Red Hat OpenShift Service on AWS 4

CLI tools

Learning how to use the command-line tools for Red Hat OpenShift Service on AWS

Last Updated: 2026-04-01

Red Hat OpenShift Service on AWS 4 CLI tools

Learning how to use the command-line tools for Red Hat OpenShift Service on AWS

Legal Notice

Copyright © Red Hat.

Except as otherwise noted below, the text of and illustrations in this documentation are licensed by Red Hat under the Creative Commons Attribution–Share Alike 3.0 Unported license . If you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, the Red Hat logo, JBoss, Hibernate, and RHCE are trademarks or registered trademarks of Red Hat, LLC. or its subsidiaries in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS is a trademark or registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and other countries.

The OpenStack ® Word Mark and OpenStack logo are trademarks or registered trademarks of the Linux Foundation, used under license.

All other trademarks are the property of their respective owners.

Abstract

This document provides information about installing, configuring, and using the command-line tools in Red Hat OpenShift Service on AWS. It also contains a reference of CLI commands and examples of how to use them.

Table of Contents

CHAPTER 1. RED HAT OPENSHIFT SERVICE ON AWS CLI TOOLS OVERVIEW 12
1.1. LIST OF CLI TOOLS 12

CHAPTER 2. OPENSHIFT CLI (OC) 13
2.1. GETTING STARTED WITH THE OPENSHIFT CLI 13
2.1.1. About the OpenShift CLI 13
2.1.2. Installing the OpenShift CLI 13
2.1.3. Installing the OpenShift CLI on Linux 13
2.1.4. Installing the OpenShift CLI on Windows 14
2.1.5. Installing the OpenShift CLI on macOS 14
2.1.5.1. Installing the OpenShift CLI by using the web console 15
2.1.5.1.1. Installing the OpenShift CLI on Linux using the web console 15
2.1.5.1.2. Installing the OpenShift CLI on Windows using the web console 16
2.1.5.1.3. Installing the OpenShift CLI on macOS using the web console 17
2.1.5.2. Installing the OpenShift CLI by using an RPM 18
2.1.5.3. Installing the OpenShift CLI by using Homebrew 19
2.1.6. Logging in to the OpenShift CLI 19
2.1.7. Logging in to the OpenShift CLI using a web browser 21
2.1.8. Using the OpenShift CLI 22
2.1.8.1. Creating a project 22
2.1.8.2. Creating a new app 22
2.1.8.3. Viewing pods 23
2.1.8.4. Viewing pod logs 23
2.1.8.5. Viewing the current project 23
2.1.8.6. Viewing the status for the current project 23
2.1.8.7. Listing supported API resources 24
2.1.9. Getting help 24
2.1.10. Logging out of the OpenShift CLI 25
2.2. CONFIGURING THE OPENSHIFT CLI 26
2.2.1. Enabling tab completion 26
2.2.1.1. Enabling tab completion for Bash 26
2.2.1.2. Enabling tab completion for Zsh 26
2.2.2. Accessing kubeconfig by using the oc CLI 27
2.3. USAGE OF OC AND KUBECTL COMMANDS 27
2.3.1. The oc binary 28
2.3.2. The kubectl binary 29
2.4. MANAGING CLI PROFILES 29
2.4.1. About switches between CLI profiles 29
2.4.2. Manual configuration of CLI profiles 31
2.4.3. Load and merge rules 33
2.5. EXTENDING THE OPENSHIFT CLI WITH PLUGINS 34
2.5.1. Writing CLI plugins 35
2.5.2. Installing and using CLI plugins 35
2.6. OPENSHIFT CLI DEVELOPER COMMAND REFERENCE 36
2.6.1. OpenShift CLI (oc) developer commands 36
2.6.1.1. oc annotate 36
2.6.1.2. oc api-resources 37
2.6.1.3. oc api-versions 37
2.6.1.4. oc apply 38
2.6.1.5. oc apply edit-last-applied 38
2.6.1.6. oc apply set-last-applied 38

2.6.1.7. oc apply view-last-applied 39
2.6.1.8. oc attach 39
2.6.1.9. oc auth can-i 39
2.6.1.10. oc auth reconcile 40
2.6.1.11. oc auth whoami 40
2.6.1.12. oc autoscale 40
2.6.1.13. oc cancel-build 41
2.6.1.14. oc cluster-info 41
2.6.1.15. oc cluster-info dump 41
2.6.1.16. oc completion 41
2.6.1.17. oc config current-context 42
2.6.1.18. oc config delete-cluster 43
2.6.1.19. oc config delete-context 43
2.6.1.20. oc config delete-user 43
2.6.1.21. oc config get-clusters 43
2.6.1.22. oc config get-contexts 43
2.6.1.23. oc config get-users 44
2.6.1.24. oc config new-admin-kubeconfig 44
2.6.1.25. oc config new-kubelet-bootstrap-kubeconfig 44
2.6.1.26. oc config refresh-ca-bundle 44
2.6.1.27. oc config rename-context 44
2.6.1.28. oc config set 45
2.6.1.29. oc config set-cluster 45
2.6.1.30. oc config set-context 45
2.6.1.31. oc config set-credentials 45
2.6.1.32. oc config unset 46
2.6.1.33. oc config use-context 46
2.6.1.34. oc config view 47
2.6.1.35. oc cp 47
2.6.1.36. oc create 47
2.6.1.37. oc create build 48
2.6.1.38. oc create clusterresourcequota 48
2.6.1.39. oc create clusterrole 48
2.6.1.40. oc create clusterrolebinding 49
2.6.1.41. oc create configmap 49
2.6.1.42. oc create cronjob 49
2.6.1.43. oc create deployment 49
2.6.1.44. oc create deploymentconfig 50
2.6.1.45. oc create identity 50
2.6.1.46. oc create imagestream 50
2.6.1.47. oc create imagestreamtag 50
2.6.1.48. oc create ingress 51
2.6.1.49. oc create job 51
2.6.1.50. oc create namespace 52
2.6.1.51. oc create poddisruptionbudget 52
2.6.1.52. oc create priorityclass 52
2.6.1.53. oc create quota 52
2.6.1.54. oc create role 53
2.6.1.55. oc create rolebinding 53
2.6.1.56. oc create route edge 53
2.6.1.57. oc create route passthrough 53
2.6.1.58. oc create route reencrypt 54
2.6.1.59. oc create secret docker-registry 54

2.6.1.60. oc create secret generic 54
2.6.1.61. oc create secret tls 55
2.6.1.62. oc create service clusterip 55
2.6.1.63. oc create service externalname 55
2.6.1.64. oc create service loadbalancer 55
2.6.1.65. oc create service nodeport 55
2.6.1.66. oc create serviceaccount 56
2.6.1.67. oc create token 56
2.6.1.68. oc create user 56
2.6.1.69. oc create useridentitymapping 56
2.6.1.70. oc debug 57
2.6.1.71. oc delete 57
2.6.1.72. oc describe 58
2.6.1.73. oc diff 58
2.6.1.74. oc edit 59
2.6.1.75. oc events 59
2.6.1.76. oc exec 59
2.6.1.77. oc explain 60
2.6.1.78. oc expose 60
2.6.1.79. oc extract 61
2.6.1.80. oc get 61
2.6.1.81. oc get-token 62
2.6.1.82. oc idle 62
2.6.1.83. oc image append 63
2.6.1.84. oc image extract 63
2.6.1.85. oc image info 64
2.6.1.86. oc image mirror 65
2.6.1.87. oc import-image 66
2.6.1.88. oc kustomize 66
2.6.1.89. oc label 66
2.6.1.90. oc login 67
2.6.1.91. oc logout 67
2.6.1.92. oc logs 68
2.6.1.93. oc new-app 68
2.6.1.94. oc new-build 69
2.6.1.95. oc new-project 70
2.6.1.96. oc observe 70
2.6.1.97. oc patch 70
2.6.1.98. oc plugin 71
2.6.1.99. oc plugin list 71
2.6.1.100. oc policy add-role-to-user 71
2.6.1.101. oc policy scc-review 71
2.6.1.102. oc policy scc-subject-review 72
2.6.1.103. oc port-forward 72
2.6.1.104. oc process 73
2.6.1.105. oc project 73
2.6.1.106. oc projects 73
2.6.1.107. oc proxy 74
2.6.1.108. oc registry login 74
2.6.1.109. oc replace 74
2.6.1.110. oc rollback 75
2.6.1.111. oc rollout 75
2.6.1.112. oc rollout cancel 75

2.6.1.113. oc rollout history 76
2.6.1.114. oc rollout latest 76
2.6.1.115. oc rollout pause 76
2.6.1.116. oc rollout restart 76
2.6.1.117. oc rollout resume 77
2.6.1.118. oc rollout retry 77
2.6.1.119. oc rollout status 77
2.6.1.120. oc rollout undo 77
2.6.1.121. oc rsh 77
2.6.1.122. oc rsync 78
2.6.1.123. oc run 78
2.6.1.124. oc scale 79
2.6.1.125. oc secrets link 79
2.6.1.126. oc secrets unlink 79
2.6.1.127. oc set build-hook 80
2.6.1.128. oc set build-secret 80
2.6.1.129. oc set data 80
2.6.1.130. oc set deployment-hook 80
2.6.1.131. oc set env 81
2.6.1.132. oc set image 81
2.6.1.133. oc set image-lookup 82
2.6.1.134. oc set probe 82
2.6.1.135. oc set resources 83
2.6.1.136. oc set route-backends 83
2.6.1.137. oc set selector 84
2.6.1.138. oc set serviceaccount 84
2.6.1.139. oc set subject 84
2.6.1.140. oc set triggers 84
2.6.1.141. oc set volumes 85
2.6.1.142. oc start-build 86
2.6.1.143. oc status 86
2.6.1.144. oc tag 86
2.6.1.145. oc version 87
2.6.1.146. oc wait 87
2.6.1.147. oc whoami 88
2.7. OPENSHIFT CLI ADMINISTRATOR COMMAND REFERENCE 88
2.7.1. OpenShift CLI (oc) administrator commands 88
2.7.1.1. oc adm build-chain 88
2.7.1.2. oc adm catalog mirror 88
2.7.1.3. oc adm certificate approve 89
2.7.1.4. oc adm certificate deny 89
2.7.1.5. oc adm copy-to-node 89
2.7.1.6. oc adm cordon 89
2.7.1.7. oc adm create-bootstrap-project-template 90
2.7.1.8. oc adm create-error-template 90
2.7.1.9. oc adm create-login-template 90
2.7.1.10. oc adm create-provider-selection-template 90
2.7.1.11. oc adm drain 90
2.7.1.12. oc adm groups add-users 91
2.7.1.13. oc adm groups new 91
2.7.1.14. oc adm groups prune 91
2.7.1.15. oc adm groups remove-users 91
2.7.1.16. oc adm groups sync 92

2.7.1.17. oc adm inspect 92
2.7.1.18. oc adm migrate icsp 92
2.7.1.19. oc adm migrate template-instances 93
2.7.1.20. oc adm must-gather 93
2.7.1.21. oc adm new-project 93
2.7.1.22. oc adm node-image create 94
2.7.1.23. oc adm node-image monitor 94
2.7.1.24. oc adm node-logs 94
2.7.1.25. oc adm ocp-certificates monitor-certificates 95
2.7.1.26. oc adm ocp-certificates regenerate-leaf 95
2.7.1.27. oc adm ocp-certificates regenerate-machine-config-server-serving-cert 95
2.7.1.28. oc adm ocp-certificates regenerate-top-level 95
2.7.1.29. oc adm ocp-certificates remove-old-trust 95
2.7.1.30. oc adm ocp-certificates update-ignition-ca-bundle-for-machine-config-server 96
2.7.1.31. oc adm policy add-cluster-role-to-group 96
2.7.1.32. oc adm policy add-cluster-role-to-user 96
2.7.1.33. oc adm policy add-role-to-user 96
2.7.1.34. oc adm policy add-scc-to-group 96
2.7.1.35. oc adm policy add-scc-to-user 97
2.7.1.36. oc adm policy remove-cluster-role-from-group 97
2.7.1.37. oc adm policy remove-cluster-role-from-user 97
2.7.1.38. oc adm policy scc-review 97
2.7.1.39. oc adm policy scc-subject-review 98
2.7.1.40. oc adm prune builds 98
2.7.1.41. oc adm prune deployments 98
2.7.1.42. oc adm prune groups 98
2.7.1.43. oc adm prune images 99
2.7.1.44. oc adm prune renderedmachineconfigs 99
2.7.1.45. oc adm prune renderedmachineconfigs list 100
2.7.1.46. oc adm reboot-machine-config-pool 100
2.7.1.47. oc adm release extract 100
2.7.1.48. oc adm release info 101
2.7.1.49. oc adm release mirror 101
2.7.1.50. oc adm release new 101
2.7.1.51. oc adm restart-kubelet 102
2.7.1.52. oc adm taint 102
2.7.1.53. oc adm top images 103
2.7.1.54. oc adm top imagestreams 103
2.7.1.55. oc adm top node 103
2.7.1.56. oc adm top persistentvolumeclaims 103
2.7.1.57. oc adm top pod 103
2.7.1.58. oc adm uncordon 104
2.7.1.59. oc adm upgrade 104
2.7.1.60. oc adm verify-image-signature 104
2.7.1.61. oc adm wait-for-node-reboot 105
2.7.1.62. oc adm wait-for-stable-cluster 105
2.7.2. Additional resources 105

CHAPTER 3. IMPORTANT UPDATE ON ODO **106**

CHAPTER 4. KNATIVE CLI FOR USE WITH OPENSHIFT SERVERLESS **107**
4.1. KEY FEATURES 107
4.2. INSTALLING THE KNATIVE CLI 107

CHAPTER 5. PIPELINES CLI (TKN) 108
5.1. INSTALLING TKN 108
5.1.1. Installing the Red Hat OpenShift Pipelines CLI on Linux 108
5.1.2. Installing the Red Hat OpenShift Pipelines CLI on Linux using an RPM 109
5.1.3. Installing the Red Hat OpenShift Pipelines CLI on Windows 110
5.1.4. Installing the Red Hat OpenShift Pipelines CLI on macOS 110
5.2. CONFIGURING THE OPENSHIFT PIPELINES TKN CLI 111
5.2.1. Enabling tab completion 111
5.3. OPENSHIFT PIPELINES TKN REFERENCE 111
5.3.1. Basic syntax 111
5.3.2. Global options 111
5.3.3. Utility commands 111
5.3.3.1. tkn 111
5.3.3.2. completion [shell] 112
5.3.3.3. version 112
5.3.4. Pipelines management commands 112
5.3.4.1. pipeline 112
5.3.4.2. pipeline delete 112
5.3.4.3. pipeline describe 112
5.3.4.4. pipeline list 112
5.3.4.5. pipeline logs 113
5.3.4.6. pipeline start 113
5.3.5. Pipeline run commands 113
5.3.5.1. pipelinerun 113
5.3.5.2. pipelinerun cancel 113
5.3.5.3. pipelinerun delete 113
5.3.5.4. pipelinerun describe 114
5.3.5.5. pipelinerun list 114
5.3.5.6. pipelinerun logs 114
5.3.6. Task management commands 114
5.3.6.1. task 114
5.3.6.2. task delete 115
5.3.6.3. task describe 115
5.3.6.4. task list 115
5.3.6.5. task logs 115
5.3.6.6. task start 115
5.3.7. Task run commands 115
5.3.7.1. taskrun 115
5.3.7.2. taskrun cancel 116
5.3.7.3. taskrun delete 116
5.3.7.4. taskrun describe 116
5.3.7.5. taskrun list 116
5.3.7.6. taskrun logs 116
5.3.8. Condition management commands 116
5.3.8.1. condition 116
5.3.8.2. condition delete 117
5.3.8.3. condition describe 117
5.3.8.4. condition list 117
5.3.9. Pipeline Resource management commands 117
5.3.9.1. resource 117
5.3.9.2. resource create 117
5.3.9.3. resource delete 118
5.3.9.4. resource describe 118

