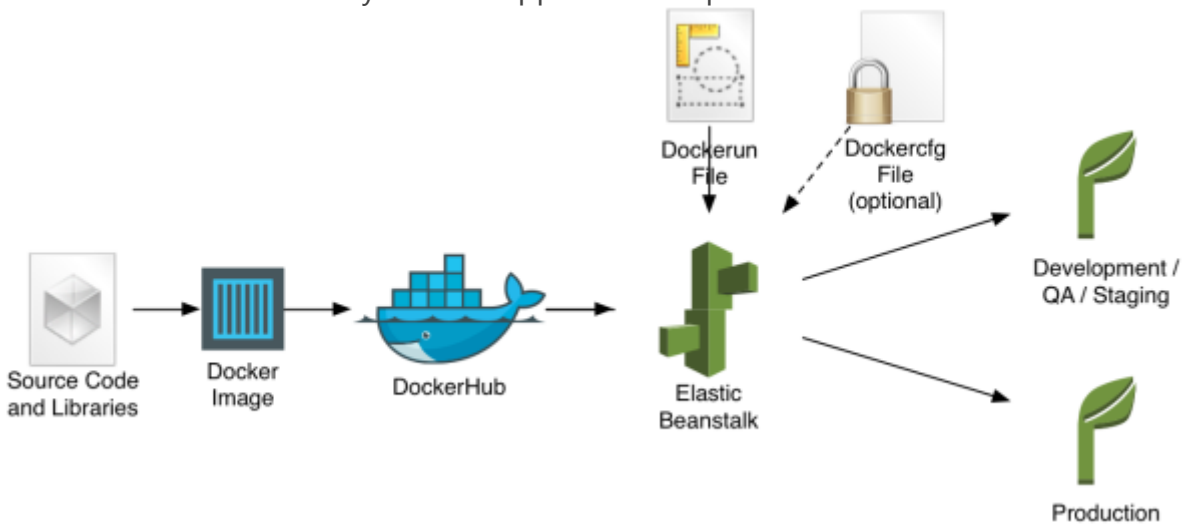
Amazon EC2 Container Service (ECS) is a highly scalable, high performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS eliminates the need for you to install, operate, and scale your own cluster management infrastructure.

With simple API calls, you can launch and stop Docker-enabled applications, query the complete state of your cluster, and access many familiar features like security groups, Elastic Load Balancing, EBS volumes, and IAM roles. You can use Amazon ECS to schedule the placement of containers across your cluster based on your resource needs and availability requirements. You can also integrate your own scheduler or third-party schedulers to meet business or application specific requirements.

10.Docker & Amazon Web Services Beanstalk

Elastic Beanstalk supports the deployment of web applications from Docker containers. With Docker containers, you can define your own runtime environment. You can choose your own platform, programming language, and any application dependencies (such as package managers or tools), that aren't supported by other platforms. Docker containers are self-contained and include all the configuration

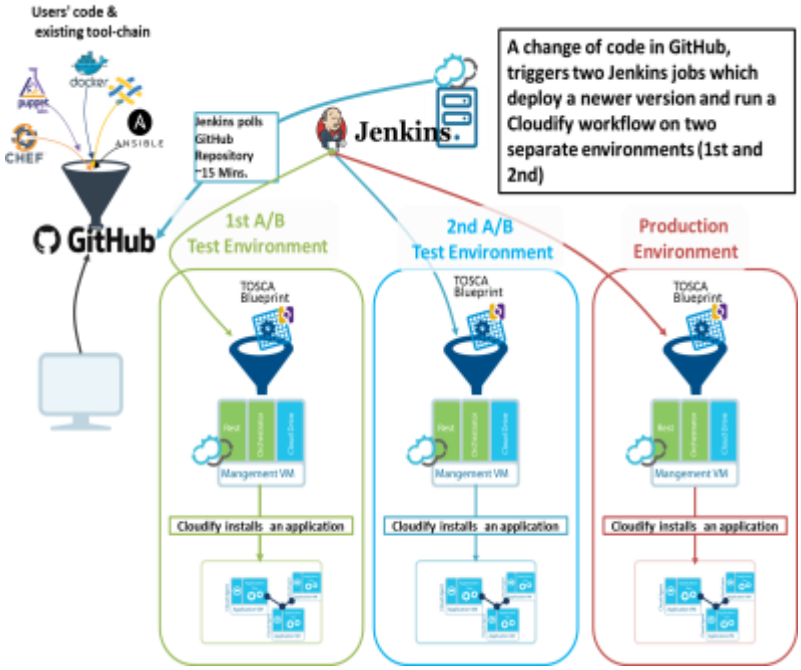
information and software your web application requires to run.



