Deployment

Ahmed Hesham

What Is Deployment?

- **Running an application on a target device**
- **Types of deployment:**
  - Local
  - Remote

Why Deploy Applications Remotely?

- Part of serverless architecture
- Abstracts work needed to managing servers
- Dependencies should be packaged with the application

Issues with deployment

- **No guarantee on where application will be deployed**
- **Hardware support is not guaranteed:**
  - Different CPU architectures (x86 vs ARM)
  - Special hardware e.g. GPUs
- **Application may be hosted locally to cater for special hardware requirements**

Packaging Applications

- **There are two main methods of packaging applications:**
  - Virtualization
    - Hosted
    - Bare Metals
  - Containerization

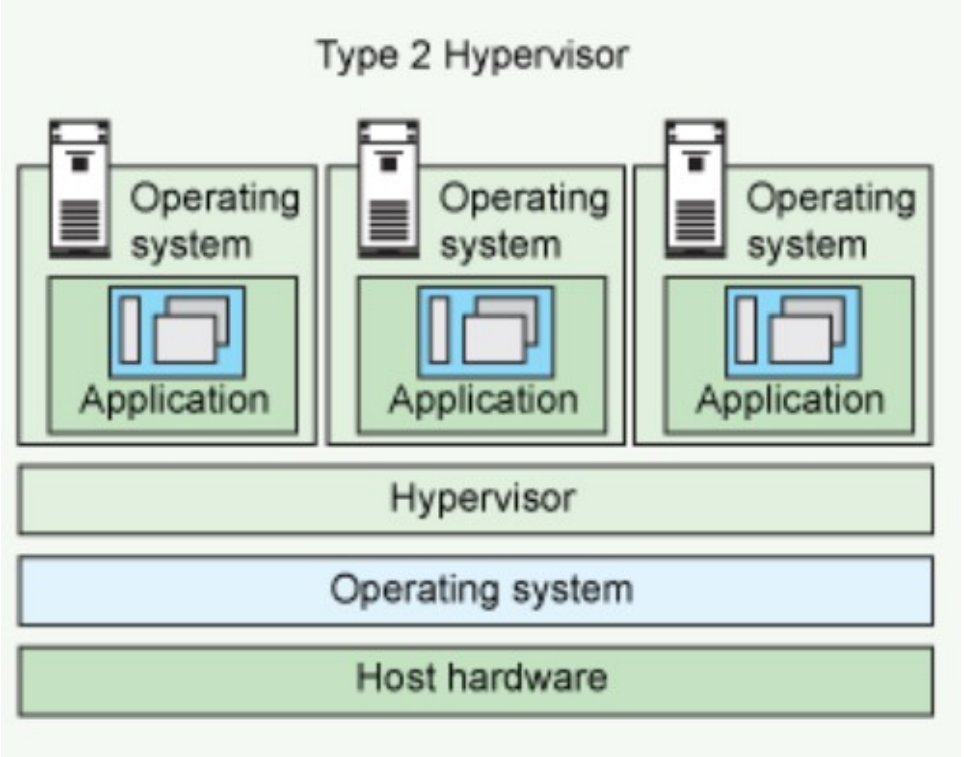
What Is Virtualization?

- **Running a virtual instance of a computer**
- **Requires an OS inside the instance**

- **Pros:**
  - High security (CPU level)
  - Multiple OS per machine
- **Cons:**
  - Large in size

What Is Hosted Virtualization?

- Uses a Hypervisor to translate system calls between guest and host OS
- Example: Virtualbox