5.3.9.5. resource list 118
5.3.10. ClusterTask management commands 118
5.3.10.1. clustertask 118
5.3.10.2. clustertask delete 118
5.3.10.3. clustertask describe 118
5.3.10.4. clustertask list 119
5.3.10.5. clustertask start 119
5.3.11. Trigger management commands 119
5.3.11.1. eventlistener 119
5.3.11.2. eventlistener delete 119
5.3.11.3. eventlistener describe 119
5.3.11.4. eventlistener list 119
5.3.11.5. eventlistener logs 120
5.3.11.6. triggerbinding 120
5.3.11.7. triggerbinding delete 120
5.3.11.8. triggerbinding describe 120
5.3.11.9. triggerbinding list 120
5.3.11.10. triggertemplate 120
5.3.11.11. triggertemplate delete 121
5.3.11.12. triggertemplate describe 121
5.3.11.13. triggertemplate list 121
5.3.11.14. clustertriggerbinding 121
5.3.11.15. clustertriggerbinding delete 121
5.3.11.16. clustertriggerbinding describe 121
5.3.11.17. clustertriggerbinding list 122
5.3.12. Hub interaction commands 122
5.3.12.1. hub 122
5.3.12.2. hub downgrade 122
5.3.12.3. hub get 122
5.3.12.4. hub info 122
5.3.12.5. hub install 123
5.3.12.6. hub reinstall 123
5.3.12.7. hub search 123
5.3.12.8. hub upgrade 123

CHAPTER 6. OPM CLI **124**
6.1. INSTALLING THE OPM CLI 124
6.1.1. About the opm CLI 124
6.1.2. Installing the opm CLI 124
6.2. OPM CLI REFERENCE 125
6.2.1. generate 126
6.2.1.1. dockerfile 126
6.2.2. index 127
6.2.2.1. add 128
6.2.2.2. prune 128
6.2.2.3. prune-stranded 129
6.2.2.4. rm 130
6.2.3. init 131
6.2.4. migrate 131
6.2.5. render 132
6.2.6. serve 132
6.2.7. validate 133

CHAPTER 7. ROSA CLI 134
7.1. GETTING STARTED WITH THE ROSA CLI 134
7.1.1. About the ROSA CLI 134
7.1.2. Setting up the ROSA CLI 134
7.1.3. Configuring the ROSA CLI 136
7.1.3.1. login 136
7.1.3.1.1. Authenticating the ROSA CLI with Red Hat single sign-on 136
7.1.3.1.2. Authenticating the ROSA CLI with a single sign-on authorization code 137
7.1.3.1.3. Authenticating the ROSA CLI with a single sign-on device code 137
7.1.3.1.4. Authenticating the ROSA CLI with an offline token 138
7.1.3.2. logout 139
7.1.3.3. verify permissions 139
7.1.3.4. verify quota 140
7.1.3.5. download rosa 141
7.1.3.6. download oc 141
7.1.3.7. verify oc 141
7.1.4. Updating the ROSA CLI 142
7.2. ROSA CLI COMMAND REFERENCE 143
7.2.1. ROSA CLI commands 143
7.2.1.1. rosa create account-roles 143
7.2.1.2. rosa create admin 143
7.2.1.3. rosa create autoscaler 143
7.2.1.4. rosa create break-glass-credential 144
7.2.1.5. rosa create cluster 144
7.2.1.6. rosa create decision 144
7.2.1.7. rosa create dns-domain 144
7.2.1.8. rosa create external-auth-provider 145
7.2.1.9. rosa create iamserviceaccount 145
7.2.1.10. rosa create idp 145
7.2.1.11. rosa create image-mirror 145
7.2.1.12. rosa create kubeletconfig 145
7.2.1.13. rosa create log-forwarder 146
7.2.1.14. rosa create machinepool 146
7.2.1.15. rosa create network 146
7.2.1.16. rosa create ocm-role 147
7.2.1.17. rosa create oidc-config 147
7.2.1.18. rosa create oidc-provider 147
7.2.1.19. rosa create operator-roles 147
7.2.1.20. rosa create tuning-configs 148
7.2.1.21. rosa create user-role 148
7.2.1.22. rosa delete account-roles 148
7.2.1.23. rosa delete admin 148
7.2.1.24. rosa delete autoscaler 148
7.2.1.25. rosa delete cluster 149
7.2.1.26. rosa delete dns-domain 149
7.2.1.27. rosa delete external-auth-provider 149
7.2.1.28. rosa delete iamserviceaccount 149
7.2.1.29. rosa delete idp 149
7.2.1.30. rosa delete image-mirror 149
7.2.1.31. rosa delete ingress 150
7.2.1.32. rosa delete kubeletconfig 150
7.2.1.33. rosa delete log-forwarder 150
7.2.1.34. rosa delete machinepool 150

7.2.1.35. rosa delete ocm-role 150
7.2.1.36. rosa delete oidc-config 151
7.2.1.37. rosa delete oidc-provider 151
7.2.1.38. rosa delete operator-roles 151
7.2.1.39. rosa delete tuning-configs 151
7.2.1.40. rosa delete user-role 151
7.2.1.41. rosa describe access-request 152
7.2.1.42. rosa describe addon 152
7.2.1.43. rosa describe addon-installation 152
7.2.1.44. rosa describe admin 152
7.2.1.45. rosa describe autoscaler 152
7.2.1.46. rosa describe break-glass-credential 152
7.2.1.47. rosa describe cluster 153
7.2.1.48. rosa describe external-auth-provider 153
7.2.1.49. rosa describe iamserviceaccount 153
7.2.1.50. rosa describe ingress 153
7.2.1.51. rosa describe kubeletconfig 153
7.2.1.52. rosa describe log-forwarder 154
7.2.1.53. rosa describe machinepool 154
7.2.1.54. rosa describe tuning-configs 154
7.2.1.55. rosa describe upgrade 154
7.2.1.56. rosa download openshift-client 154
7.2.1.57. rosa download rosa-client 154
7.2.1.58. rosa edit addon 155
7.2.1.59. rosa edit autoscaler 155
7.2.1.60. rosa edit cluster 155
7.2.1.61. rosa edit image-mirror 155
7.2.1.62. rosa edit ingress 156
7.2.1.63. rosa edit kubeletconfig 156
7.2.1.64. rosa edit machinepool 156
7.2.1.65. rosa edit tuning-configs 156
7.2.1.66. rosa grant user 157
7.2.1.67. rosa init 157
7.2.1.68. rosa install addon 157
7.2.1.69. rosa link ocm-role 157
7.2.1.70. rosa link user-role 157
7.2.1.71. rosa list access-request 158
7.2.1.72. rosa list account-roles 158
7.2.1.73. rosa list addons 158
7.2.1.74. rosa list break-glass-credentials 158
7.2.1.75. rosa list clusters 158
7.2.1.76. rosa list dns-domain 158
7.2.1.77. rosa list external-auth-providers 159
7.2.1.78. rosa list gates 159
7.2.1.79. rosa list iamserviceaccounts 159
7.2.1.80. rosa list idps 159
7.2.1.81. rosa list image-mirrors 159
7.2.1.82. rosa list ingresses 160
7.2.1.83. rosa list instance-types 160
7.2.1.84. rosa list kubeletconfigs 160
7.2.1.85. rosa list log-forwarders 160
7.2.1.86. rosa list machinepools 160
7.2.1.87. rosa list ocm-roles 161

7.2.1.88. rosa list oidc-config 161
7.2.1.89. rosa list oidc-providers 161
7.2.1.90. rosa list operator-roles 161
7.2.1.91. rosa list regions 161
7.2.1.92. rosa list tuning-configs 161
7.2.1.93. rosa list user-roles 162
7.2.1.94. rosa list users 162
7.2.1.95. rosa list versions 162
7.2.1.96. rosa login 162
7.2.1.97. rosa logs 162
7.2.1.98. rosa logs install 163
7.2.1.99. rosa logs uninstall 163
7.2.1.100. rosa register oidc-config 163
7.2.1.101. rosa revoke break-glass-credentials 163
7.2.1.102. rosa revoke user 163
7.2.1.103. rosa uninstall addon 164
7.2.1.104. rosa unlink ocm-role 164
7.2.1.105. rosa unlink user-role 164
7.2.1.106. rosa upgrade account-roles 164
7.2.1.107. rosa upgrade cluster 164
7.2.1.108. rosa upgrade machinepool 165
7.2.1.109. rosa upgrade operator-roles 165
7.2.1.110. rosa upgrade roles 165
7.2.1.111. rosa verify network 165
7.2.1.112. rosa verify openshift-client 165
7.2.1.113. rosa verify permissions 166
7.2.1.114. rosa verify quota 166
7.2.1.115. rosa verify rosa-client 166
7.2.1.116. rosa whoami 166
7.3. LEAST PRIVILEGE PERMISSIONS FOR ROSA CLI COMMANDS 166
7.3.1. Least privilege permissions for common Red Hat OpenShift Service on AWS CLI commands 167
7.3.1.1. Create a managed OpenID Connect (OIDC) provider 167
7.3.1.2. Create an unmanaged OpenID Connect provider 167
7.3.1.3. List your account roles 168
7.3.1.4. List your Operator roles 168
7.3.1.5. List your OIDC providers 169
7.3.1.6. Verify your quota 169
7.3.1.7. Delete your managed OIDC configuration 170
7.3.1.8. Delete your unmanaged OIDC configuration 170
7.3.1.9. Create a cluster 171
7.3.1.10. Create your account roles and Operator roles 172
7.3.1.11. Delete your account roles 172
7.3.1.12. Delete your Operator roles 173
7.3.2. ROSA CLI commands with no required permissions 173
7.3.3. Additional resources 174
7.4. MANAGING BILLING ACCOUNTS FOR RED HAT OPENSHIFT SERVICE ON AWS CLUSTERS 174
7.4.1. Update billing accounts for Red Hat OpenShift Service on AWS clusters 174

CHAPTER 1. RED HAT OPENSHIFT SERVICE ON AWS CLI TOOLS OVERVIEW

A user performs a range of operations while working on Red Hat OpenShift Service on AWS such as the following:

- Managing clusters
- Building, deploying, and managing applications
- Managing deployment processes
- Creating and maintaining Operator catalogs

Red Hat OpenShift Service on AWS offers a set of command-line interface (CLI) tools that simplify these tasks by enabling users to perform various administration and development operations from the terminal. These tools expose simple commands to manage the applications, as well as interact with each component of the system.

1.1. LIST OF CLI TOOLS

The following set of CLI tools are available in Red Hat OpenShift Service on AWS:

- OpenShift CLI (**oc**): This is the most commonly used CLI tool by Red Hat OpenShift Service on AWS users. It helps both cluster administrators and developers to perform end-to-end operations across Red Hat OpenShift Service on AWS using the terminal. Unlike the web console, it allows the user to work directly with the project source code using command scripts.
- Knative CLI (kn): The Knative (**kn**) CLI tool provides simple and intuitive terminal commands that can be used to interact with OpenShift Serverless components, such as Knative Serving and Eventing.
- Pipelines CLI (tkn): OpenShift Pipelines is a continuous integration and continuous delivery (CI/CD) solution in Red Hat OpenShift Service on AWS, which internally uses Tekton. The **tkn** CLI tool provides simple and intuitive commands to interact with OpenShift Pipelines using the terminal.
- opm CLI: The **opm** CLI tool helps the Operator developers and cluster administrators to create and maintain the catalogs of Operators from the terminal.
- ROSA CLI (**rosa**): Use the **rosa** CLI to create, update, manage, and delete Red Hat OpenShift Service on AWS clusters and resources.

CHAPTER 2. OPENSHIFT CLI (OC)

2.1. GETTING STARTED WITH THE OPENSHIFT CLI

2.1.1. About the OpenShift CLI

With the OpenShift CLI (**oc**), you can create applications and manage Red Hat OpenShift Service on AWS projects from a terminal. The OpenShift CLI is ideal in the following situations:

- Working directly with project source code.
- Scripting Red Hat OpenShift Service on AWS operations
- Managing projects while restricted by bandwidth resources and the web console is unavailable.

2.1.2. Installing the OpenShift CLI

You can install the OpenShift CLI (**oc**) either by downloading the binary or by using an RPM.

2.1.3. Installing the OpenShift CLI on Linux

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on Linux.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat OpenShift Service on AWS.

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the architecture from the **Product Variant** list.
3. Select the appropriate version from the **Version** list.
4. Click **Download Now** next to the **OpenShift v4 Linux Clients** entry and save the file.
5. Unpack the archive:

```
$ tar xvf <file>
```

6. Place the **oc** binary in a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.1.4. Installing the OpenShift CLI on Windows

To manage your cluster and deploy applications from the command line, install OpenShift CLI (**oc**) binary on Windows.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat OpenShift Service on AWS.

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform page on the Red Hat Customer Portal.
2. Select the appropriate version from the **Version** list.
3. Click **Download Now** next to the **OpenShift v4 Windows Client** entry and save the file.
4. Extract the archive with a ZIP program.
5. Move the **oc** binary to a directory that is on your **PATH** variable.
   To check your **PATH** variable, open the command prompt and execute the following command:

```
C:\> path
```

Verification

- After you install the OpenShift CLI, it is available using the **oc** command:

```
C:\> oc <command>
```

2.1.5. Installing the OpenShift CLI on macOS

To manage your cluster and deploy applications from the command line, install the OpenShift CLI (**oc**) binary on macOS.

IMPORTANT

If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat OpenShift Service on AWS.

Download and install the new version of **oc**.

