MESOSPHERE

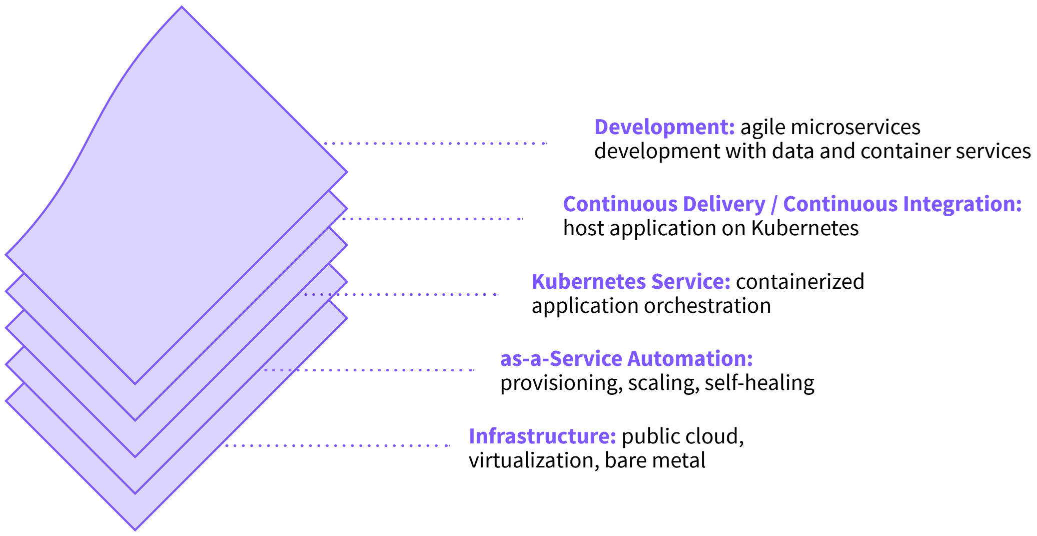
# KUBERNETES CHEATSHEET

## 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:



**Development:** agile microservices development with data and container services
**Continuous Delivery / Continuous Integration:** host application on Kubernetes
**Kubernetes Service:** containerized application orchestration
**as-a-Service Automation:** provisioning, scaling, self-healing
**Infrastructure:** public cloud, virtualization, bare metal

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.

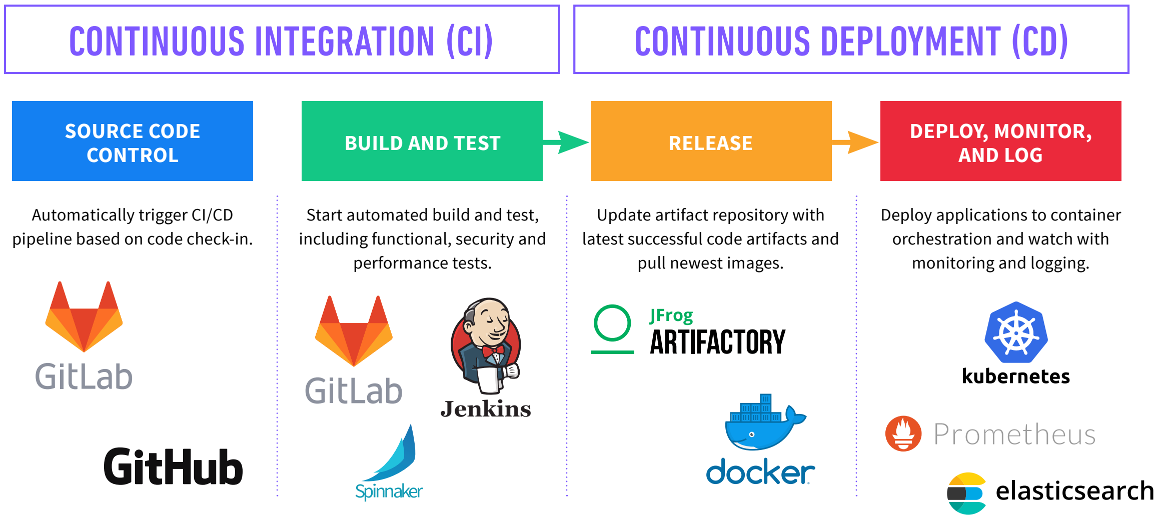
### Kubernetes Features vs. Community Projects