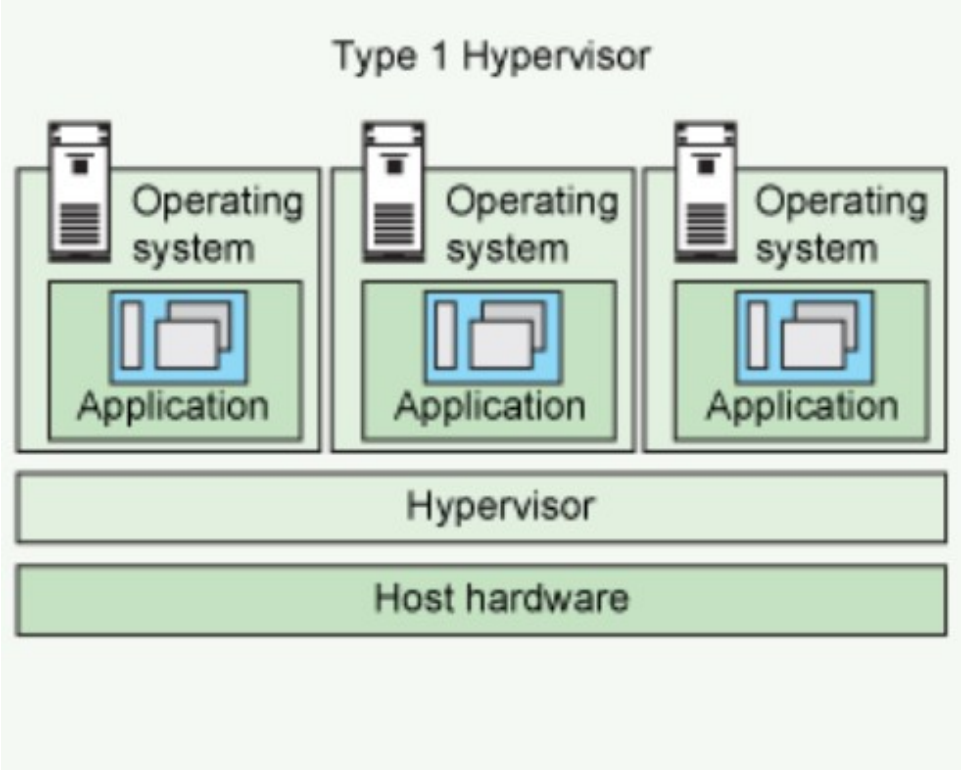


Hosted Virtualization Pros/Cons

- **Pros:**
  - Greater host platform support through translation layer (Hypervisor)
  - Allows for different architectures OS
- **Cons:**
  - Limited CPU functionality
  - Requires a hosting OS to run guest OS

What Is Bare Metals Virtualization?

- **Direct access to hardware**
- **Hypervisor provides drivers needed for hardware**
- **Example:**
  - Linux KVM
  - Windows Hyper-V



Bare Metals Pros/Cons

- **Pros:**
  - Higher performance than hosted
  - Set allocated hardware, no sharing
- **Cons:**
  - Management overhead
  - Guest is limited by hardware architecture

Why Virtualize?

- Application is dependent on a specific OS
- OS must be packaged with application

What Is Containerization?

- **OS level virtualization**
- **Applications partitioned in isolated user spaces**
- **Examples:**
  - Docker
  - Kata Containers
  - Apache Mesos

Containerization Pros/Cons

- **Pros:**
  - Lightweight, no OS present
  - Scheduled by OS as a normal process

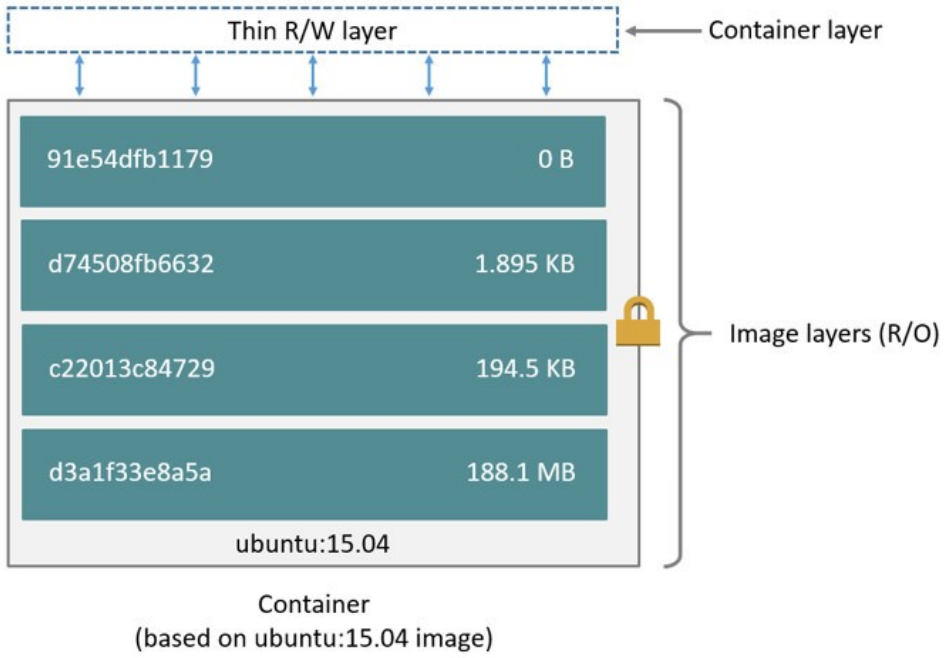
- **Cons:**
  - Depends on OS to provide isolation
  - Less secure than VM

