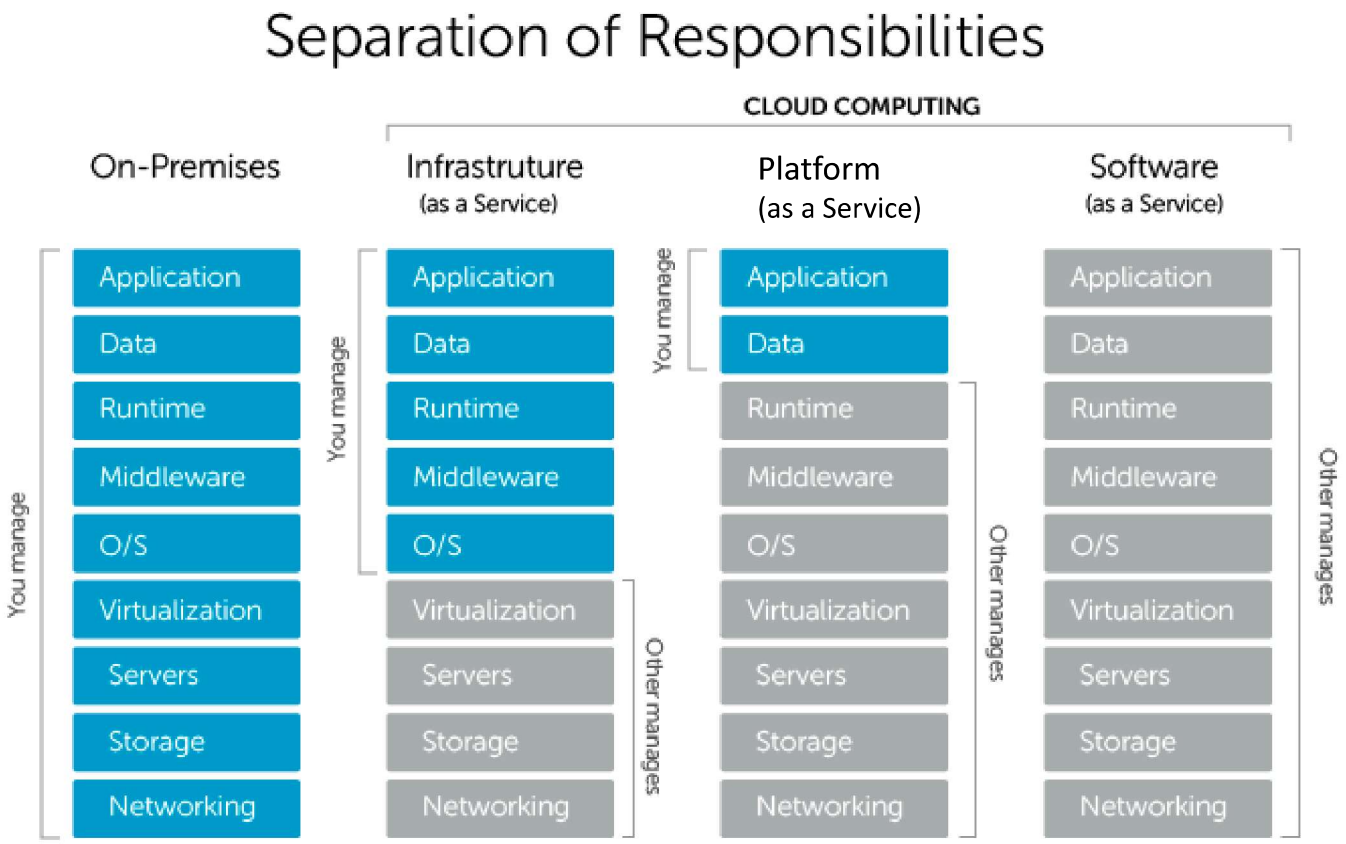
NYU COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Data Communications & Networks

Session 3 – Introduction to K8s

Dr. Jean-Claude Franchitti

Cloud Computing Models



Monolithic vs Microservice

Monolithic Service:

- Built as a single unit
- Difficult to understand
  - Can become too big to manage (no developer can understand the entire code)
- Fixes are difficult
  - (now you need to re-deploy the whole app)

Microservice:

- Built as multiple units, separated by logical functionalities
- Easier to understand and manage
- Fixes are easier and only requires deployment of specific microservices

