KUBERNETES CHEAT SHEET

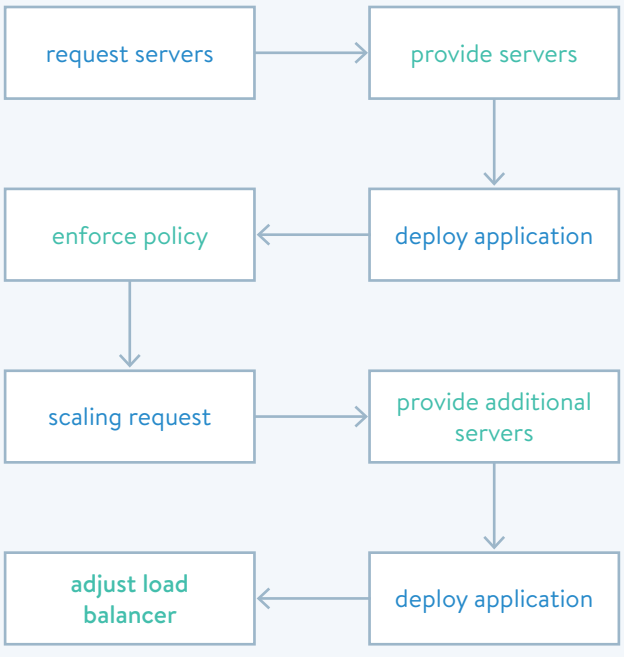
Kubernetes is the de-facto open source standard for managing application containers from local to global scale. Originally developed by Google employing 15 years of R&D, Kubernetes now flourishes in the open source community and has become the foundation for producing cloud native solutions.

Kubernetes is nicknamed "K8s" - K[ubernete]s –> K[8]s –> K8s.

WHO USES KUBERNETES AND WHY

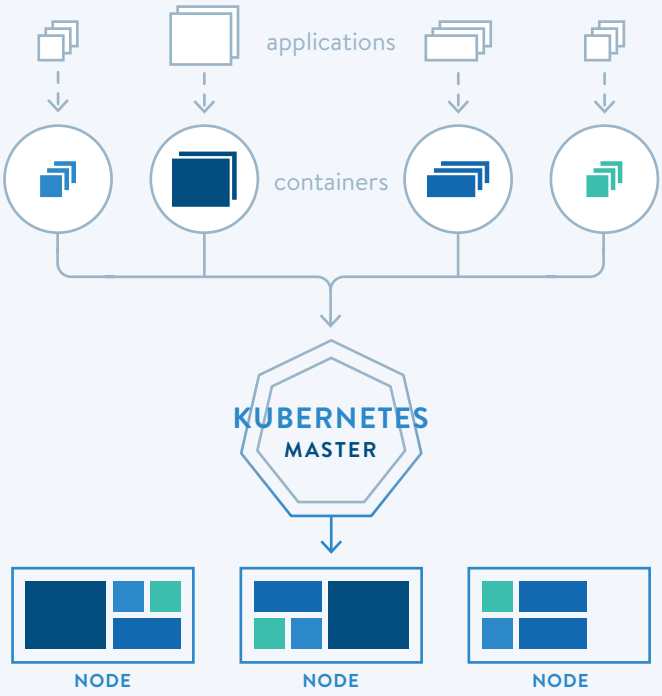
- **Anyone** managing containers - e.g. Docker
- **Developers** building microservices or cloud native apps at scale
- **Operations** hosting one or more microservice or cloud native application
- **Operations** building or hosting data analytics or distributed database systems - e.g. Cassandra
- **Operations** building or offering a centralized, shared, self service platform to developers (PaaS)

