

D2
IQ

KUBERNETES CHEAT SHEET



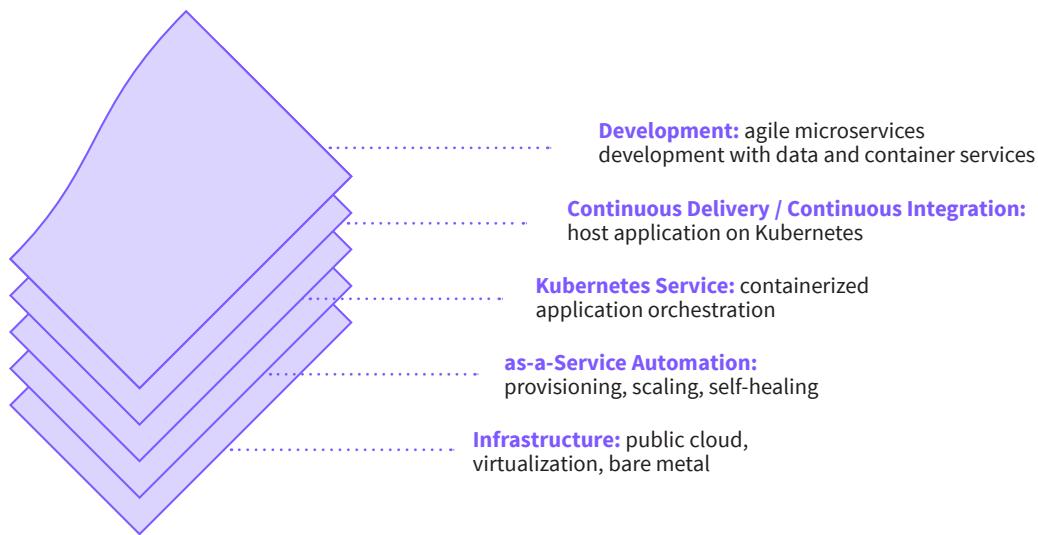
D2
IQ

Executive Summary

Kubernetes is a leading container management solution. For an organization to deliver Kubernetes-as-a-Service to every line of business and developer group, operations needs to architect and manage both the core Kubernetes container orchestration and the necessary auxiliary solutions — e.g. monitoring, logging, and CI/CD pipeline. This cheat sheet offers guidance on end-to-end architecture and ongoing management.

What is Kubernetes?

Kubernetes is a container management solution with several logical layers:



Kubernetes differs from the orchestration offered by configuration management solutions in several ways:

Abstraction	Declarative	Immutable
Kubernetes abstracts the application orchestration from the infrastructure resource and as-a-Service automation.	Kubernetes master decides how the hosted application is deployed and scaled on the underlying fabric.	Different versions of services running on Kubernetes are completely new and not swapped out.

Kubernetes Solution Design Considerations



Automated Management	True Interoperability	Evergreen Cluster
Plan to automate ongoing management of an end-to-end solution — Kubernetes, CI/CD, etc.	Pure Kubernetes with stock user interface and command line is the current industry standard.	Kubernetes is relatively new and versions with critical patches and desired features are released frequently.

Kubernetes success relies on conformance and alleviates the burden created by other solutions' open-endedness and lack of interoperability from ancillary projects.

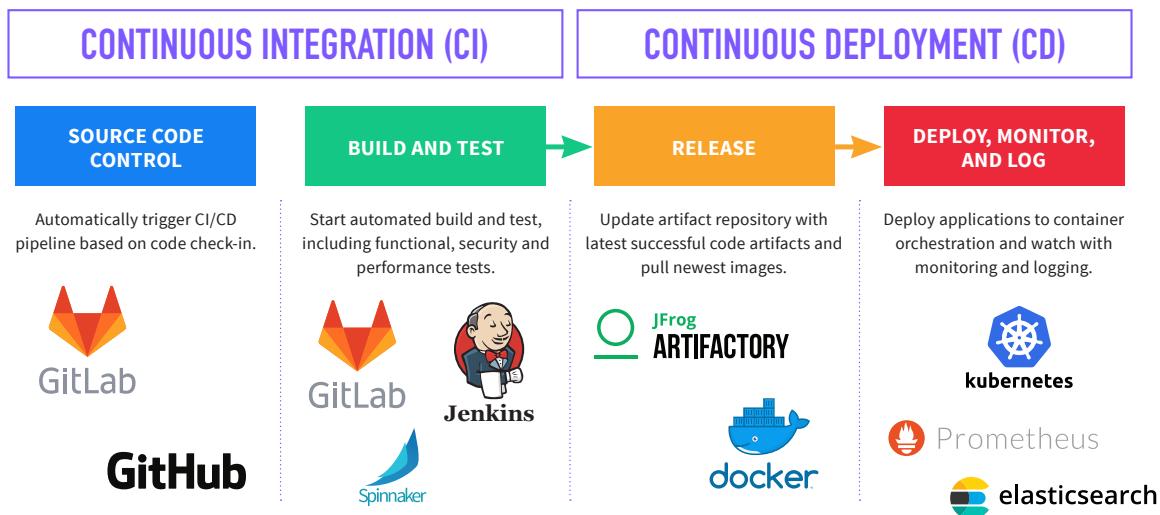
Kubernetes Features vs. Community Projects

