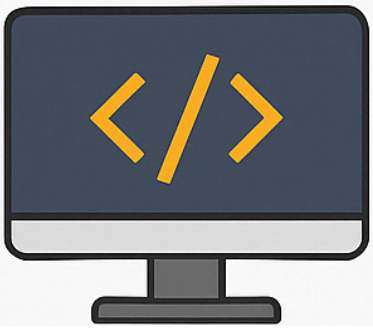
# CKAD Core Concepts – Important Flags Cheatsheet

## Master Kubernetes Core Concepts for CKAD Success!



# CKAD Core Concepts — Important Flags (Full Details)

| Flag | Used with | Description | Example |
|---|---|---|---|
| --image | kubectl run, create deployment, create job, create cronjob | Set the container image | --image=nginx:1.19 |
| --port | kubectl run, create deployment, expose | Define the container's port internally | --port=80 |
| --target-port | kubectl expose | The backend container port the service routes to | --target-port=8080 |
| --type | kubectl expose | Type of service (ClusterIP, NodePort, LoadBalancer) | --type=NodePort |
| --restart | kubectl run | Set Pod restart policy (Always, OnFailure, Never) | --restart=Never |
| --replicas | kubectl create deployment | Number of replicas for Deployment | --replicas=3 |
| --env | kubectl run, create job, create cronjob | Set environment variables inside container | --env="ENV=prod" |
| --limits | kubectl run | Set CPU and memory limits | --limits=cpu=500m,memory=128Mi |
| --requests | kubectl run | Set CPU and memory requests | --requests=cpu=200m,memory=64Mi |
| --schedule | kubectl create cronjob | Set schedule for CronJob in crontab format | --schedule="*/5 * * * *" |
| --command | kubectl run, create job, create cronjob | Treat arguments as a command instead of args | --command -- sleep 3600 |
| --labels | Any create or run | Attach labels to the object | --labels="app=nginx,env=prod" |
| --field-selector | kubectl get | Filter resources based on fields | --field-selector=status.phase=Running |
| --selector | kubectl get | Filter based on labels | --selector=env=prod |

| Flag | Used with | Description | Example |
|---|---|---|---|
| --output, -o | Any kubectl get, describe, create | Format output as yaml, json, wide, name | -o yaml |
| --dry-run=client | Any create, run, expose | Simulate and print resource without creating | --dry-run=client |
| --grace-period | kubectl delete | Time to wait before force killing the pod | --grace-period=0 |
| --force | kubectl delete | Force delete even if finalizers present | --force |
| --overrides | kubectl run | Pass raw JSON overrides for spec (advanced) | --overrides='{"spec": {"nodeName": "worker1"}}' |
| --record | kubectl set | Record the change cause in rollout history | --record |
| --stdin, --tty, -i, -t | kubectl exec | Attach input and allocate TTY | kubectl exec -it mypod -- sh |
| --namespace, -n | Any kubectl command | Operate in a specific namespace | -n mynamespace |

---

# Expanded Explanation:

- `--image`: Tells the pod/container which image to pull from registry.
- `--port` vs `--target-port`: External vs internal ports.
- `--restart`: Pod's restart behavior.
- `--replicas`: Set number of Deployment pods.
- `--env`: Define environment variables inside containers.
- `--limits` & `--requests`: Define resource usage boundaries.
- `--schedule`: CronJob specific schedule.
- `--command`: Run custom commands in containers.
- `--labels`: Metadata organization.
- `--field-selector` & `--selector`: Filtering during `kubectl get`.
- `--dry-run=client`: YAML generation without creation.
- `--grace-period` & `--force`: Instant resource deletion.
- `--overrides`: Advanced JSON spec injection.
- `--record`: Keep track of Deployment change history.
- `--stdin`, `--tty`, `-i`, `-t`: Attach interactive shells.
- `--namespace`: Operate under different namespaces.

