

Version - **2025.01**

Last update : 2025/01/17 16:38

DOE303 - The kubectl, krew and kustomize commands

Curriculum

- **DOE303 - The kubectl, krew and kustomize commands**
 - Curriculum
 - LAB #1 - Using the kubectl command
 - 1.1 - Getting help with kubectl commands
 - 1.2 - Obtaining information about the Cluster
 - The version command
 - The cluster-info command
 - The api-versions command
 - The api-resources command
 - 1.3 - Obtaining information about nodes
 - The describe node command
 - The top command
 - 1.4 - Obtaining information about Pods
 - The describe pod command
 - The top command
 - 1.5 - Working with the kubectl command
 - The apply command
 - The create command
 - The get command
 - Using Options
 - The exec command
 - Imperative commands
 - LAB #2 - Managing kubectl plugins with the krew command
-

- 2.1 - Installing krew
- 2.2 - Viewing the list of plugins
- 2.3 - Installing and using plugins
- 2.4 - Updating and deleting plugins
- LAB #3 - Managing patches with the kustomize command

LAB #1 - Using the kubectl command

1.1 - Getting help with kubectl commands

The **kubectl** commands are grouped by category:

```
root@kubemaster:~# kubectl --help
kubectl controls the Kubernetes cluster manager.
```

Find more information at: <https://kubernetes.io/docs/reference/kubectl/>

Basic Commands (Beginner):

```
create          Create a resource from a file or from stdin
expose          Take a replication controller, service, deployment or pod and expose it as a new Kubernetes
service
run             Run a particular image on the cluster
set            Set specific features on objects
```

Basic Commands (Intermediate):

```
explain        Get documentation for a resource
get            Display one or many resources
edit           Edit a resource on the server
delete         Delete resources by file names, stdin, resources and names, or by resources and label selector
```

Deploy Commands:

```
rollout        Manage the rollout of a resource
```

scale	Set a new size for a deployment, replica set, or replication controller
autoscale	Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:

certificate	Modify certificate resources.
cluster-info	Display cluster information
top	Display resource (CPU/memory) usage
cordons	Mark node as unschedulable
uncordon	Mark node as schedulable
drain	Drain node in preparation for maintenance
taint	Update the taints on one or more nodes

Troubleshooting and Debugging Commands:

describe	Show details of a specific resource or group of resources
logs	Print the logs for a container in a pod
attach	Attach to a running container
exec	Execute a command in a container
port-forward	Forward one or more local ports to a pod
proxy	Run a proxy to the Kubernetes API server
cp	Copy files and directories to and from containers
auth	Inspect authorization
debug	Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:

diff	Diff the live version against a would-be applied version
apply	Apply a configuration to a resource by file name or stdin
patch	Update fields of a resource
replace	Replace a resource by file name or stdin
wait	Experimental: Wait for a specific condition on one or many resources
kustomize	Build a kustomization target from a directory or URL.

Settings Commands:

label	Update the labels on a resource
annotate	Mettre à jour les annotations d'une ressource

```
completion      Output shell completion code for the specified shell (bash, zsh, fish, or powershell)
```

Other Commands:

```
alpha           Commands for features in alpha
api-resources   Print the supported API resources on the server
api-versions    Print the supported API versions on the server, in the form of "group/version"
config         Modifier des fichiers kubeconfig
plugin          Provides utilities for interacting with plugins
version        Print the client and server version information
```

Usage:

```
kubectl [flags] [options]
```

Use "kubectl <command> --help" for more information about a given command.

Use "kubectl options" for a list of global command-line options (applies to all commands).

More information about each command can be obtained by passing the **-help** option, for example :

```
root@kubemaster:~# kubectl create --help
Create a resource from a file or from stdin.
```

```
JSON and YAML formats are accepted.
```

Examples:

```
# Create a pod using the data in pod.json
kubectl create -f ./pod.json
# Create a pod based on the JSON passed into stdin
cat pod.json | kubectl create -f -
# Edit the data in registry.yaml in JSON then create the resource using the edited data
kubectl create -f registry.yaml --edit -o json
```

Available Commands:

```
clusterrole      Create a cluster role
clusterrolebinding  Create a cluster role binding for a particular cluster role
```

configmap	Create a config map from a local file, directory or literal value
cronjob	Create a cron job with the specified name
deployment	Create a deployment with the specified name
ingress	Create an ingress with the specified name
job	Create a job with the specified name
namespace	Create a namespace with the specified name
poddisruptionbudget	Create a pod disruption budget with the specified name
priorityclass	Create a priority class with the specified name
quota	Create a quota with the specified name
role	Create a role with single rule
rolebinding	Create a role binding for a particular role or cluster role
secret	Create a secret using specified subcommand
service	Create a service using a specified subcommand
serviceaccount	Create a service account with the specified name
token	Request a service account token

Options:

`--allow-missing-template-keys=true:`

If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to

golang and jsonpath output formats.

`--dry-run='none':`

Must be "none", "server", or "client". If client strategy, only print the object that would be sent, without

sending it. If server strategy, submit server-side request without persisting the resource.

`--edit=false:`

Edit the API resource before creating

`--field-manager='kubectl-create':`

Name of the manager used to track field ownership.

`-f, --filename=[]:`

Filename, directory, or URL to files to use to create the resource

-k, --kustomize='':

Process the kustomization directory. This flag can't be used together with -f or -R.

-o, --output='':

Output format. One of: (json, yaml, name, go-template, go-template-file, template, templatefile, jsonpath, jsonpath-as-json, jsonpath-file).

--raw='':

Raw URI to POST to the server. Uses the transport specified by the kubeconfig file.

-R, --recursive=false:

Process the directory used in -f, --filename recursively. Useful when you want to manage related manifests organized within the same directory.

--save-config=false:

If true, the configuration of current object will be saved in its annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to perform kubectl apply on this object in the future.

-l, --selector='':

Selector (label query) to filter on, supports '=', '==', and '!='.(e.g. -l key1=value1,key2=value2). Matching objects must satisfy all of the specified label constraints.

--show-managed-fields=false:

If true, keep the managedFields when printing objects in JSON or YAML format.

--template='':

Template string or path to template file to use when -o=go-template, -o=go-template-file. The template format

```
is golang templates [http://golang.org/pkg/text/template/#pkg-overview].

--validate='strict':
    Must be one of: strict (or true), warn, ignore (or false).          "true" or "strict" will use a
schema to validate
    the input and fail the request if invalid. It will perform server side validation if
ServerSideFieldValidation
    is enabled on the api-server, but will fall back to less reliable client-side validation if not.
"warn" will
    warn about unknown or duplicate fields without blocking the request if server-side field validation is
enabled
    on the API server, and behave as "ignore" otherwise.          "false" or "ignore" will not perform any
schema
    validation, silently dropping any unknown or duplicate fields.

--windows-line-endings=false:
    Only relevant if --edit=true. Defaults to the line ending native to your platform.

Usage:
  kubectl create -f FILENAME [options]

Use "kubectl <command> --help" for more information about a given command.
Use "kubectl options" for a list of global command-line options (applies to all commands).
```

Lastly, kubectl commands can be given options. To view the options that can be passed to all kubectl commands, enter the following command:

```
root@kubemaster:~# kubectl options
The following options can be passed to any command:

--add-dir-header=false:
    If true, adds the file directory to the header of the log messages (DEPRECATED: will be removed in a
future
    release, see
https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flag
```

s-in-k8s-components)

--alsologtostderr=false:

log to standard error as well as files (no effect when -logtostderr=true) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--as='':

Username to impersonate for the operation. User could be a regular user or a service account in a namespace.

--as-group=[]:

Group to impersonate for the operation, this flag can be repeated to specify multiple groups.

--as-uid='':

UID to impersonate for the operation.

--cache-dir='/root/.kube/cache':

Default cache directory

--certificate-authority='':

Path to a cert file for the certificate authority

--client-certificate='':

Path to a client certificate file for TLS

--client-key='':

Path to a client key file for TLS

--cluster='':

The name of the kubeconfig cluster to use

--context='':

The name of the kubeconfig context to use

--insecure-skip-tls-verify=false:

If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure

--kubeconfig='':

Path to the kubeconfig file to use for CLI requests.

--log-backtrace-at=:0:

when logging hits line file:N, emit a stack trace (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--log-dir='':

If non-empty, write log files in this directory (no effect when -logtostderr=true) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--log-file='':

If non-empty, use this log file (no effect when -logtostderr=true) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--log-file-max-size=1800:

Defines the maximum size a log file can grow to (no effect when -logtostderr=true). Unit is megabytes. If the value is 0, the maximum file size is unlimited. (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--log-flush-frequency=5s:

Maximum number of seconds between log flushes

--logtostderr=true:

log to standard error instead of files (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--match-server-version=false:

Require server version to match client version

-n, --namespace='':

If present, the namespace scope for this CLI request

--one-output=false:

If true, only write logs to their native severity level (vs also writing to each lower severity level; no effect when -logtostderr=true) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

--password='':

Password for basic authentication to the API server

--profile='none':

Name of profile to capture. One of (none|cpu|heap|goroutine|threadcreate|block|mutex)

--profile-output='profile.pprof':

Name of the file to write the profile to

--request-timeout='0':

The length of time to wait before giving up on a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.