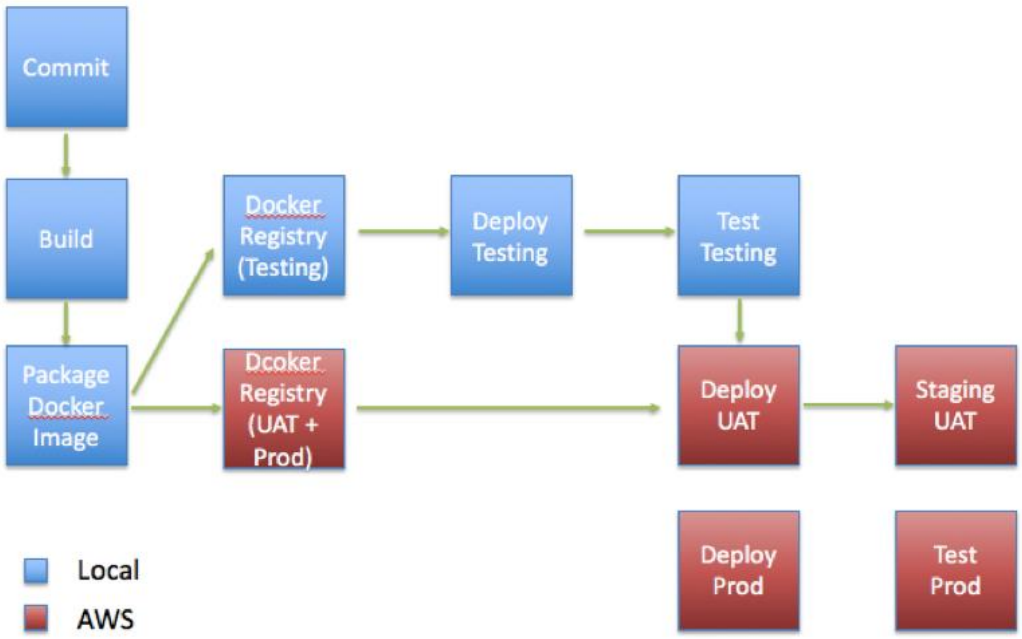
By using Docker with Elastic Beanstalk, you have an infrastructure that automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