Docker

- **A containerization solution**
- **Uses Linux kernel containers (LXC) to operate**
- **Modes of operation:**
  - Container Host: this container uses the libraries and tools provided by the host OS normally
  - Container OS: this container comes packaged with the tools of a specific OS to use as a dependency for example: CentOS on an Ubuntu host

How Docker Works

- A Docker image is a set of layers stacked up to form the container
- Each layer is *Read Only* with the exception of the last layer for writing small files



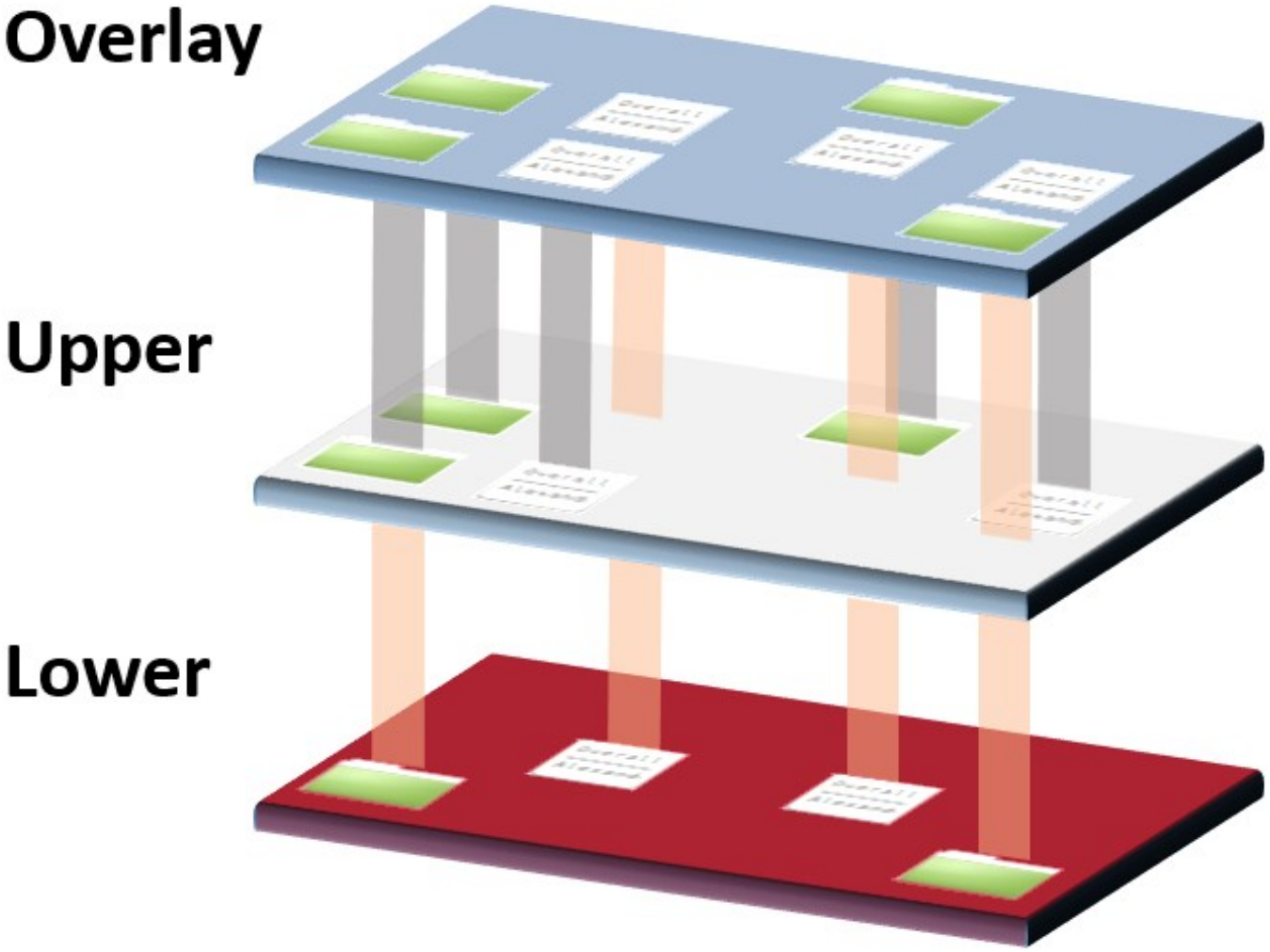
How Docker Works – continued

- Reduces layer duplication on disk through using Copy On Write (CoW)
- Docker works best with CoW filesystems e.g. BTRFS, ZFS and AppleFS
- In the case of other filesystems being used, OverlayFS is used to mimic CoW

What Is OverlayFS?