Procedure

1. Navigate to the Download OpenShift Container Platform on the Red Hat Customer Portal.

2. Select the appropriate version from the **Version** drop-down list.
3. Click **Download Now** next to the **OpenShift v4 macOS Clients** entry and save the file.
4. Unpack and unzip the archive.
5. Move the **oc** binary to a directory on your **PATH** variable.
   To check your **PATH** variable, open a terminal and execute the following command:

```
$ echo $PATH
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

2.1.5.1. Installing the OpenShift CLI by using the web console

You can install the OpenShift CLI (**oc**) to interact with Red Hat OpenShift Service on AWS clusters from a web console. You can install **oc** on Linux, Windows, or macOS.

IMPORTANT

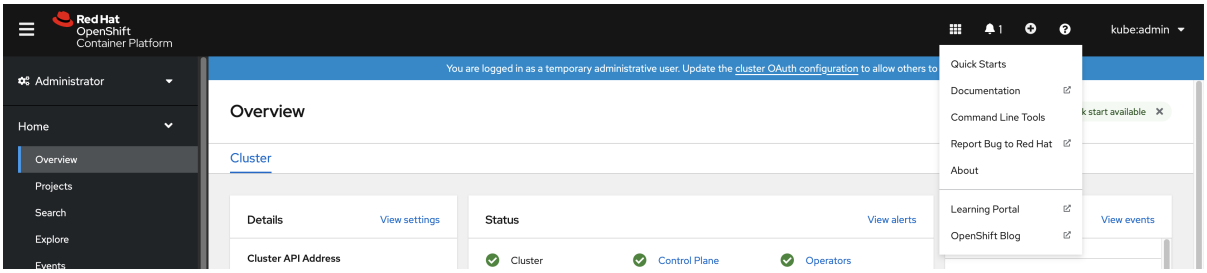
If you installed an earlier version of **oc**, you cannot use it to complete all of the commands in Red Hat OpenShift Service on AWS. Download and install the new version of **oc**.

2.1.5.1.1. Installing the OpenShift CLI on Linux using the web console

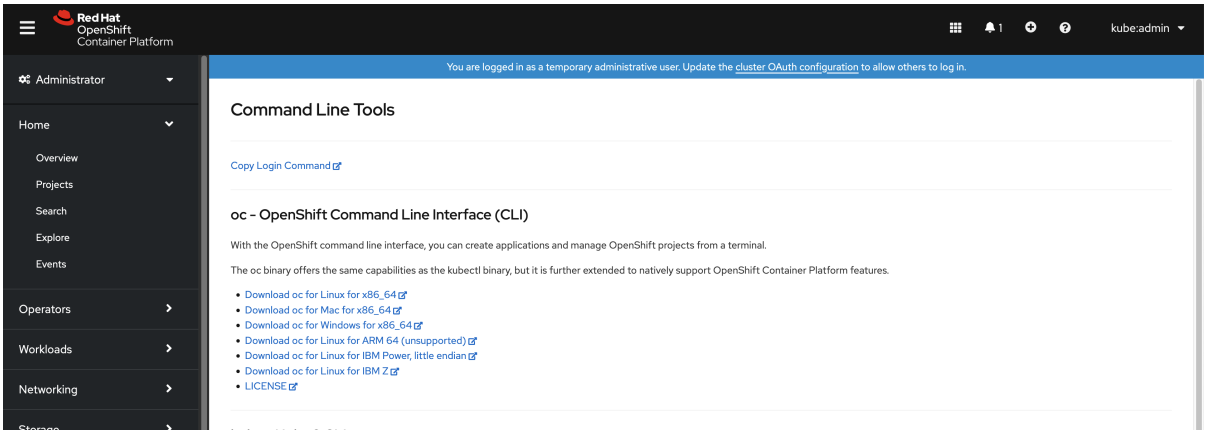
You can install the OpenShift CLI (**oc**) binary on Linux by using the following procedure.

Procedure

1. From the web console, click **?**.



2. Click **Command Line Tools**.



3. Select appropriate **oc** binary for your Linux platform, and then click **Download oc for Linux**
4. Save the file.
5. Unpack the archive.

```
$ tar xvf <file>
```

6. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, execute the following command:

```
$ echo $PATH
```

After you install the OpenShift CLI, it is available using the **oc** command:

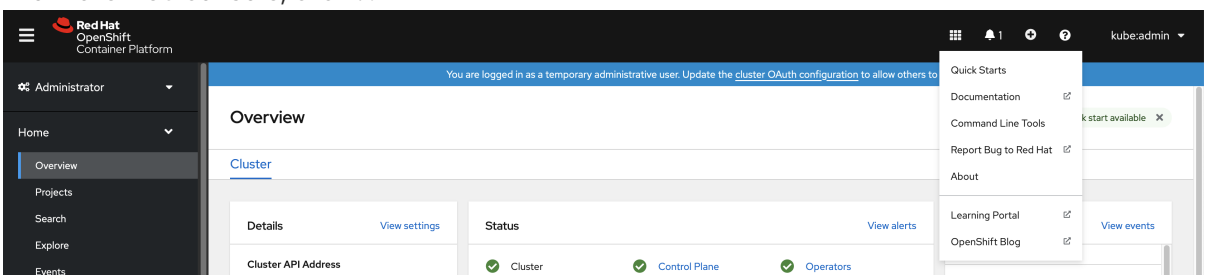
```
$ oc <command>
```

2.1.5.1.2. Installing the OpenShift CLI on Windows using the web console

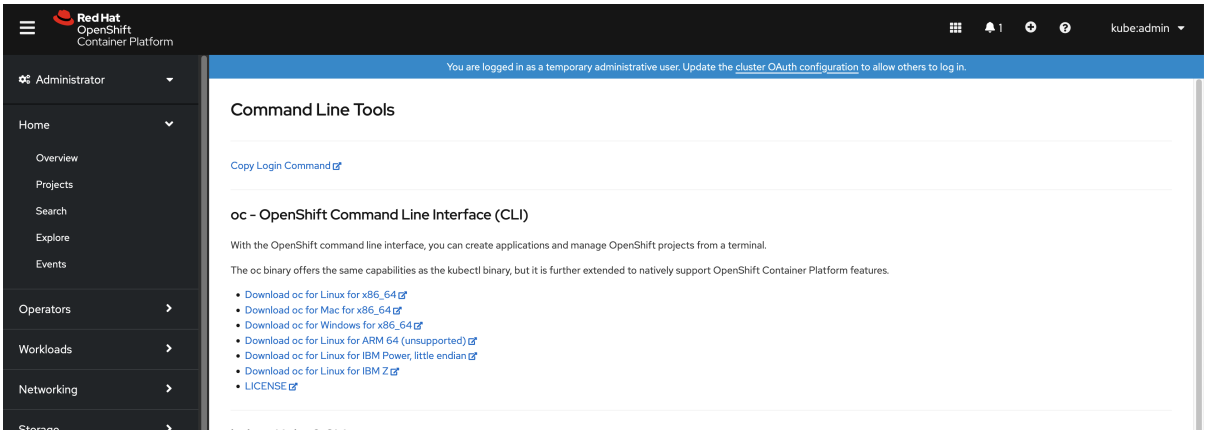
You can install the OpenShift CLI (**oc**) binary on Windows by using the following procedure.

Procedure

1. From the web console, click **?**.



2. Click **Command Line Tools**.



3. Select the **oc** binary for Windows platform, and then click **Download oc for Windows for x86_64**.

4. Save the file.

5. Unzip the archive with a ZIP program.

6. Move the **oc** binary to a directory that is on your **PATH**.
   To check your **PATH**, open the command prompt and execute the following command:

```
C:\> path
```

After you install the OpenShift CLI, it is available using the **oc** command:

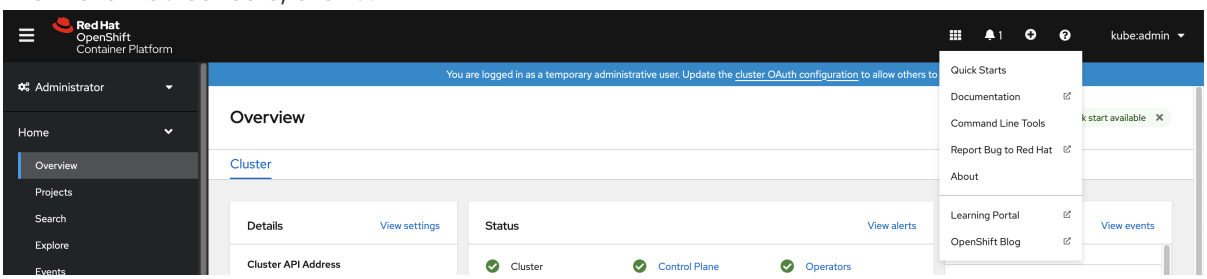
```
C:\> oc <command>
```

2.1.5.1.3. Installing the OpenShift CLI on macOS using the web console

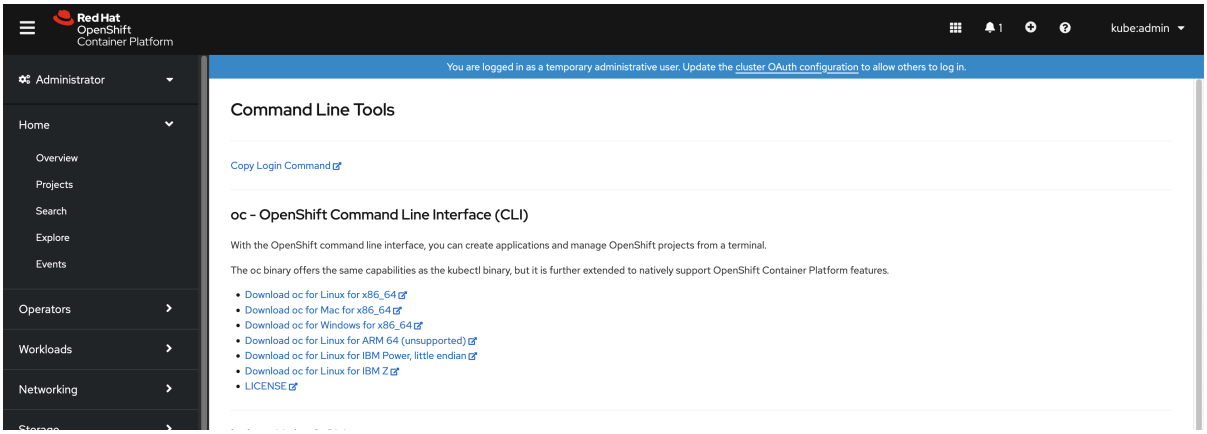
You can install the OpenShift CLI (**oc**) binary on macOS by using the following procedure.

Procedure

1. From the web console, click **?**.



2. Click **Command Line Tools**.



3. Select the **oc** binary for macOS platform, and then click **Download oc for Mac for x86_64**

> **NOTE**
>
> For macOS arm64, click **Download oc for Mac for ARM 64**

4. Save the file.
5. Unpack and unzip the archive.
6. Move the **oc** binary to a directory on your PATH.
   To check your **PATH**, open a terminal and execute the following command:

```
$ echo $PATH
```

After you install the OpenShift CLI, it is available using the **oc** command:

```
$ oc <command>
```

2.1.5.2. Installing the OpenShift CLI by using an RPM

For Red Hat Enterprise Linux (RHEL), you can install the OpenShift CLI (**oc**) as an RPM if you have an active Red Hat OpenShift Service on AWS subscription on your Red Hat account.

> **IMPORTANT**
>
> You must install **oc** for RHEL 9 by downloading the binary. Installing **oc** by using an RPM package is not supported on Red Hat Enterprise Linux (RHEL) 9.

Prerequisites

- You must have root or sudo privileges.

Procedure

1. Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

2. Pull the latest subscription data:

```
# subscription-manager refresh
```

3. List the available subscriptions:

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. In the output for the previous command, find the pool ID for an Red Hat OpenShift Service on AWS subscription and attach the subscription to the registered system:

```
# subscription-manager attach --pool=<pool_id>
```

5. Enable the repositories required by Red Hat OpenShift Service on AWS 4.

```
# subscription-manager repos --enable="rhocp-4-for-rhel-8-x86_64-rpms"
```

6. Install the **openshift-clients** package:

```
# yum install openshift-clients
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

2.1.5.3. Installing the OpenShift CLI by using Homebrew

For macOS, you can install the OpenShift CLI (**oc**) by using the Homebrew package manager.

Prerequisites

- You must have Homebrew (**brew**) installed.

Procedure

- Install the openshift-cli package by running the following command:

```
$ brew install openshift-cli
```

Verification

- Verify your installation by using an **oc** command:

```
$ oc <command>
```

2.1.6. Logging in to the OpenShift CLI

You can log in to the OpenShift CLI (**oc**) to access and manage your cluster.

Prerequisites

- You must have access to a Red Hat OpenShift Service on AWS cluster.
- The OpenShift CLI (**oc**) is installed.

> **NOTE**
>
> To access a cluster that is accessible only over an HTTP proxy server, you can set the **HTTP_PROXY**, **HTTPS_PROXY** and **NO_PROXY** variables. These environment variables are respected by the **oc** CLI so that all communication with the cluster goes through the HTTP proxy.
>
> Authentication headers are sent only when using HTTPS transport.

Procedure

1. Enter the **oc login** command and pass in a user name:

```
$ oc login -u user1
```

2. When prompted, enter the required information:

Example output

```
Server [https://localhost:8443]: https://openshift.example.com:6443 (1)
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could be
intercepted by others.
Use insecure connections? (y/n): y (2)

Authentication required for https://openshift.example.com:6443 (openshift)
Username: user1
Password: (3)
Login successful.

You don't have any projects. You can try to create a new project, by running

    oc new-project <projectname>

Welcome! See 'oc help' to get started.
```

1. Enter the Red Hat OpenShift Service on AWS server URL.
2. Enter whether to use insecure connections.
3. Enter the user's password.

NOTE

If you are logged in to the web console, you can generate an **oc login** command that includes your token and server information. You can use the command to log in to the OpenShift CLI (**oc**) without the interactive prompts. To generate the command, select **Copy login command** from the username drop-down menu at the top right of the web console.

You can now create a project or issue other commands for managing your cluster.

2.1.7. Logging in to the OpenShift CLI using a web browser

You can log in to the OpenShift CLI (**oc**) with the help of a web browser to access and manage your cluster. This allows users to avoid inserting their access token into the command line.

WARNING

Logging in to the CLI through the web browser runs a server on localhost with HTTP, not HTTPS; use with caution on multi-user workstations.

Prerequisites

- You must have access to an Red Hat OpenShift Service on AWS cluster.
- You must have installed the OpenShift CLI (**oc**).
- You must have a browser installed.

Procedure

1. Enter the **oc login** command with the **--web** flag:

```
$ oc login <cluster_url> --web (1)
```

(1) Optionally, you can specify the server URL and callback port. For example, **oc login <cluster_url> --web --callback-port 8280 localhost:8443**.

2. The web browser opens automatically. If it does not, click the link in the command output. If you do not specify the Red Hat OpenShift Service on AWS server **oc** tries to open the web console of the cluster specified in the current **oc** configuration file. If no **oc** configuration exists, **oc** prompts interactively for the server URL.

