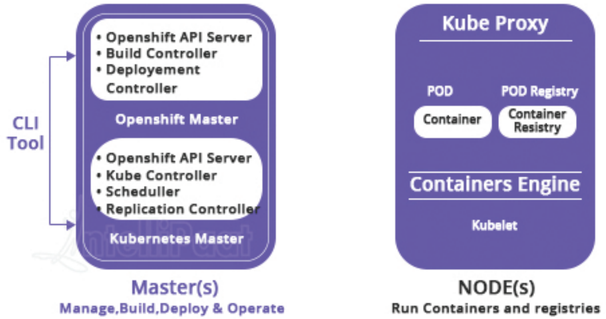
KUBERNETES CHEAT SHEET

KUBERNETES

- It is an open source platform for automating deployment and scaling of containers across clusters of hosts providing container centric infrastructure.
- It is a container orchestrator and can run Linux containers:
- Launch container.
- Maintain and monitor container site.
- Performs container-oriented networking



Master(s)
Manage,Build,Deploy & Operate

NODE(s)
Run Containers and registries

Key Concepts

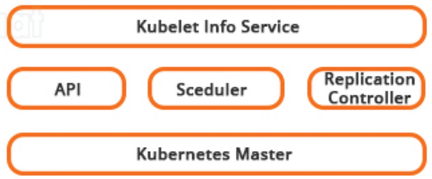
Now let's discuss the key points of this architecture.

- Pod: These are the group of containers.
- Labels: These are used to identify the pods.
- Kubelet: They are container agents, responsible for maintaining the set of pods.
- Proxy: They are the Load balancer for pods, helping in distributing tasks across the pods.
- ETCD: A Metadata service.
- Cadvisor: For resource usage and performance stats.
- Replication controller: It manages pod replication.
- Scheduler: Used for pod scheduling in worker nodes.
- API server: Kubernetes API server.

Now let's understand the role Master and Node play in the Kubernetes Architecture.

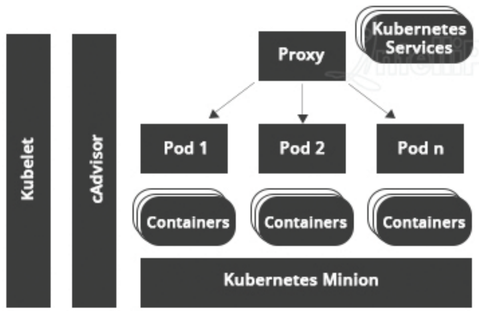
Master

- It is responsible for maintaining the desired state for the cluster you are working on.
- "Master" indicates a set of processes that are used to manage the cluster.
- Contains info, API, scheduler, replication controllers, and master.



Worker Nodes / Minions

- Also called as a minion. It contains the services necessary to run the pods that are managed by the master.
- Some services include: container runtime, Kubelet, kube-proxy.
- Contains: Kubelet, cAdvisor, services, pods and containers.



Features

- Automated scheduling- provides an advanced scheduler that helps launch container on cluster nodes
- Self healing- reschedule, replace and restart dead containers.
- Automated rollouts and rollbacks- supports rollback for systems incase of a failure. Enables rollout and rollback for the desired state.
- Horizontal scaling- can scale up and down the app as per required. Can also be automated wrt CPU usage.
- Service discovery and load balancing- uses unique ip and dns name to containers. This helps identify them across different containers.

• Pods and Container Introspection

COMMANDS	FUNCTION
Kubectl get pods	Lists all current pods
Kubectl describe pod<name>	Describes the pod names
Kubectl get rc	List all replication controllers
Kubectl get rc --namespace="namespace"	Lists all current pods
Kubectl describe rc <name>	Shows the replication controller name
Kubectl get cvc	Lists the services
Kubectl describe svc <name>	Shows the service name
Kubectl delete pod <name>	Deletes the pod
Kubectl get nodes -w	Watch nodes continuously

• Debugging

COMMANDS	FUNCTION
Execute command on service by selecting container.	Kubectl exec<service><commands>[-c< $container>]
Get logs from service for a container	Kubectl logs -f<name>>[-c< $container>]
Watch the kubelet logs	Watch -n 2 cat/var/log/kublet.log
Show metrics for node	Kubectl top node
Show metrics for pods	Kubectl top pod

• Objects

All	Clusterroleblindings	FUNCTION
cm= conf gmaps c	controllerrevisions	crd=custom resource definition
Cronjobs	cs=component status	csr= certificate signing requests
Deploy=deployments	ds= daemon sets	ep=end points
ev= events	hpa= autoscaling	ing= ingress
jobs	limits=limit ranges	Netpol- network policies
No = nodes	ns= namespaces	pdb= pod
po= pods	Pod preset	Pod templates
Psp= pod security policies	Pv= persistent volumes	pvc= persistent volume claims
quota= resource quotas	rc= replication controllers	Role bindings
roles	rs= replica sets	sa=service account
sc= storage classes	secrets	sts= stateful sets

• Cluster Introspection

COMMANDS	FUNCTION
Get version information	Kubectl version
Get cluster information	Kubectl cluster-info
Get the configuration	Kubectl config g view
Output info about a node	Kubectl describe node<node>

Other Quick Commands

Launch a pod with a name an image : Kubectl run<name> --image=<image-name>

Create a service in <manifest.yaml> : Kubectl create -f <manifest.yaml>

Scale replication counter to count the number of instances : Kubectl scale --replicas=<count>

Map external port to internal replication port : Expose rc<name> --port=<external>--target-port=<internal>

To stop all pod in <n> : Kubectl drain<n>-- delete-local-data--force--ignore-daemonset

Allow master nodes to run pods : Kubectltaintnodes --all-noderole.kuernetes.io/master-

IntelliPaat