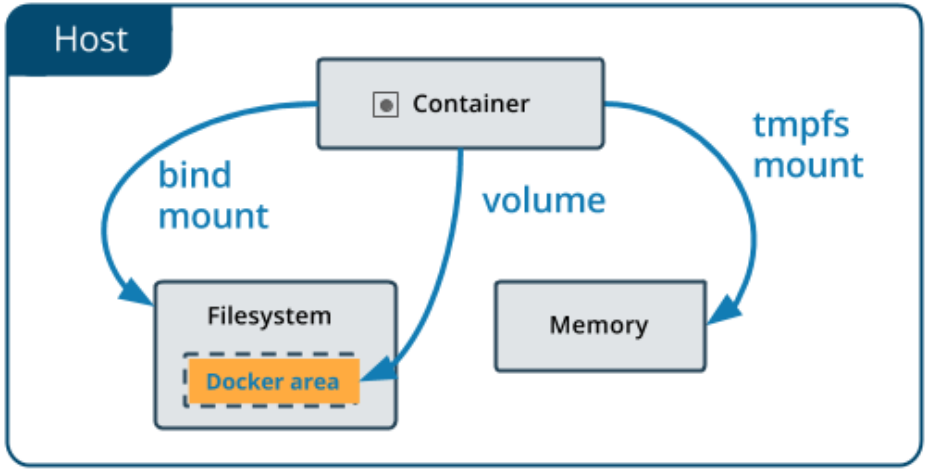
- **Union mount filesystem: can mount multiple directories to appear as one**
- **It emulates CoW by separating directories into layers:**
  - Lower layer: not modified
  - Upper layer: stores any new files
  - Overlay layer: temporary working layer

OverlayFS – continued



Docker Volumes

- Containers have only a limited writing space
- To allow for larger storage areas, volumes are used
- Volumes can be shared between host machines as they are managed by Docker itself



Use Case Of Docker Volumes

- Media Server
- Databases

Managing Docker Containers

- Server can host hundreds of Docker containers
- Unfeasible to manage each by hand
- Orchestration tools automate management of containers

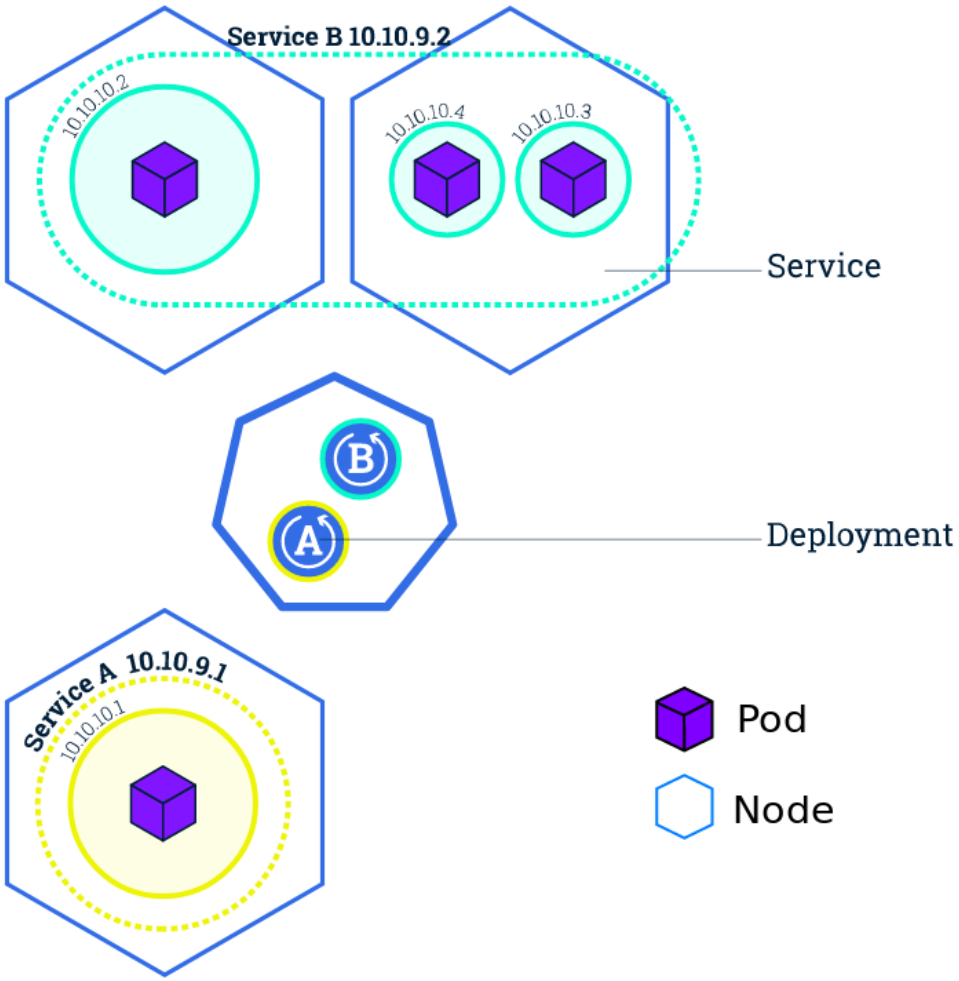
What Is Kuberenetes?

