OpenShift DO180 Command Cheat Sheet

> This cheat sheet contains the most useful commands from the RedHat OpenShift DO180 course, organized by category for easy navigation in Obsidian.

Table of Contents

Authentication & Basic Setup

Login Commands

```
# Login with username and password
oc login https://api.ocp4.example.com:6443 -u developer -p developer

# Login with token (get from web console)
oc login --token=sha256-BW...rA8 --server=https://api.ocp4.example.com:6443

# Get console URL
oc whoami --show-console

# Show current user
oc whoami

# Show current context
```

```
oc whoami --show-context

# Show current server
oc whoami --show-server
```

Basic CLI Setup

```
# Get help
oc help
oc <command> --help

# Show version
oc version

# Show cluster info
oc cluster-info

# Show supported API versions
oc api-versions

# Show cluster operators
oc get clusteroperator
```

Project Management

Project Commands

```
# Create new project
oc new-project myapp

# List projects
oc projects

# Switch to project
oc project myproject

# Get current project
oc project

# Delete project
oc delete project myproject

# Show project status
oc status
```

```
# Show project status with suggestions
oc status --suggest
```

Basic Information & Inspection

Resource Listing

```
# List all resource types
oc api-resources

# List namespaced resources only
oc api-resources --namespaced=true

# List resources by API group
oc api-resources --api-group=apps

# Get all resources in current project
oc get all

# Get all resources with wide output
oc get all -o wide

# Get resources across all namespaces
oc get pods --all-namespaces
oc get pods -A
```

Resource Information

```
# Describe resource
oc describe pod podname
oc describe deployment deploymentname
oc describe service servicename

# Get resource in YAML format
oc get pod podname -o yaml

# Get resource in JSON format
oc get pod podname -o json

# Explain resource fields
oc explain pod
oc explain deployment.spec
oc explain pod.spec.containers.resources
```

```
# Show resource with labels
oc get pods --show-labels
```

Custom Output Formatting

```
# Custom columns output
oc get pods -o custom-
columns=NAME:.metadata.name,STATUS:.status.phase,IP:.status.podIP

# JSONPath queries
oc get pods -o jsonpath='{.items[0].metadata.name}'
oc get pods -o jsonpath='{range .items[*]}{.metadata.name}{" "}
{.status.podIP}{"\n"}{end}'

# Using jq for JSON parsing
oc get pod podname -o json | jq '.status.containerStatuses[].name'
```

Pod Management

Creating Pods

```
# Run simple pod
oc run mypod --image=registry.access.redhat.com/ubi9/ubi

# Run interactive pod
oc run shell -it --image=registry.access.redhat.com/ubi9/ubi -- /bin/bash

# Run pod with environment variables
oc run mypod --image=nginx --env="VAR1=value1" --env="VAR2=value2"

# Run pod and delete when finished
oc run test-pod -it --rm --image=registry.access.redhat.com/ubi9/ubi --
restart=Never -- /bin/bash

# Run pod with specific restart policy
oc run mypod --image=nginx --restart=Never
```

Pod Operations

```
# List pods
oc get pods
oc get pods -o wide

# Get pod with specific selector
```

```
oc get pods -l app=myapp
oc get pods --selector app=myapp

# Watch pod status
oc get pods -w

# Get pod logs
oc logs podname

# Get logs with tail
oc logs podname --tail=10

# Follow logs
oc logs -f podname

# Get logs from specific container
oc logs podname -c containername

# Execute command in pod
oc exec podname -- command
oc exec -it podname -- /bin/bash

# Attach to running pod
oc attach podname -it

# Copy files to/from pods
oc cp file.txt podname:/path/to/destination
oc cp podname:/path/to/file ./local-file

# Port forward
oc port-forward pod/podname 8080:80
```

Deployment Management

Creating Deployments

```
# Create deployment
oc create deployment myapp --image=nginx

# Create deployment with port
oc create deployment myapp --image=nginx --port=80

# Create deployment with replicas
oc create deployment myapp --image=nginx --replicas=3
```

```
# Create deployment from file
oc create -f deployment.yaml
```