Example output

```
Opening login URL in the default browser: https://openshift.example.com
Opening in existing browser session.
```

3. If more than one identity provider is available, select your choice from the options provided.

4. Enter your username and password into the corresponding browser fields. After you are logged in, the browser displays the text **access token received successfully; please return to your terminal**.

5. Check the CLI for a login confirmation.

   Example output

   ```
   Login successful.

   You don't have any projects. You can try to create a new project, by running

     oc new-project <projectname>
   ```

> **NOTE**
>
> The web console defaults to the profile used in the previous session. To switch between Administrator and Developer profiles, log out of the Red Hat OpenShift Service on AWS web console and clear the cache.

You can now create a project or issue other commands for managing your cluster.

2.1.8. Using the OpenShift CLI

Review the following sections to learn how to complete common tasks using the CLI.

2.1.8.1. Creating a project

Use the **oc new-project** command to create a new project.

```
$ oc new-project my-project
```

Example output

```
Now using project "my-project" on server "https://openshift.example.com:6443".
```

2.1.8.2. Creating a new app

Use the **oc new-app** command to create a new application.

```
$ oc new-app https://github.com/sclorg/cakephp-ex
```

Example output

```
--> Found image 40de956 (9 days old) in imagestream "openshift/php" under tag "7.2" for "php"

...

    Run 'oc status' to view your app.
```

2.1.8.3. Viewing pods

Use the **oc get pods** command to view the pods for the current project.

> **NOTE**
>
> When you run **oc** inside a pod and do not specify a namespace, the namespace of the pod is used by default.

```
$ oc get pods -o wide
```

Example output

```
NAME                    READY   STATUS      RESTARTS   AGE     IP           NODE
NOMINATED NODE
cakephp-ex-1-build      0/1     Completed   0          5m45s   10.131.0.10  ip-10-0-141-74.ec2.internal
<none>
cakephp-ex-1-deploy     0/1     Completed   0          3m44s   10.129.2.9   ip-10-0-147-65.ec2.internal
<none>
cakephp-ex-1-ktz97      1/1     Running     0          3m33s   10.128.2.11  ip-10-0-168-105.ec2.internal
<none>
```

2.1.8.4. Viewing pod logs

Use the **oc logs** command to view logs for a particular pod.

```
$ oc logs cakephp-ex-1-deploy
```

Example output

```
--> Scaling cakephp-ex-1 to 1
--> Success
```

2.1.8.5. Viewing the current project

Use the **oc project** command to view the current project.

```
$ oc project
```

Example output

```
Using project "my-project" on server "https://openshift.example.com:6443".
```

2.1.8.6. Viewing the status for the current project

Use the **oc status** command to view information about the current project, such as services, deployments, and build configs.

```
$ oc status
```

Example output

```
In project my-project on server https://openshift.example.com:6443

svc/cakephp-ex - 172.30.236.80 ports 8080, 8443
  dc/cakephp-ex deploys istag/cakephp-ex:latest <-
    bc/cakephp-ex source builds https://github.com/sclorg/cakephp-ex on openshift/php:7.2
    deployment #1 deployed 2 minutes ago - 1 pod

3 infos identified, use 'oc status --suggest' to see details.
```

2.1.8.7. Listing supported API resources

Use the **oc api-resources** command to view the list of supported API resources on the server.

```
$ oc api-resources
```

Example output

```
NAME                              SHORTNAMES   APIGROUP                       NAMESPACED   KIND
bindings                                                                      true         Binding
componentstatuses                 cs                                          false        ComponentStatus
configmaps                        cm                                          true         ConfigMap
...
```

2.1.9. Getting help

You can get help with CLI commands and Red Hat OpenShift Service on AWS resources in the following ways:

- Use **oc help** to get a list and description of all available CLI commands:

  #### Example: Get general help for the CLI

  ```
  $ oc help
  ```

  #### Example output

  ```
  OpenShift Client

  This client helps you develop, build, deploy, and run your applications on any OpenShift or
  Kubernetes compatible
  platform. It also includes the administrative commands for managing a cluster under the 'adm'
  subcommand.

  Usage:
    oc [flags]

  Basic Commands:
    login           Log in to a server
    new-project     Request a new project
    new-app         Create a new application
  ```

```
...
```

- Use the **--help** flag to get help about a specific CLI command:

Example: Get help for the oc create command

```
$ oc create --help
```

Example output

```
Create a resource by filename or stdin

JSON and YAML formats are accepted.

Usage:
  oc create -f FILENAME [flags]

...
```

- Use the **oc explain** command to view the description and fields for a particular resource:

Example: View documentation for the Pod resource

```
$ oc explain pods
```

Example output

```
KIND:     Pod
VERSION:  v1

DESCRIPTION:
     Pod is a collection of containers that can run on a host. This resource is
     created by clients and scheduled onto hosts.

FIELDS:
   apiVersion <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/api-conventions.md#resources

...
```

2.1.10. Logging out of the OpenShift CLI

You can log out the OpenShift CLI to end your current session.

- Use the **oc logout** command.

```
$ oc logout
```

Example output

```
Logged "user1" out on "https://openshift.example.com"
```

This deletes the saved authentication token from the server and removes it from your configuration file.

2.2. CONFIGURING THE OPENSHIFT CLI

2.2.1. Enabling tab completion

You can enable tab completion for the Bash or Zsh shells.

2.2.1.1. Enabling tab completion for Bash

After you install the OpenShift CLI (**oc**), you can enable tab completion to automatically complete **oc** commands or suggest options when you press Tab. The following procedure enables tab completion for the Bash shell.

Prerequisites

- You must have the OpenShift CLI (**oc**) installed.
- You must have the package **bash-completion** installed.

Procedure

1. Save the Bash completion code to a file:

```
$ oc completion bash > oc_bash_completion
```

2. Copy the file to **/etc/bash_completion.d/**:

```
$ sudo cp oc_bash_completion /etc/bash_completion.d/
```

You can also save the file to a local directory and source it from your **.bashrc** file instead. Tab completion is enabled when you open a new terminal.

2.2.1.2. Enabling tab completion for Zsh

After you install the OpenShift CLI (**oc**), you can enable tab completion to automatically complete **oc** commands or suggest options when you press Tab. The following procedure enables tab completion for the Zsh shell.

Prerequisites

- You must have the OpenShift CLI (**oc**) installed.

Procedure

- To add tab completion for **oc** to your **.zshrc** file, run the following command:

```
$ cat >>~/.zshrc<<EOF
autoload -Uz compinit
compinit
```

```
if [ $commands[oc] ]; then
  source <(oc completion zsh)
  compdef _oc oc
fi
EOF
```

Tab completion is enabled when you open a new terminal.

2.2.2. Accessing kubeconfig by using the oc CLI

You can use the **oc** CLI to log in to your OpenShift cluster and retrieve a kubeconfig file for accessing the cluster from the command line.

Prerequisites

- You have access to the Red Hat OpenShift Service on AWS web console or API server endpoint.

Procedure

1. Log in to your OpenShift cluster by running the following command:

```
$ oc login <api-server-url> -u <username> -p <password> (1)(2)(3)
```

(1) Specify the full API server URL. For example: **https://api.my-cluster.example.com:6443**.

(2) Specify a valid username. For example: **kubeadmin**.

(3) Provide the password for the specified user. For example, the **kubeadmin** password generated during cluster installation.

2. Save the cluster configuration to a local file by running the following command:

```
$ oc config view --raw > kubeconfig
```

3. Set the **KUBECONFIG** environment variable to point to the exported file by running the following command:

```
$ export KUBECONFIG=./kubeconfig
```

4. Use **oc** to interact with your OpenShift cluster by running the following command:

```
$ oc get nodes
```

NOTE

If you plan to reuse the exported **kubeconfig** file across sessions or machines, store it securely and avoid committing it to source control.

2.3. USAGE OF OC AND KUBECTL COMMANDS

The Kubernetes command-line interface (CLI), **kubectl**, can be used to run commands against a Kubernetes cluster. Because Red Hat OpenShift Service on AWS is a certified Kubernetes distribution, you can use the supported **kubectl** binaries that ship with Red Hat OpenShift Service on AWS, or you can gain extended functionality by using the **oc** binary.

2.3.1. The oc binary

The **oc** binary offers the same capabilities as the **kubectl** binary, but it extends to natively support additional Red Hat OpenShift Service on AWS features, including:

- **Full support for Red Hat OpenShift Service on AWS resources**
  Resources such as **DeploymentConfig**, **BuildConfig**, **Route**, **ImageStream**, and **ImageStreamTag** objects are specific to Red Hat OpenShift Service on AWS distributions, and build upon standard Kubernetes primitives.
- **Authentication**
- **Additional commands**
  The additional command **oc new-app**, for example, makes it easier to get new applications started using existing source code or pre-built images. Similarly, the additional command **oc new-project** makes it easier to start a project that you can switch to as your default.

> **IMPORTANT**
>
> If you installed an earlier version of the **oc** binary, you cannot use it to complete all of the commands in Red Hat OpenShift Service on AWS . If you want the latest features, you must download and install the latest version of the **oc** binary corresponding to your Red Hat OpenShift Service on AWS server version.

Non-security API changes will involve, at minimum, two minor releases (4.1 to 4.2 to 4.3, for example) to allow older **oc** binaries to update. Using new capabilities might require newer **oc** binaries. A 4.3 server might have additional capabilities that a 4.2 **oc** binary cannot use and a 4.3 **oc** binary might have additional capabilities that are unsupported by a 4.2 server.

Table 2.1. Compatibility Matrix

	X.Y (**oc** Client)	X.Y+N [a] (**oc** Client)
X.Y (Server)	1	3
X.Y+N [a] (Server)	2	1
[a] Where **N** is a number greater than or equal to 1.		

1 Fully compatible.

2 **oc** client might not be able to access server features.

3 **oc** client might provide options and features that might not be compatible with the accessed server.

2.3.2. The kubectl binary

The **kubectl** binary is provided as a means to support existing workflows and scripts for new Red Hat OpenShift Service on AWS users coming from a standard Kubernetes environment, or for those who prefer to use the **kubectl** CLI. Existing users of **kubectl** can continue to use the binary to interact with Kubernetes primitives, with no changes required to the Red Hat OpenShift Service on AWS cluster.

You can install the supported **kubectl** binary by following the steps to Install the OpenShift CLI. The **kubectl** binary is included in the archive if you download the binary, or is installed when you install the CLI by using an RPM.

For more information, see the kubectl documentation.

2.4. MANAGING CLI PROFILES

A CLI configuration file allows you to configure different profiles, or contexts, for use with the CLI tools overview. A context consists of a Red Hat OpenShift Service on AWS server information associated with a *nickname*.

2.4.1. About switches between CLI profiles

Contexts allow you to easily switch between multiple users across multiple Red Hat OpenShift Service on AWS servers, or clusters, when using CLI operations. Nicknames make managing CLI configurations easier by providing short-hand references to contexts, user credentials, and cluster details. After a user logs in with the **oc** CLI for the first time, Red Hat OpenShift Service on AWS creates a **~/.kube/config** file if one does not already exist. As more authentication and connection details are provided to the CLI, either automatically during an **oc login** operation or by manually configuring CLI profiles, the updated information is stored in the configuration file:

CLI config file

```
apiVersion: v1
clusters: 1
- cluster:
    insecure-skip-tls-verify: true
    server: https://openshift1.example.com:8443
  name: openshift1.example.com:8443
- cluster:
    insecure-skip-tls-verify: true
    server: https://openshift2.example.com:8443
  name: openshift2.example.com:8443
contexts: 2
- context:
    cluster: openshift1.example.com:8443
    namespace: alice-project
    user: alice/openshift1.example.com:8443
  name: alice-project/openshift1.example.com:8443/alice
- context:
    cluster: openshift1.example.com:8443
    namespace: joe-project
```

```
    user: alice/openshift1.example.com:8443
  name: joe-project/openshift1/alice
current-context: joe-project/openshift1.example.com:8443/alice (3)
kind: Config
preferences: {}
users: (4)
- name: alice/openshift1.example.com:8443
  user:
    token: xZHd2piv5_9vQrg-SKXRJ2Dsl9SceNJdhNTljEKTb8k
```

1. The **clusters** section defines connection details for Red Hat OpenShift Service on AWS clusters, including the address for their master server. In this example, one cluster is nicknamed **openshift1.example.com:8443** and another is nicknamed **openshift2.example.com:8443**.
2. This **contexts** section defines two contexts: one nicknamed **alice-project/openshift1.example.com:8443/alice**, using the **alice-project** project, **openshift1.example.com:8443** cluster, and **alice** user, and another nicknamed **joe-project/openshift1.example.com:8443/alice**, using the **joe-project** project, **openshift1.example.com:8443** cluster and **alice** user.
3. The **current-context** parameter shows that the **joe-project/openshift1.example.com:8443/alice** context is currently in use, allowing the **alice** user to work in the **joe-project** project on the **openshift1.example.com:8443** cluster.
4. The **users** section defines user credentials. In this example, the user nickname **alice/openshift1.example.com:8443** uses an access token.

The CLI can support multiple configuration files which are loaded at runtime and merged together along with any override options specified from the command line. After you are logged in, you can use the **oc status** or **oc project** command to verify your current working environment:

Verify the current working environment

```
$ oc status
```

Example output

```
oc status
In project Joe's Project (joe-project)

service database (172.30.43.12:5434 -> 3306)
  database deploys docker.io/openshift/mysql-55-centos7:latest
    #1 deployed 25 minutes ago - 1 pod

service frontend (172.30.159.137:5432 -> 8080)
  frontend deploys origin-ruby-sample:latest <-
    builds https://github.com/openshift/ruby-hello-world with joe-project/ruby-20-centos7:latest
    #1 deployed 22 minutes ago - 2 pods

To see more information about a service or deployment, use 'oc describe service <name>' or 'oc
describe dc <name>'.
You can use 'oc get all' to see lists of each of the types described in this example.
```

List the current project

```
$ oc project
```

Example output

```
Using project "joe-project" from context named "joe-project/openshift1.example.com:8443/alice" on
server "https://openshift1.example.com:8443".
```

You can run the **oc login** command again and supply the required information during the interactive process, to log in using any other combination of user credentials and cluster details. A context is constructed based on the supplied information if one does not already exist. If you are already logged in and want to switch to another project the current user already has access to, use the **oc project** command and enter the name of the project:

```
$ oc project alice-project
```

Example output

```
Now using project "alice-project" on server "https://openshift1.example.com:8443".
```

At any time, you can use the **oc config view** command to view your current CLI configuration, as seen in the output. Additional CLI configuration commands are also available for more advanced usage.

NOTE

If you have access to administrator credentials but are no longer logged in as the default system user **system:admin**, you can log back in as this user at any time as long as the credentials are still present in your CLI config file. The following command logs in and switches to the default project:

```
$ oc login -u system:admin -n default
```

2.4.2. Manual configuration of CLI profiles

NOTE

This section covers more advanced usage of CLI configurations. In most situations, you can use the **oc login** and **oc project** commands to log in and switch between contexts and projects.