Kubernetes Features

- Rigorous Testing & Integration
 - Stable
 - Versioned
 - Discoverable
 - Included in apiserver
 - Include client support
- Included in Kubernetes API & Documentation
- Avoids OpenStack's open-endedness & prevents snowflakes

	FEATURES	PROJECTS
EXAMPLES	Pod Horizontal Autoscaling, ReplicaSet	IaaS autoscaling, VM orchestration
PART OF KUBERNETES	Yes	No
VETTED BY KUBERNETES STAKEHOLDERS	Yes	No
TESTED AS PART OF KUBERNETES	Yes	No
STANDARD COMMERCIAL SUPPORT	Yes	No
VERSION RISK	Low	High
API CHANGES OR DEPRECIATION RISK	Low	High

From Developer to Platform: Hosting Applications on Kubernetes



Standard Components of Kubernetes

These are the minimum components required for a Kubernetes cluster:

Master Nodes	Worker Nodes
API SERVER <ul style="list-style-type: none">Entry point for clusterProcesses requests and updates etcdPerforms authentication/authorizationMore: https://goo.gl/KL8WfQ	KUBELET — AGENT ON EVERY WORKER <ul style="list-style-type: none">Instantiate pods (group of one or more containers) using PodSpec and insures all pods are running and healthyInteracts with containers - e.g. DockerMore: https://goo.gl/FEKN43
CONTROLLER MANAGER <ul style="list-style-type: none">Daemon process that implements the control loops built into Kubernetes — e.g. rolling deploymentsMore: https://goo.gl/NJyRP3	KUBE PROXY — AGENT ON EVERY WORKER <ul style="list-style-type: none">Network proxy and load balancer for Kubernetes ServicesMore: https://goo.gl/ph4sAs
SCHEDULER <ul style="list-style-type: none">Decides where pods should run based on multiple factors - affinity, available resources, labels, QoS, etc.More: https://goo.gl/nvLDE9	

Standard Add-ons for Kubernetes

These are the Kubernetes add-ons that are required for all but Hello World solutions.

Kube-DNS	Kubectl
<ul style="list-style-type: none">Provisioned as a pod and a service on KubernetesEvery service gets a DNS entry in KubernetesKube-DNS resolves DNS of all services in the clusters	<ul style="list-style-type: none">Official command line for KubernetesIndustry standard Kubernetes commands start with “Kubectl”
Metrics Server	Web UI (Dashboard)
<ul style="list-style-type: none">Provides API for cluster wide usage metrics like CPU and memory utilizationFeeds the usage graphs in the Kubernetes Dashboard (GUI) — see Dashboard image under “Kubernetes Constructs” section.	<ul style="list-style-type: none">Official GUI of KubernetesIndustry standard GUI for a Kubernetes clusters

Required for Container Solution

These are the ecosystem components required for any production Kubernetes solution but not included with Kubernetes.