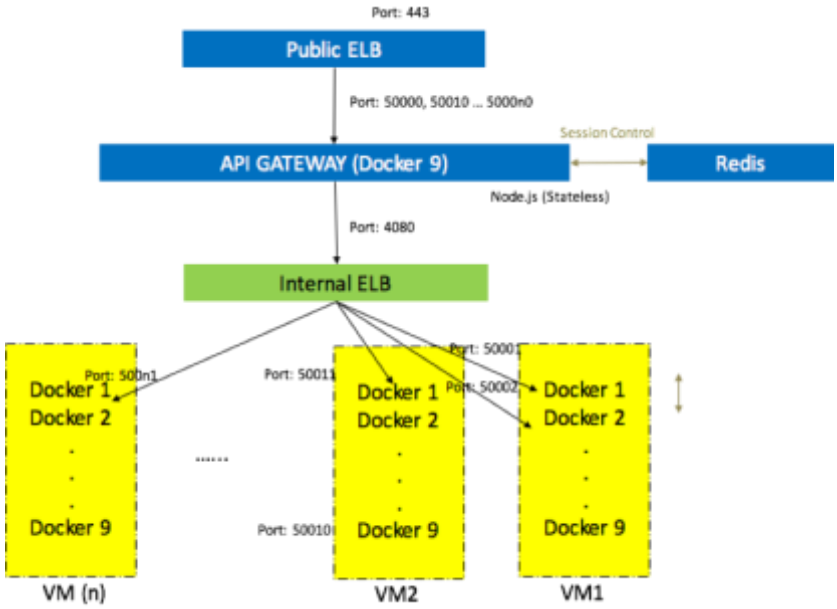
11.Docker & Cloud Platform Integration

12.Docker & CI Tools



13.Docker in DevOPS Practices





14.Practical Hands On Lab (2 Hours)

15.MCQ Exam (30 Questions)

* * * * * * * * * *

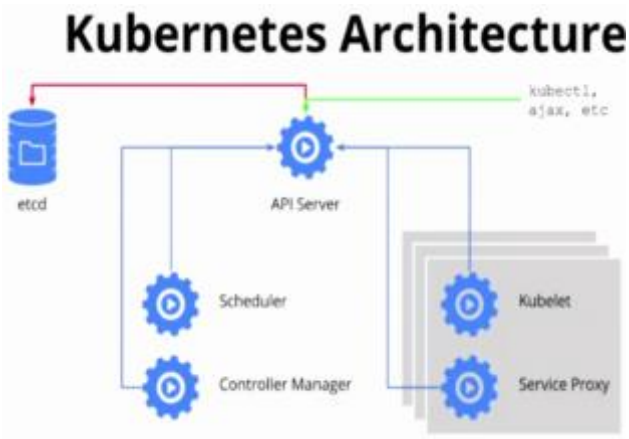
2. Kubernetes Training content

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

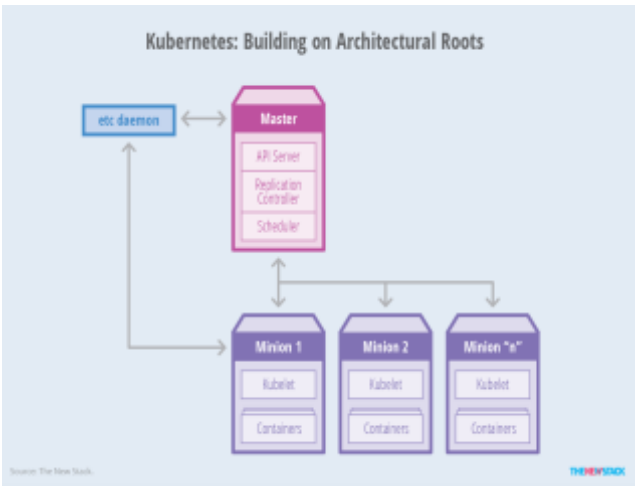
It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