**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


CONTINUOUS INTEGRATION (CI)
CONTINUOUS DEPLOYMENT (CD)
SOURCE CODE CONTROL
BUILD AND TEST
RELEASE
DEPLOY, MONITOR, AND LOG
Automatically trigger CI/CD pipeline based on code check-in.
Start automated build and test, including functional, security and performance tests.
Update artifact repository with latest successful code artifacts and pull newest images.
Deploy applications to container orchestration and watch with monitoring and logging.
GitLab
GitHub
GitLab
Jenkins
Spinnaker
JFrog ARTIFACTORY
docker
kubernetes
Prometheus
elasticsearch

### Standard Components of Kubernetes

These are the minimum components required for a Kubernetes cluster:

| Master Nodes | Worker Nodes |
|---|---|
| **API SERVER**<br>• Entry point for cluster<br>• Processes requests and updates etcd<br>• Performs authentication/ authorization<br>• More: https://goo.gl/KL8WfQ | **KUBELET — AGENT ON EVERY WORKER**<br>• Instantiate pods (group of one or more containers) using PodSpec and insures all pods are running and healthy<br>• Interacts with containers - e.g. Docker<br>• More: https://goo.gl/FEKN43 |
| **CONTROLLER MANAGER**<br>• Daemon process that implements the control loops built into Kubernetes — e.g. rolling deployments<br>• More: https://goo.gl/NJyRP3 | **KUBE PROXY — AGENT ON EVERY WORKER**<br>• Network proxy and load balancer for Kubernetes Services<br>• More: https://goo.gl/ph4sAs |
| **SCHEDULER**<br>• Decides where pods should run based on multiple factors - affinity, available resources, labels, QoS, etc.<br>• 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.