Infrastructure	as-a-Service Automation (DC/OS)
<ul style="list-style-type: none">Kubernetes can be installed on bare metal, public cloud instances or virtual machines	<ul style="list-style-type: none">Required management layer for Kubernetes CI/CD, and data servicesDC/OS provides intelligent as-a-Service automation on any infrastructureDC/OS features abstraction, declarative, and immutable management
Ingress Controller	Private Container Registry
<ul style="list-style-type: none">HTTP traffic access control for Kubernetes servicesInteracts with Kubernetes API for state changesApplies ingress rules to service load balancer	<ul style="list-style-type: none">Registry for an organization's standard container imagesRequire access credentials (from IDM or secrets located in Kubernetes pod)
Monitoring	Logging & Auditing
<ul style="list-style-type: none">Metrics collected on Kubernetes infrastructure and hosted objectsTypical options: Prometheus, Sysdig, Datadog	<ul style="list-style-type: none">Centralized logging for KubernetesTypical options: FluentD, Logstash
Network Plugin	Secrets Management
<ul style="list-style-type: none">Network overlay for policy and software defined networkingNetwork overlays use the Container Network Interface (CNI) standard that works with all Kubernetes clusters	<ul style="list-style-type: none">Holds sensitive information such as passwords, OAuth tokens, and ssh keys required for services, developers and operations
Load Balancing	Container Runtime
<ul style="list-style-type: none">Software load balancing to each Kubernetes services	<ul style="list-style-type: none">Specific containers used in KubernetesCurrently Kubernetes supports Docker

Kubernetes Constructs:

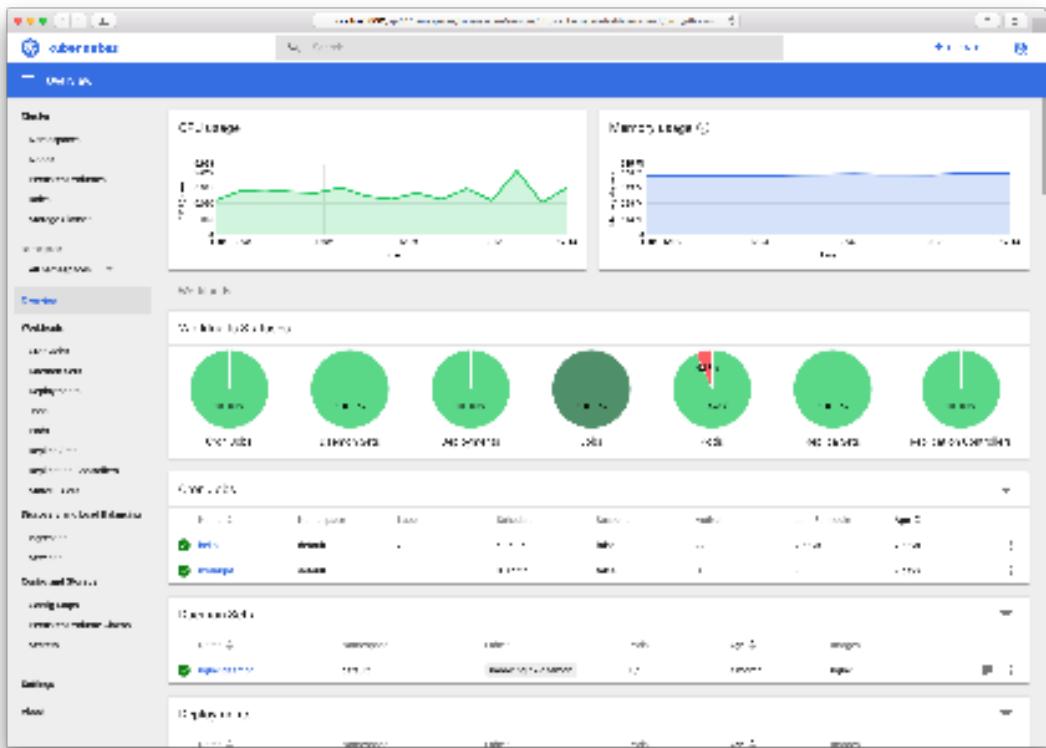


Image via the Kubernetes Dashboard Github: <https://github.com/kubernetes/dashboard>

Namespaces — Virtual segmentation of single clusters.	Pods — A logical grouping of one or more containers that is managed by Kubernetes
Nodes — Infrastructure fabric of Kubernetes (host of worker and master components)	ReplicaSet — continuous loop that ensures given number of pods are running
Roles — role based access controls for Kubernetes cluster	Ingresses — manages external HTTP traffic to hosted service
Deployments — manages a ReplicaSet, pod definitions/updates and other concepts	Services — a logical layer that provides IP/DNS/etc. persistence to dynamic pods

Commands

Below is some commands useful for IT professionals getting started with Kubernetes. A full list of Kubectl commands can be found at the reference documentation <https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands>