LEGACY DEVELOPMENT & HOSTING



Repeat process for patching and upgrading

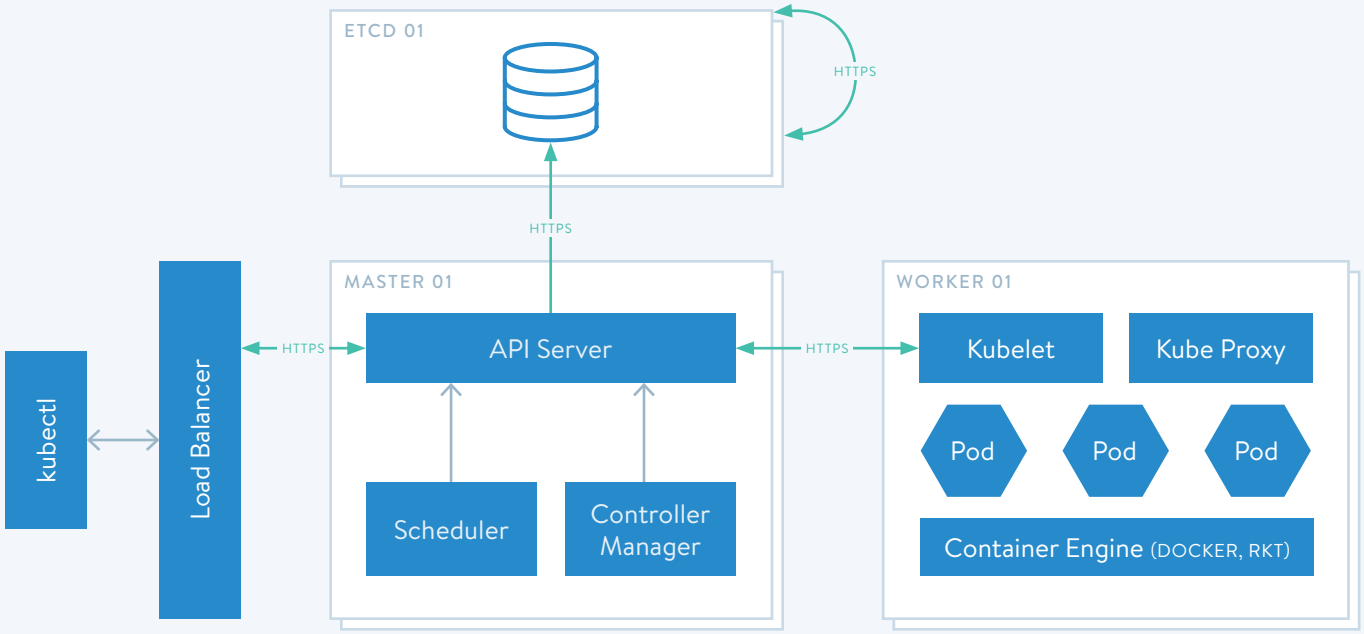
KUBERNETES DEVELOPMENT & HOSTING



Scheduled and packed dynamically onto nodes

INSTALLING OR RUNNING KUBERNETES

OPTION	FOCUS	WHERE TO FIND	STRENGTH	WEAKNESS
PUBLIC CLOUD	Managed cluster	Google GKE Microsoft Azure ACS IBM Bluemix BCS	• Easy to use • Recommended for developers getting started • Includes all ancillary features needed to run K8s - e.g. monitoring	Doesn't work in data center
KISMATIC ENTERPRISE TOOLKIT	Kubernetes inside corporate firewall or public cloud	Kismatic Github	• Easy to use • Community OSS • Recommended for operations or developers	Requires cluster planning
KOPS	AWS deployments	KOPS Github	• Easy to use • Community OSS	Takes decent amount of work for production
MINIKUBE	For laptops	Minikube Github	• Easy to use • Community OSS	It is not a real cluster and used only for basic dev work
VENDOR DISTROS	Deployments managed ops	Apprenda Cloud Platform CoreOS Tectonic Red Hat OpenShift (includes more than just Kubernetes)	Robust installation option with vendor support	Single vendor "OSS" (propensity for lock-in)



COMPONENTS

Master(s)

Authority or brain of cluster (often managed by provider when using public cloud - e.g. GKE)

API SERVER (CRUD SERVER)

- Entry point for cluster - i.e. what you talk to when writing commands
- Processes requests and updates etcd
- Does authentication/authorization
- More

CONTROLLER MANAGER

- Daemon process that implements the control loops built into Kubernetes - e.g. rolling deployments
- More

SCHEDULER

- Decides where pods (i.e. one or more containers) should run based on multiple factors - affinity, available resources, labels, QoS, etc.
- Expert mode

Worker(s)

1-1,000s of servers or virtual machines (nodes) that host distributed workloads in cluster

KUBELET - AGENT ON EVERY WORKER

- Instantiate pods (group of one or more containers) using PodSpec and insures all pods are running and healthy
- Interacts with containers - e.g. Docker
- More

KUBE PROXY - AGENT ON EVERY WORKER

- Network proxy and load balancer for Kubernetes Services
- More

kubectl

Kubernetes command line

Other Components

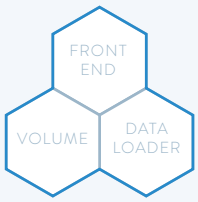
- Persistent storage
- Network overlay
- Cluster monitoring & app monitoring
- Log aggregation & analysis
- Load balancers (e.g. F5)
- Ingress nodes (HTTP traffic)
- Helm for installing and managing applications
- ...and more

etcd(s)