1. Why Kubernetes
2. Kubernetes vs Docker

3. Design
3.1. Control Plane
3.2. Kubernetes Nodes

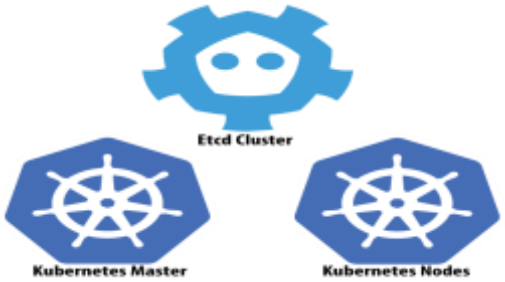


5. Components
5.1. Consistent Data Store
5.2. Controller Services
5.3. Worker Services

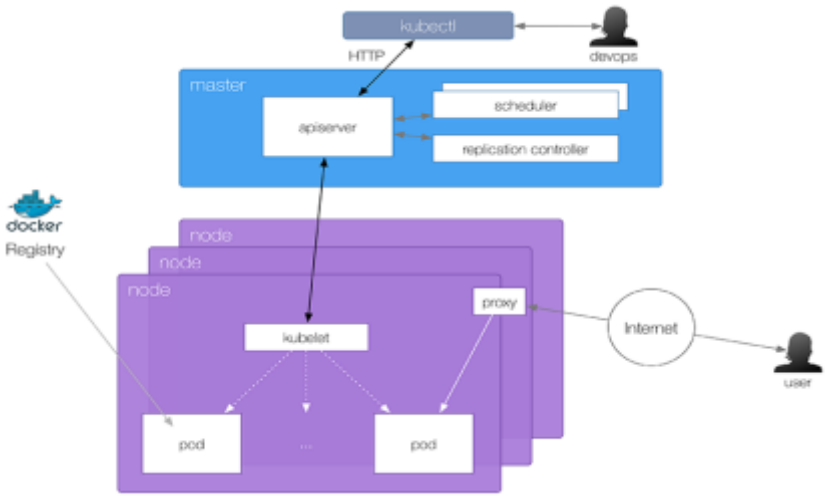


6. Kubernetes Nodes
6.1. Requirements
6.2. Kubernetes Machine Image 6.3. Launching Nodes
6.4. Configuring Docker Daemon
6.5. Kubernetes Kubelet
6.6. Advanced Network Mgmt.

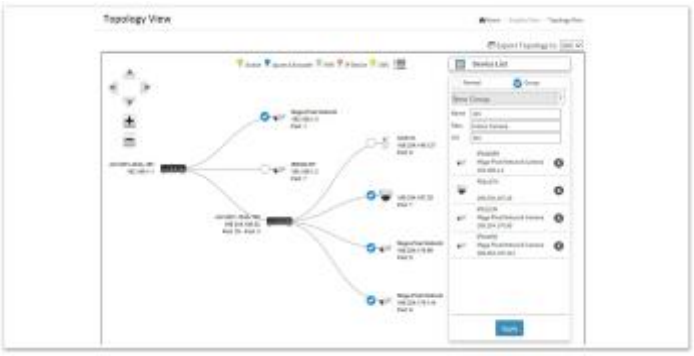
7. Bootstrapping
7.1. Etcd
7.2. API Server
7.3. Controller Manager
7.4. Scheduler
7.5. Health Checks