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

Kubectl Command	Format
Kubernetes abstracts the application orchestration from the infrastructure resource and as-a-Service automation.	Find the version of the Kubectl command line.
\$ kubectl version	Find the version of the Kubectl command line.
\$ kubectl API version	Print the version of the API Server.
\$ kubectl cluster-info	IP addresses of master and services.
\$ kubectl cluster-info dump --namespaces	List all the namespace used in Kubernetes.
\$ kubectl cordon NODE	Mark node as unschedulable. Used for maintenance of cluster.
\$ kubectl uncordon NODE	Mark node as scheduled. Used after maintenance.
\$ kubectl drain NODE	Removes pods from node via graceful termination for maintenance.
\$ kubectl drain NODE --dry-run=true	Find the names of the objects that will be removed
\$ kubectl drain NODE --force=true	Removes pods even if they are not managed by controller
\$ kubectl taint nodes node1 key=value:NoSchedule	Taint a node so they can only run dedicated workloads or certain pods that need specialized hardware.
\$ kubectl run nginx --image=nginx --port=8080	Start instance of nginx
\$ kubectl expose rc nginx --port=80 --target-port=8080	

Kubectl Command	Format
\$ kubectl get RESOURCE	<p>Print information on Kubernetes resources including:</p> <ul style="list-style-type: none"> • all • certificatesigningrequests (aka ‘csr’) • clusterrolebindings • clusterroles • componentstatuses (aka ‘cs’) • configmaps (aka ‘cm’) • controllerrevisions • cronjobs • customresourcedefinition (aka ‘crd’) • daemonsets (aka ‘ds’) • deployments (aka ‘deploy’) • endpoints (aka ‘ep’) • events (aka ‘ev’) • horizontalpodautoscalers (aka ‘hpa’) • ingresses (aka ‘ing’) • jobs • limitranges (aka ‘limits’) • namespaces (aka ‘ns’) • networkpolicies (aka ‘netpol’) • nodes (aka ‘no’) • persistentvolumeclaims (aka ‘pvc’) • persistentvolumes (aka ‘pv’) • poddisruptionbudgets (aka ‘pdb’) • podpreset • pods (aka ‘po’) • podsecuritypolicies (aka ‘psp’) • podtemplates • replicaset (aka ‘rs’) • replicationcontrollers (aka ‘rc’) • resourcequotas (aka ‘quota’) • rolebindings • roles • secrets • serviceaccounts (aka ‘sa’) • services (aka ‘svc’) • statefulsets (aka ‘sts’) • storageclasses (aka ‘sc’)
\$ kubectl explain RESOURCE	Print documentation of resources
\$ kubectl scale --replicas=COUNT rs/foo	<p>Scale a ReplicaSet (rs) named foo</p> <p>Can also scale a Replication Controller, or StatefulSet</p>

Kubectl Command	Format
\$ kubectl rolling-update frontend-v1 -f frontend-v2.json	Perform rolling update
\$ kubectl label pods foo GPU=true	Update the labels of resources
\$ kubectl delete pod foo	Delete foo pods
\$ kubectl delete svc foo	Delete foo services
\$ kubectl create service clusterip foo --tcp=5678:8080	Create a clusterIP for a service named foo
\$ kubectl autoscale deployment foo --min=2 --max=10 --cpu-percent=70	Autoscale pod foo with a minimum of 2 and maximum of 10 replicas when CPU utilization is equal to or greater than 70%