- An open source container orchestration tool developed by Google
- Can replicate or migrate containers to other servers depending on the resource requirements
- Clusters servers to view them as one instance

Kubernetes Terminology

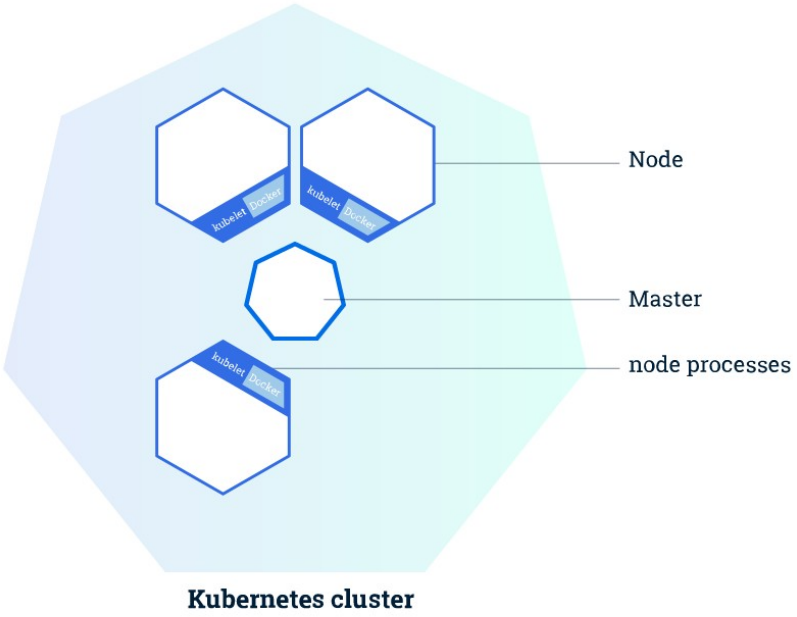
- **Pods:**
  - Most basic unit of computation
  - Can host multiple containers
  - Guarantees colocation of containers for network access
- **Services:**
  - Set of pods that together form an application
  - Example: media micro-service might be composed of media server and media micro-service

Kubernetes Architecture



Kubernetes Cluster

- **Master and worker architecture**
- **A node can be VM or a physical computer**
- **Each node has:**
  - Kubelet: interface to node
  - Docker interface