Containerization

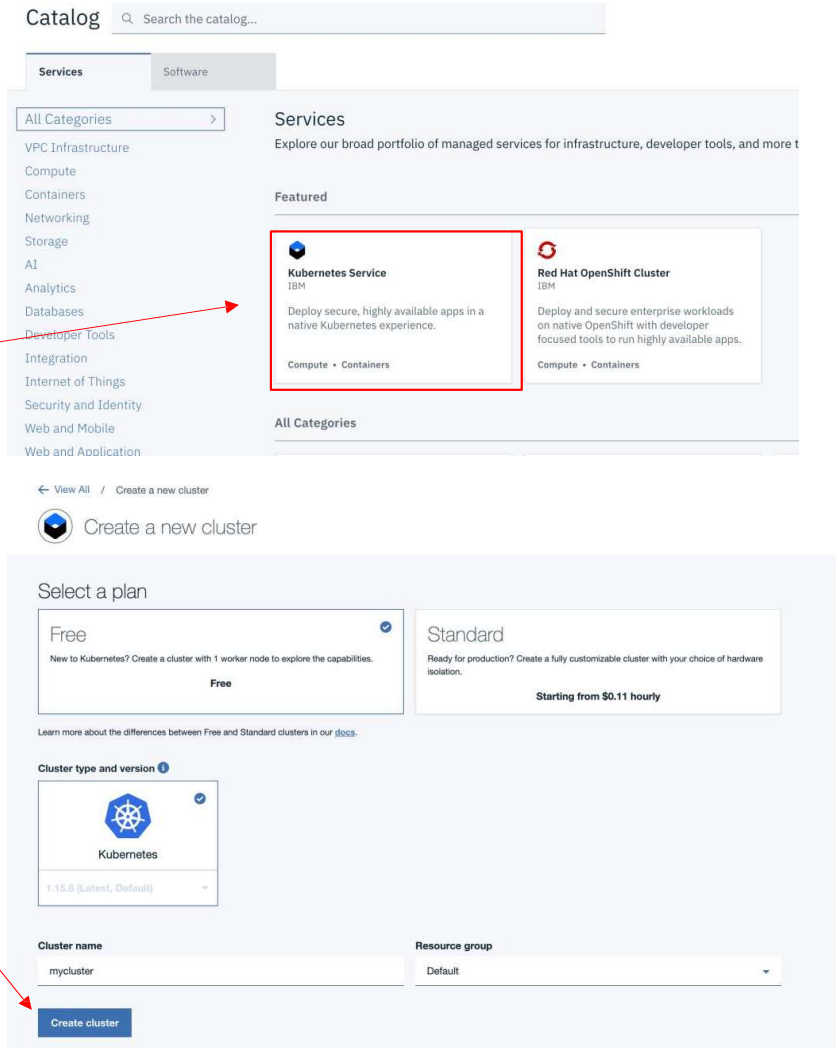
- Traditionally applications are deployed in VMs
  - Difficult to reproduce (configuration change from VM to VM)
  - Any change in VM results in affecting your application
- Containerization allows packaging of your application into a lightweight container with its own operating system
- Container can run in anywhere (no specific VM)
- Docker is the most popular containerization platform
- Containerization paves the way for microservice-based application development and deployment

Kubernetes (K8s)

- Open Source
- Automates deployment
- Scales up and down
- Facilitates updates
- Facilitates management