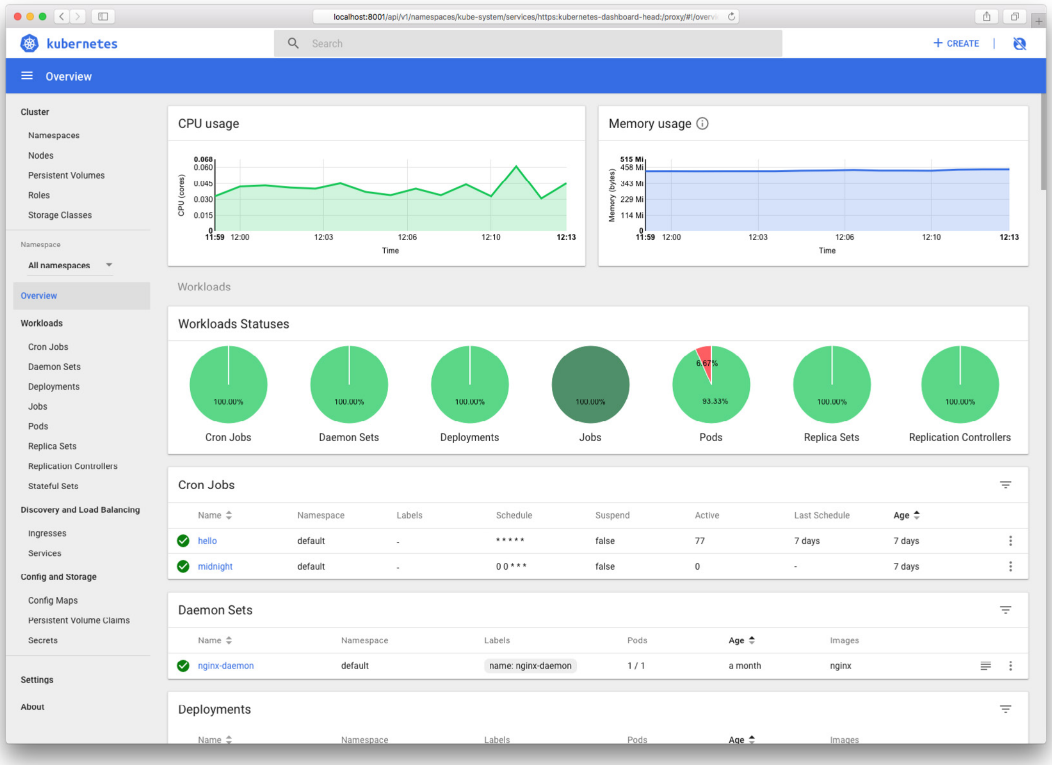
| Kube-DNS | Kubectl |
|---|---|
| • Provisioned as a pod and a service on Kubernetes<br>• Every service gets a DNS entry in Kubernetes<br>• Kube-DNS resolves DNS of all services in the clusters | • Official command line for Kubernetes<br>• Industry standard Kubernetes commands start with "Kubectl" |
| **Metrics Server** | **Web UI (Dashboard)** |
| • Provides API for cluster wide usage metrics like CPU and memory utilization<br>• Feeds the usage graphs in the Kubernetes Dashboard (GUI) — see Dashboard image under "Kubernetes Constructs" section. | • Official GUI of Kubernetes<br>• Industry 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 (Mesosphere DC/OS)** |
|---|---|
| • Kubernetes can be installed on bare metal, public cloud instances or virtual machines | • Required management layer for Kubernetes CI/CD and data services<br>• Mesosphere DC/OS provides intelligent as-a-Service automation on any infrastructure<br>• DC/OS features abstraction, declarative, and immutable management |
| **Ingress Controller** | **Private Container Registry** |
| • HTTP traffic access control for Kubernetes services<br>• Interacts with Kubernetes API for state changes<br>• Applies ingress rules to service load balancer | • Registry for an organization's standard container images<br>• Require access credentials (from IDM or secrets located in Kubernetes pod) |
| **Monitoring** | **Logging & Auditing** |
| • Metrics collected on Kubernetes infrastructure and hosted objects<br>• Typical options: Prometheus, Sysdig, Datadog | • Centralized logging for Kubernetes<br>• Typical options: FluentD, Logstash |
| **Network Plugin** | **Secrets Management** |
| • Network overlay for policy and software defined networking<br>• Network overlays use the Container Network Interface (CNI) standard that works with all Kubernetes clusters | • Holds sensitive information such as passwords, OAuth tokens, and ssh keys required for services, developers and operations |
| **Load Balancing** | **Container Runtime** |
| • Software load balancing to each Kubernetes services | • Specific containers used in Kubernetes<br>• Currently Kubernetes supports Docker |

## Kubernetes Constructs:



Cron Jobs

| Name | Namespace | Labels | Schedule | Suspend | Active | Last Schedule | Age |
|---|---|---|---|---|---|---|---|
| hello | default | - | * * * * * | false | 77 | 7 days | 7 days |
| midnight | default | - | 0 0 * * * | false | 0 | - | 7 days |

Daemon Sets

| Name | Namespace | Labels | Pods | Age | Images |
|---|---|---|---|---|---|
| nginx-daemon | default | name: nginx-daemon | 1 / 1 | a month | nginx |

Deployments

| Name | Namespace | Labels | Pods | Age | Images |
|---|---|---|---|---|---|

*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 are 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` | |