---

# Common Full Command Examples

| Resource | Full Command |
|---|---|
| Pod | `kubectl run mypod --image=nginx --restart=Never --port=80 --env="MODE=dev" --labels="app=web" --dry-run=client -o yaml > mypod.yaml` |
| Deployment | `kubectl create deployment mydep --image=nginx:1.19 --replicas=2 --port=80 --dry-run=client -o yaml > deploy.yaml` |
| Service | `kubectl expose deployment mydep --port=80 --target-port=8080 --type=ClusterIP --name=mydep-svc --dry-run=client -o yaml > svc.yaml` |
| Job | `kubectl create job myjob --image=busybox --command -- sleep 300 --dry-run=client -o yaml > job.yaml` |
| CronJob | `kubectl create cronjob mycron --image=busybox --schedule="*/5 * * * *" --command -- echo "Hi" --dry-run=client -o yaml > cronjob.yaml` |
| | |

---

# Summary Table

| Flag | Pod | Deployment | Job | CronJob | Service | Namespace |
|---|---|---|---|---|---|---|
| --image | ✅ | ✅ | ✅ | ✅ | ❌ | ❌ |
| --port | ✅ | ✅ | ❌ | ❌ | ✅ | ❌ |
| --target-port | ❌ | ❌ | ❌ | ❌ | ✅ | ❌ |
| --restart | ✅ | ❌ | ❌ | ❌ | ❌ | ❌ |
| --replicas | ❌ | ✅ | ❌ | ❌ | ❌ | ❌ |
| --env | ✅ | ✅ | ✅ | ✅ | ❌ | ❌ |
| --schedule | ❌ | ❌ | ❌ | ✅ | ❌ | ❌ |
| --dry-run | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| --labels | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |
| --namespace | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |

---

# Special "20 Fast CKAD Exercises" practice set next? It will boost your hands-on speed Naturally!!!

**1.**Create a Pod named `busy-pod` using the image `busybox`, with restart policy set to `Never`, and expose port `8080`.

**2.**Create a Deployment named `nginx-deploy` with `nginx:1.21` image, 3 replicas, and expose port `80`.

**3.**Expose the Deployment `nginx-deploy` internally as a `ClusterIP` service on port `80`, target port `80`.

**4.**Create a Job named `echo-job` that runs a busybox container and executes `echo CKAD Practice`.

**5.**Create a CronJob named `echo-cron` scheduled every 5 minutes that echoes `Hello CKAD!`.

**6.**Create a Pod `limit-pod` using `nginx`, requesting `200m` CPU and `128Mi` memory, and setting limits to `500m` CPU and `256Mi` memory.

**7.**Create a ConfigMap `app-config` with keys `ENV=production` and `VERSION=1.0`.

**8.**Create a Secret `db-secret` with username `admin` and password `Passw0rd!`.

**9.**Patch the Deployment `nginx-deploy` to scale replicas from 3 to 5.

**10.**Set a new image `nginx:1.23` for the Deployment `nginx-deploy`.

**11.**Create a ServiceAccount named `custom-sa` and launch a pod that uses it.

**12.**Force delete the Pod `stuck-pod` immediately.

**13.**Create a PVC named `myclaim` requesting `500Mi` storage.

**14.**Create a Pod `mount-pvc-pod` using nginx, and mount the above PVC at `/usr/share/nginx/html`.

**15.**Create a NetworkPolicy `allow-dns` that only allows pods to make DNS (port 53 TCP/UDP) egress traffic.

**16.**Create a Pod `label-pod` with labels `tier=frontend` and `env=prod`.

**17.**List all pods running in namespace `kube-system` filtered by field `status.phase=Running`.

**18.**Create a Pod `override-node-pod` scheduled to a node labeled with `zone=us-east1-b`.

**19.**Exec inside the Pod `busy-pod` and run `hostname`.

**20.**Create a Deployment `httpd-deploy` using `httpd:2.4` image, with environment variable `MODE=testing`.