Single source of truth of cluster (often managed by provider when using public cloud - e.g. GKE)

- Distributed, consistent, key-value store
- Stores all K8s API objects
- Supports revisions & event streams
- Requires more than one for production (typically 1, 3 or 5 - needs odd number)

CONSTRUCTS



Pods

Group of one or more related containers that are treated as single unit for Kubernetes to orchestrate

Labels / Selectors

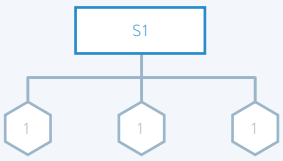
LABELS: Description of Kubernetes resource - e.g. front end

SELECTORS: Chooses resources with labels defined by K8s users

Controllers

Continuous loop where the user's desired state of the application is reconciled with current state

There are many types of controllers possible in K8s



Services

Microservices or cloud native application

(Persistent) Volumes

Persistent data storage for ephemeral containers

Namespaces

Virtual segmentation of single cluster

KUBECTL (KUBERNETES COMMAND LINE) GETTING STARTED

Source and more information (Kubernetes v 1.7) and --help command line output.

GENERALIZED COMMAND FORMAT:

kubectl [command] [TYPE] [NAME] [flags]

IN ENGLISH:

"Operation performed" **[command]** *"on this resource type - e.g. Pods, nodes"* **[TYPE]** *"with this name we gave it"* **[NAME]** *"with these considerations when running the operation"* **[flags]**

Just Getting Started

OPERATION	FORMAT	EXAMPLE
Display about master and cluster services	kubectl cluster-info	kubectl cluster-info
Display EVERYTHING about the cluster state into file	kubectl cluster-info dump --output-directory=/path/to-directory	kubectl cluster-info dump --output-directory=/chrisgaun/cluster-state
Check version of kubectl	kubectl version	kubectl version
Display documentation on resources	kubectl explain [RESOURCE]	kubectl explain nodes
Display current context	kubectl config current-context	kubectl config current-context
Display the Kubernetes contexts	kubectl config get-context	kubectl config get-context
Start a local proxy to access Kubernetes Dashboard (UI) - paste printed URL and port on local host to browser and add "/ui" to end e.g. "127.0.0.1:8001/ui" "Ctrl + c" to exit in command line	kubectl proxy	kubectl proxy

Just Getting Started (continued)

OPERATION	FORMAT	EXAMPLE
Start a pod from image	kubectl run [NAME] --image=[SOURCE] --port=[PORT]	kubectl run mynginx --image=nginx --port=8080
Create a resource (e.g. service, namespace, secret, etc) from a URL (busybox as example)	kubectl create -f [URL]	kubectl create -f https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.4/hack/testdata/recursive/pod/busybox.yaml
Start a replicated instance of Nginx	kubectl run [NAME] --image=[SOURCE] --replicas=[N]	kubectl run nginxserver --image=nginx --replicas=5
Add a label to an existing pod (use “kubectl get pods” to find name)	kubectl label pods [NAME] [KEY]=[VALUE]	kubectl label pods mynginx unhealthy=true
Delete pods with specific name (use “ kubectl get pods” to find name)	kubectl delete pod mynginx	kubectl delete pod mynginx
Delete all pods	kubectl delete pods --all	kubectl delete pods --all

Information on Resources

Types of Kubernetes resources

OPERATION	FORMAT	EXAMPLE
List all pods	kubectl get pods	kubectl get pods
Display logs of named pod (use “kubectl get pods” to find name of pod)	kubectl logs [NAME]	kubectl logs mynginx

Information on Resources (continued)

Types of Kubernetes resources

OPERATION	FORMAT	EXAMPLE
Display CPU and memory usage of pods	kubectl top pods	kubectl top pods
List all pods in json format (this may be a long output)	kubectl get pods -o json	kubectl get pods -o json
Display detailed state of all pods (this may be a long output)	kubectl describe pods	kubectl describe pods
List all services	kubectl get services	kubectl get services
List all namespaces	kubectl get namespaces	kubectl get namespaces
Display detailed state of namespace	kubectl describe namespaces	kubectl describe namespaces
List all nodes	kubectl get nodes	kubectl get nodes
Display CPU, memory and storage of nodes	kubectl top pods	kubectl top pods
List all pods with more information (such as node name)	kubectl get pods -o wide	kubectl get pods -o wide
List all resources	kubectl get all	kubectl get all