Managing Deployments

```
# List deployments
oc get deployment
oc get deploy

# Get deployment details
oc describe deployment myapp

# Update deployment image
oc set image deployment/myapp containername=newimage:tag

# Set environment variables
oc set env deployment/myapp VAR1=value1 VAR2=value2

# Set environment from secret
oc set env deployment/myapp --from=secret/mysecret

# Set environment from configmap
oc set env deployment/myapp --from=configmap/myconfig

# Add resource limits
oc set resources deployment/myapp --limits=cpu=500m,memory=512Mi --
requests=cpu=200m,memory=256Mi

# Add probes
oc set probe deployment/myapp --readiness --get-url=http://:8080/health
oc set probe deployment/myapp --liveness --get-url=http://:8080/health

# Set image triggers
oc set triggers deployment/myapp --from-image=imagestreamname:latest
```

Service & Networking

Service Management

```
# Expose deployment as service
oc expose deployment myapp
oc expose deployment myapp --port=80 --target-port=8080

# Create service with specific type
oc create service clusterip myapp --tcp=80:8080
```

```
# Get services
oc get services
oc get svc

# Get service endpoints
oc get endpoints

# Describe service
oc describe service myapp
```

Route Management

```
# Create route from service
oc expose service myapp

# Create route with custom hostname
oc expose service myapp --hostname=myapp.apps.example.com

# Create route with path
oc expose service myapp --path=/api

# Get routes
oc get routes

# Describe route
oc describe route myapp
```

Ingress Management

```
# Create ingress
oc create ingress myingress --rule="host.example.com/path=service:port"

# Get ingress
oc get ingress

# Describe ingress
oc describe ingress myingress
```

Network Testing

```
# Test connectivity using nc (netcat)
oc run test-pod -it --rm --
image=registry.ocp4.example.com:8443/openshift4/network-tools-rhel8 --
restart=Never -- nc -z service-name port
```

```
# Test DNS resolution
oc run test-pod -it --rm --image=registry.access.redhat.com/ubi9/ubi --
restart=Never -- nslookup service-name

# Test HTTP connectivity
oc run test-pod -it --rm --image=registry.access.redhat.com/ubi9/ubi --
restart=Never -- curl service-name:port
```

Storage Management

Persistent Volume Claims

```
# Create PVC
oc create -f pvc.yaml

# List PVCs
oc get pvc

# Describe PVC
oc describe pvc mypvc

# Get PV
oc get pv
```

ConfigMaps

```
# Create configmap from literal values
oc create configmap myconfig --from-literal=key1=value1 --from-
literal=key2=value2

# Create configmap from file
oc create configmap myconfig --from-file=config.txt

# Create configmap from directory
oc create configmap myconfig --from-file=config-dir/

# Get configmaps
oc get configmap
oc get cm

# Describe configmap
oc describe configmap myconfig
```

Secrets

```
# Create secret from literal values
oc create secret generic mysecret --from-literal=username=admin --from-
literal=password=secret

# Create secret from files
oc create secret generic mysecret --from-file=username.txt --from-
file=password.txt

# Create TLS secret
oc create secret tls tls-secret --cert=tls.crt --key=tls.key

# Get secrets
oc get secrets

# Describe secret (values are hidden)
oc describe secret mysecret

# Get secret in YAML (base64 encoded values)
oc get secret mysecret -o yaml
```

Configuration Management

Jobs and CronJobs

```
# Create job
oc create job myjob --image=busybox -- /bin/sh -c "echo hello"

# Create job from cronjob
oc create job manual-job --from=cronjob/mycronjob

# Create cronjob
oc create cronjob mycronjob --image=busybox --schedule="0 1 * * *" --
/bin/sh -c "echo hello"

# Get jobs
oc get jobs

# Get cronjobs
oc get cronjobs
oc get cj

# Get job pods
oc get pods --selector=job-name=myjob
```