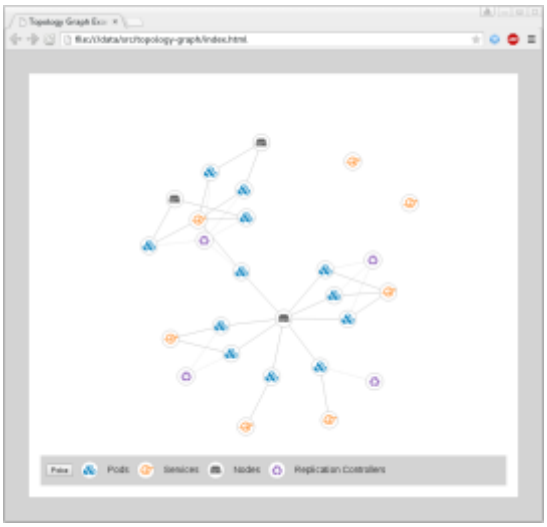


8.Working with Kubernetes Client

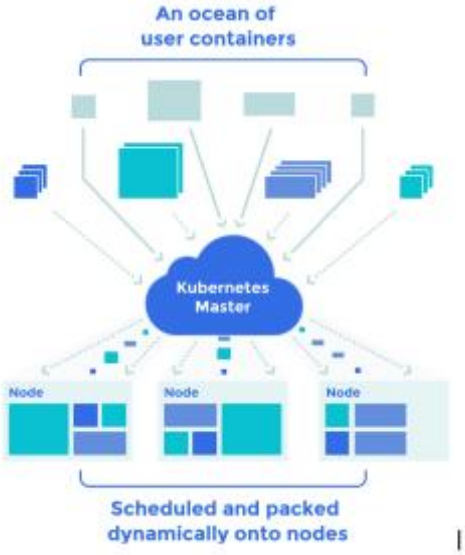


9.Cluster Addons
9.1. Kubernetes UI
9.2. Securely Exposing API Server





10.Container Resources

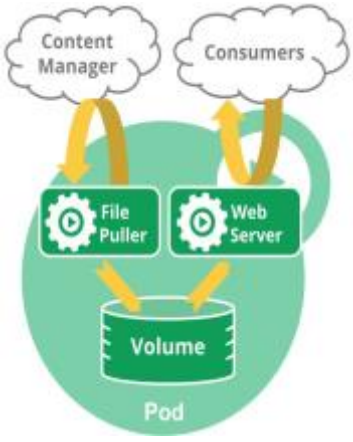


Container hooks provide information to the container about events in its management lifecycle. For example, immediately after a container is started, it receives a *PostStart* hook. These hooks are broadcast *into* the container with information about the life-cycle of the container. They are different from the events provided by Docker and other systems which are *output* from the container. Output events provide a log of what has already happened. Input hooks provide real-time notification about things that are happening, but no historical log.

10.1. System Containers
10.2. Application Containers
10.3. Building & Storing Images
10.4. Exploring influx db API
10.5. Limiting Resource Usage
10.6. Persistent Data with Volumes

3. Working with PODS

A *pod* (as in a pod of whales or pea pod) is a group of one or more containers (such as Docker containers), the shared storage for those containers, and options about how to run the containers. Pods are always co-located and co-scheduled, and run in a shared context. A pod models an application-specific “logical host” - it contains one or more application containers which are relatively tightly coupled — in a pre-container world, they would have executed on the same physical or virtual machine.



While Kubernetes supports more container runtimes than just Docker, Docker is the most commonly known runtime, and it helps to describe pods in Docker terms.

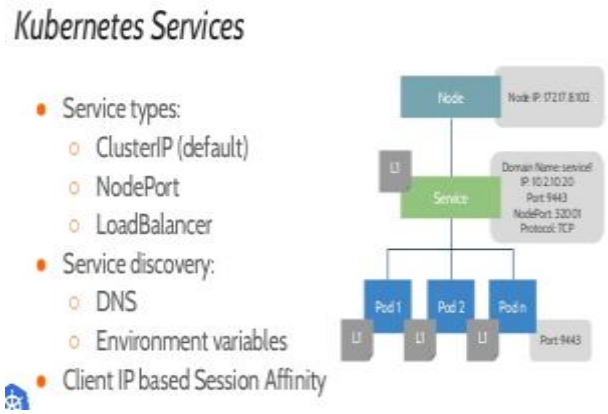
11.1. POD manifest
11.2. Running PODS
11.3. POD Health Checks
11.4. Resource Management
11.5. Volume Plugins

12.Labels & Annotations

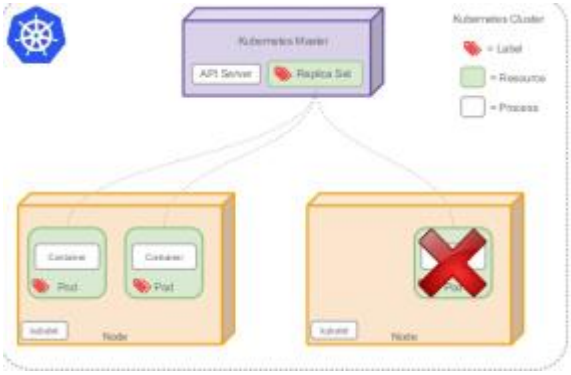
Labels are key/value pairs that are attached to objects, such as pods. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but which do not directly imply semantics to the core system. Labels can be used to organize

and to select subsets of objects. Labels can be attached to objects at creation time and subsequently added and modified at any time.

