OpenShift Client (oc)

Usage:

```
oc COMMAND_TYPE
```

Where,

COMMAND_TYPE	BASIC, BUILD/DEPLOY, MANAGEMENT, TROUBLESHOOTING, SETTINGS, ADVANCED, OTHERS.
BASIC	login, new-project, new-app, status, project, projects, explain
BUILD/DEPLOY	new-build, start-build, cancel-build, rollout, rollback, cancel-build, import-image, tag
MANAGEMENT	create, apply, get, describe, edit, set, label, annotate, expose, delete, scale, autoscale, secrets, serviceaccounts
TROUBLESHOOTING	logs, rsh, rsync, port-forward, debug, exec, proxy, attach, run, cp, wait
SETTINGS	logout, config, whoami, completion
ADVANCED	adm, replace, patch, process, extract, observe, policy, auth, image, registry, idle, api-versions, api-resources, cluster-info, diff, kustomize
OTHERS	ex, help, plugin, version

For a description of all the subcommand listed above, type `oc --help`

Best method for getting help: `oc COMMAND --help`

Configuration Files:

```
~/.kube/config
```

Most often used common syntax:

```
oc COMMAND RESOURCE [NAME] [-n PROJECT]
                   ↥
          space or /
```

Where,

COMMAND	get, delete, describe, edit, rsh, logs, port-forward, etc.
RESOURCE	node, project, pod, services(svc), route, persistentvolume(pv), persistentvolumeclaim(pvc), secret, configmap(cm), deploymentconfig(dc), deployment(deploy), replicationcontroller(rc), replicaset(rs), etc.

Basic Operations:

Login Operations:

```
oc login [-u USER] [-p PASSWORD] API_URL
oc whoami [--show-console] [--show-token]          # --show-token = -t
oc logout
```

```
oc login -u devuser https://api.mycluster.example.com:6443
```

Project Operations:

```
oc new-project PROJECT                        # Create a new project
oc project [PROJECT]                          # List/Change current project
oc projects OR oc get project                 # list all projects
oc delete project PROJECT                     # Delete project PROJECT
```

Creating Application Resources:

```
oc create -f FILE                             # create resource from YAML/JSON file
```

Create App using existing Image/ImageStream

```
oc new-app [--as-deployment-config] \              # create app using IMAGE
       {[--docker-image] IMAGE} [--name NAME] \
       [-e KEY=VALUE]...
oc new-app [--as-deployment-config] \              # create app using IMAGE_STREAM
       IMAGE_STREAM
```

Create App using Source-To-Image (S2I/STI) or Dockerfile

```
oc new-app [--as-deployment-config] \              # create app using SOURCE_CODE
       [-i BUILDER_IS] URL [--name NAME] \
       [--strategy source|docker] [-e KEY=VALUE]...
oc new-app [--as-deployment-config] \              # create app using SOURCE_CODE
       BUILDER_IS~URL [--name NAME] \              # force S2I to use BUILDER_IS
       [--strategy source|docker] [-e KEY=VALUE]...
```

```
oc new-app -i php https://github.com/user/myapp#branch --context-dir mydb
oc new-app -i php:7.1 https://github.com/user/yourapp
oc new-app php:7.1~https://github.com/user/superapp
```

Create App using Template

```
oc new-app --template TEMPLATE \                   # create app using TEMPLATE
       [-p PARAM=VALUE]... [--param-file PARAM_FILE] \
       [-e KEY=VALUE]
```

API Resources (Resource Types) Operations:

```
oc api-resources                    # List all resource types. Any namespaced resource
                                    command, accepts -n PROJECT option

oc explain RESOURCE[.FIELD]{[.FIELD]}...          # Learn resource structure
oc get RESOURCE [NAME] [-n PROJECT] [-o (json|yaml)|wide]       # Show resource info
oc describe RESOURCE NAME                         # Show more info

oc edit RESOURCE NAME [-o json]                   # Edit resource definition
oc patch ....                                     # Refer to "oc help patch"
oc set env|probe|volumes|sa|triggers|...          # refer to "oc help set RESOURCE"

oc delete RESOURCE NAME
oc delete all --all                               # delete all resource from project
oc delete all -l LABEL                            # delete all resources having LABEL
```

Scaling Pods:

```
oc scale --replicas=VALUE dc|rc NAME          # only scale rc if there's no dc !!!!
oc autoscale dc|deployment NAME \             # only works if METRICS available
       --min VALUE --max VALUE \
       --cpu-percent VALUE
oc get hpa                                    # list HorizontalPodAutoscaler
```

Image Operation:

```
oc import-image NAME [--confirm] --from IMAGE_URI [--insecure]
```

Troubleshooting:

```
oc logs bc|dc|build|pod NAME [-f]
oc get events
oc rsh POD_NAME [CMD]
oc cp FROM TO
```

where,

```
       FROM = TO = [POD_NAME:]PATH
```

```
oc cp ./file1 mypod-1-ab123:/mnt/testing/fileX
oc cp mypod-1-12345:/index.html /tmp/backup.html
```

Creating Resources:

```
oc expose dc|rc|pod|svc NAME        # expose dc/rc/pod gets svc, expose svc gets a route
oc create secret generic NAME {--from-literal KEY=VALUE}...      # oc create secret -h
oc create cm NAME {--from-literal KEY=VALUE}...                  # oc create cm -h
```

```
oc expose dc/myapp
oc expose svc/myapp --name myroute --hostname myroute.apps.example.com
```