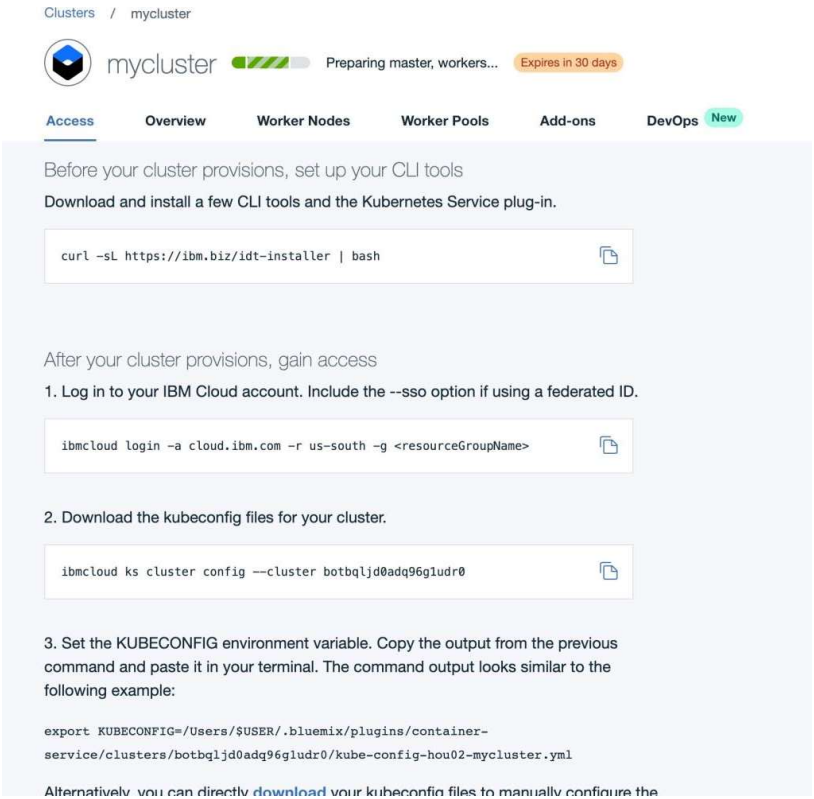
Cluster Creation

- Visit cloud.ibm.com
- Click “Catalog” from the top menu
- Select K8s service
- Click “Create” button in the bottom
- Select “Free” and hit “Create Cluster”



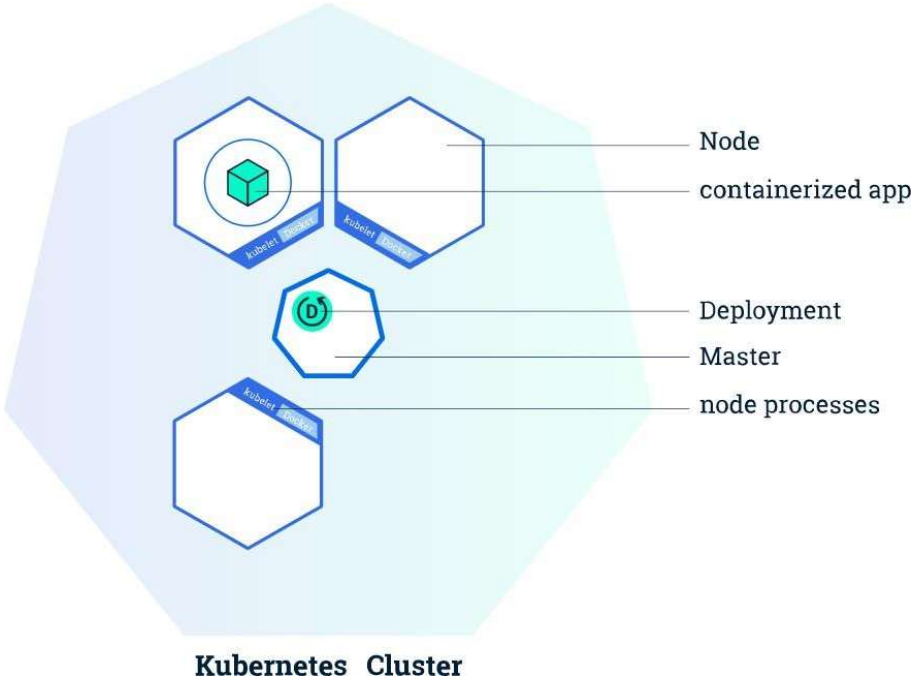
Environment Setup

- Follow the instructions in the “Access” tab:



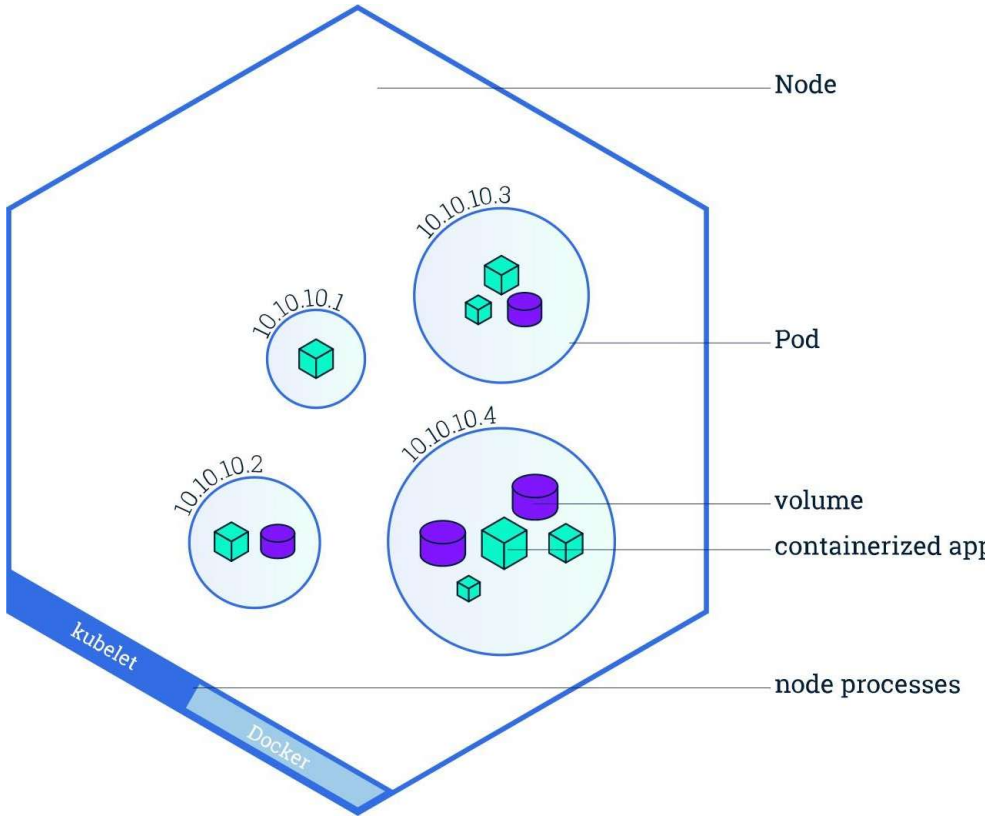
Deployment

- kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1
- kubectl get deployments
- kubectl describe pod [POD_NAME]
- kubectl excec [POD_NAME]



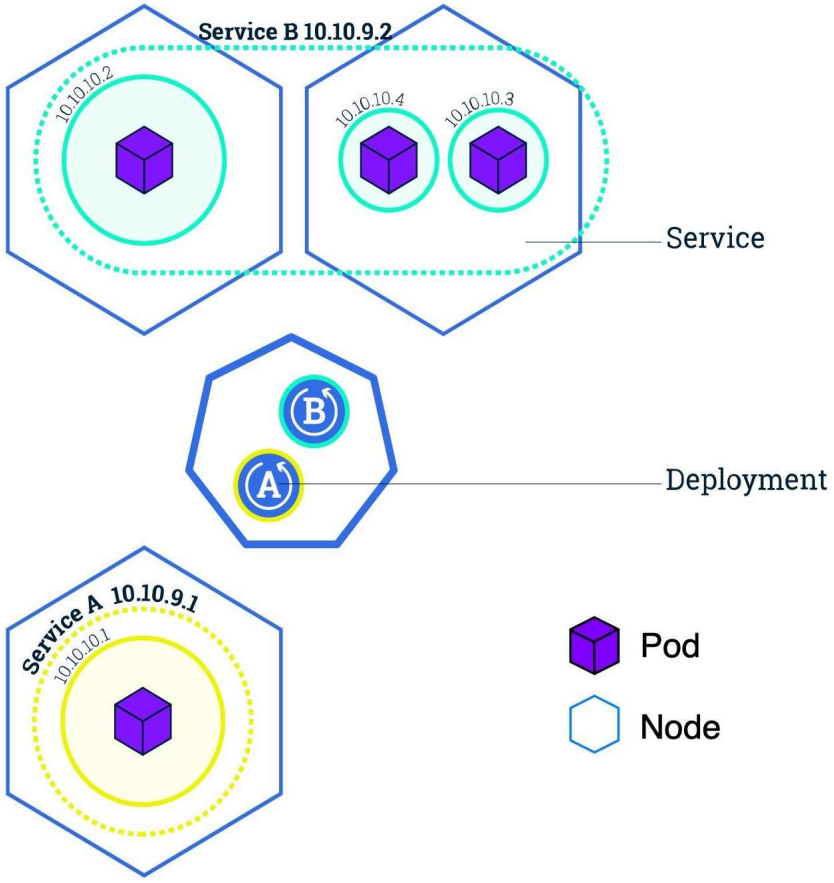
Exposing Services

- Pods are mortal
- A ReplicaSet is used to create new ones
- However when pod dies, IP will be lost as well
- Services are used to keep things functioning as configured
  - Set of pods and policy to access them as .yaml file



Services

- *ClusterIP* (default) - Exposes the Service on an internal IP in the cluster.
- *NodePort* - Exposes the Service on the same port of each selected Node in the cluster using NAT. Makes a Service accessible from outside the cluster using <NodeIP>:<NodePort> (superset of ClusterIP).
- *LoadBalancer* - Creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service (superset of NodePort(.
- *ExternalName* - Exposes the Service using an arbitrary name (specified by externalName in the spec) by returning a CNAME record with the name. No proxy is used. This type requires v1.7 or higher of kube-dns.



NYU COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Data Communications & Networks