If you want to manually configure your CLI config files, you can use the **oc config** command instead of directly modifying the files. The **oc config** command includes a number of helpful sub-commands for this purpose:

Table 2.2. CLI configuration subcommands

Subcommand	Usage
set-cluster	Sets a cluster entry in the CLI config file. If the referenced cluster nickname already exists, the specified information is merged in. `$ oc config set-cluster <cluster_nickname> [--server=<master_ip_or_fqdn>] [--certificate-authority=<path/to/certificate/authority>] [--api-version=<apiversion>] [--insecure-skip-tls-verify=true]`
set-context	Sets a context entry in the CLI config file. If the referenced context nickname already exists, the specified information is merged in. `$ oc config set-context <context_nickname> [--cluster=<cluster_nickname>] [--user=<user_nickname>] [--namespace=<namespace>]`
use-context	Sets the current context using the specified context nickname. `$ oc config use-context <context_nickname>`
set	Sets an individual value in the CLI config file. `$ oc config set <property_name> <property_value>` The **<property_name>** is a dot-delimited name where each token represents either an attribute name or a map key. The **<property_value>** is the new value being set.
unset	Unsets individual values in the CLI config file. `$ oc config unset <property_name>` The **<property_name>** is a dot-delimited name where each token represents either an attribute name or a map key.

view	Displays the merged CLI configuration currently in use.
	
	`$ oc config view`
	
	Displays the result of the specified CLI config file.
	
	`$ oc config view --config=<specific_filename>`

Example usage

- Log in as a user that uses an access token. This token is used by the **alice** user:

```
$ oc login https://openshift1.example.com --
token=ns7yVhuRNpDM9cgzfhhxQ7bM5s7N2ZVrkZepSRf4LC0
```

- View the cluster entry automatically created:

```
$ oc config view
```

Example output

```
apiVersion: v1
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https://openshift1.example.com
  name: openshift1-example-com
contexts:
- context:
    cluster: openshift1-example-com
    namespace: default
    user: alice/openshift1-example-com
  name: default/openshift1-example-com/alice
current-context: default/openshift1-example-com/alice
kind: Config
preferences: {}
users:
- name: alice/openshift1.example.com
  user:
    token: ns7yVhuRNpDM9cgzfhhxQ7bM5s7N2ZVrkZepSRf4LC0
```

- Update the current context to have users log in to the desired namespace:

```
$ oc config set-context `oc config current-context` --namespace=<project_name>
```

- Examine the current context, to confirm that the changes are implemented:

```
$ oc whoami -c
```

All subsequent CLI operations uses the new context, unless otherwise specified by overriding CLI options or until the context is switched.

2.4.3. Load and merge rules

You can follow these rules, when issuing CLI operations for the loading and merging order for the CLI configuration:

- CLI config files are retrieved from your workstation, using the following hierarchy and merge rules:
  - If the **--config** option is set, then only that file is loaded. The flag is set once and no merging takes place.
  - If the **$KUBECONFIG** environment variable is set, then it is used. The variable can be a list of paths, and if so the paths are merged together. When a value is modified, it is modified in the file that defines the stanza. When a value is created, it is created in the first file that exists. If no files in the chain exist, then it creates the last file in the list.
  - Otherwise, the ***~/.kube/config*** file is used and no merging takes place.

- The context to use is determined based on the first match in the following flow:
  - The value of the **--context** option.
  - The **current-context** value from the CLI config file.
  - An empty value is allowed at this stage.
- The user and cluster to use is determined. At this point, you may or may not have a context; they are built based on the first match in the following flow, which is run once for the user and once for the cluster:
  - The value of the **--user** for user name and **--cluster** option for cluster name.
  - If the **--context** option is present, then use the context's value.
  - An empty value is allowed at this stage.
- The actual cluster information to use is determined. At this point, you may or may not have cluster information. Each piece of the cluster information is built based on the first match in the following flow:
  - The values of any of the following command-line options:
    - **--server**,
    - **--api-version**
    - **--certificate-authority**
    - **--insecure-skip-tls-verify**
  - If cluster information and a value for the attribute is present, then use it.
  - If you do not have a server location, then there is an error.
- The actual user information to use is determined. Users are built using the same rules as clusters, except that you can only have one authentication technique per user; conflicting techniques cause the operation to fail. Command-line options take precedence over config file values. Valid command-line options are:
  - **--auth-path**
  - **--client-certificate**
  - **--client-key**
  - **--token**
- For any information that is still missing, default values are used and prompts are given for additional information.

2.5. EXTENDING THE OPENSHIFT CLI WITH PLUGINS

You can write and install plugins to build on the default **oc** commands, allowing you to perform new and more complex tasks with the OpenShift CLI.

2.5.1. Writing CLI plugins

You can write a plugin for the OpenShift CLI in any programming language or script that allows you to write command-line commands. Note that you can not use a plugin to overwrite an existing **oc** command.

Procedure

This procedure creates a simple Bash plugin that prints a message to the terminal when the **oc foo** command is issued.

1. Create a file called **oc-foo**.
   When naming your plugin file, keep the following in mind:
   - The file must begin with **oc-** or **kubectl-** to be recognized as a plugin.
   - The file name determines the command that invokes the plugin. For example, a plugin with the file name **oc-foo-bar** can be invoked by a command of **oc foo bar**. You can also use underscores if you want the command to contain dashes. For example, a plugin with the file name **oc-foo_bar** can be invoked by a command of **oc foo-bar**.
2. Add the following contents to the file.

```
#!/bin/bash

# optional argument handling
if [[ "$1" == "version" ]]
then
    echo "1.0.0"
    exit 0
fi

# optional argument handling
if [[ "$1" == "config" ]]
then
    echo $KUBECONFIG
    exit 0
fi

echo "I am a plugin named kubectl-foo"
```

After you install this plugin for the OpenShift CLI, it can be invoked using the **oc foo** command.

Additional resources

- Review the Sample plugin repository for an example of a plugin written in Go.
- Review the CLI runtime repository for a set of utilities to assist in writing plugins in Go.

2.5.2. Installing and using CLI plugins

After you write a custom plugin for the OpenShift CLI, you must install the plugin before use.

Prerequisites

- You must have the **oc** CLI tool installed.
- You must have a CLI plugin file that begins with **oc-** or **kubectl-**.

Procedure

1. If necessary, update the plugin file to be executable.

```
$ chmod +x <plugin_file>
```

2. Place the file anywhere in your **PATH**, such as **/usr/local/bin/**.

```
$ sudo mv <plugin_file> /usr/local/bin/.
```

3. Run **oc plugin list** to make sure that the plugin is listed.

```
$ oc plugin list
```

Example output

```
The following compatible plugins are available:

/usr/local/bin/<plugin_file>
```

If your plugin is not listed here, verify that the file begins with **oc-** or **kubectl-**, is executable, and is on your **PATH**.

4. Invoke the new command or option introduced by the plugin.
For example, if you built and installed the **kubectl-ns** plugin from the Sample plugin repository, you can use the following command to view the current namespace.

```
$ oc ns
```

Note that the command to invoke the plugin depends on the plugin file name. For example, a plugin with the file name of **oc-foo-bar** is invoked by the **oc foo bar** command.

2.6. OPENSHIFT CLI DEVELOPER COMMAND REFERENCE

This reference provides descriptions and example commands for OpenShift CLI (**oc**) developer commands.

Run **oc help** to list all commands or run **oc <command> --help** to get additional details for a specific command.

2.6.1. OpenShift CLI (oc) developer commands

2.6.1.1. oc annotate

Update the annotations on a resource

Example usage

-

```
# Update pod 'foo' with the annotation 'description' and the value 'my frontend'
# If the same annotation is set multiple times, only the last value will be applied
oc annotate pods foo description='my frontend'

# Update a pod identified by type and name in "pod.json"
oc annotate -f pod.json description='my frontend'

# Update pod 'foo' with the annotation 'description' and the value 'my frontend running nginx',
overwriting any existing value
oc annotate --overwrite pods foo description='my frontend running nginx'

# Update all pods in the namespace
oc annotate pods --all description='my frontend running nginx'

# Update pod 'foo' only if the resource is unchanged from version 1
oc annotate pods foo description='my frontend running nginx' --resource-version=1

# Update pod 'foo' by removing an annotation named 'description' if it exists
# Does not require the --overwrite flag
oc annotate pods foo description-
```

2.6.1.2. oc api-resources

Print the supported API resources on the server

Example usage

```
# Print the supported API resources
oc api-resources

# Print the supported API resources with more information
oc api-resources -o wide

# Print the supported API resources sorted by a column
oc api-resources --sort-by=name

# Print the supported namespaced resources
oc api-resources --namespaced=true

# Print the supported non-namespaced resources
oc api-resources --namespaced=false

# Print the supported API resources with a specific APIGroup
oc api-resources --api-group=rbac.authorization.k8s.io
```

2.6.1.3. oc api-versions

Print the supported API versions on the server, in the form of "group/version"

Example usage

```
# Print the supported API versions
oc api-versions
```

2.6.1.4. oc apply

Apply a configuration to a resource by file name or stdin

Example usage

```
# Apply the configuration in pod.json to a pod
oc apply -f ./pod.json

# Apply resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
oc apply -k dir/

# Apply the JSON passed into stdin to a pod
cat pod.json | oc apply -f -

# Apply the configuration from all files that end with '.json'
oc apply -f '*.json'

# Note: --prune is still in Alpha
# Apply the configuration in manifest.yaml that matches label app=nginx and delete all other
resources that are not in the file and match label app=nginx
oc apply --prune -f manifest.yaml -l app=nginx

# Apply the configuration in manifest.yaml and delete all the other config maps that are not in the file
oc apply --prune -f manifest.yaml --all --prune-allowlist=core/v1/ConfigMap
```

2.6.1.5. oc apply edit-last-applied

Edit latest last-applied-configuration annotations of a resource/object

Example usage

```
# Edit the last-applied-configuration annotations by type/name in YAML
oc apply edit-last-applied deployment/nginx

# Edit the last-applied-configuration annotations by file in JSON
oc apply edit-last-applied -f deploy.yaml -o json
```

2.6.1.6. oc apply set-last-applied

Set the last-applied-configuration annotation on a live object to match the contents of a file

Example usage

```
# Set the last-applied-configuration of a resource to match the contents of a file
oc apply set-last-applied -f deploy.yaml

# Execute set-last-applied against each configuration file in a directory
oc apply set-last-applied -f path/

# Set the last-applied-configuration of a resource to match the contents of a file; will create the
annotation if it does not already exist
oc apply set-last-applied -f deploy.yaml --create-annotation=true
```

2.6.1.7. oc apply view-last-applied

View the latest last-applied-configuration annotations of a resource/object

Example usage

```
# View the last-applied-configuration annotations by type/name in YAML
oc apply view-last-applied deployment/nginx

# View the last-applied-configuration annotations by file in JSON
oc apply view-last-applied -f deploy.yaml -o json
```

2.6.1.8. oc attach

Attach to a running container

Example usage

```
# Get output from running pod mypod; use the 'oc.kubernetes.io/default-container' annotation
# for selecting the container to be attached or the first container in the pod will be chosen
oc attach mypod

# Get output from ruby-container from pod mypod
oc attach mypod -c ruby-container

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc attach mypod -c ruby-container -i -t

# Get output from the first pod of a replica set named nginx
oc attach rs/nginx
```

2.6.1.9. oc auth can-i

Check whether an action is allowed

Example usage

```
# Check to see if I can create pods in any namespace
oc auth can-i create pods --all-namespaces

# Check to see if I can list deployments in my current namespace
oc auth can-i list deployments.apps

# Check to see if service account "foo" of namespace "dev" can list pods in the namespace "prod"
# You must be allowed to use impersonation for the global option "--as"
oc auth can-i list pods --as=system:serviceaccount:dev:foo -n prod

# Check to see if I can do everything in my current namespace ("*" means all)
oc auth can-i '*' '*'

# Check to see if I can get the job named "bar" in namespace "foo"
oc auth can-i list jobs.batch/bar -n foo
```

```
# Check to see if I can read pod logs
oc auth can-i get pods --subresource=log

# Check to see if I can access the URL /logs/
oc auth can-i get /logs/

# Check to see if I can approve certificates.k8s.io
oc auth can-i approve certificates.k8s.io

# List all allowed actions in namespace "foo"
oc auth can-i --list --namespace=foo
```

2.6.1.10. oc auth reconcile

Reconciles rules for RBAC role, role binding, cluster role, and cluster role binding objects

Example usage

```
# Reconcile RBAC resources from a file
oc auth reconcile -f my-rbac-rules.yaml
```

2.6.1.11. oc auth whoami

Experimental: Check self subject attributes

Example usage

```
# Get your subject attributes
oc auth whoami

# Get your subject attributes in JSON format
oc auth whoami -o json
```

2.6.1.12. oc autoscale

Autoscale a deployment config, deployment, replica set, stateful set, or replication controller

Example usage

```
# Auto scale a deployment "foo", with the number of pods between 2 and 10, no target CPU
utilization specified so a default autoscaling policy will be used
oc autoscale deployment foo --min=2 --max=10

# Auto scale a replication controller "foo", with the number of pods between 1 and 5, target CPU
utilization at 80%
oc autoscale rc foo --max=5 --cpu=80%

# Auto scale a deployment "bar", with the number of pods between 3 and 6, target average CPU of
500m and memory of 200Mi
oc autoscale deployment bar --min=3 --max=6 --cpu=500m --memory=200Mi

# Auto scale a deployment "bar", with the number of pods between 2 and 8, target CPU utilization
60% and memory utilization 70%
oc autoscale deployment bar --min=3 --max=6 --cpu=60% --memory=70%
```

#

2.6.1.13. oc cancel-build

Cancel running, pending, or new builds

Example usage

```
# Cancel the build with the given name
oc cancel-build ruby-build-2

# Cancel the named build and print the build logs
oc cancel-build ruby-build-2 --dump-logs

# Cancel the named build and create a new one with the same parameters
oc cancel-build ruby-build-2 --restart

# Cancel multiple builds
oc cancel-build ruby-build-1 ruby-build-2 ruby-build-3

# Cancel all builds created from the 'ruby-build' build config that are in the 'new' state
oc cancel-build bc/ruby-build --state=new
```

2.6.1.14. oc cluster-info

Display cluster information

Example usage

```
# Print the address of the control plane and cluster services
oc cluster-info
```

2.6.1.15. oc cluster-info dump

Dump relevant information for debugging and diagnosis

Example usage

```
# Dump current cluster state to stdout
oc cluster-info dump

# Dump current cluster state to /path/to/cluster-state
oc cluster-info dump --output-directory=/path/to/cluster-state

# Dump all namespaces to stdout
oc cluster-info dump --all-namespaces

# Dump a set of namespaces to /path/to/cluster-state
oc cluster-info dump --namespaces default,kube-system --output-directory=/path/to/cluster-state
```