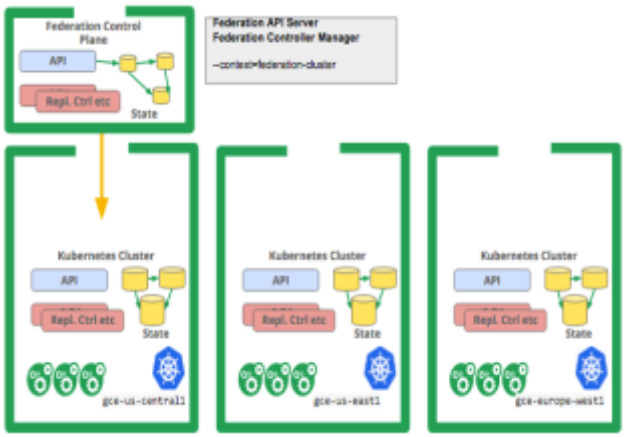
13. Using Services
    - 13.1 Endpoint Controller
    - 13.2 Service Proxies
    - 13.3 Service Specification
    - 13.4 Environment Variables
    - 13.5 DNS
    - 13.6 Node Port
    - 13.7 Load Balancer
    - 13.8. External IPs



14. Working with Replica Sets
14.1. Replica Set Specs
14.2. POD Template & Labels
14.3. Creating Replica Sets
14.4. Inspecting Replica Sets
14.5. Scaling
14.6. Deleting Replica Sets

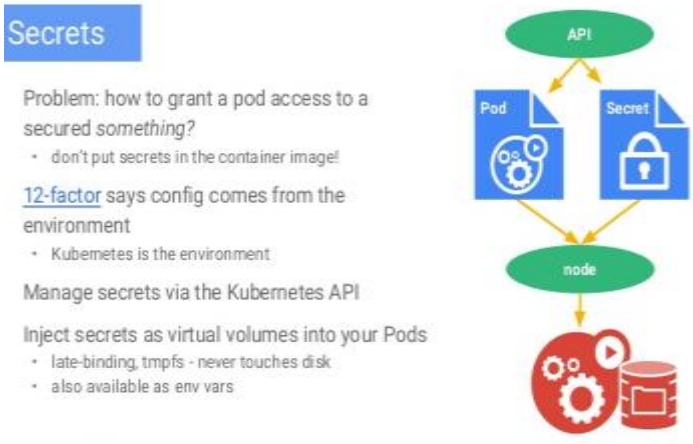


15. DaemonSets
15.1. DaemonSet Scheduler
15.2. Creating DaemonSet 15.3. Limiting Nodes
15.4. Updating DaemonSet
15.5. Deleting DaemonSet



16. Job Controller & Job Patterns

17. Working with Secrets



ask@eact-tech.com

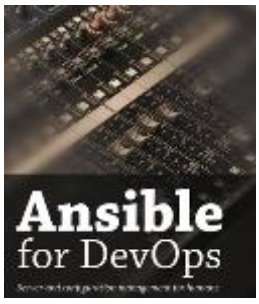
17.1 Creating Secrets
17.2 Consuming Secrets
17.3 Managing Secrets