-s, --server='':

The address and port of the Kubernetes API server

`--skip-headers=false:`

If true, avoid header prefixes in the log messages (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

`--skip-log-headers=false:`

If true, avoid headers when opening log files (no effect when `-logtostderr=true`) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

`--stderrthreshold=2:`

logs at or above this threshold go to stderr when writing to files and stderr (no effect when `-logtostderr=true` or `-alsologtostderr=false`) (DEPRECATED: will be removed in a future release, see <https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/2845-deprecate-klog-specific-flags-in-k8s-components>)

`--tls-server-name='':`

Server name to use for server certificate validation. If it is not provided, the hostname used to contact the server is used

`--token='':`

Bearer token for authentication to the API server

`--user='':`

The name of the kubeconfig user to use

`--username='':`

Username for basic authentication to the API server

```
-v, --v=0:  
    number for the log level verbosity  
  
--vmodule=:  
    comma-separated list of pattern=N settings for file-filtered logging  
  
--warnings-as-errors=false:  
    Treat warnings received from the server as errors and exit with a non-zero exit code
```

1.2 - Obtaining information about the Cluster

The version Command

Start by obtaining the client and server version information:

```
root@kubemaster:~# kubectl version --short  
Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.  
Client Version: v1.25.0  
Kustomize Version: v4.5.7  
Server Version: v1.25.0
```

The cluster-info command

Then view the cluster information:

```
root@kubemaster:~# kubectl cluster-info  
Kubernetes control plane is running at https://192.168.56.2:6443  
CoreDNS is running at https://192.168.56.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy  
  
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

The api-versions command

To find out which API versions are compatible with the Kubernetes version you have installed, run the **api-versions** command:

```
root@kubemaster:~# kubectl api-versions
admissionregistration.k8s.io/v1
apiextensions.k8s.io/v1
apiregistration.k8s.io/v1
apps/v1
authentication.k8s.io/v1
authorization.k8s.io/v1
autoscaling/v1
autoscaling/v2
autoscaling/v2beta2
batch/v1
certificates.k8s.io/v1
coordination.k8s.io/v1
crd.projectcalico.org/v1
discovery.k8s.io/v1
events.k8s.io/v1
flowcontrol.apiserver.k8s.io/v1beta1
flowcontrol.apiserver.k8s.io/v1beta2
networking.k8s.io/v1
node.k8s.io/v1
policy/v1
rbac.authorization.k8s.io/v1
scheduling.k8s.io/v1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
```

The api-resources command

The **api-resources** command displays the list of cluster resources, including :

- the resource name - **NAME**,
- the short name to be used with kubectl - **SHORTNAMES**,
- the API group to which the resource belongs - **APIVERSION**,
- whether or not the resource is linked to a namespace - **NAMESPACED**,
- the resource's KIND type - **KIND**.

```
root@kubemaster:~# kubectl api-resources
NAME                SHORTNAMES  APIVERSION  NAMESPACED  KIND
bindings            v1          v1          true        Binding
componentstatuses   cs          v1          false
ComponentStatus
configmaps          cm          v1          true        ConfigMap
endpoints            ep          v1          true        Endpoints
events              ev          v1          true        Event
limitranges         limits      v1          true        LimitRange
namespaces          ns          v1          false       Namespace
nodes               no          v1          false       Node
persistentvolumeclaims  pvc        v1          true
PersistentVolumeClaim
persistentvolumes    pv          v1          false
PersistentVolume
pods                po          v1          true        Pod
podtemplates        v1          v1          true        PodTemplate
replicationcontrollers  rc          v1          true
ReplicationController
resourcequotas       quota       v1          true        ResourceQuota
secrets              v1          v1          true        Secret
serviceaccounts      sa          v1          true        ServiceAccount
services             svc          v1          true        Service
mutatingwebhookconfigurations  admissionregistration.k8s.io/v1  false
```

MutatingWebhookConfiguration					
validatingwebhookconfigurations			admissionregistration.k8s.io/v1	false	
ValidatingWebhookConfiguration					
customresourcedefinitions	crd,crds		apiextensions.k8s.io/v1	false	
CustomResourceDefinition					
apiservices			apiregistration.k8s.io/v1	false	APIService
controllerrevisions			apps/v1	true	
ControllerRevision					
daemonsets	ds		apps/v1	true	DaemonSet
deployments	deploy		apps/v1	true	Deployment
replicasets	rs		apps/v1	true	ReplicaSet
statefulsets	sts		apps/v1	true	StatefulSet
tokenreviews			authentication.k8s.io/v1	false	TokenReview
localsubjectaccessreviews			authorization.k8s.io/v1	true	
LocalSubjectAccessReview					
selfsubjectaccessreviews			authorization.k8s.io/v1	false	
SelfSubjectAccessReview					
selfsubjectrulesreviews			authorization.k8s.io/v1	false	
SelfSubjectRulesReview					
subjectaccessreviews			authorization.k8s.io/v1	false	
SubjectAccessReview					
horizontalpodautoscalers	hpa		autoscaling/v2	true	
HorizontalPodAutoscaler					
cronjobs	cj		batch/v1	true	CronJob
jobs			batch/v1	true	Job
certificatesigningrequests	csr		certificates.k8s.io/v1	false	
CertificateSigningRequest					
leases			coordination.k8s.io/v1	true	Lease
bgpconfigurations			crd.projectcalico.org/v1	false	
BGPConfiguration					
bgppeers			crd.projectcalico.org/v1	false	BGPPeer
blockaffinities			crd.projectcalico.org/v1	false	BlockAffinity
caliconodestatuses			crd.projectcalico.org/v1	false	
CalicoNodeStatus					

clusterinformations		crd.projectcalico.org/v1	false	
ClusterInformation				
felixconfigurations		crd.projectcalico.org/v1	false	
FelixConfiguration				
globalnetworkpolicies		crd.projectcalico.org/v1	false	
GlobalNetworkPolicy				
globalnetworksets		crd.projectcalico.org/v1	false	
GlobalNetworkSet				
hostendpoints		crd.projectcalico.org/v1	false	HostEndpoint
ipamblocks		crd.projectcalico.org/v1	false	IPAMBlock
ipamconfigs		crd.projectcalico.org/v1	false	IPAMConfig
ipamhandles		crd.projectcalico.org/v1	false	IPAMHandle
ippools		crd.projectcalico.org/v1	false	IPPool
ipreservations		crd.projectcalico.org/v1	false	IPReservation
kubecontrollersconfigurations		crd.projectcalico.org/v1	false	
KubeControllersConfiguration				
networkpolicies		crd.projectcalico.org/v1	true	NetworkPolicy
networksets		crd.projectcalico.org/v1	true	NetworkSet
endpointslices		discovery.k8s.io/v1	true	EndpointSlice
events	ev	events.k8s.io/v1	true	Event
flowschemas		flowcontrol.apiserver.k8s.io/v1beta2	false	FlowSchema
prioritylevelconfigurations		flowcontrol.apiserver.k8s.io/v1beta2	false	
PriorityLevelConfiguration				
ingressclasses		networking.k8s.io/v1	false	IngressClass
ingresses	ing	networking.k8s.io/v1	true	Ingress
networkpolicies	netpol	networking.k8s.io/v1	true	NetworkPolicy
runtimeclasses		node.k8s.io/v1	false	RuntimeClass
poddisruptionbudgets	pdb	policy/v1	true	
PodDisruptionBudget				
clusterrolebindings		rbac.authorization.k8s.io/v1	false	
ClusterRoleBinding				
clusterroles		rbac.authorization.k8s.io/v1	false	ClusterRole
rolebindings		rbac.authorization.k8s.io/v1	true	RoleBinding
roles		rbac.authorization.k8s.io/v1	true	Role

priorityclasses	pc	scheduling.k8s.io/v1	false	PriorityClass
csidrivers		storage.k8s.io/v1	false	CSIDriver
csinodes		storage.k8s.io/v1	false	CSINode
csistoragecapacities		storage.k8s.io/v1	true	
CSIStorageCapacity				
storageclasses	sc	storage.k8s.io/v1	false	StorageClass
volumeattachments		storage.k8s.io/v1	false	
VolumeAttachment				

1.3 - Obtaining information about nodes

The describe node command

Node information can be obtained with the **describe node** command. In the first part of the command output, you can see:

- the **Labels:** section. Labels can be used to manage the affinity of a pod, i.e. on which node a pod can be scheduled according to the labels associated with the pod,
- the **Unschedulable: false** line, which indicates that the node accepts pods.