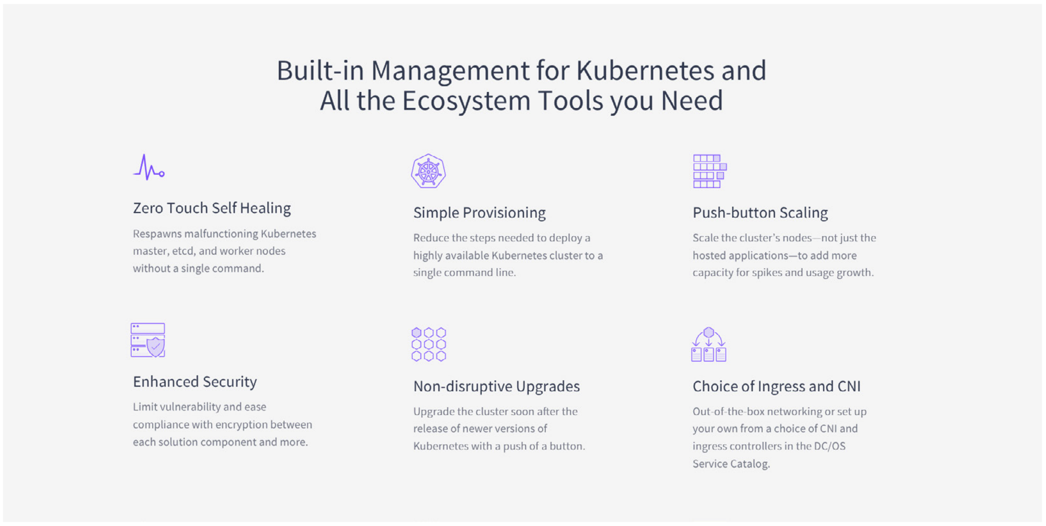
| Kubectl Command | Format |
|---|---|
| `$ kubectl get RESOURCE` | Print information on Kubernetes resources including:<br><br>• all<br>• certificatesigningrequests (aka 'csr')<br>• clusterrolebindings<br>• clusterroles<br>• componentstatuses (aka 'cs')<br>• configmaps (aka 'cm')<br>• controllerrevisions<br>• cronjobs<br>• customresourcedefinition (aka 'crd')<br>• daemonsets (aka 'ds')<br>• deployments (aka 'deploy')<br>• endpoints (aka 'ep')<br>• events (aka 'ev')<br>• horizontalpodautoscalers (aka 'hpa')<br>• ingresses (aka 'ing')<br>• jobs<br>• limitranges (aka 'limits')<br>• namespaces (aka 'ns')<br>• networkpolicies (aka 'netpol')<br>• nodes (aka 'no')<br>• persistentvolumeclaims (aka 'pvc')<br>• persistentvolumes (aka 'pv')<br>• poddisruptionbudgets (aka 'pdb')<br>• podpreset<br>• pods (aka 'po')<br>• podsecuritypolicies (aka 'psp')<br>• podtemplates<br>• replicasets (aka 'rs')<br>• replicationcontrollers (aka 'rc')<br>• resourcequotas (aka 'quota')<br>• rolebindings<br>• roles<br>• secrets<br>• serviceaccounts (aka 'sa')<br>• services (aka 'svc')<br>• statefulsets (aka 'sts')<br>• storageclasses (aka 'sc') |
| `$ kubectl explain RESOURCE` | Print documentation of resources |
| `$ kubectl scale --replicas=COUNT rs/foo` | Scale a ReplicaSet (rs) named foo<br><br>Can also scale a Replication Controller, or StatefulSet |

| 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 Mesosphere DC/OS

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

Mesosphere 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

Zero Touch Self Healing

Respawns malfunctioning Kubernetes master, etcd, and worker nodes without a single command.

Simple Provisioning

Reduce the steps needed to deploy a highly available Kubernetes cluster to a single command line.

Push-button Scaling

Scale the cluster's nodes—not just the hosted applications—to add more capacity for spikes and usage growth.