18. Practical Lab - 1 & 2

19. MCQ

* * * * * * * *

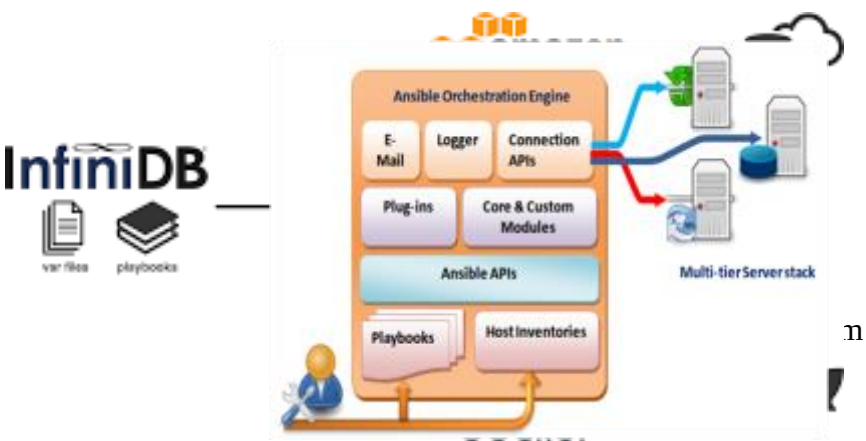
3. ANSIBLE training content



Ansible is a simple, but powerful, server and configuration management tool (with a few other tricks up its sleeve). This training helps those familiar with the command line and basic shell scripting start using Ansible to provision and manage anywhere from one to thousands of servers.

The training begins with fundamentals, like installing Ansible, setting up a basic inventory file, and basic concepts, then guides you through Ansible's many uses, including ad-hoc commands, basic and advanced playbooks, application deployments, multiple-provider server provisioning, and even Docker orchestration!

You need a tool that can act as the glue layer automating across



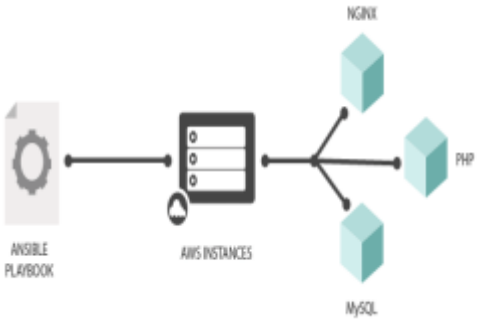
services and applications no matter where they are. Once one person on your team learns how to do something, they can capture their solution in an Ansible Playbook and enable everyone to use it.

DAY-1 to 3

1. Introduction to Ansible
2. Ansible & Competition
3. Introduction to YAML
4. Test Environment Set Up
5. Download & Installation
6. Ansible Configuration File
7. Python Dependencies
8. The Hosts File

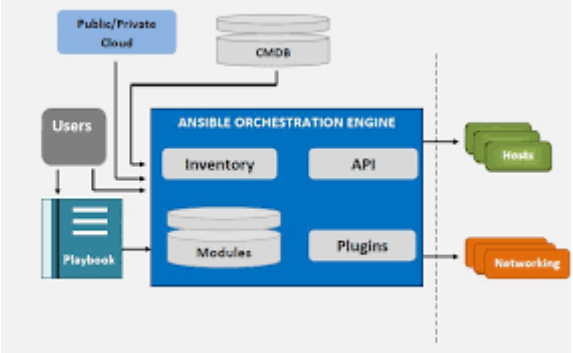
9. Hosts, System Ansible, Roles-override

10. Ansible Playbooks
    - 10.1. Ansible Account
    - 10.2. Ansible Command Line
    - 10.3. System Facts
    - 10.4. First Playbook
    - 10.5. Variables
    - 10.6. Target Section
    - 10.7. Hangles, Lookup, RunOnce
    - 10.8. Loops, Conditions, Notify, Vault
    - 10.9. Playbook Optimization



11. Ansible Modules
    - 11.1. SetUp
    - 11.2. Files
    - 11.3. Pause
    - 11.4. Yum
    - 11.5. Apt
    - 11.6. Cron
    - 11.7. Debug
    - 11.8. User
    - 11.9. More Modules

12. Ansible Roles
    - 12.1. Directory Structure
    - 12.2. Conditional Execution
    - 12.3. Variable Substitution
    - 12.4. Handlers
    - 12.5. Using Notification
    - 12.6. Waiting for Events



12.7. Delegate to
12.8. Local Action

13.Ansible Command Line

13.1. Installing Packages
13.2. Services & Hosts
13.3. Commands & Shells
13.4. Managing Users
13.5. Managing Cron Jobs

14. Labs

* * * * * * * *