```

root@kubemaster:~# kubectl describe node kubemaster.ittraining.loc
Name:          kubemaster.ittraining.loc
Roles:        control-plane
Labels:       beta.kubernetes.io/arch=amd64
              beta.kubernetes.io/os=linux
              kubernetes.io/arch=amd64
              kubernetes.io/hostname=kubemaster.ittraining.loc
              kubernetes.io/os=linux
              node-role.kubernetes.io/control-plane=
              node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:  kubeadm.alpha.kubernetes.io/cni-socket: unix:///var/run/containerd/containerd.sock
              node.alpha.kubernetes.io/ttl: 0
              projectcalico.org/IPv4Address: 192.168.56.2/24

```

```

projectcalico.org/IPv4IPIPTunnelAddr: 192.168.55.192
volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Sun, 04 Sep 2022 09:36:00 +0200
Taints:
node-role.kubernetes.io/control-plane:NoSchedule
Unschedulable: false
Lease:
  HolderIdentity: kubemaster.ittraining.loc
  AcquireTime: <unset>
  RenewTime: Sun, 04 Sep 2022 16:56:54 +0200
Conditions:
  Type                Status  LastHeartbeatTime               LastTransitionTime              Reason
  Message
  ----                -
  -----
  NetworkUnavailable  False   Sun, 04 Sep 2022 09:44:21 +0200   Sun, 04 Sep 2022 09:44:21 +0200   CalicoIsUp
  Calico is running on this node
  MemoryPressure      False   Sun, 04 Sep 2022 16:52:48 +0200   Sun, 04 Sep 2022 09:35:59 +0200
  KubeletHasSufficientMemory kubelet has sufficient memory available
  DiskPressure        False   Sun, 04 Sep 2022 16:52:48 +0200   Sun, 04 Sep 2022 09:35:59 +0200
  KubeletHasNoDiskPressure kubelet has no disk pressure
  PIDPressure         False   Sun, 04 Sep 2022 16:52:48 +0200   Sun, 04 Sep 2022 09:35:59 +0200
  KubeletHasSufficientPID kubelet has sufficient PID available
  Ready               True    Sun, 04 Sep 2022 16:52:48 +0200   Sun, 04 Sep 2022 12:15:32 +0200   KubeletReady
  kubelet is posting ready status
  ...

```

In the second part of the output, we can see :

- the **Addresses:** section containing the node's IP address and host name.

```

...
Addresses:
  InternalIP: 10.0.2.65
  Hostname:   kubemaster.ittraining.loc

```

```
Capacity:
  cpu: 4
  ephemeral-storage: 18400976Ki
  hugepages-2Mi: 0
  memory: 8181164Ki
  pods: 110
Allocatable:
  cpu: 4
  ephemeral-storage: 16958339454
  hugepages-2Mi: 0
  memory: 8078764Ki
  pods: 110
...
```

In the third part of the output, we can see :

- the **System Info:** section, containing information on the operating system, Docker and Kubernetes versions,
- the **Non-terminated Pods** section, containing information on the CPU and memory limits of each running POD.

```
...
System Info:
  Machine ID: 243c6f9d604e4aba852a482a1936be23
  System UUID: 68639C3A-D77A-4C61-B7E8-4F4F70419B8A
  Boot ID: 9bd56aa5-b94c-40d3-804a-a54bd8daf305
  Kernel Version: 4.9.0-19-amd64
  OS Image: Debian GNU/Linux 9 (stretch)
  Operating System: linux
  Architecture: amd64
  Container Runtime Version: containerd://1.4.3
  Kubelet Version: v1.25.0
  Kube-Proxy Version: v1.25.0
PodCIDR: 192.168.0.0/24
PodCIDRs: 192.168.0.0/24
Non-terminated Pods: (7 in total)
```

Namespace	Name	CPU Requests	CPU Limits
Memory Requests	Memory Limits	Age	
-----	-----	-----	-----
kube-system	calico-node-dc7hd	250m (6%)	0 (0%)
(0%)	7h18m		0
kube-system	coredns-565d847f94-tqd8z	100m (2%)	0 (0%)
(0%)	3h56m		70Mi
kube-system	etcd-kubemaster.ittraining.loc	100m (2%)	0 (0%)
100Mi (1%)	4h19m		
kube-system	kube-apiserver-kubemaster.ittraining.loc	250m (6%)	0 (0%)
(0%)	4h19m		0
kube-system	kube-controller-manager-kubemaster.ittraining.loc	200m (5%)	0 (0%)
(0%)	4h19m		0
kube-system	kube-proxy-x7fpc	0 (0%)	0 (0%)
(0%)	4h25m		0
kube-system	kube-scheduler-kubemaster.ittraining.loc	100m (2%)	0 (0%)
(0%)	4h19m		0
...			

In the last part of the output, we can see :

- the **Allocated resources:** section, which indicates the resources allocated to the node.

```
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
Resource           Requests    Limits
-----
cpu                 1 (25%)    0 (0%)
memory              170Mi (2%) 170Mi (2%)
ephemeral-storage  0 (0%)     0 (0%)
hugepages-2Mi      0 (0%)     0 (0%)
Events:
Type   Reason          Age   From          Message
```

```
-----  
Normal RegisteredNode 37m node-controller Node kubemaster.ittraining.loc event: Registered Node  
kubemaster.ittraining.loc in Controller
```

The top command

The **top** command requires the **Metrics** API to be available in the cluster. To deploy the Metrics server, download the file **components.yaml** :

```
root@kubemaster:~# wget  
https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.4.1/components.yaml
```



Important: If the above URL does not work, click [here](#). Copy the contents of the file and create it in the current directory.

Modify the **containers** section of the **components.yaml** file:

```
root@kubemaster:~# vi components.yaml  
...  
spec:  
  containers:  
  - args:  
    - --cert-dir=/tmp  
    - --secure-port=4443  
    - --kubelet-insecure-tls  
    - --kubelet-preferred-address-types=InternalIP,Hostname,InternalDNS,ExternalDNS,ExternalIP  
    - --kubelet-use-node-status-port  
  ...
```

Deploy the Metrics server :

```
root@kubemaster:~# kubectl apply -f components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

Check the status of deployment :

```
root@kubemaster:~# kubectl get deployments --all-namespaces
NAMESPACE      NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
default        myapp-deployment                  3/3     3             3           6h50m
kube-system    calico-kube-controllers           1/1     1             1           7h22m
kube-system    coredns                           2/2     2             2           7h25m
kube-system    metrics-server                    1/1     1             1           28s
```

To find out the node's resource usage, use the **top nodes** command:

```
root@kubemaster:~# kubectl top nodes
NAME                               CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
kubemaster.ittraining.loc          182m         4%    1901Mi          24%
kubenode1.ittraining.loc           68m          1%    898Mi           23%
kubenode2.ittraining.loc           104m         2%    819Mi           21%
```

To see the evolution of the resources used by the node, use the **watch** command.

```
root@kubemaster:~# watch kubectl top nodes
Every 2,0s: kubectl top nodes
kubemaster.ittraining.loc: Sun Sep  4 17:02:45 2022
```

```
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
kubemaster.ittraining.loc  142m        3%    1951Mi          24%
kubenode1.ittraining.loc   71m         1%    899Mi           23%
kubenode2.ittraining.loc   52m         1%    742Mi           19%
...
^C
root@kubemaster:~#
```



Important: Note the use of **^C** to exit the **watch** command.

The output can be sorted in descending order of the CPU usage:

```
root@kubemaster:~# kubectl top nodes --sort-by cpu
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
kubemaster.ittraining.loc  132m        3%    1916Mi          24%
kubenode1.ittraining.loc   65m         1%    952Mi           24%
kubenode2.ittraining.loc   50m         1%    887Mi           23%
```

Finally, it is possible to sort the output in descending order of the memory usage:

```
root@kubemaster:~# kubectl top nodes --sort-by memory
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
kubemaster.ittraining.loc  139m        3%    1909Mi          24%
kubenode1.ittraining.loc   70m         1%    951Mi           24%
kubenode2.ittraining.loc   52m         1%    885Mi           23%
```

1.4 - Obtaining information about Pods

The describe pod command

As with nodes, information about a specific pod can be obtained using the **kubectl describe** command:

```
root@kubemaster:~# kubectl describe pod myapp-deployment-689f9d59-c25f9
Name:          myapp-deployment-689f9d59-c25f9
Namespace:     default
Priority:       0
Service Account: default
Node:          kubenode1.ittraining.loc/192.168.56.3
Start Time:    Sun, 04 Sep 2022 13:23:12 +0200
Labels:        app=myapp
                pod-template-hash=689f9d59
                type=front-end
Annotations:   cni.projectcalico.org/containerID:
0d234054b43a4bd5c8a3c8f0a9e0b8594a8d1abdccdad8b656c311ad31731a54
                cni.projectcalico.org/podIP: 192.168.239.9/32
                cni.projectcalico.org/podIPs: 192.168.239.9/32
Status:        Running
IP:            192.168.239.9
IPs:
  IP:          192.168.239.9
Controlled By: ReplicaSet/myapp-deployment-689f9d59
Containers:
  nginx-container:
    Container ID:  containerd://b0367fe494be444f98facd069f5a6e48fadce9236ad5a1baa5feb31d2a08760a
    Image:         nginx
    Image ID:
docker.io/library/nginx@sha256:b95a99feebf7797479e0c5eb5ec0bdfa5d9f504bc94da550c2f58e839ea6914f
    Port:         <none>
    Host Port:    <none>
    State:        Running
    Started:      Sun, 04 Sep 2022 13:23:21 +0200
```

```
Ready:      True
Restart Count: 0
Environment: <none>
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-fjdsd (ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  kube-api-access-fjdsd:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:    true
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:          <none>
```

The top command

It is possible to view the resource usage by pod:

```
root@kubemaster:~# kubectl top pods
NAME                                CPU(cores)   MEMORY(bytes)
myapp-deployment-689f9d59-c25f9    0m           3Mi
myapp-deployment-689f9d59-nn9sw    0m           4Mi
```

```
myapp-deployment-689f9d59-rnc4r 0m 4Mi
```

