Chapter1: Introduction

Monolithic vs Microservice Condition to declare a service as microservice. (must have db Connection ) How to microservices are communicated with each other.(by API Call http or curl) strangler pattern Monolithic vs microservice Architecture(Note: In microservice each component needs to follow Devops Cycle)

Chapter2: CNI

What is CNI(Container networking interface) CNI (Container Network Interface), a [Cloud Native Computing Foundation](https://cncf.io) project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins.

* All Pods can communicate with all other Pods without using NAT (network address

translation)

* All Nodes can communicate with all Pods without NAT
* The IP that Pod sees itself is the same IP that others see it as

Rule for creating container networking interface.

NAT (Network address translator)>> Public to private communication CIDR (Classless inter domain routing) range > Sizing your network Kubernetes network example>> calico, fannel

Chapter3: Pod to pod communication

ARP(Address resolution protocal, and how it works) The Address Resolution Protocol is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address

Ethernet(Ethernet is the traditional technology for connecting devices in a wired local area network (LAN) (root ethernet and virtual ethernet)

Localhost for communication between two container withing same namespace. Container bridge for communication between pod with host.

Chapter4: Pod to pod communication between different host

Container overlay network and ARP protocol

Chapter5: How communication happen between service and pods

USERSPACE

IVPS: it's Kubernetes native load balancing with round robin algorithm (you can change algorithm)

IPTABLES: it's Linux feature for networking communication.

Note: IVPS is the best, and we follow it.

IPTABLES: Based on user condition accept/reject traffic.

Chapter6: Foundational Pattern – Predictable Demands

Compressible resource – Which can be added dynamically.>> CPU Incompressible – Which can't be added dynamically.>> Memory

QOS: 1) Best Effort, 2) Burstable and 3) Guaranteed. (Param: resource and limit)

Kill Criteria: Best then Burstable and then Guaranteed (Very less chances to kill)

Chapter7: Docker image revision

* Creating docker image.
* running created docker image and validate its utilization.

Chapter8: Health check probe

* Liveness probe => ps -ef | grep java
* readiness probe => HTTP 200 code

Chapter9: Managed Life Cycle

* SIGTERM signal:
* SIGKILL signal:

Chapter10: Setting up cluster

* Node selector
* Node affinity<Node selector with operator>

Chapter11: Container placement

Pod affinity: container in same pod.

anti-affinity: opposite to pod affinity.

taints: not allowed pod to schedule if present in node. (K8 master)

tolerance: 1) effect=NoSchedule

2) effect=PreferNoSchedule)

Bare pods that are not managed by a replication controller will be not rescheduled upon node disruption. (Like if I create a pod manually)

Chapter12: Kubernetes volume

Data, Database, CURD operation, DBMS, Need of volume, PV, PVC, storage class, configmap, secrets.

Kubernetes volume are resource like RAM and CPU. There is a 5-step process-

1. Application team demand for storage
2. Admin team create volume, and ask storage team to provide space.
3. Application team can use that volume by PVC.
4. Storage class provision volume dynamically. (- When persistent volume claim it)

Chapter13: Application and it’s type

* What is Stateful application
* Example of stateful application? >> Splunk, ELK
* Combination of stateful and stateless services

Chapter14: Comparison between deployment set and replica set.

A ReplicaSet ensures that a specified number of pod replicas are running at any given time. However, **a Deployment is a higher-level concept that manages ReplicaSets and provides declarative updates to Pods along with a lot of other useful features**.

Chapter15: Singleton services: Updating database. observing message from Q in sequential order.

Developer concept for providing high availability with 1 pod.

Chapter16: Services: Cluster-ip, Nodeport, headless and loadbalancer.

Chapter17: Init, sidecar and Adaptor(Multiple microservices generating multipe logs> making them in one formate)

Chapter18: Ambasdor pattern > it's kind of proxy services.

Chapter19: Config map and secrets

Chapter20: Concept of developing microservices application for invetory for e-commerce application.

Chapter21:

Understanding components of microservices (Database and front-end)

Chapter22:

Docker compose (introduction and depends on instruction)

Chapter23: RBAC- Authentication, autherization,

Namespace role and binding,

Cluster role and binding.

Chapter24: Aggregating ClusterRoles(combination of 2 role)

Deployer, Admin, monitor, ISA security admin

Chapter25: Helm, installation manager

Installing helm, installing my-sql by helm.

Chapter26: Helm2 vs Helm3 comparision