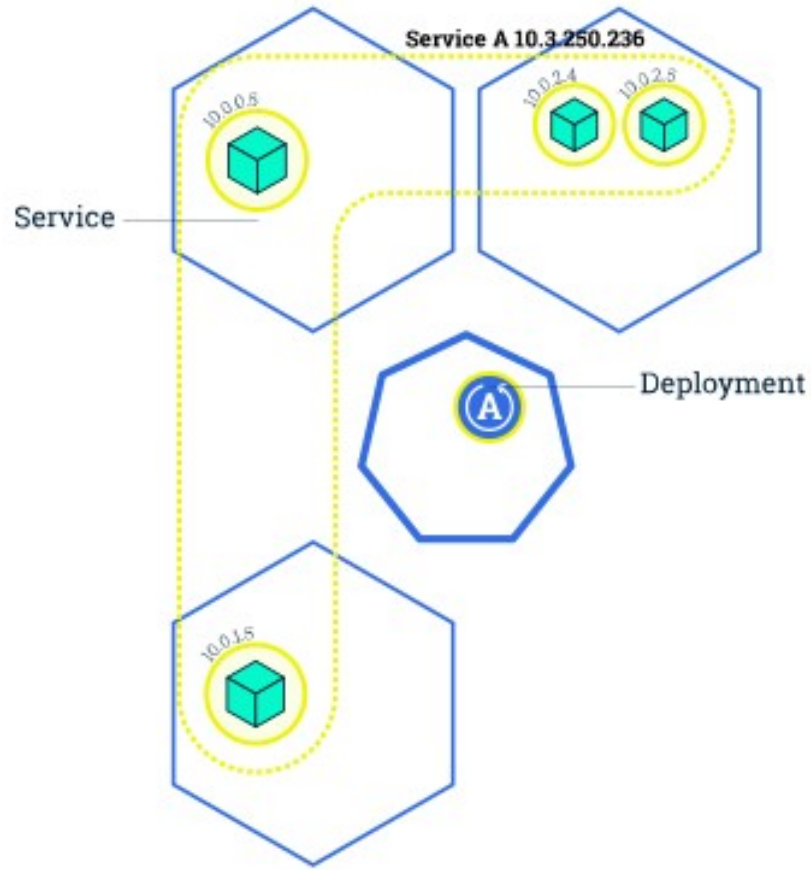
Deploying On Kubernetes

- Allows developers to specify the resources per container to be used
- Master would find the most appropriate node to be used and deploy on it
- If a node fails, the master has a registry of deployed pods and redeploys them on live nodes (self-healing)

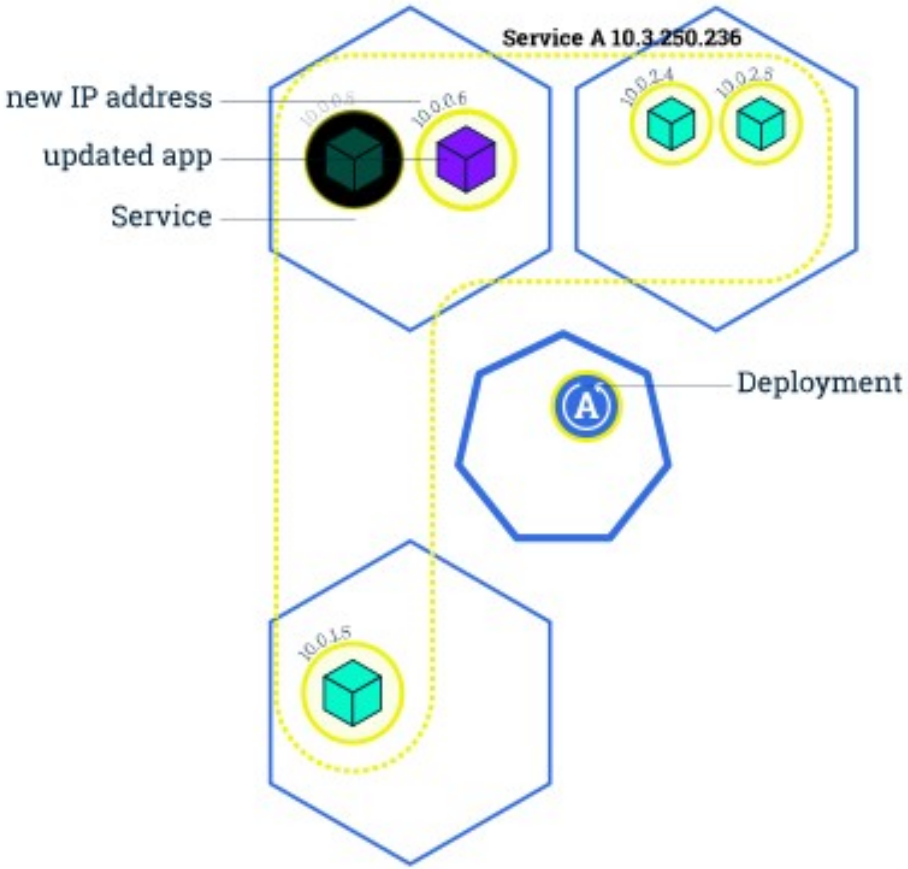
Rolling Updates

- Allows for rolling updates to happen to allow for zero down time
- Instances are updated incrementally till all the instances are updated