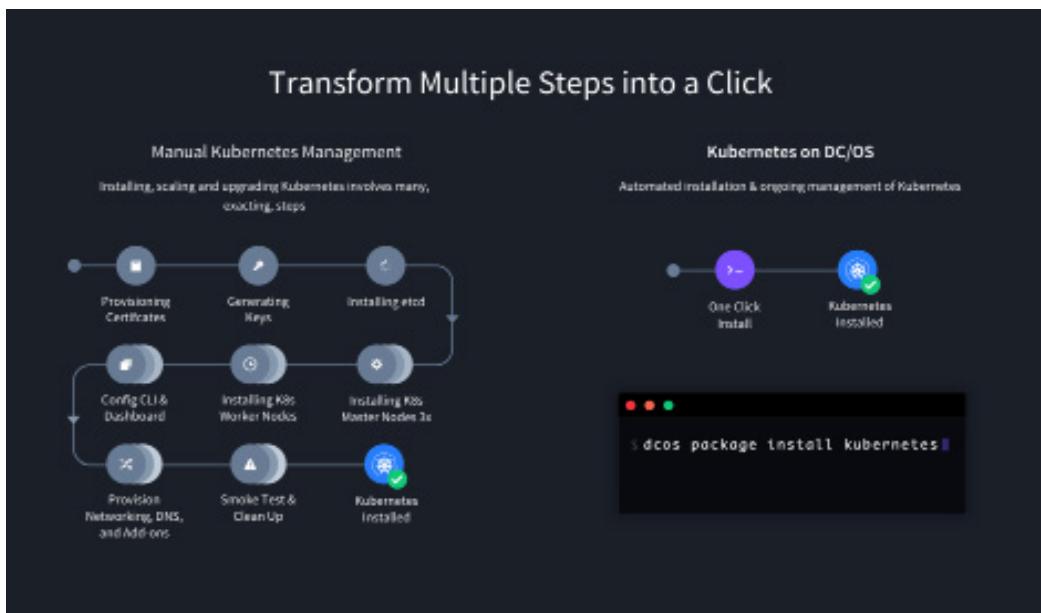
Kubernetes-as-a-Service Anywhere with DC/OS

Deliver Kubernetes on any infrastructure with push-button control and automated self-healing.

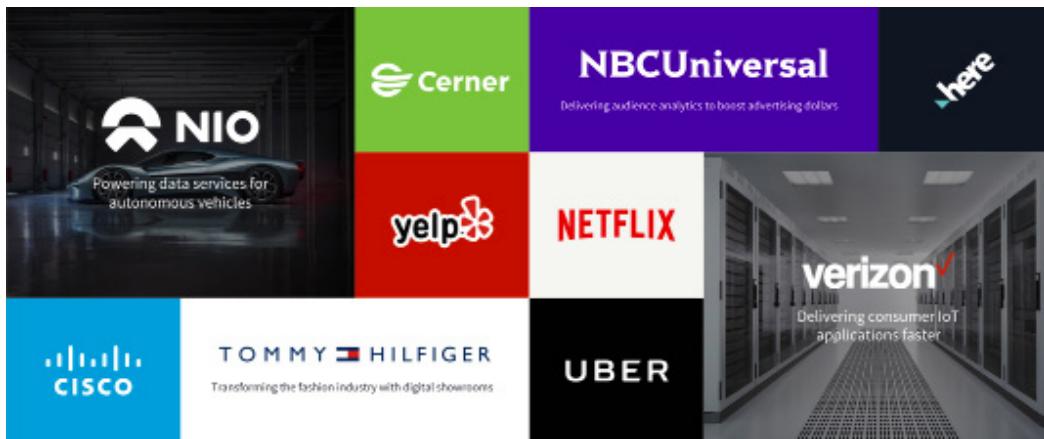
DC/OS automates the end-to-end management of Kubernetes, developer tools, and Big Data services so they can be delivered as-a-Service on any infrastructure. DC/OS provides the management layer organizations need to deliver Kubernetes to developer groups and lines of business:

Built-in Management for Kubernetes and All the Ecosystem Tools you Need

 <p>Zero Touch Self Healing Reapawn malfunctioning Kubernetes master, etcd, and worker nodes without a single command.</p>	 <p>Simple Provisioning Reduce the steps needed to deploy a highly available Kubernetes cluster to a single command line.</p>	 <p>Push-button Scaling Scale the cluster's nodes—not just the hosted applications—to add more capacity for spikes and usage growth.</p>
 <p>Enhanced Security Limit vulnerability and ease compliance with encryption between each solution component and more.</p>	 <p>Non-disruptive Upgrades Upgrade the cluster soon after the release of newer versions of Kubernetes with a push of a button.</p>	 <p>Choice of Ingress and CNI Out-of-the-box networking or set up your own from a choice of CNI and Ingress controllers in the DC/OS Service Catalog.</p>



D2iQ Proven Success



D2iQ is leading the enterprise transformation toward distributed computing and hybrid cloud portability. DC/OS is the premier platform for building, deploying, and elastically scaling modern, containerized applications and big data without compromise. DC/OS makes running containers, data services, and microservices easy, across any infrastructure — datacenter or cloud — without lock-in

Learn More

[Ready to see how D2iQ can power Kubernetes in your organization?](#)

Contact sales@d2iq.com today to get started. From weekly touch-base meetings to biweekly roadmap calls, customer success managers and solution architects work lockstep with your technology organization to eliminate the learning curve.