KUBERNETES NOTES FOR SELF-HELP

1. Control Plane
   1. API Server
   2. ETCD
   3. Kube Scheduler
   4. Kube Controller Manager
2. Worker Node
   1. Kube-proxy
   2. Kubelet
   3. Container Runtime
3. Deployments -> Manages Replicaset
4. Replicaset
5. Daemon sets -> Monitoring Pods typically running 1 per Worker Node
6. Stateful sets
   1. Creation of pods in an order (Important in Master Slave Architecture)
   2. Specific / Unique name based on Index
   3. DB’s, Cache’s
7. Nodes
   1. Taints -> Helps unwanted pods to not get scheduled
   2. Affinity v/s Node Selector
      1. Based on Node Selectors
      2. Required (HARD) v/s Preferred (SOFT – Will still schedule if Node isn’t available)
8. Pods
   1. Affinity -> Running in Same Node
   2. Anti-Affinity -> Running in Separate Nodes (High Availability - NGINX)
   3. Tolerations -> Helps it to get accepted into a Node which has taints on them. This way only nodes with certain Taints accept Pods with certain Tolerations.
   4. Ephemeral Containers -> Useful for interactive troubleshooting
   5. Static Pods -> Mostly used for Cluster Component Bootstrapping. E.g.: Kube scheduler, controller manager are all static pods due to the fact that API Server itself is not available to monitor their lifecycle. It’s tied to a Kubelet & not controlled by the API Server.
9. Services
   1. Nodeport -> Can be accessed from outside of Kubernetes Cluster
   2. ClusterIP -> Internal to Kubernetes Cluster. Needs Ingress for it to be exposed.
   3. LoadBalancer -> Less efficient way to exposing multiple services on a Kubernetes Cluster.
   4. Headless Service
      1. Inter-pod communication without going through a service.
      2. Use cases: RabbitMQ, Kafka, Relational DBS (useful with Stateful Sets)
      3. Useful when the requirement is to get all Pods supporting a certain functionality bypassing the proxy (Service Layer)
      4. Excellent examples - <https://kodekloud.com/blog/kubernetes-headless-service/#>
10. Ingress
    1. Ingress Controllers
       1. Supports TLS, HTTPS
       2. Path based routing.
    2. Examples
       1. NGINX
       2. Istio
          1. Supports mTLS
       3. Kubernetes-ingress
11. Autoscaling:
    1. Horizontal Pod Autoscaler ->
    2. Cluster Autoscaler
    3. Vertical Pod Autoscaler
12. ConfigMap
    1. Stores typical non sensitive items like -> connection strings etc.
13. Secrets
    1. Stores encrypted items such as passwords & other sensitive values.
14. Custom Kubernetes Resources
    1. CR
    2. CRD’s
    3. Custom Controllers – Extending Kubernetes Controller functionality
15. Kubernetes Operators
    1. Replicaset Controller – In built/native controller
    2. Ngnix/Ingress Ingress Controller – custom
    3. Bundle, Package & Manage Kubernetes Custom Controllers -> **Is mostly compared with Helm Charts**
       1. Can perform reconciliation / auto-healing
       2. Version automatic upgrade
       3. Telemetric System for informational purposes
16. Pod Disruption Budget
    1. defines how much disruption is acceptable.
    2. minAvailable – For quorum based applications requiring minimum no. of pods for successful execution.
    3. maxUnavailable -Ensures pods are drained 1 at a time to ensure.
17. RBAC
    1. Service Account
    2. Role
    3. RoleBinding
    4. ClusterRole
    5. ClusterRoleBinding
18. Pod Security Admission
19. Network Policies
    1. Calico
    2. Eks-vpc-cni
20. K8s Monitoring – Metric Server, Prometheus, Grafana
21. Helm Charts
22. Service Mesh
    1. Mostly used for pod-to-pod communication using a sidecar/proxy. Proxies take responsibility of communication.
    2. Useful for experimental canary releases
    3. Service Authorization
    4. Sharing metrics
    5. Tools – AWS App Mesh, Istio
    6. Created in k8s using Virtual Service