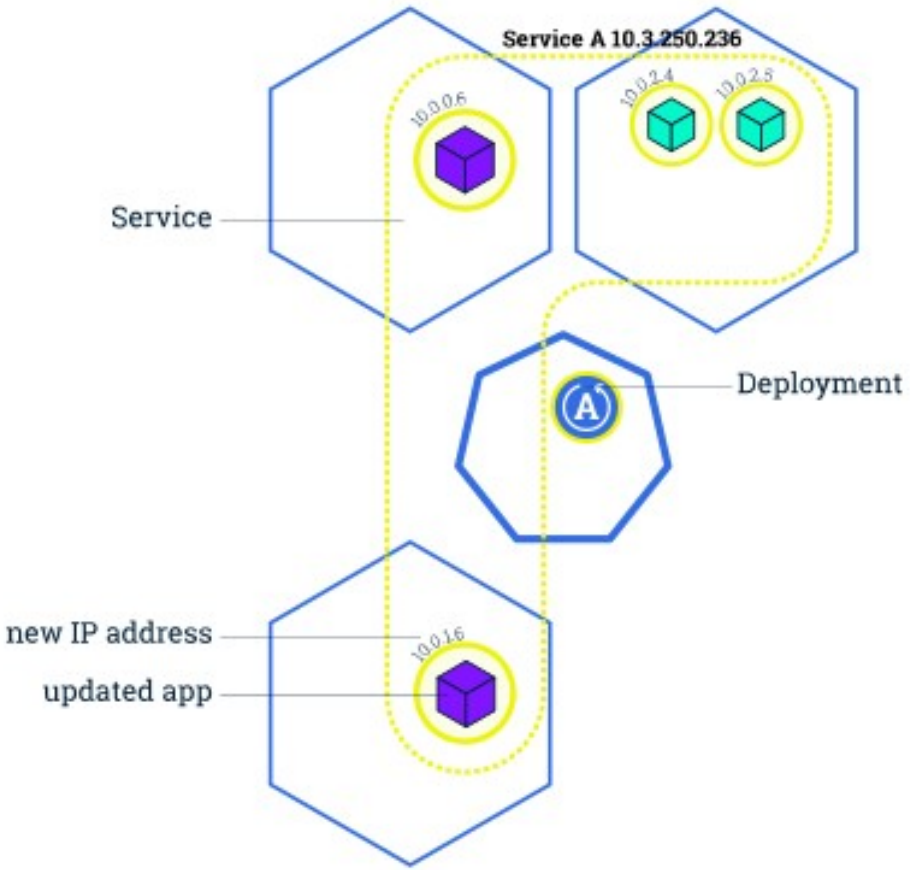
Rolling Updates – I



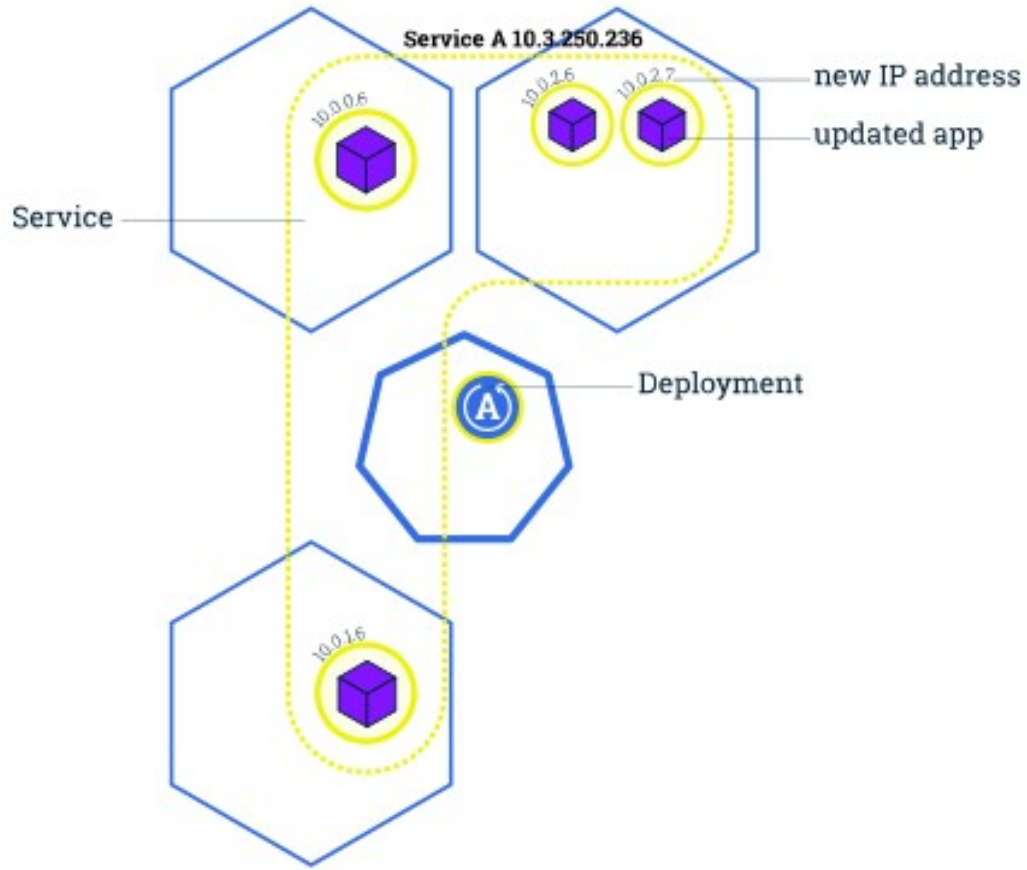
Rolling Updates – II



Rolling Updates – III



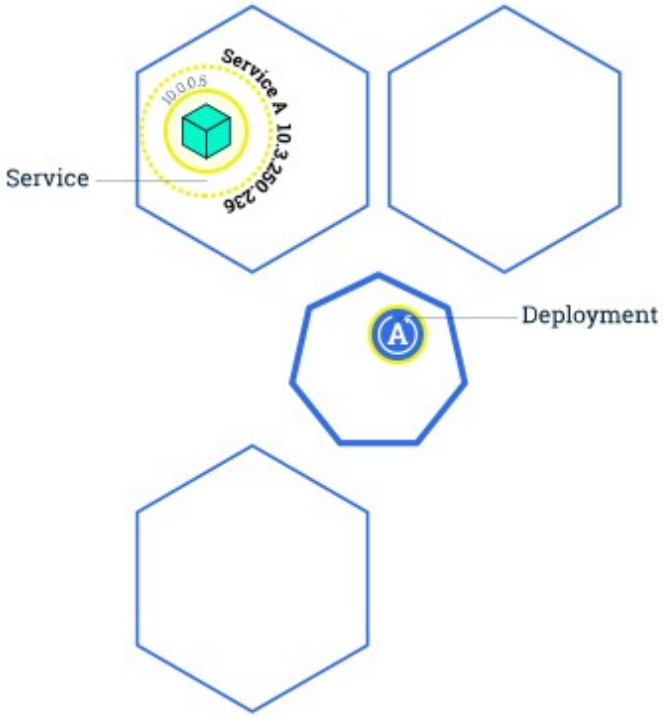
Rolling Updates – IV

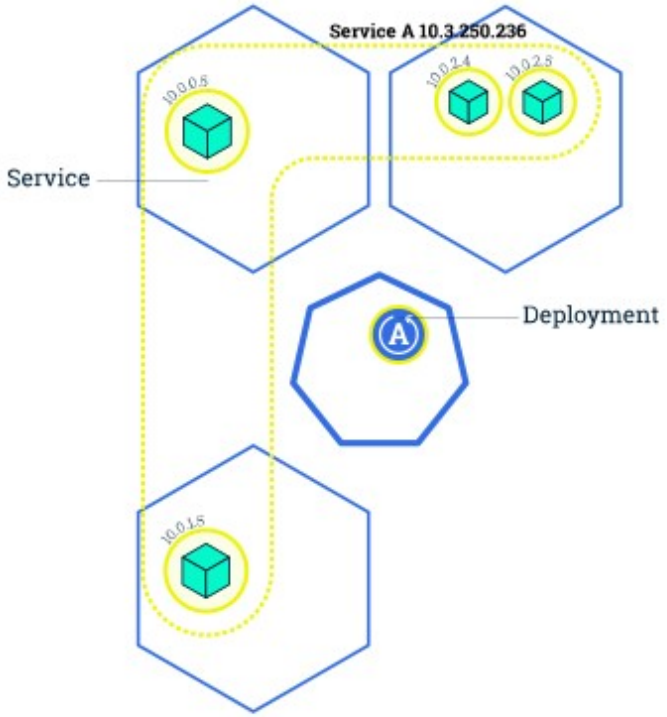


Scaling With Kubernetes

- **Kubernetes provides two IPs to access an instance:**
  - Public IP: accessed through the internet
  - Private IP: only visible on the local subnet between instances
- **When replicating:**
  - New private is assigned to the instance
  - Public IP unchanged
  - Proxy routes connections internally

Scaling With Kubernetes





Any questions?