2.6.1.16. oc completion

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

Example usage

```
# Installing bash completion on macOS using homebrew
## If running Bash 3.2 included with macOS
brew install bash-completion
## or, if running Bash 4.1+
brew install bash-completion@2
## If oc is installed via homebrew, this should start working immediately
## If you've installed via other means, you may need add the completion to your completion directory
oc completion bash > $(brew --prefix)/etc/bash_completion.d/oc


# Installing bash completion on Linux
## If bash-completion is not installed on Linux, install the 'bash-completion' package
## via your distribution's package manager.
## Load the oc completion code for bash into the current shell
source <(oc completion bash)
## Write bash completion code to a file and source it from .bash_profile
oc completion bash > ~/.kube/completion.bash.inc
printf "
# oc shell completion
source '$HOME/.kube/completion.bash.inc'
" >> $HOME/.bash_profile
source $HOME/.bash_profile

# Load the oc completion code for zsh[1] into the current shell
source <(oc completion zsh)
# Set the oc completion code for zsh[1] to autoload on startup
oc completion zsh > "${fpath[1]}/_oc"


# Load the oc completion code for fish[2] into the current shell
oc completion fish | source
# To load completions for each session, execute once:
oc completion fish > ~/.config/fish/completions/oc.fish

# Load the oc completion code for powershell into the current shell
oc completion powershell | Out-String | Invoke-Expression
# Set oc completion code for powershell to run on startup
## Save completion code to a script and execute in the profile
oc completion powershell > $HOME\.kube\completion.ps1
Add-Content $PROFILE "$HOME\.kube\completion.ps1"
## Execute completion code in the profile
Add-Content $PROFILE "if (Get-Command oc -ErrorAction SilentlyContinue) {
oc completion powershell | Out-String | Invoke-Expression
}"
## Add completion code directly to the $PROFILE script
oc completion powershell >> $PROFILE
```

2.6.1.17. oc config current-context

Display the current-context

Example usage

```
# Display the current-context
oc config current-context
```

2.6.1.18. oc config delete-cluster

Delete the specified cluster from the kubeconfig

Example usage

```
# Delete the minikube cluster
oc config delete-cluster minikube
```

2.6.1.19. oc config delete-context

Delete the specified context from the kubeconfig

Example usage

```
# Delete the context for the minikube cluster
oc config delete-context minikube
```

2.6.1.20. oc config delete-user

Delete the specified user from the kubeconfig

Example usage

```
# Delete the minikube user
oc config delete-user minikube
```

2.6.1.21. oc config get-clusters

Display clusters defined in the kubeconfig

Example usage

```
# List the clusters that oc knows about
oc config get-clusters
```

2.6.1.22. oc config get-contexts

Describe one or many contexts

Example usage

```
# List all the contexts in your kubeconfig file
oc config get-contexts

# Describe one context in your kubeconfig file
oc config get-contexts my-context
```

2.6.1.23. oc config get-users

Display users defined in the kubeconfig

Example usage

```
# List the users that oc knows about
oc config get-users
```

2.6.1.24. oc config new-admin-kubeconfig

Generate, make the server trust, and display a new admin.kubeconfig

Example usage

```
# Generate a new admin kubeconfig
oc config new-admin-kubeconfig
```

2.6.1.25. oc config new-kubelet-bootstrap-kubeconfig

Generate, make the server trust, and display a new kubelet /etc/kubernetes/kubeconfig

Example usage

```
# Generate a new kubelet bootstrap kubeconfig
oc config new-kubelet-bootstrap-kubeconfig
```

2.6.1.26. oc config refresh-ca-bundle

Update the OpenShift CA bundle by contacting the API server

Example usage

```
# Refresh the CA bundle for the current context's cluster
oc config refresh-ca-bundle

# Refresh the CA bundle for the cluster named e2e in your kubeconfig
oc config refresh-ca-bundle e2e

# Print the CA bundle from the current OpenShift cluster's API server
oc config refresh-ca-bundle --dry-run
```

2.6.1.27. oc config rename-context

Rename a context from the kubeconfig file

Example usage

```
# Rename the context 'old-name' to 'new-name' in your kubeconfig file
oc config rename-context old-name new-name
```

2.6.1.28. oc config set

Set an individual value in a kubeconfig file

Example usage

```
# Set the server field on the my-cluster cluster to https://1.2.3.4
oc config set clusters.my-cluster.server https://1.2.3.4

# Set the certificate-authority-data field on the my-cluster cluster
oc config set clusters.my-cluster.certificate-authority-data $(echo "cert_data_here" | base64 -i -)

# Set the cluster field in the my-context context to my-cluster
oc config set contexts.my-context.cluster my-cluster

# Set the client-key-data field in the cluster-admin user using --set-raw-bytes option
oc config set users.cluster-admin.client-key-data cert_data_here --set-raw-bytes=true
```

2.6.1.29. oc config set-cluster

Set a cluster entry in kubeconfig

Example usage

```
# Set only the server field on the e2e cluster entry without touching other values
oc config set-cluster e2e --server=https://1.2.3.4

# Embed certificate authority data for the e2e cluster entry
oc config set-cluster e2e --embed-certs --certificate-authority=~/.kube/e2e/kubernetes.ca.crt

# Disable cert checking for the e2e cluster entry
oc config set-cluster e2e --insecure-skip-tls-verify=true

# Set the custom TLS server name to use for validation for the e2e cluster entry
oc config set-cluster e2e --tls-server-name=my-cluster-name

# Set the proxy URL for the e2e cluster entry
oc config set-cluster e2e --proxy-url=https://1.2.3.4
```

2.6.1.30. oc config set-context

Set a context entry in kubeconfig

Example usage

```
# Set the user field on the gce context entry without touching other values
oc config set-context gce --user=cluster-admin
```

2.6.1.31. oc config set-credentials

Set a user entry in kubeconfig

Example usage

-

```
# Set only the "client-key" field on the "cluster-admin"
# entry, without touching other values
oc config set-credentials cluster-admin --client-key=~/.kube/admin.key

# Set basic auth for the "cluster-admin" entry
oc config set-credentials cluster-admin --username=admin --password=uXFGweU9l35qcif

# Embed client certificate data in the "cluster-admin" entry
oc config set-credentials cluster-admin --client-certificate=~/.kube/admin.crt --embed-certs=true

# Enable the Google Compute Platform auth provider for the "cluster-admin" entry
oc config set-credentials cluster-admin --auth-provider=gcp

# Enable the OpenID Connect auth provider for the "cluster-admin" entry with additional arguments
oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-id=foo --auth-
provider-arg=client-secret=bar

# Remove the "client-secret" config value for the OpenID Connect auth provider for the "cluster-
admin" entry
oc config set-credentials cluster-admin --auth-provider=oidc --auth-provider-arg=client-secret-

# Enable new exec auth plugin for the "cluster-admin" entry
oc config set-credentials cluster-admin --exec-command=/path/to/the/executable --exec-api-
version=client.authentication.k8s.io/v1beta1

# Enable new exec auth plugin for the "cluster-admin" entry with interactive mode
oc config set-credentials cluster-admin --exec-command=/path/to/the/executable --exec-api-
version=client.authentication.k8s.io/v1beta1 --exec-interactive-mode=Never

# Define new exec auth plugin arguments for the "cluster-admin" entry
oc config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

# Create or update exec auth plugin environment variables for the "cluster-admin" entry
oc config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=key2=val2

# Remove exec auth plugin environment variables for the "cluster-admin" entry
oc config set-credentials cluster-admin --exec-env=var-to-remove-
```

2.6.1.32. oc config unset

Unset an individual value in a kubeconfig file

Example usage

```
# Unset the current-context
oc config unset current-context

# Unset namespace in foo context
oc config unset contexts.foo.namespace
```

2.6.1.33. oc config use-context

Set the current-context in a kubeconfig file

Example usage

```
# Use the context for the minikube cluster
oc config use-context minikube
```

2.6.1.34. oc config view

Display merged kubeconfig settings or a specified kubeconfig file

Example usage

```
# Show merged kubeconfig settings
oc config view

# Show merged kubeconfig settings, raw certificate data, and exposed secrets
oc config view --raw

# Get the password for the e2e user
oc config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

2.6.1.35. oc cp

Copy files and directories to and from containers

Example usage

```
# !!!Important Note!!!
# Requires that the 'tar' binary is present in your container
# image.  If 'tar' is not present, 'oc cp' will fail.
#
# For advanced use cases, such as symlinks, wildcard expansion or
# file mode preservation, consider using 'oc exec'.

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
tar cf - /tmp/foo | oc exec -i -n <some-namespace> <some-pod> -- tar xf - -C /tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc exec -n <some-namespace> <some-pod> -- tar cf - /tmp/foo | tar xf - -C /tmp/bar

# Copy /tmp/foo_dir local directory to /tmp/bar_dir in a remote pod in the default namespace
oc cp /tmp/foo_dir <some-pod>:/tmp/bar_dir

# Copy /tmp/foo local file to /tmp/bar in a remote pod in a specific container
oc cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some-namespace>
oc cp /tmp/foo <some-namespace>/<some-pod>:/tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
oc cp <some-namespace>/<some-pod>:/tmp/foo /tmp/bar
```

2.6.1.36. oc create

Create a resource from a file or from stdin

Example usage

```
# Create a pod using the data in pod.json
oc create -f ./pod.json

# Create a pod based on the JSON passed into stdin
cat pod.json | oc create -f -

# Edit the data in registry.yaml in JSON then create the resource using the edited data
oc create -f registry.yaml --edit -o json
```

2.6.1.37. oc create build

Create a new build

Example usage

```
# Create a new build
oc create build myapp
```

2.6.1.38. oc create clusterresourcequota

Create a cluster resource quota

Example usage

```
# Create a cluster resource quota limited to 10 pods
oc create clusterresourcequota limit-bob --project-annotation-selector=openshift.io/requester=user-bob --hard=pods=10
```

2.6.1.39. oc create clusterrole

Create a cluster role

Example usage

```
# Create a cluster role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
oc create clusterrole pod-reader --verb=get,list,watch --resource=pods

# Create a cluster role named "pod-reader" with ResourceName specified
oc create clusterrole pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-name=anotherpod

# Create a cluster role named "foo" with API Group specified
oc create clusterrole foo --verb=get,list,watch --resource=rs.apps

# Create a cluster role named "foo" with SubResource specified
oc create clusterrole foo --verb=get,list,watch --resource=pods,pods/status

# Create a cluster role name "foo" with NonResourceURL specified
```

```
oc create clusterrole "foo" --verb=get --non-resource-url=/logs/*

# Create a cluster role name "monitoring" with AggregationRule specified
oc create clusterrole monitoring --aggregation-rule="rbac.example.com/aggregate-to-monitoring=true"
```

2.6.1.40. oc create clusterrolebinding

Create a cluster role binding for a particular cluster role

Example usage

```
# Create a cluster role binding for user1, user2, and group1 using the cluster-admin cluster role
oc create clusterrolebinding cluster-admin --clusterrole=cluster-admin --user=user1 --user=user2 --group=group1
```

2.6.1.41. oc create configmap

Create a config map from a local file, directory or literal value

Example usage

```
# Create a new config map named my-config based on folder bar
oc create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config with specified keys instead of file basenames on disk
oc create configmap my-config --from-file=key1=/path/to/bar/file1.txt --from-file=key2=/path/to/bar/file2.txt

# Create a new config map named my-config with key1=config1 and key2=config2
oc create configmap my-config --from-literal=key1=config1 --from-literal=key2=config2

# Create a new config map named my-config from the key=value pairs in the file
oc create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config from an env file
oc create configmap my-config --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

2.6.1.42. oc create cronjob

Create a cron job with the specified name

Example usage

```
# Create a cron job
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *"

# Create a cron job with a command
oc create cronjob my-job --image=busybox --schedule="*/1 * * * *" -- date
```

2.6.1.43. oc create deployment

Create a deployment with the specified name

Example usage

```
# Create a deployment named my-dep that runs the busybox image
oc create deployment my-dep --image=busybox

# Create a deployment with a command
oc create deployment my-dep --image=busybox -- date

# Create a deployment named my-dep that runs the nginx image with 3 replicas
oc create deployment my-dep --image=nginx --replicas=3

# Create a deployment named my-dep that runs the busybox image and expose port 5701
oc create deployment my-dep --image=busybox --port=5701

# Create a deployment named my-dep that runs multiple containers
oc create deployment my-dep --image=busybox:latest --image=ubuntu:latest --image=nginx
```

2.6.1.44. oc create deploymentconfig

Create a deployment config with default options that uses a given image

Example usage

```
# Create an nginx deployment config named my-nginx
oc create deploymentconfig my-nginx --image=nginx
```

2.6.1.45. oc create identity

Manually create an identity (only needed if automatic creation is disabled)

Example usage

```
# Create an identity with identity provider "acme_ldap" and the identity provider username
"adamjones"
oc create identity acme_ldap:adamjones
```

2.6.1.46. oc create imagestream

Create a new empty image stream

Example usage

```
# Create a new image stream
oc create imagestream mysql
```

2.6.1.47. oc create imagestreamtag

Create a new image stream tag

Example usage

```
# Create a new image stream tag based on an image in a remote registry
oc create imagestreamtag mysql:latest --from-image=myregistry.local/mysql/mysql:5.0
```

2.6.1.48. oc create ingress

Create an ingress with the specified name

Example usage

```
# Create a single ingress called 'simple' that directs requests to foo.com/bar to svc
# svc1:8080 with a TLS secret "my-cert"
oc create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"

# Create a catch all ingress of "/path" pointing to service svc:port and Ingress Class as
"otheringress"
oc create ingress catch-all --class=otheringress --rule="/path=svc:port"

# Create an ingress with two annotations: ingress.annotation1 and ingress.annotations2
oc create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla

# Create an ingress with the same host and multiple paths
oc create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
--rule="foo.com/admin/=svcadmin:portadmin"

# Create an ingress with multiple hosts and the pathType as Prefix
oc create ingress ingress1 --class=default \
--rule="foo.com/path*=svc:8080" \
--rule="bar.com/admin*=svc2:http"

# Create an ingress with TLS enabled using the default ingress certificate and different path types
oc create ingress ingtls --class=default \
--rule="foo.com/=svc:https,tls" \
--rule="foo.com/path/subpath*=othersvc:8080"

# Create an ingress with TLS enabled using a specific secret and pathType as Prefix
oc create ingress ingsecret --class=default \
--rule="foo.com/*=svc:8080,tls=secret1"

# Create an ingress with a default backend
oc create ingress ingdefault --class=default \
--default-backend=defaultsvc:http \
--rule="foo.com/*=svc:8080,tls=secret1"
```