Image Streams

```
# Create image stream
oc create imagestream myapp

# Import image to image stream
oc import-image myapp --from=registry.access.redhat.com/ubi9/httpd-24 --
confirm

# Get image streams
oc get imagestream
oc get is

# Describe image stream
oc describe is myapp

# Get image stream tags
oc get imagestreamtag
oc get istag

# Tag image
oc tag source-image:tag target-is:tag
```

Debugging & Troubleshooting

Debug Commands

```
# Debug node (requires admin privileges)
oc debug node/nodename

# Debug pod
oc debug pod/podname

# Debug deployment
oc debug deployment/myapp

# Run debug pod with specific image
oc debug pod/podname --image=registry.access.redhat.com/ubi9/ubi
```

Node Operations (Admin Required)

```
# List nodes
oc get nodes

# Describe node
oc describe node nodename
```

```
# Get node logs
oc adm node-logs nodename

# Get node resources
oc adm top nodes
```

Troubleshooting Commands

```
# Get events
oc get events

# Get events sorted by time
oc get events --sort-by='.lastTimestamp'

# Describe events for specific resource
oc describe events --field-selector involvedObject.name=podname

# Check resource usage
oc adm top pods
oc adm top nodes

# Get logs from all containers in pod
oc logs podname --all-containers=true

# Get previous container logs (after restart)
oc logs podname --previous
```

Image Management

Image Information

```
# Get image info
oc image info registry.access.redhat.com/ubi9/httpd-24:latest

# Get image info for specific architecture
oc image info registry.access.redhat.com/ubi9/httpd-24:latest --filter-by-
os=linux/amd64

# Mirror images
oc image mirror source-registry/image:tag dest-registry/image:tag

# Extract files from image
```

```
oc image extract registry.access.redhat.com/ubi9/httpd-24:latest --path
/etc/httpd:./config
```

Application Updates & Rollbacks

Rollout Management

```
# Check rollout status
oc rollout status deployment/myapp

# Pause rollout
oc rollout pause deployment/myapp

# Resume rollout
oc rollout resume deployment/myapp

# Restart deployment (rollout)
oc rollout restart deployment/myapp

# View rollout history
oc rollout history deployment/myapp

# Rollback to previous version
oc rollout undo deployment/myapp

# Rollback to specific revision
oc rollout undo deployment/myapp --to-revision=2
```

Update Strategies

```
# Set deployment strategy (in YAML or via edit)
# strategy:
#   type: RollingUpdate
#   rollingUpdate:
#     maxSurge: 25%
#     maxUnavailable: 25%

# Set recreate strategy
# strategy:
#   type: Recreate
```

Scaling & Performance

Scaling Applications

```
# Scale deployment
oc scale deployment/myapp --replicas=5

# Scale replicaset
oc scale rs/myapp-12345 --replicas=3

# Autoscale deployment
oc autoscale deployment/myapp --min=2 --max=10 --cpu-percent=70

# Get horizontal pod autoscalers
oc get hpa

# Describe hpa
oc describe hpa myapp
```

Resource Management

```
# Set resource requests and limits
oc set resources deployment/myapp --requests=cpu=100m,memory=128Mi --
limits=cpu=500m,memory=512Mi

# Create limit range
oc create -f limitrange.yaml

# Create resource quota
oc create quota myquota --
hard=pods=10,requests.cpu=1,requests.memory=1Gi,limits.cpu=2,limits.memory=2
Gi

# Get quotas
oc get quota

# Describe quota
oc describe quota myquota
```

Monitoring & Observability

Health Checks

```
# Add readiness probe
oc set probe deployment/myapp --readiness --get-url=http://:8080/health --
initial-delay-seconds=30
```

```
# Add liveness probe
oc set probe deployment/myapp --liveness --get-url=http://:8080/health --
initial-delay-seconds=30

# Add probe with exec command
oc set probe deployment/myapp --readiness --open-tcp=8080 --initial-delay-
seconds=10

# Remove probe
oc set probe deployment/myapp --remove --readiness
```

