Client Configuration

Setup autocomplete in bash; bash-completion package should be installed first

```
source <(kubectl completion bash)
```

View Kubernetes config

```
kubectl config view
```

View specific config items by json path

```
kubectl config view -o jsonpath='{.users[?(@.name == "k8s")].user.password}'
kubectl config set-credentials kubeuser/foo.kubernetes.com \
--username=kubeuser --password=kubepassword
```

Viewing, Finding Resources

List all services in the namespace

```
kubectl get services
```

List all pods in all namespaces in wide format

```
kubectl get pods -o wide --all-namespaces
```

List all pods in json (or yaml) format

```
kubectl get pods -o json
```

Describe resource details (node, pod, svc)

```
kubectl describe nodes my-node
```

List services sorted by name

```
kubectl get services --sort-by=.metadata.name
```

List pods sorted by restart count

```
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'
```

Rolling update pods for frontend-v1

```
kubectl rolling-update frontend-v1 -f frontend-v2.json
```

Scale a replicaset named 'foo' to 3

```
kubectl scale --replicas=3 rs/foo
```

Scale a resource specified in "foo.yaml" to 3

```
kubectl scale --replicas=3 -f foo.yaml
```

Execute a command in every pod / replica

```
for i in 0 1; do kubectl exec foo-$i -- sh -c 'echo $(hostname) > /usr/share/nginx/html/index.html'; done
```

Manage Resources

Get documentation for pod or service

```
kubectl explain pods,svc
```

Create resource(s) like pods, services or daemonsets

```
kubectl create -f ./my-manifest.yaml
```

Apply a configuration to a resource

```
kubectl apply -f ./my-manifest.yaml
```

Start a single instance of Nginx

```
kubectl run nginx --image=nginx
```

Create a secret with several keys

```
cat <<EOF | kubectl create -f -
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo "s33msi4" | base64)
  username: $(echo "jane"| base64)
EOF
```

Delete a resource

```
kubectl delete -f ./my-manifest.yaml
```

Monitoring & Logging

Deploy Heapster from Github repository https://github.com/kubernetes/heapster

```
kubectl create -f deploy/kube-config/standalone/
```

Show metrics for nodes

```
kubectl top node
```

Show metrics for pods

```
kubectl top pod
```

Show metrics for a given pod and its containers

```
kubectl top pod pod_name --containers
```

Dump pod logs (stdout)

```
kubectl logs pod_name
```

Stream pod container logs (stdout, multi-container case)

```
kubectl logs -f pod_name -c my-container
```

Kubernetes Cheatsheet

Create a daemonset from stdin. The example deploys Sematext Docker Agent to all nodes for the cluster-wide collection of metrics, logs and events. There is NO need to deploy cAdvisor, Heapster, Prometheus, Elasticsearch, Grafana, InfluxDb on your local nodes.

```
cat <<EOF | kubectl create -f -
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
 name: sematext-agent
spec:
 template:
  metadata:
   labels:
    app: sematext-agent
  spec:
   nodeSelector: {}
   hostNetwork: true
   dnsPolicy: "ClusterFirst"
   restartPolicy: "Always"
   containers:
   - name: sematext-agent
    image: sematext/sematext-agent-docker:latest
    imagePullPolicy: "Always"
    env:
    - name: SPM_TOKEN
     value: "YOUR_SPM_TOKEN"
    - name: LOGSENE_TOKEN
     value: "YOUR_LOGSENE_TOKEN"
    volumeMounts:
     - mountPath: /var/run/docker.sock
      name: docker-sock
     - mountPath: /etc/localtime
      name: localtime
    securityContext:
     privileged: true
   volumes:
    - name: docker-sock
     hostPath:
      path: /var/run/docker.sock
    - name: localtime
     hostPath:
      path: /etc/localtime
EOF
```

brought to you by

sematext