2.6.1.49. oc create job

Create a job with the specified name

Example usage

```
# Create a job
oc create job my-job --image=busybox
```

```
# Create a job with a command
oc create job my-job --image=busybox -- date

# Create a job from a cron job named "a-cronjob"
oc create job test-job --from=cronjob/a-cronjob
```

2.6.1.50. oc create namespace

Create a namespace with the specified name

Example usage

```
# Create a new namespace named my-namespace
oc create namespace my-namespace
```

2.6.1.51. oc create poddisruptionbudget

Create a pod disruption budget with the specified name

Example usage

```
# Create a pod disruption budget named my-pdb that will select all pods with the app=rails label
# and require at least one of them being available at any point in time
oc create poddisruptionbudget my-pdb --selector=app=rails --min-available=1

# Create a pod disruption budget named my-pdb that will select all pods with the app=nginx label
# and require at least half of the pods selected to be available at any point in time
oc create pdb my-pdb --selector=app=nginx --min-available=50%
```

2.6.1.52. oc create priorityclass

Create a priority class with the specified name

Example usage

```
# Create a priority class named high-priority
oc create priorityclass high-priority --value=1000 --description="high priority"

# Create a priority class named default-priority that is considered as the global default priority
oc create priorityclass default-priority --value=1000 --global-default=true --description="default
priority"

# Create a priority class named high-priority that cannot preempt pods with lower priority
oc create priorityclass high-priority --value=1000 --description="high priority" --preemption-
policy="Never"
```

2.6.1.53. oc create quota

Create a quota with the specified name

Example usage

```
# Create a new resource quota named my-quota
oc create quota my-quota --
hard=cpu=1,memory=1G,pods=2,services=3,replicationcontrollers=2,resourcequotas=1,secrets=5,persi
tentvolumeclaims=10

# Create a new resource quota named best-effort
oc create quota best-effort --hard=pods=100 --scopes=BestEffort
```

2.6.1.54. oc create role

Create a role with single rule

Example usage

```
# Create a role named "pod-reader" that allows user to perform "get", "watch" and "list" on pods
oc create role pod-reader --verb=get --verb=list --verb=watch --resource=pods

# Create a role named "pod-reader" with ResourceName specified
oc create role pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-
name=anotherpod

# Create a role named "foo" with API Group specified
oc create role foo --verb=get,list,watch --resource=rs.apps

# Create a role named "foo" with SubResource specified
oc create role foo --verb=get,list,watch --resource=pods,pods/status
```

2.6.1.55. oc create rolebinding

Create a role binding for a particular role or cluster role

Example usage

```
# Create a role binding for user1, user2, and group1 using the admin cluster role
oc create rolebinding admin --clusterrole=admin --user=user1 --user=user2 --group=group1

# Create a role binding for service account monitoring:sa-dev using the admin role
oc create rolebinding admin-binding --role=admin --serviceaccount=monitoring:sa-dev
```

2.6.1.56. oc create route edge

Create a route that uses edge TLS termination

Example usage

```
# Create an edge route named "my-route" that exposes the frontend service
oc create route edge my-route --service=frontend

# Create an edge route that exposes the frontend service and specify a path
# If the route name is omitted, the service name will be used
oc create route edge --service=frontend --path /assets
```

2.6.1.57. oc create route passthrough

Create a route that uses passthrough TLS termination

Example usage

```
# Create a passthrough route named "my-route" that exposes the frontend service
oc create route passthrough my-route --service=frontend

# Create a passthrough route that exposes the frontend service and specify
# a host name. If the route name is omitted, the service name will be used
oc create route passthrough --service=frontend --hostname=www.example.com
```

2.6.1.58. oc create route reencrypt

Create a route that uses reencrypt TLS termination

Example usage

```
# Create a route named "my-route" that exposes the frontend service
oc create route reencrypt my-route --service=frontend --dest-ca-cert cert.cert

# Create a reencrypt route that exposes the frontend service, letting the
# route name default to the service name and the destination CA certificate
# default to the service CA
oc create route reencrypt --service=frontend
```

2.6.1.59. oc create secret docker-registry

Create a secret for use with a Docker registry

Example usage

```
# If you do not already have a .dockercfg file, create a dockercfg secret directly
oc create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY_SERVER --
docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-
email=DOCKER_EMAIL

# Create a new secret named my-secret from ~/.docker/config.json
oc create secret docker-registry my-secret --from-file=path/to/.docker/config.json
```

2.6.1.60. oc create secret generic

Create a secret from a local file, directory, or literal value

Example usage

```
# Create a new secret named my-secret with keys for each file in folder bar
oc create secret generic my-secret --from-file=path/to/bar

# Create a new secret named my-secret with specified keys instead of names on disk
oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-file=ssh-
publickey=path/to/id_rsa.pub

# Create a new secret named my-secret with key1=supersecret and key2=topsecret
```

```
oc create secret generic my-secret --from-literal=key1=supersecret --from-literal=key2=topsecret

# Create a new secret named my-secret using a combination of a file and a literal
oc create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-literal=passphrase=topsecret

# Create a new secret named my-secret from env files
oc create secret generic my-secret --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

2.6.1.61. oc create secret tls

Create a TLS secret

Example usage

```
# Create a new TLS secret named tls-secret with the given key pair
oc create secret tls tls-secret --cert=path/to/tls.crt --key=path/to/tls.key
```

2.6.1.62. oc create service clusterip

Create a ClusterIP service

Example usage

```
# Create a new ClusterIP service named my-cs
oc create service clusterip my-cs --tcp=5678:8080

# Create a new ClusterIP service named my-cs (in headless mode)
oc create service clusterip my-cs --clusterip="None"
```

2.6.1.63. oc create service externalname

Create an ExternalName service

Example usage

```
# Create a new ExternalName service named my-ns
oc create service externalname my-ns --external-name bar.com
```

2.6.1.64. oc create service loadbalancer

Create a LoadBalancer service

Example usage

```
# Create a new LoadBalancer service named my-lbs
oc create service loadbalancer my-lbs --tcp=5678:8080
```

2.6.1.65. oc create service nodeport

Create a NodePort service

Example usage

```
# Create a new NodePort service named my-ns
oc create service nodeport my-ns --tcp=5678:8080
```

2.6.1.66. oc create serviceaccount

Create a service account with the specified name

Example usage

```
# Create a new service account named my-service-account
oc create serviceaccount my-service-account
```

2.6.1.67. oc create token

Request a service account token

Example usage

```
# Request a token to authenticate to the kube-apiserver as the service account "myapp" in the current namespace
oc create token myapp

# Request a token for a service account in a custom namespace
oc create token myapp --namespace myns

# Request a token with a custom expiration
oc create token myapp --duration 10m

# Request a token with a custom audience
oc create token myapp --audience https://example.com

# Request a token bound to an instance of a Secret object
oc create token myapp --bound-object-kind Secret --bound-object-name mysecret

# Request a token bound to an instance of a Secret object with a specific UID
oc create token myapp --bound-object-kind Secret --bound-object-name mysecret --bound-object-uid 0d4691ed-659b-4935-a832-355f77ee47cc
```

2.6.1.68. oc create user

Manually create a user (only needed if automatic creation is disabled)

Example usage

```
# Create a user with the username "ajones" and the display name "Adam Jones"
oc create user ajones --full-name="Adam Jones"
```

2.6.1.69. oc create useridentitymapping

Manually map an identity to a user

Example usage

```
# Map the identity "acme_ldap:adamjones" to the user "ajones"
oc create useridentitymapping acme_ldap:adamjones ajones
```

2.6.1.70. oc debug

Launch a new instance of a pod for debugging

Example usage

```
# Start a shell session into a pod using the OpenShift tools image
oc debug

# Debug a currently running deployment by creating a new pod
oc debug deploy/test

# Debug a node as an administrator
oc debug node/master-1

# Debug a Windows node
# Note: the chosen image must match the Windows Server version (2019, 2022) of the node
oc debug node/win-worker-1 --image=mcr.microsoft.com/powershell:lts-nanoserver-ltsc2022

# Launch a shell in a pod using the provided image stream tag
oc debug istag/mysql:latest -n openshift

# Test running a job as a non-root user
oc debug job/test --as-user=1000000

# Debug a specific failing container by running the env command in the 'second' container
oc debug daemonset/test -c second -- /bin/env

# See the pod that would be created to debug
oc debug mypod-9xbc -o yaml

# Debug a resource but launch the debug pod in another namespace
# Note: Not all resources can be debugged using --to-namespace without modification. For
example,
# volumes and service accounts are namespace-dependent. Add '-o yaml' to output the debug pod
definition
# to disk.  If necessary, edit the definition then run 'oc debug -f -' or run without --to-namespace
oc debug mypod-9xbc --to-namespace testns
```

2.6.1.71. oc delete

Delete resources by file names, stdin, resources and names, or by resources and label selector

Example usage

```
# Delete a pod using the type and name specified in pod.json
oc delete -f ./pod.json

# Delete resources from a directory containing kustomization.yaml - e.g. dir/kustomization.yaml
```

```
oc delete -k dir

# Delete resources from all files that end with '.json'
oc delete -f '*.json'

# Delete a pod based on the type and name in the JSON passed into stdin
cat pod.json | oc delete -f -

# Delete pods and services with same names "baz" and "foo"
oc delete pod,service baz foo

# Delete pods and services with label name=myLabel
oc delete pods,services -l name=myLabel

# Delete a pod with minimal delay
oc delete pod foo --now

# Force delete a pod on a dead node
oc delete pod foo --force

# Delete all pods
oc delete pods --all

# Delete all pods only if the user confirms the deletion
oc delete pods --all --interactive
```

2.6.1.72. oc describe

Show details of a specific resource or group of resources

Example usage

```
# Describe a node
oc describe nodes kubernetes-node-emt8.c.myproject.internal

# Describe a pod
oc describe pods/nginx

# Describe a pod identified by type and name in "pod.json"
oc describe -f pod.json

# Describe all pods
oc describe pods

# Describe pods by label name=myLabel
oc describe pods -l name=myLabel

# Describe all pods managed by the 'frontend' replication controller
# (rc-created pods get the name of the rc as a prefix in the pod name)
oc describe pods frontend
```

2.6.1.73. oc diff

Diff the live version against a would-be applied version

Example usage

```
# Diff resources included in pod.json
oc diff -f pod.json

# Diff file read from stdin
cat service.yaml | oc diff -f -
```

2.6.1.74. oc edit

Edit a resource on the server

Example usage

```
# Edit the service named 'registry'
oc edit svc/registry

# Use an alternative editor
KUBE_EDITOR="nano" oc edit svc/registry

# Edit the job 'myjob' in JSON using the v1 API format
oc edit job.v1.batch/myjob -o json

# Edit the deployment 'mydeployment' in YAML and save the modified config in its annotation
oc edit deployment/mydeployment -o yaml --save-config

# Edit the 'status' subresource for the 'mydeployment' deployment
oc edit deployment mydeployment --subresource='status'
```

2.6.1.75. oc events

List events

Example usage

```
# List recent events in the default namespace
oc events

# List recent events in all namespaces
oc events --all-namespaces

# List recent events for the specified pod, then wait for more events and list them as they arrive
oc events --for pod/web-pod-13je7 --watch

# List recent events in YAML format
oc events -oyaml

# List recent only events of type 'Warning' or 'Normal'
oc events --types=Warning,Normal
```

2.6.1.76. oc exec

Execute a command in a container

Example usage

```
# Get output from running the 'date' command from pod mypod, using the first container by default
oc exec mypod -- date

# Get output from running the 'date' command in ruby-container from pod mypod
oc exec mypod -c ruby-container -- date

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container from pod mypod
# and sends stdout/stderr from 'bash' back to the client
oc exec mypod -c ruby-container -i -t -- bash -il

# List contents of /usr from the first container of pod mypod and sort by modification time
# If the command you want to execute in the pod has any flags in common (e.g. -i),
# you must use two dashes (--) to separate your command's flags/arguments
# Also note, do not surround your command and its flags/arguments with quotes
# unless that is how you would execute it normally (i.e., do ls -t /usr, not "ls -t /usr")
oc exec mypod -i -t -- ls -t /usr

# Get output from running 'date' command from the first pod of the deployment mydeployment,
using the first container by default
oc exec deploy/mydeployment -- date

# Get output from running 'date' command from the first pod of the service myservice, using the first
container by default
oc exec svc/myservice -- date
```

2.6.1.77. oc explain

Get documentation for a resource

Example usage

```
# Get the documentation of the resource and its fields
oc explain pods

# Get all the fields in the resource
oc explain pods --recursive

# Get the explanation for deployment in supported api versions
oc explain deployments --api-version=apps/v1

# Get the documentation of a specific field of a resource
oc explain pods.spec.containers

# Get the documentation of resources in different format
oc explain deployment --output=plaintext-openapiv2
```

2.6.1.78. oc expose

Expose a replicated application as a service or route

Example usage

```
# Create a route based on service nginx. The new route will reuse nginx's labels
oc expose service nginx

# Create a route and specify your own label and route name
oc expose service nginx -l name=myroute --name=fromdowntown

# Create a route and specify a host name
oc expose service nginx --hostname=www.example.com

# Create a route with a wildcard
oc expose service nginx --hostname=x.example.com --wildcard-policy=Subdomain
# This would be equivalent to *.example.com. NOTE: only hosts are matched by the wildcard;
subdomains would not be included

# Expose a deployment configuration as a service and use the specified port
oc expose dc ruby-hello-world --port=8080

# Expose a service as a route in the specified path
oc expose service nginx --path=/nginx
```

2.6.1.79. oc extract

Extract secrets or config maps to disk

Example usage

```
# Extract the secret "test" to the current directory
oc extract secret/test

# Extract the config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp

# Extract the config map "nginx" to STDOUT
oc extract configmap/nginx --to=-

# Extract only the key "nginx.conf" from config map "nginx" to the /tmp directory
oc extract configmap/nginx --to=/tmp --keys=nginx.conf
```

2.6.1.80. oc get

Display one or many resources

Example usage

```
# List all pods in ps output format
oc get pods

# List all pods in ps output format with more information (such as node name)
oc get pods -o wide

# List a single replication controller with specified NAME in ps output format
oc get replicationcontroller web

# List deployments in JSON output format, in the "v1" version of the "apps" API group
```

```
oc get deployments.v1.apps -o json

# List a single pod in JSON output format
oc get -o json pod web-pod-13je7

# List a pod identified by type and name specified in "pod.yaml" in JSON output format
oc get -f pod.yaml -o json

# List resources from a directory with kustomization.yaml - e.g. dir/kustomization.yaml
oc get -k dir/

# Return only the phase value of the specified pod
oc get -o template pod/web-pod-13je7 --template={{.status.phase}}

# List resource information in custom columns
oc get pod test-pod -o custom-
columns=CONTAINER:.spec.containers[0].name,IMAGE:.spec.containers[0].image