Now sort the output in descending order of processor usage:

```
root@kubemaster:~# kubectl top pods
NAME                                CPU(cores)  MEMORY(bytes)
myapp-deployment-689f9d59-c25f9    0m          3Mi
myapp-deployment-689f9d59-nn9sw    0m          4Mi
myapp-deployment-689f9d59-rnc4r    0m          4Mi
```

Now sort the output in descending order of memory usage:

```
root@kubemaster:~# kubectl top pods --sort-by memory
NAME                                CPU(cores)  MEMORY(bytes)
myapp-deployment-689f9d59-nn9sw    0m          4Mi
myapp-deployment-689f9d59-rnc4r    0m          4Mi
myapp-deployment-689f9d59-c25f9    0m          3Mi
```

1.5 - Working with the kubectl command

Create the file **pod.yaml** :



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi pod.yaml
root@kubemaster:~# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
```

```
name: my-pod
spec:
  containers:
  - name: busybox
    image: radial/busyboxplus:curl
    command: ['sh', '-c', 'while true; do sleep 3600; done']
```

The apply command

Now create the pod using the **pod.yaml** file:

```
root@kubemaster:~# kubectl apply -f pod.yaml
pod/my-pod created
```

The create command

The **create** command can only be used if an object with the same name does not already exist in the cluster:

```
root@kubemaster:~# kubectl create -f pod.yaml
Error from server (AlreadyExists): error when creating "pod.yaml": pods "my-pod" already exists
```

To view the list of objects that can be created, use the **kubectl create** command:

```
root@kubemaster:~# kubectl create
Error: must specify one of -f and -k
```

Create a resource from a file or from stdin.

JSON and YAML formats are accepted.

Examples:

```
# Create a pod using the data in pod.json
kubectl create -f ./pod.json
# Create a pod based on the JSON passed into stdin
cat pod.json | kubectl create -f -
# Edit the data in registry.yaml in JSON then create the resource using the edited data
kubectl create -f registry.yaml --edit -o json
```

Available Commands:

clusterrole	Create a cluster role
clusterrolebinding	Create a cluster role binding for a particular cluster role
configmap	Create a config map from a local file, directory or literal value
cronjob	Create a cron job with the specified name
deployment	Create a deployment with the specified name
ingress	Create an ingress with the specified name
job	Create a job with the specified name
namespace	Create a namespace with the specified name
poddisruptionbudget	Create a pod disruption budget with the specified name
priorityclass	Create a priority class with the specified name
quota	Create a quota with the specified name
role	Create a role with single rule
rolebinding	Create a role binding for a particular role or cluster role
secret	Create a secret using specified subcommand
service	Create a service using a specified subcommand
serviceaccount	Create a service account with the specified name
token	Request a service account token

Options:

--allow-missing-template-keys=true:

If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to
golang and jsonpath output formats.

--dry-run='none':

Must be "none", "server", or "client". If client strategy, only print the object that would be sent,

without

sending it. If server strategy, submit server-side request without persisting the resource.

--edit=false:

Edit the API resource before creating

--field-manager='kubectl-create':

Name of the manager used to track field ownership.

-f, --filename=[]:

Filename, directory, or URL to files to use to create the resource

-k, --kustomize='':

Process the kustomization directory. This flag can't be used together with -f or -R.

-o, --output='':

Output format. One of: (json, yaml, name, go-template, go-template-file, template, templatefile, jsonpath, jsonpath-as-json, jsonpath-file).

--raw='':

Raw URI to POST to the server. Uses the transport specified by the kubeconfig file.

-R, --recursive=false:

Process the directory used in -f, --filename recursively. Useful when you want to manage related manifests organized within the same directory.

--save-config=false:

If true, the configuration of current object will be saved in its annotation. Otherwise, the annotation will be unchanged. This flag is useful when you want to perform kubectl apply on this object in the future.

-l, --selector='':

Selector (label query) to filter on, supports '=', '==', and '!='.(e.g. -l key1=value1,key2=value2).

Matching

objects must satisfy all of the specified label constraints.

`--show-managed-fields=false:`

If true, keep the managedFields when printing objects in JSON or YAML format.

`--template='':`

Template string or path to template file to use when `-o=go-template`, `-o=go-template-file`. The template format

is go lang templates [<http://golang.org/pkg/text/template/#pkg-overview>].

`--validate='strict':`

Must be one of: strict (or true), warn, ignore (or false). "true" or "strict" will use a schema to validate

the input and fail the request if invalid. It will perform server side validation if

ServerSideFieldValidation

is enabled on the api-server, but will fall back to less reliable client-side validation if not.

"warn" will

warn about unknown or duplicate fields without blocking the request if server-side field validation is enabled

on the API server, and behave as "ignore" otherwise. "false" or "ignore" will not perform any

schema

validation, silently dropping any unknown or duplicate fields.

`--windows-line-endings=false:`

Only relevant if `--edit=true`. Defaults to the line ending native to your platform.

Usage:

```
kubectl create -f FILENAME [options]
```

Use "kubectl <command> --help" for more information about a given command.

Use "kubectl options" for a list of global command-line options (applies to all commands).

The **apply** command is then used to apply changes made to the yaml file:

```
root@kubemaster:~# kubectl apply -f pod.yaml
pod/my-pod unchanged
```

The get command

Check pod status:

```
root@kubemaster:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-pod                              1/1     Running   0           10s
myapp-deployment-689f9d59-c25f9     1/1     Running   0           6d1h
myapp-deployment-689f9d59-nn9sw     1/1     Running   0           6d1h
myapp-deployment-689f9d59-rnc4r     1/1     Running   0           6d1h
```

Remember that you can use a short code for the word pods:

```
root@kubemaster:~# kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
my-pod                              1/1     Running   0           54s
myapp-deployment-689f9d59-c25f9     1/1     Running   0           6d1h
myapp-deployment-689f9d59-nn9sw     1/1     Running   0           6d1h
myapp-deployment-689f9d59-rnc4r     1/1     Running   0           6d1h
```

To see the status of only one particular pod, specify its name as an argument:

```
root@kubemaster:~# kubectl get po my-pod
NAME    READY   STATUS    RESTARTS   AGE
my-pod  1/1     Running   0           109s
```

Using Options

Remember that the **wide** option lets you view pod IP addresses and the nodes hosting them:

```
root@kubemaster:~# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE   READINESS GATES
my-pod                                1/1    Running   0           115s  192.168.150.9   kubenode2.ittraining.loc
<none>                <none>
myapp-deployment-689f9d59-c25f9      1/1    Running   0           6d1h  192.168.239.9   kubenode1.ittraining.loc
<none>                <none>
myapp-deployment-689f9d59-nn9sw      1/1    Running   0           6d1h  192.168.239.13  kubenode1.ittraining.loc
<none>                <none>
myapp-deployment-689f9d59-rnc4r     1/1    Running   0           6d1h  192.168.239.12  kubenode1.ittraining.loc
<none>                <none>
```

Using the **json** option displays the same information in json format:

```
root@kubemaster:~# kubectl get pods -o json | more
{
  "apiVersion": "v1",
  "items": [
    {
      "apiVersion": "v1",
      "kind": "Pod",
      "metadata": {
        "annotations": {
          "cni.projectcalico.org/containerID":
"584cf2663957e8a6d5628c7f316e5858629ea646ec890bd5d6f9d1e217963b52",
          "cni.projectcalico.org/podIP": "192.168.150.9/32",
          "cni.projectcalico.org/podIPs": "192.168.150.9/32",
          "kubectl.kubernetes.io/last-applied-configuration":
"{\"apiVersion\": \"v1\", \"kind\": \"Pod\", \"metadata\"
```

```
":{"annotations\":{}},"name\":"my-  
pod","\namespace\":"default"},"spec\":{"containers\":[{"command\":["sh","\-c","\n  
"while true; do sleep 3600; done"],"image\":"radial/busyboxplus:curl","\name\":"busybox"}]}}\n"  
  },  
  "creationTimestamp": "2022-09-10T13:03:20Z",  
  "name": "my-pod",  
  "namespace": "default",  
  "resourceVersion": "755938",  
  "uid": "628ca9e4-2fbe-4fc9-b0fa-9a05ef942a07"  
},  
"spec": {  
  "containers": [  
    {  
      "command": [  
        "sh",  
        "-c",  
        "while true; do sleep 3600; done"  
      ],  
      "image": "radial/busyboxplus:curl",  
      "imagePullPolicy": "IfNotPresent",  
      "name": "busybox",  
      "resources": {},  
      "terminationMessagePath": "/dev/termination-log",  
      "terminationMessagePolicy": "File",  
      "volumeMounts": [  
        {  
          "mountPath": "/var/run/secrets/kubernetes.io/serviceaccount",  
          "name": "kube-api-access-qwzzv",  
          "readOnly": true,  
          "recursive": false,  
          "subPath": ""  
        }  
      ]  
    }  
  ]  
}
```

--More--

Use the **yaml** option to view the same information in yaml format:

```
root@kubemaster:~# kubectl get pods -o yaml | more  
apiVersion: v1
```

```
items:
- apiVersion: v1
  kind: Pod
  metadata:
    annotations:
      cni.projectcalico.org/containerID: 584cf2663957e8a6d5628c7f316e5858629ea646ec890bd5d6f9d1e217963b52
      cni.projectcalico.org/podIP: 192.168.150.9/32
      cni.projectcalico.org/podIPs: 192.168.150.9/32
      kubectrl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"my-
pod","namespace":"default"},"spec":{"contain
ers":[{"command":["sh","-c","while true; do sleep 3600;
done"],"image":"radial/busyboxplus:curl","name":"busybox"}]}
      creationTimestamp: "2022-09-10T13:03:20Z"
      name: my-pod
      namespace: default
      resourceVersion: "755938"
      uid: 628ca9e4-2fbe-4fc9-b0fa-9a05ef942a07
  spec:
    containers:
    - command:
      - sh
      - -c
      - while true; do sleep 3600; done
      image: radial/busyboxplus:curl
      imagePullPolicy: IfNotPresent
      name: busybox
      resources: {}
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
        name: kube-api-access-qwzzv
        readOnly: true
```

```

  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: kubenode2.ittraining.loc
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
--More--

```

The **-sort-by** option is used to sort the output according to a yaml key:

```

root@kubemaster:~# kubectl get pods -o wide --sort-by .spec.nodeName
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE   READINESS GATES
myapp-deployment-689f9d59-c25f9    1/1    Running   0           6d1h  192.168.239.9   kubenode1.ittraining.loc
<none>                <none>
myapp-deployment-689f9d59-nn9sw    1/1    Running   0           6d1h  192.168.239.13  kubenode1.ittraining.loc
<none>                <none>
myapp-deployment-689f9d59-rnc4r    1/1    Running   0           6d1h  192.168.239.12  kubenode1.ittraining.loc
<none>                <none>
my-pod                             1/1    Running   0           3m22s  192.168.150.9   kubenode2.ittraining.loc
<none>                <none>

```

The **-selector** option allows you to see only those pods that match the specified label, for example, **k8s-app** :

```

root@kubemaster:~# kubectl get pods -n kube-system --selector k8s-app=calico-node
NAME                READY   STATUS    RESTARTS   AGE
calico-node-5htrc   1/1    Running   0           6d5h
calico-node-dc7hd   1/1    Running   0           6d5h
calico-node-qk5kt   1/1    Running   0           6d5h

```

The exec command

The **exec** command is used to execute a command in the container. The command is preceded by the characters `--` :

```
root@kubemaster:~# kubectl exec my-pod -c busybox -- echo "Hello, world!"
Hello, world!
```



Important: Note the use of **-c** to indicate the container name.

1.6 - Imperative commands

Before continuing, delete the **my-pod** pod:

```
root@kubemaster:~# kubectl delete pod my-pod
pod "my-pod" deleted
```

Next, create a deployment with the following imperative command :

```
root@kubemaster:~# kubectl create deployment my-deployment --image=nginx
deployment.apps/my-deployment created
```

By executing the same imperative command, it is possible to create instructions in yaml format using the **-dry-run** option:

```
root@kubemaster:~# kubectl create deployment my-deployment --image=nginx --dry-run -o yaml
W0910 15:28:49.797172 17135 helpers.go:639] --dry-run is deprecated and can be replaced with --dry-run=client.
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: my-deployment
```

```
name: my-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: my-deployment
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

These instructions can then be injected into a file in order to use it to create an identical deployment:

```
root@kubemaster:~# kubectl create deployment my-deployment --image=nginx --dry-run -o yaml > deployment.yaml
W0910 15:29:05.006256 17242 helpers.go:639] --dry-run is deprecated and can be replaced with --dry-run=client.
```

```
root@kubemaster:~# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: my-deployment
  name: my-deployment
spec:
  replicas: 1
```

```
selector:
  matchLabels:
    app: my-deployment
strategy: {}
template:
  metadata:
    creationTimestamp: null
    labels:
      app: my-deployment
  spec:
    containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

LAB #2 - Managing kubectl plugins with the krew command

kubectl plugins extend its functionality. The **krew** plugin manager is available for macOS™, Windows™ and Linux. A plugin is a simple executable written, for example, in **bash** or **Go**.

2.1 - Installing krew

To install the **krew** command, you must first install **git** :

```
root@kubemaster:~# apt install git-all
```

Then install krew with the following command:

```
(
set -x; cd "$(mktemp -d)" &&
```

```
curl -fsSLO "https://github.com/kubernetes-sigs/krew/releases/download/v0.4.3/krew-linux_amd64.tar.gz" &&
tar zxvf krew-linux_amd64.tar.gz &&
KREW=./krew-"$(uname | tr '[:upper:]' '[:lower:]')_$(uname -m | sed -e 's/x86_64/amd64/' -e 's/arm.*$/arm/')" &&
"$KREW" install krew
)
```

You will see this output:

```
root@kubemaster:~# (
> set -x; cd "$(mktemp -d)" &&
> curl -fsSLO "https://github.com/kubernetes-sigs/krew/releases/download/v0.4.3/krew-linux_amd64.tar.gz" &&
> tar zxvf krew-linux_amd64.tar.gz &&
> KREW=./krew-"$(uname | tr '[:upper:]' '[:lower:]')_$(uname -m | sed -e 's/x86_64/amd64/' -e 's/arm.*$/arm/')"
&&
> "$KREW" install krew
> )
++ mktemp -d
+ cd /tmp/tmp.eA3ZH8tKRg
+ curl -fsSLO https://github.com/kubernetes-sigs/krew/releases/download/v0.4.3/krew-linux_amd64.tar.gz
+ tar zxvf krew-linux_amd64.tar.gz
./LICENSE
./krew-linux_amd64
++ uname
++ tr '[:upper:]' '[:lower:]'
++ uname -m
++ sed -e s/x86_64/amd64/ -e 's/arm.*$/arm/'
+ KREW=./krew-linux_amd64
+ ./krew-linux_amd64 install krew
Adding "default" plugin index from https://github.com/kubernetes-sigs/krew-index.git.
Updated the local copy of plugin index.
Installing plugin: krew
Installed plugin: krew
\
| Use this plugin:
```

```
| kubectl krew
| Documentation:
| https://krew.sigs.k8s.io/
| Caveats:
| \
| krew is now installed! To start using kubectl plugins, you need to add
| krew's installation directory to your PATH:
| |
| | * macOS/Linux:
| | - Add the following to your ~/.bashrc or ~/.zshrc:
| | export PATH="{KREW_ROOT:-$HOME/.krew}/bin:$PATH"
| | - Restart your shell.
| |
| | * Windows: Add %USERPROFILE%\krew\bin to your PATH environment variable
| |
| To list krew commands and to get help, run:
| $ kubectl krew
| For a full list of available plugins, run:
| $ kubectl krew search
| |
| You can find documentation at
| https://krew.sigs.k8s.io/docs/user-guide/quickstart/.
| /
| /
```

Important: If you cannot download the file, download it from [here](#) and then use the following code:



```
(
set -x; cd "$(mktemp -d)" &&
tar zxvf krew-linux_amd64.tar.gz &&
KREW=./krew-"$(uname | tr '[:upper:]' '[:lower:]')_$(uname -m | sed
-e 's/x86_64/amd64/' -e 's/arm.*$/arm/')" &&
```



```
"$KREW" install krew  
)
```

Now add **\$HOME/.krew/bin** to your PATH:

```
root@kubemaster:~# export PATH="${KREW_ROOT:-$HOME/.krew}/bin:$PATH"
```

To avoid having to redefine the PATH after each login, add the line at the end of your **.bashrc** file:

```
root@kubemaster:~# echo 'export PATH="${KREW_ROOT:-$HOME/.krew}/bin:$PATH"' >> .bashrc
```

2.2 - Viewing the list of plugins

Update the list of plugins with the following command:

```
root@kubemaster:~# kubectl krew update  
Updated the local copy of plugin index.
```

To view the list of plugins, use the **search** command:

```
root@kubemaster:~# kubectl krew search
```

NAME	DESCRIPTION	INSTALLED
access-matrix	Show an RBAC access matrix for server resources	no
accurate	Manage Accurate, a multi-tenancy controller	no
advise-policy	Suggests PodSecurityPolicies and OPA Policies f...	no
advise-psp	Suggests PodSecurityPolicies for cluster.	no
allctx	Run commands on contexts in your kubeconfig	no
apparmor-manager	Manage AppArmor profiles for cluster.	no
assert	Assert Kubernetes resources	no
auth-proxy	Authentication proxy to a pod or service	no
aws-auth	Manage aws-auth ConfigMap	no

azad-proxy	Generate and handle authentication for azad-kub...	no
bd-xray	Run Black Duck Image Scans	no
blame	Show who edited resource fields.	no
bulk-action	Do bulk actions on Kubernetes resources.	no
ca-cert	Print the PEM CA certificate of the current clu...	no
capture	Triggers a Sysdig capture to troubleshoot the r...	no
cert-manager	Manage cert-manager resources inside your cluster	no
change-ns	View or change the current namespace via kubectl.	no
cilium	Easily interact with Cilium agents.	no
cluster-group	Exec commands across a group of contexts.	no
clusternet	Wrap multiple kubectl calls to Clusternet	no
cm	Provides commands for OCM/MCE/ACM.	no
cnpq	Manage your CloudNativePG clusters	no
config-cleanup	Automatically clean up your kubeconfig	no
config-registry	Switch between registered kubeconfigs	no
cost	View cluster cost information	no
creyaml	Generate custom resource YAML manifest	no
ctx	Switch between contexts in your kubeconfig	no
custom-cols	A "kubectl get" replacement with customizable c...	no
cyclonus	NetworkPolicy analysis tool suite	no
datadog	Manage the Datadog Operator	no
datree	Scan your cluster resources for misconfigurations	no
dds	Detect if workloads are mounting the docker socket	no
debug-shell	Create pod with interactive kube-shell.	no
deprecations	Checks for deprecated objects in a cluster	no
df-pv	Show disk usage (like unix df) for persistent v...	no
direct-csi	CSI driver to manage drives in k8s cluster as v...	no
directpv	Deploys and manages the lifecycle of DirectPV C...	no
doctor	Scans your cluster and reports anomalies.	no
dtlogin	Login to a cluster via openid-connect	no
duck	List custom resources with ducktype support	no
edit-status	Edit /status subresources of CRs	no
eds	Interact and manage ExtendedDaemonset resources	no
eksporter	Export resources and removes a pre-defined set ...	no

emit-event	Emit Kubernetes Events for the requested object	no
evict-pod	Evicts the given pod	no
example	Prints out example manifest YAMLS	no
exec-as	Like kubectl exec, but offers a `user` flag to ...	no
exec-cronjob	Run a CronJob immediately as Job	no
explore	A better kubectl explain with the fuzzy finder	no
fields	Grep resources hierarchy by field name	no
flame	Generate CPU flame graphs from pods	no
fleet	Shows config and resources of a fleet of clusters	no
flyte	Monitor, launch and manage flyte executions	no
fuzzy	Fuzzy and partial string search for kubectl	no
gadget	Gadgets for debugging and introspecting apps	no
get-all	Like `kubectl get all` but <u>really</u> everything	no
gke-credentials	Fetch credentials for GKE clusters	no
gopass	Imports secrets from gopass	no
graph	Visualize Kubernetes resources and relationships.	no
grep	Filter Kubernetes resources by matching their n...	no
gs	Handle custom resources with Giant Swarm	no
hlf	Deploy and manage Hyperledger Fabric components	no
hns	Manage hierarchical namespaces (part of HNC)	no
htpasswd	Create nginx-ingress compatible basic-auth secrets	no
ice	View configuration settings of containers insid...	no
iexec	Interactive selection tool for `kubectl exec`	no
images	Show container images used in the cluster.	no
ingress-nginx	Interact with ingress-nginx	no
ingress-rule	Update Ingress rules via command line	no
ipick	A kubectl wrapper for interactive resource sele...	no
istiolog	Manipulate istio-proxy logging level without is...	no
janitor	Lists objects in a problematic state	no
kadalu	Manage Kadalu Operator, CSI and Storage pods	no
karbon	Connect to Nutanix Karbon cluster	no
karmada	Manage clusters with Karmada federation.	no
konfig	Merge, split or import kubeconfig files	no
krew	Package manager for kubectl plugins.	yes

kruise	Easily handle OpenKruise workloads	no
ks	Simple management of KubeSphere components	no
ktop	A top tool to display workload metrics	no
kubesecc-scan	Scan Kubernetes resources with kubesecc.io.	no
kudo	Declaratively build, install, and run operators...	no
kuota-calc	Calculate needed quota to perform rolling updates.	no
kurt	Find what's restarting and why	no
kuttl	Declaratively run and test operators	no
kyverno	Kyverno is a policy engine for kubernetes	no
lineage	Display all dependent resources or resource dep...	no
linstor	View and manage LINSTOR storage resources	no
liqo	Install and manage Liqo on your clusters	no
log2rbac	Fine-tune your RBAC using log2rbac operator	no
match-name	Match names of pods and other API objects	no
mc	Run kubectl commands against multiple clusters ...	no
minio	Deploy and manage MinIO Operator and Tenant(s)	no
moco	Interact with MySQL operator MOCO.	no
modify-secret	modify secret with implicit base64 translations	no
mtail	Tail logs from multiple pods matching label sel...	no
multiforward	Port Forward to multiple Kubernetes Services	no
multinet	Shows pods' network-status of multi-net-spec	no
neat	Remove clutter from Kubernetes manifests to mak...	no
net-forward	Proxy to arbitrary TCP services on a cluster ne...	no
node-admin	List nodes and run privileged pod with chroot	no
node-restart	Restart cluster nodes sequentially and gracefully	no
node-shell	Spawn a root shell on a node via kubectl	no
np-viewer	Network Policies rules viewer	no
ns	Switch between Kubernetes namespaces	no
nsenter	Run shell command in Pod's namespace on the nod...	no
oidc-login	Log in to the OpenID Connect provider	no
open-svc	Open the Kubernetes URL(s) for the specified se...	no
openebs	View and debug OpenEBS storage resources	no
operator	Manage operators with Operator Lifecycle Manager	no
oulogin	Login to a cluster via OpenUnison	no

outdated	Finds outdated container images running in a cl...	no
passman	Store kubeconfig credentials in keychains or pa...	no
pexec	Execute process with privileges in a pod	no
pod-dive	Shows a pod's workload tree and info inside a node	no
pod-inspect	Get all of a pod's details at a glance	no
pod-lens	Show pod-related resources	no
pod-logs	Display a list of pods to get logs from	no
pod-shell	Display a list of pods to execute a shell in	no
podevents	Show events for pods	no
popeye	Scans your clusters for potential resource issues	no
preflight	Executes application preflight tests in a cluster	no
print-env	Build config files from k8s environments.	no
profefe	Gather and manage pprof profiles from running pods	no
promdump	Dumps the head and persistent blocks of Prometh...	no
prompt	Prompts for user confirmation when executing co...	no
prune-unused	Prune unused resources	no
psp-util	Manage Pod Security Policy(PSP) and the related...	no
pv-migrate	Migrate data across persistent volumes	no
pvmigrate	Migrates PVs between StorageClasses	no
rabbitmq	Manage RabbitMQ clusters	no
rbac-lookup	Reverse lookup for RBAC	no
rbac-tool	Plugin to analyze RBAC permissions and generate...	no
rbac-view	A tool to visualize your RBAC permissions.	no
realname-diff	Diffs live and local resources ignoring Kustomi...	no
reap	Delete unused Kubernetes resources.	no
relay	Drop-in "port-forward" replacement with UDP and...	no
reliably	Surfaces reliability issues in Kubernetes	no
rename-pvc	Rename a PersistentVolumeClaim (PVC)	no
resource-capacity	Provides an overview of resource requests, limi...	no
resource-snapshot	Prints a snapshot of nodes, pods and HPAs resou...	no
resource-versions	Print supported API resource versions	no
restart	Restarts a pod with the given name	no
rm-standalone-pods	Remove all pods without owner references	no
rolesum	Summarize RBAC roles for subjects	no

roll	Rolling restart of all persistent pods in a nam...	no
rook-ceph	Rook plugin for Ceph management	no
safe	Prompts before running edit commands	no
schemahero	Declarative database schema migrations via YAML	no
score	Kubernetes static code analysis.	no
secretdata	Viewing decoded Secret data with search flags	no
service-tree	Status for ingresses, services, and their backends	no
shovel	Gather diagnostics for .NET Core applications	no
sick-pods	Find and debug Pods that are "Not Ready"	no
skew	Find if your cluster/kubectl version is skewed	no
slice	Split a multi-YAML file into individual files.	no
snap	Delete half of the pods in a namespace or cluster	no
sniff	Start a remote packet capture on pods using tcp...	no
socks5-proxy	SOCKS5 proxy to Services or Pods in the cluster	no
sort-manifests	Sort manifest files in a proper order by Kind	no
split-yaml	Split YAML output into one file per resource.	no
spy	pod debugging tool for kubernetes clusters with...	no
sql	Query the cluster via pseudo-SQL	no
ssh-jump	Access nodes or services using SSH jump Pod	no
sshd	Run SSH server in a Pod	no
ssm-secret	Import/export secrets from/to AWS SSM param store	no
starboard	Toolkit for finding risks in kubernetes resources	no
status	Show status details of a given resource.	no
stern	Multi pod and container log tailing	no
strace	Capture strace logs from a running workload	no
sudo	Run Kubernetes commands impersonated as group s...	no
support-bundle	Creates support bundles for off-cluster analysis	no
switch-config	Switches between kubeconfig files	no
tail	Stream logs from multiple pods and containers u...	no
tap	Interactively proxy Kubernetes Services with ease	no
tmux-exec	An exec multiplexer using Tmux	no
topology	Explore region topology for nodes or pods	no
trace	Trace Kubernetes pods and nodes with system tools	no
tree	Show a tree of object hierarchies through owner...	no

tunnel	Reverse tunneling between cluster and your machine	no
unused-volumes	List unused PVCs	no
vela	Easily interact with KubeVela	no
view-allocations	List allocations per resources, nodes, pods.	no
view-cert	View certificate information stored in secrets	no
view-secret	Decode Kubernetes secrets	no
view-serviceaccount-kubeconfig	Show a kubeconfig setting to access the apiserv...	no
view-utilization	Shows cluster cpu and memory utilization	no
view-webhook	Visualize your webhook configurations	no
viewnode	Displays nodes with their pods and containers a...	no
virt	Control KubeVirt virtual machines using virtctl	no
volsync	Manage replication with the VolSync operator	no
vpa-recommendation	Compare VPA recommendations to actual resources...	no
warp	Sync and execute local files in Pod	no
whisper-secret	Create secrets with improved privacy	no
who-can	Shows who has RBAC permissions to access Kubern...	no
whoami	Show the subject that's currently authenticated...	no

2.3 - Installing and using plugins

Install the **ctx**, **ns**, **view-allocations** and **pod-logs** plugins:

```
root@kubemaster:~# kubectl krew install ctx ns view-allocations pod-logs
Updated the local copy of plugin index.
Installing plugin: ctx
Installed plugin: ctx
\
| Use this plugin:
|   kubectl ctx
| Documentation:
|   https://github.com/ahmetb/kubectx
| Caveats:
| \
```

```
| | If fzf is installed on your machine, you can interactively choose
| | between the entries using the arrow keys, or by fuzzy searching
| | as you type.
| | See https://github.com/ahmetb/kubectx for customization and details.
| /
/
WARNING: You installed plugin "ctx" from the krew-index plugin repository.
  These plugins are not audited for security by the Krew maintainers.
  Run them at your own risk.
Installing plugin: ns
Installed plugin: ns
\
| Use this plugin:
|   kubectl ns
| Documentation:
|   https://github.com/ahmetb/kubectx
| Caveats:
| \
| | If fzf is installed on your machine, you can interactively choose
| | between the entries using the arrow keys, or by fuzzy searching
| | as you type.
| /
/
WARNING: You installed plugin "ns" from the krew-index plugin repository.
  These plugins are not audited for security by the Krew maintainers.
  Run them at your own risk.
Installing plugin: view-allocations
Installed plugin: view-allocations
\
| Use this plugin:
|   kubectl view-allocations
| Documentation:
|   https://github.com/davidB/kubectl-view-allocations
/
```

```
WARNING: You installed plugin "view-allocations" from the krew-index plugin repository.
  These plugins are not audited for security by the Krew maintainers.
  Run them at your own risk.
Installing plugin: pod-logs
Installed plugin: pod-logs
\
| Use this plugin:
|   kubectl pod-logs
| Documentation:
|   https://github.com/danisla/kubefunc
/
WARNING: You installed plugin "pod-logs" from the krew-index plugin repository.
  These plugins are not audited for security by the Krew maintainers.
  Run them at your own risk.
```

The **ctx** plugin makes it easy to switch between **contexts**. A context is an element that groups access parameters under a name. There are three access parameters: cluster, namespace and user. The kubectl command uses the parameters of the current context to communicate with the cluster.

List the contexts in the cluster:

```
root@kubemaster:~# kubectl ctx
kubernetes-admin@kubernetes
```

The **ns** plugin makes it easy to switch between **namespaces**.

Namespaces :

- can be considered as virtual clusters,
- allow isolation and logical segmentation,
- allow users, roles and resources to be grouped together,
- are used with applications, customers, projects or teams.

List the namespaces in the cluster:

```

root@kubemaster:~# kubectl ns
default
kube-node-lease
kube-public
kube-system

```

The **view-allocations** plugin allows you to view resource allocations such as CPU, memory, storage etc :

```

root@kubemaster:~# kubectl view-allocations
Resource
cpu
├─ kubemaster.ittraining.loc
│   │   (13%) 1.6
│   │   ─── 12.0
│   │   ─── 10.4
│   └─ (28%) 1.1
│       ─── 4.0
│       ─── 2.9
│   └─ calico-node-688lw
│       ─── 250.0m
│       ─── ───
│   └─ coredns-6d4b75cb6d-dw4ph
│       ─── 100.0m
│       ─── ───
│   └─ coredns-6d4b75cb6d-ms2jm
│       ─── 100.0m
│       ─── ───
│   └─ etcd-kubemaster.ittraining.loc
│       ─── 100.0m
│       ─── ───
│   └─ kube-apiserver-kubemaster.ittraining.loc
│       ─── 250.0m
│       ─── ───
│   └─ kube-controller-manager-kubemaster.ittraining.loc
│       ─── 200.0m
│       ─── ───
│   └─ kube-scheduler-kubemaster.ittraining.loc
│       ─── 100.0m
│       ─── ───
├─ kubenode1.ittraining.loc
│   │   (6%) 250.0m
│   │   ─── 4.0
│   │   ─── 3.8
│   └─ calico-node-5mrjl
│       ─── 250.0m
│       ─── ───
├─ kubenode2.ittraining.loc
│   │   (6%) 250.0m
│   │   ─── 4.0
│   │   ─── 3.8
│   └─ calico-node-j25xd
│       ─── 250.0m
│       ─── ───
ephemeral-storage
├─ kubemaster.ittraining.loc
│   ─── ─── 50.9G
│   ─── ─── ───
├─ kubenode1.ittraining.loc
│   ─── ─── 17.0G
│   ─── ─── ───
├─ kubenode2.ittraining.loc
│   ─── ─── 17.0G
│   ─── ─── ───
memory
├─ kubemaster.ittraining.loc
│   │   (1%) 240.0Mi
│   │   (1%) 340.0Mi
│   │   31.0Gi
│   │   30.7Gi
│   └─ (2%) 240.0Mi
│       │   (2%) 340.0Mi
│       │   15.6Gi
│       │   15.2Gi
│       └─ 70.0Mi
│           ─── 170.0Mi
│           ─── ───
│       └─ 70.0Mi
│           ─── 170.0Mi
│           ─── ───
│       └─ 100.0Mi
│           ─── ───
├─ kubenode1.ittraining.loc
│   ─── ─── 7.7Gi
│   ─── ─── ───
├─ kubenode2.ittraining.loc
│   ─── ─── 7.7Gi
│   ─── ─── ───

```

pods	(5%)	17.0	(5%)	17.0	330.0	313.0
├ kubemaster.ittraining.loc	(7%)	8.0	(7%)	8.0	110.0	102.0
├ kubenode1.ittraining.loc	(4%)	4.0	(4%)	4.0	110.0	106.0
└ kubenode2.ittraining.loc	(5%)	5.0	(5%)	5.0	110.0	105.0

The **pod-logs** plugin provides you with a list of running pods and asks you to choose one:

```

root@kubemaster:~# kubectl pod-logs
1)  myapp-deployment-57c6cb89d9-dh4cb      default      Running
2)  myapp-deployment-57c6cb89d9-f69nk      default      Running
3)  myapp-deployment-57c6cb89d9-q7d4p      default      Running
4)  calico-kube-controllers-6766647d54-v4hrm kube-system  Running
5)  calico-node-5mrjl                      kube-system  Running
6)  calico-node-688lw                      kube-system  Running
7)  calico-node-j25xd                      kube-system  Running
8)  coredns-6d4b75cb6d-dw4ph              kube-system  Running
9)  coredns-6d4b75cb6d-ms2jm              kube-system  Running
10) etcd-kubemaster.ittraining.loc         kube-system  Running
11) kube-apiserver-kubemaster.ittraining.loc kube-system  Running
12) kube-controller-manager-kubemaster.ittraining.loc kube-system  Running
13) kube-proxy-bwctz                      kube-system  Running
14) kube-proxy-j89vg                      kube-system  Running
15) kube-proxy-jx76x                      kube-system  Running
16) kube-scheduler-kubemaster.ittraining.loc kube-system  Running
17) metrics-server-7cb867d5dc-g55k5      kube-system  Running
Select a Pod:

```

Select pod **17**. You will see the output of the logs command:

```

Select a Pod: 17
I0713 03:28:27.452157      1 serving.go:325] Generated self-signed cert (/tmp/apiserver.crt, /tmp/apiserver.key)
I0713 03:28:28.433807      1 secure_serving.go:197] Serving securely on [::]:4443
I0713 03:28:28.433876      1 requestheader_controller.go:169] Starting RequestHeaderAuthRequestController
I0713 03:28:28.433901      1 shared_informer.go:240] Waiting for caches to sync for

```

```
RequestHeaderAuthRequestController
I0713 03:28:28.433938      1 dynamic_serving_content.go:130] Starting serving-
cert::/tmp/apiserver.crt::/tmp/apiserver.key
I0713 03:28:28.433984      1 tlsconfig.go:240] Starting DynamicServingCertificateController
I0713 03:28:28.435681      1 configmap_cafile_content.go:202] Starting client-ca::kube-system::extension-
apiserver-authentication::client-ca-file
I0713 03:28:28.435702      1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-
system::extension-apiserver-authentication::client-ca-file
I0713 03:28:28.435727      1 configmap_cafile_content.go:202] Starting client-ca::kube-system::extension-
apiserver-authentication::requestheader-client-ca-file
I0713 03:28:28.435735      1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-
system::extension-apiserver-authentication::requestheader-client-ca-file
I0713 03:28:28.534094      1 shared_informer.go:247] Caches are synced for RequestHeaderAuthRequestController
I0713 03:28:28.535893      1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-
apiserver-authentication::requestheader-client-ca-file
I0713 03:28:28.535937      1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-
apiserver-authentication::client-ca-file
```

To list installed plugins, use the **list** command:

```
root@kubemaster:~# kubectl krew list
PLUGIN          VERSION
ctx             v0.9.4
krew            v0.4.3
ns              v0.9.4
pod-logs        v1.0.1
view-allocations v0.14.8
```

2.4 - Updating and deleting plugins

To update installed plugins, use the **upgrade** command:

```
root@kubemaster:~# kubectl krew upgrade
```

```
Updated the local copy of plugin index.  
Upgrading plugin: ctx  
Skipping plugin ctx, it is already on the newest version  
Upgrading plugin: krew  
Skipping plugin krew, it is already on the newest version  
Upgrading plugin: ns  
Skipping plugin ns, it is already on the newest version  
Upgrading plugin: pod-logs  
Skipping plugin pod-logs, it is already on the newest version  
Upgrading plugin: view-allocations  
Skipping plugin view-allocations, it is already on the newest version
```

To delete a plugin, use the **remove** command:

```
root@kubemaster:~# kubectl krew remove pod-logs  
Uninstalled plugin: pod-logs  
  
root@kubemaster:~# kubectl krew list  
PLUGIN          VERSION  
ctx              v0.9.4  
krew            v0.4.3  
ns              v0.9.4  
view-allocations v0.14.8
```

LAB #3 - Managing patches with the kustomize command

Start by installing the **tree** executable, which you'll use later to view the directory and file tree you are about to create:

```
root@kubemaster:~# apt install tree
```

Next, create the **kustomize** directory containing the **base** directory and move into it:

```
root@kubemaster:~# mkdir -p kustomize/base
root@kubemaster:~# cd kustomize/base/
root@kubemaster:~/kustomize/base#
```

Create the **deployment.yaml** manifest:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~/kustomize/base# vi deployment.yaml
root@kubemaster:~/kustomize/base# cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
```

```
- key: app
  operator: In
  values:
  - nginx
  topologyKey: "kubernetes.io/hostname"
containers:
- image: nginx:1.18.0
  imagePullPolicy: IfNotPresent
  name: nginx
```



Important - the content of this file creates a **deployment** of 1 **replica** of the **nginx** pod from the **nginx:1.18.0** image.

Next, create the **service.yaml** manifest:



To do: Copy the content from [here](#) and **paste it into your file.**

```
root@kubemaster:~/kustomize/base# vi service.yaml
root@kubemaster:~/kustomize/base# cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  type: ClusterIP
  ports:
```

```
- port: 80
  protocol: TCP
  targetPort: 80
selector:
  app: nginx
```



Important - the contents of this file create a **ClusterIP service** using the previous **deployment**. The ClusterIP service enables PODs offering the same service to be grouped together for easier communication.

Finally, create the **kustomization.yaml** manifest:



To do: Copy the content from [here](#) and **paste it into your file**.

```
root@kubemaster:~/kustomize/base# vi kustomization.yaml
root@kubemaster:~/kustomize/base# cat kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

images:
- name: nginx
  newTag: 1.19.1

resources:
- deployment.yaml
- service.yaml
```





Important - the contents of this file contain a **patch** for the **nginx** application created by the two previous files. Note the **newTag** in the **images** section. The **resources** section lists the manifests affected by the patch. Note that only the **deployment.yaml** manifest refers to an image. However, the **service.yaml** file is included here, as it will be needed later.

Use the **tree** command to view the **kustomize** directory:

```
root@kubemaster:~/kustomize/base# cd ..
root@kubemaster:~/kustomize# tree
.
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
```

1 directory, 3 files

Now run the **kustomize** command to create a **patch** for the files in the **base** directory:

```
root@kubemaster:~/kustomize# kubectl kustomize base
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
    name: nginx
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
```

```
  app: nginx
  type: ClusterIP
  ---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
    name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - nginx
              topologyKey: kubernetes.io/hostname
      containers:
        - image: nginx:1.19.1
          imagePullPolicy: IfNotPresent
          name: nginx
```



Important - note that the generated file contains the contents of the **two** files **deployment.yaml** and **service.yaml** separated by the **—** characters. The contents of the **service.yaml** file have not been modified, whereas the image has been changed from **image: nginx:1.18.0** to **image: nginx:1.19.1** in the contents of the **deployment.yaml** file. Note that the two original files have **not** been modified.

Now let's imagine you want to deploy two **different** environments of the same application, one for production and one for development. The **kustomize** command lets you do this using **overlays**.

Create the **kustomize/overlays/development** and **kustomize/overlays/production** directories:

```
root@kubemaster:~/kustomize# mkdir -p overlays/development
root@kubemaster:~/kustomize# mkdir overlays/production
```

View the **kustomize** directory tree:

```
root@kubemaster:~/kustomize# tree
.
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
├── overlays
├── development
└── production

4 directories, 3 files
```

Create the file **dev_kustomization.yaml**:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~/kustomize# vi overlays/development/kustomization.yaml
root@kubemaster:~/kustomize# cat overlays/development/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

bases:
- ../../base # <-----indicates where the main manifests are stored

nameSuffix: -development # <-----updates the service/deployment name

commonLabels:
environment: development # <-----adds an additional label

namespace: nginx-dev # <-----indicates the namespace name
```

Apply these changes :

```
root@kubemaster:~/kustomize# kubectl kustomize overlays/development/
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
    environment: development # <-----additional label
    name: nginx-development # <-----updated service name
    namespace: nginx-dev # <-----namespace name
spec:
  ports:
  - port: 80
```

```
  protocol: TCP
  targetPort: 80
  selector:
    app: nginx
    environment: development # <-----additional label
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
    environment: development
    name: nginx-development
    namespace: nginx-dev
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
      environment: development
  template:
    metadata:
      labels:
        app: nginx
        environment: development
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchExpressions:
                  - key: app
                    operator: In
```

```
    values:
      - nginx
    topologyKey: kubernetes.io/hostname
  containers:
  - image: nginx:1.19.1 # <-----uses the image specified in /kustomize/base/kustomization.yaml
    imagePullPolicy: IfNotPresent
    name: nginx
```

Now create the file **prod_kustomization.yaml**:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~/kustomize# vi overlays/production/kustomization.yaml
root@kubemaster:~/kustomize# cat overlays/production/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

bases:
- ../../base # <-----indicates where the main manifests are stored

nameSuffix: -production # <-----updates the service/deployment name

commonLabels:
  environment: production # <-----adds an additional label

namespace: nginx-prod # <-----indicates the namespace name

images:
- name: nginx
  newTag: 1.19.2 # <-----modifies the image specified in /kustomize/base/kustomization.yaml
```

Apply these modifications :

```
root@kubemaster:~/kustomize# kubectl kustomize overlays/production/
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
    environment: production # <-----additional label
    name: nginx-production # <-----updated service name
    namespace: nginx-prod # <-----namespace name
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
    environment: production # <-----additional label
  type: ClusterIP
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
    environment: production
    name: nginx-production
    namespace: nginx-prod
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
```

```
  environment: production
template:
  metadata:
    labels:
      app: nginx
      environment: production
  spec:
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
                - key: app
                  operator: In
                  values:
                    - nginx
            topologyKey: kubernetes.io/hostname
    containers:
      - image: nginx:1.19.2 # <-----uses the image specified in overlays/production/kustomization.yaml
        imagePullPolicy: IfNotPresent
        name: nginx
```

Now create the **nginx-prod** namespace:

```
root@kubemaster:~/kustomize# kubectl create ns nginx-prod
namespace/nginx-prod created
```

Install the **production** application:

```
root@kubemaster:~/kustomize# kubectl apply -k overlays/production/
service/nginx-production created
deployment.apps/nginx-production created
```

Check the result of the installation:

```
root@kubemaster:~/kustomize# kubectl get pods -n nginx-prod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-production-75d9486bb9-7xpr6  1/1    Running   0           45s

root@kubemaster:~/kustomize# kubectl get deployments -n nginx-prod
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-production    1/1     1             1           62s

root@kubemaster:~/kustomize# kubectl get services -n nginx-prod
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nginx-production    ClusterIP      10.97.147.125   <none>        80/TCP     79s
```

Remove the nginx-production deployment and service:

```
root@kubemaster:~/kustomize# kubectl delete deployments/nginx-production -n nginx-prod
deployment.apps "nginx-production" deleted

root@kubemaster:~/kustomize# kubectl get deployments -n nginx-prod
No resources found in nginx-prod namespace.

root@kubemaster:~/kustomize# kubectl get services -n nginx-prod
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nginx-production    ClusterIP      10.97.147.125   <none>        80/TCP     2m54s

root@kubemaster:~/kustomize# kubectl get pods -n nginx-prod
No resources found in nginx-prod namespace.

root@kubemaster:~/kustomize# kubectl delete services/nginx-production -n nginx-prod
service "nginx-production" deleted

root@kubemaster:~/kustomize# kubectl get services -n nginx-prod
No resources found in nginx-prod namespace.
```

Install the **development** application:

```
root@kubemaster:~/kustomize# kubectl create ns nginx-dev
namespace/nginx-dev created
root@kubemaster:~/kustomize# kubectl apply -k overlays/development/
service/nginx-development created
deployment.apps/nginx-development created
```

Check the result:

```
root@kubemaster:~/kustomize# kubectl get pods -n nginx-dev
NAME                                READY   STATUS    RESTARTS   AGE
nginx-development-5f8d7bdd88-fsnc6  1/1    Running   0           37s

root@kubemaster:~/kustomize# kubectl get deployments -n nginx-dev
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-development   1/1     1             1           58s

root@kubemaster:~/kustomize# kubectl get services -n nginx-dev
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nginx-development   ClusterIP   10.98.227.222   <none>        80/TCP     70s
```