Logging

```
# Get application logs
oc logs deployment/myapp

# Get logs from all pods in deployment
oc logs -l app=myapp

# Follow logs from multiple pods
oc logs -f -l app=myapp

# Get logs with timestamps
oc logs podname --timestamps

# Get logs since specific time
oc logs podname --since=1h
oc logs podname --since-time=2023-01-01T10:00:00Z
```

Resource Cleanup

Deletion Commands

```
# Delete specific resource
oc delete pod podname
oc delete deployment myapp
oc delete service myapp
oc delete route myapp

# Delete by label
oc delete pods -l app=myapp
oc delete all -l app=myapp

# Delete from file
```

```
oc delete -f deployment.yaml

# Force delete pod
oc delete pod podname --force --grace-period=0

# Delete all resources of a type
oc delete pods --all
oc delete deployments --all

# Cleanup completed jobs
oc delete jobs --field-selector=status.successful=1
```

Advanced Commands

Templating and Processing

```
# Process template
oc process -f template.yaml -p PARAM1=value1 | oc create -f -

# Create from template
oc new-app --template=templatename -p PARAM1=value1

# Export resources
oc get deployment myapp -o yaml --export > myapp-deployment.yaml
```

Annotation and Labels

```
# Add labels
oc label pod podname key=value
oc label deployment myapp environment=production

# Add annotations
oc annotate pod podname description="This is my pod"
oc annotate route myapp router.openshift.io/cookie_name=myapp

# Remove label
oc label pod podname key-

# Remove annotation
oc annotate pod podname description-
```

Patching Resources

```
# Patch deployment
oc patch deployment myapp -p '{"spec":{"replicas":3}}'

# Patch with merge strategy
oc patch configmap myconfig --type merge -p '{"data":{"key":"new-value"}}'

# Strategic merge patch
oc patch deployment myapp --type strategic -p '{"spec":{"template":{"spec":
{"containers":[{"name":"myapp","image":"nginx:latest"}]}}}}'
```

Administrative Commands

```
# Get cluster version
oc get clusterversion

# Get cluster operators status
oc get co

# Get machine config pools (admin required)
oc get mcp

# Get certificate signing requests
oc get csr

# Approve CSR (admin required)
oc adm certificate approve csr-name

# Cordon/uncordon nodes (admin required)
oc adm cordon nodename
oc adm uncordon nodename

# Drain node (admin required)
oc adm drain nodename --ignore-daemonsets --delete-emptydir-data
```

Quick Reference Tables

Common Resource Short Names

Resource	Short Name
pods	po
services	svc
deployments	deploy

Resource	Short Name
replicasets	rs
configmaps	cm
secrets	(none)
persistentvolumes	pv
persistentvolumeclaims	pvc
namespaces	ns
nodes	no
cronjobs	cj

Output Formats

Format	Flag
YAML	-o yaml
JSON	-o json
Wide	-o wide
Custom columns	-o custom-columns=...
JSONPath	-o jsonpath=...
Name only	-o name

Selector Examples

```
# Equality-based selectors
oc get pods -l app=myapp
oc get pods -l 'app in (myapp,apping)'
oc get pods -l 'app!=myapp'

# Set-based selectors
oc get pods -l 'environment in (production,qa)'
oc get pods -l 'tier notin (frontend,backend)'

# Multiple selectors
oc get pods -l app=myapp,environment=production
```

Tips and Best Practices

General Tips

- Use `--dry-run=client -o yaml` to generate YAML templates
- Use `--help` with any command to get detailed information
- Use tab completion for command and resource names
- Use `oc explain` to understand resource structure before creating YAML
- Always use specific image tags in production (avoid `latest`)

Performance Tips

- Use `-o wide` for more information without full describe
- Use `--selector` instead of `grep` for filtering
- Use `--field-selector` for server-side filtering
- Use `--chunk-size` for large list operations

Safety Tips

- Always backup important resources before making changes
- Test commands with `--dry-run` first
- Use specific selectors instead of deleting all resources
- Verify resource names before deletion operations

This cheat sheet covers the most commonly used OpenShift commands from the DO180 course. Keep it handy as a quick reference while working with OpenShift clusters!