Enhanced Security

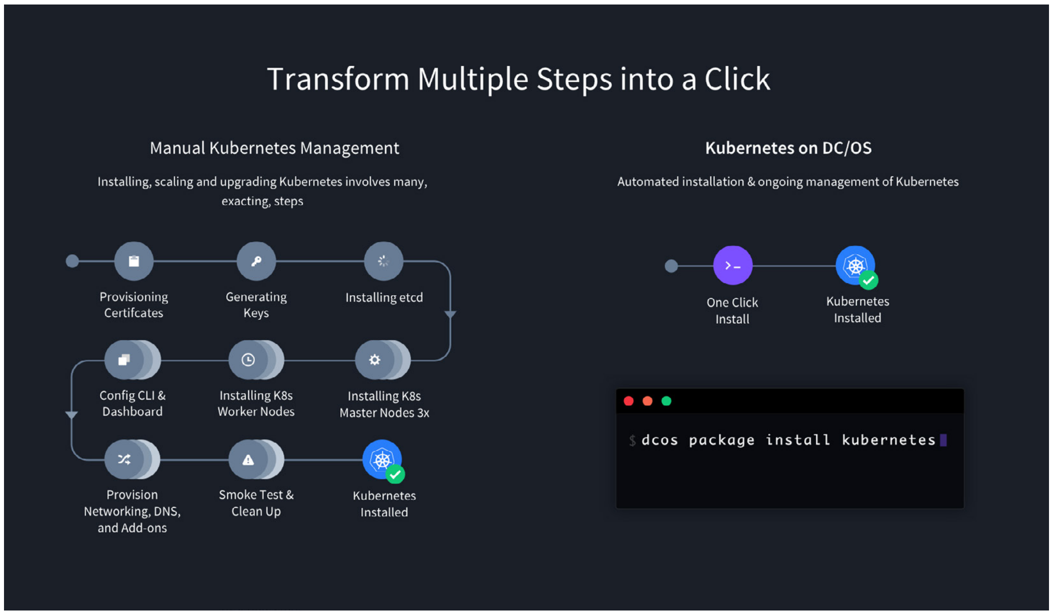
Limit vulnerability and ease compliance with encryption between each solution component and more.

Non-disruptive Upgrades

Upgrade the cluster soon after the release of newer versions of Kubernetes with a push of a button.

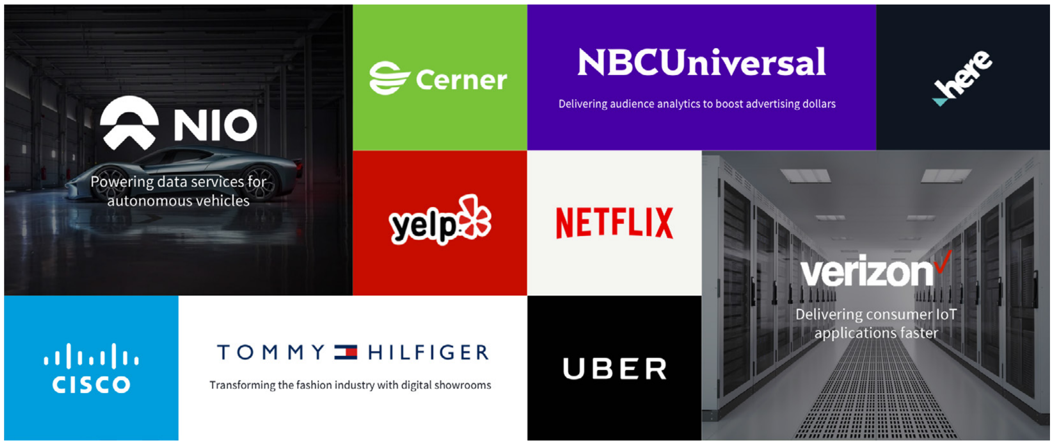
Choice of Ingress and CNI

Out-of-the-box networking or set up your own from a choice of CNI and ingress controllers in the DC/OS Service Catalog.


Transform Multiple Steps into a Click
Manual Kubernetes Management
Installing, scaling and upgrading Kubernetes involves many, exacting, steps
Provisioning Certifcates
Generating Keys
Installing etcd
Installing K8s Master Nodes 3x
Installing K8s Worker Nodes
Config CLI & Dashboard
Provision Networking, DNS, and Add-ons
Smoke Test & Clean Up
Kubernetes Installed
Kubernetes on DC/OS
Automated installation & ongoing management of Kubernetes
One Click Install
Kubernetes Installed

```
$ dcos package install kubernetes
```

## Mesosphere Proven Success


NIO
Powering data services for autonomous vehicles
Cerner
NBCUniversal
Delivering audience analytics to boost advertising dollars
here
yelp
NETFLIX
verizon
Delivering consumer IoT applications faster
cisco
TOMMY HILFIGER
Transforming the fashion industry with digital showrooms
UBER

Mesosphere is leading the enterprise transformation toward distributed computing and hybrid cloud portability. Mesosphere 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 Mesosphere can power Kubernetes in your organization?**
Contact sales@mesosphere.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.