# List all replication controllers and services together in ps output format
oc get rc,services

# List one or more resources by their type and names
oc get rc/web service/frontend pods/web-pod-13je7

# List the 'status' subresource for a single pod
oc get pod web-pod-13je7 --subresource status

# List all deployments in namespace 'backend'
oc get deployments.apps --namespace backend

# List all pods existing in all namespaces
oc get pods --all-namespaces
```

2.6.1.81. oc get-token

Experimental: Get token from external OIDC issuer as credentials exec plugin

Example usage

```
# Starts an auth code flow to the issuer URL with the client ID and the given extra scopes
oc get-token --client-id=client-id --issuer-url=test.issuer.url --extra-scopes=email,profile

# Starts an auth code flow to the issuer URL with a different callback address
oc get-token --client-id=client-id --issuer-url=test.issuer.url --callback-address=127.0.0.1:8343
```

2.6.1.82. oc idle

Idle scalable resources

Example usage

```
# Idle the scalable controllers associated with the services listed in to-idle.txt
$ oc idle --resource-names-file to-idle.txt
```

2.6.1.83. oc image append

Add layers to images and push them to a registry

Example usage

```
# Remove the entrypoint on the mysql:latest image
oc image append --from mysql:latest --to myregistry.com/myimage:latest --image '{"Entrypoint":null}'

# Add a new layer to the image
oc image append --from mysql:latest --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to the image and store the result on disk
# This results in $(pwd)/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local layer.tar.gz

# Add a new layer to the image and store the result on disk in a designated directory
# This will result in $(pwd)/mysql-local/v2/mysql/blobs,manifests
oc image append --from mysql:latest --to file://mysql:local --dir mysql-local layer.tar.gz

# Add a new layer to an image that is stored on disk (~/mysql-local/v2/image exists)
oc image append --from-dir ~/mysql-local --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to an image that was mirrored to the current directory on disk ($(pwd)/v2/image
exists)
oc image append --from-dir v2 --to myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to a multi-architecture image for an os/arch that is different from the system's
os/arch
# Note: The first image in the manifest list that matches the filter will be returned when --keep-
manifest-list is not specified
oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --to
myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to a multi-architecture image for all the os/arch manifests when keep-manifest-list
is specified
oc image append --from docker.io/library/busybox:latest --keep-manifest-list --to
myregistry.com/myimage:latest layer.tar.gz

# Add a new layer to a multi-architecture image for all the os/arch manifests that is specified by the
filter, while preserving the manifestlist
oc image append --from docker.io/library/busybox:latest --filter-by-os=linux/s390x --keep-manifest-
list --to myregistry.com/myimage:latest layer.tar.gz
```

2.6.1.84. oc image extract

Copy files from an image to the file system

Example usage

```
# Extract the busybox image into the current directory
oc image extract docker.io/library/busybox:latest

# Extract the busybox image into a designated directory (must exist)
oc image extract docker.io/library/busybox:latest --path /:/tmp/busybox
```

```
# Extract the busybox image into the current directory for linux/s390x platform
# Note: Wildcard filter is not supported with extract; pass a single os/arch to extract
oc image extract docker.io/library/busybox:latest --filter-by-os=linux/s390x

# Extract a single file from the image into the current directory
oc image extract docker.io/library/centos:7 --path /bin/bash:.

# Extract all .repo files from the image's /etc/yum.repos.d/ folder into the current directory
oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:.

# Extract all .repo files from the image's /etc/yum.repos.d/ folder into a designated directory (must exist)
# This results in /tmp/yum.repos.d/*.repo on local system
oc image extract docker.io/library/centos:7 --path /etc/yum.repos.d/*.repo:/tmp/yum.repos.d

# Extract an image stored on disk into the current directory ($(pwd)/v2/busybox/blobs,manifests exists)
# --confirm is required because the current directory is not empty
oc image extract file://busybox:local --confirm

# Extract an image stored on disk in a directory other than $(pwd)/v2 into the current directory
# --confirm is required because the current directory is not empty ($(pwd)/busybox-mirror-dir/v2/busybox exists)
oc image extract file://busybox:local --dir busybox-mirror-dir --confirm

# Extract an image stored on disk in a directory other than $(pwd)/v2 into a designated directory (must exist)
oc image extract file://busybox:local --dir busybox-mirror-dir --path /:/tmp/busybox

# Extract the last layer in the image
oc image extract docker.io/library/centos:7[-1]

# Extract the first three layers of the image
oc image extract docker.io/library/centos:7[:3]

# Extract the last three layers of the image
oc image extract docker.io/library/centos:7[-3:]
```

2.6.1.85. oc image info

Display information about an image

Example usage

```
# Show information about an image
oc image info quay.io/openshift/cli:latest

# Show information about images matching a wildcard
oc image info quay.io/openshift/cli:4.*

# Show information about a file mirrored to disk under DIR
oc image info --dir=DIR file://library/busybox:latest
```

```
# Select which image from a multi-OS image to show
oc image info library/busybox:latest --filter-by-os=linux/arm64
```

2.6.1.86. oc image mirror

Mirror images from one repository to another

Example usage

```
# Copy image to another tag
oc image mirror myregistry.com/myimage:latest myregistry.com/myimage:stable

# Copy image to another registry
oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable

# Copy all tags starting with mysql to the destination repository
oc image mirror myregistry.com/myimage:mysql* docker.io/myrepository/myimage

# Copy image to disk, creating a directory structure that can be served as a registry
oc image mirror myregistry.com/myimage:latest file://myrepository/myimage:latest

# Copy image to S3 (pull from <bucket>.s3.amazonaws.com/image:latest)
oc image mirror myregistry.com/myimage:latest
s3://s3.amazonaws.com/<region>/<bucket>/image:latest

# Copy image to S3 without setting a tag (pull via @<digest>)
oc image mirror myregistry.com/myimage:latest s3://s3.amazonaws.com/<region>/<bucket>/image

# Copy image to multiple locations
oc image mirror myregistry.com/myimage:latest docker.io/myrepository/myimage:stable \
docker.io/myrepository/myimage:dev

# Copy multiple images
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
myregistry.com/myimage:new=myregistry.com/other:target

# Copy manifest list of a multi-architecture image, even if only a single image is found
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
--keep-manifest-list=true

# Copy specific os/arch manifest of a multi-architecture image
# Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
# Note that with multi-arch images, this results in a new manifest list digest that includes only the
filtered manifests
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
--filter-by-os=os/arch

# Copy all os/arch manifests of a multi-architecture image
# Run 'oc image info myregistry.com/myimage:latest' to see list of os/arch manifests that will be
mirrored
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
--keep-manifest-list=true

# Note the above command is equivalent to
```

```
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
--filter-by-os=.*

# Copy specific os/arch manifest of a multi-architecture image
# Run 'oc image info myregistry.com/myimage:latest' to see available os/arch for multi-arch images
# Note that the target registry may reject a manifest list if the platform specific images do not all exist
# You must use a registry with sparse registry support enabled
oc image mirror myregistry.com/myimage:latest=myregistry.com/other:test \
--filter-by-os=linux/386 \
--keep-manifest-list=true
```

2.6.1.87. oc import-image

Import images from a container image registry

Example usage

```
# Import tag latest into a new image stream
oc import-image mystream --from=registry.io/repo/image:latest --confirm

# Update imported data for tag latest in an already existing image stream
oc import-image mystream

# Update imported data for tag stable in an already existing image stream
oc import-image mystream:stable

# Update imported data for all tags in an existing image stream
oc import-image mystream --all

# Update imported data for a tag that points to a manifest list to include the full manifest list
oc import-image mystream --import-mode=PreserveOriginal

# Import all tags into a new image stream
oc import-image mystream --from=registry.io/repo/image --all --confirm

# Import all tags into a new image stream using a custom timeout
oc --request-timeout=5m import-image mystream --from=registry.io/repo/image --all --confirm
```

2.6.1.88. oc kustomize

Build a kustomization target from a directory or URL

Example usage

```
# Build the current working directory
oc kustomize

# Build some shared configuration directory
oc kustomize /home/config/production

# Build from github
oc kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/helloWorld?ref=v1.0.6
```

2.6.1.89. oc label

Update the labels on a resource

Example usage

```
# Update pod 'foo' with the label 'unhealthy' and the value 'true'
oc label pods foo unhealthy=true

# Update pod 'foo' with the label 'status' and the value 'unhealthy', overwriting any existing value
oc label --overwrite pods foo status=unhealthy

# Update all pods in the namespace
oc label pods --all status=unhealthy

# Update a pod identified by the type and name in "pod.json"
oc label -f pod.json status=unhealthy

# Update pod 'foo' only if the resource is unchanged from version 1
oc label pods foo status=unhealthy --resource-version=1

# Update pod 'foo' by removing a label named 'bar' if it exists
# Does not require the --overwrite flag
oc label pods foo bar-
```

2.6.1.90. oc login

Log in to a server

Example usage

```
# Log in interactively
oc login --username=myuser

# Log in to the given server with the given certificate authority file
oc login localhost:8443 --certificate-authority=/path/to/cert.crt

# Log in to the given server with the given credentials (will not prompt interactively)
oc login localhost:8443 --username=myuser --password=mypass

# Log in to the given server through a browser
oc login localhost:8443 --web --callback-port 8280

# Log in to the given server through a browser without opening it automatically (print URL only)
oc login localhost:8443 --web --auto-open-browser=false --callback-port 8280

# Log in to the external OIDC issuer through Auth Code + PKCE by starting a local server listening
on port 8080
oc login localhost:8443 --exec-plugin=oc-oidc --client-id=client-id --extra-scopes=email,profile --
callback-port=8080
```

2.6.1.91. oc logout

End the current server session

Example usage

```
# Log out
oc logout
```

2.6.1.92. oc logs

Print the logs for a container in a pod

Example usage

```
# Start streaming the logs of the most recent build of the openldap build config
oc logs -f bc/openldap

# Start streaming the logs of the latest deployment of the mysql deployment config
oc logs -f dc/mysql

# Get the logs of the first deployment for the mysql deployment config. Note that logs
# from older deployments may not exist either because the deployment was successful
# or due to deployment pruning or manual deletion of the deployment
oc logs --version=1 dc/mysql

# Return a snapshot of ruby-container logs from pod backend
oc logs backend -c ruby-container

# Start streaming of ruby-container logs from pod backend
oc logs -f pod/backend -c ruby-container
```

2.6.1.93. oc new-app

Create a new application

Example usage

```
# List all local templates and image streams that can be used to create an app
oc new-app --list

# Create an application based on the source code in the current git repository (with a public remote)
and a container image
oc new-app . --image=registry/repo/langimage

# Create an application myapp with Docker based build strategy expecting binary input
oc new-app  --strategy=docker --binary --name myapp

# Create a Ruby application based on the provided [image]~[source code] combination
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

# Use the public container registry MySQL image to create an app. Generated artifacts will be
labeled with db=mysql
oc new-app mysql MYSQL_USER=user MYSQL_PASSWORD=pass MYSQL_DATABASE=testdb -
l db=mysql

# Use a MySQL image in a private registry to create an app and override application artifacts'
names
oc new-app --image=myregistry.com/mycompany/mysql --name=private
```

```
# Use an image with the full manifest list to create an app and override application artifacts' names
oc new-app --image=myregistry.com/mycompany/image --name=private --import-
mode=PreserveOriginal

# Create an application from a remote repository using its beta4 branch
oc new-app https://github.com/openshift/ruby-hello-world#beta4

# Create an application based on a stored template, explicitly setting a parameter value
oc new-app --template=ruby-helloworld-sample --param=MYSQL_USER=admin

# Create an application from a remote repository and specify a context directory
oc new-app https://github.com/youruser/yourgitrepo --context-dir=src/build

# Create an application from a remote private repository and specify which existing secret to use
oc new-app https://github.com/youruser/yourgitrepo --source-secret=yoursecret

# Create an application based on a template file, explicitly setting a parameter value
oc new-app --file=./example/myapp/template.json --param=MYSQL_USER=admin

# Search all templates, image streams, and container images for the ones that match "ruby"
oc new-app --search ruby

# Search for "ruby", but only in stored templates (--template, --image-stream and --image
# can be used to filter search results)
oc new-app --search --template=ruby

# Search for "ruby" in stored templates and print the output as YAML
oc new-app --search --template=ruby --output=yaml
```

2.6.1.94. oc new-build

Create a new build configuration

Example usage

```
# Create a build config based on the source code in the current git repository (with a public
# remote) and a container image
oc new-build . --image=repo/langimage

# Create a NodeJS build config based on the provided [image]~[source code] combination
oc new-build centos/nodejs-8-centos7~https://github.com/sclorg/nodejs-ex.git

# Create a build config from a remote repository using its beta2 branch
oc new-build https://github.com/openshift/ruby-hello-world#beta2

# Create a build config using a Dockerfile specified as an argument
oc new-build -D $'FROM centos:7\nRUN yum install -y httpd'

# Create a build config from a remote repository and add custom environment variables
oc new-build https://github.com/openshift/ruby-hello-world -e RACK_ENV=development

# Create a build config from a remote private repository and specify which existing secret to use
oc new-build https://github.com/youruser/yourgitrepo --source-secret=yoursecret

# Create a build config using  an image with the full manifest list to create an app and override
```

```
application artifacts' names
  oc new-build --image=myregistry.com/mycompany/image --name=private --import-
mode=PreserveOriginal

  # Create a build config from a remote repository and inject the npmrc into a build
  oc new-build https://github.com/openshift/ruby-hello-world --build-secret npmrc:.npmrc

  # Create a build config from a remote repository and inject environment data into a build
  oc new-build https://github.com/openshift/ruby-hello-world --build-config-map env:config

  # Create a build config that gets its input from a remote repository and another container image
  oc new-build https://github.com/openshift/ruby-hello-world --source-image=openshift/jenkins-1-
centos7 --source-image-path=/var/lib/jenkins:tmp
```

2.6.1.95. oc new-project

Request a new project

Example usage

```
  # Create a new project with minimal information
  oc new-project web-team-dev

  # Create a new project with a display name and description
  oc new-project web-team-dev --display-name="Web Team Development" --
description="Development project for the web team."
```

2.6.1.96. oc observe

Observe changes to resources and react to them (experimental)

Example usage

```
  # Observe changes to services
  oc observe services

  # Observe changes to services, including the clusterIP and invoke a script for each
  oc observe services --template '{ .spec.clusterIP }' -- register_dns.sh

  # Observe changes to services filtered by a label selector
  oc observe services -l regist-dns=true --template '{ .spec.clusterIP }' -- register_dns.sh
```

2.6.1.97. oc patch

Update fields of a resource

Example usage

```
  # Partially update a node using a strategic merge patch, specifying the patch as JSON
  oc patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

  # Partially update a node using a strategic merge patch, specifying the patch as YAML
  oc patch node k8s-node-1 -p $'spec:\n unschedulable: true'
```

```
# Partially update a node identified by the type and name specified in "node.json" using strategic merge patch
oc patch -f node.json -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because it's a merge key
oc patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-hostname","image":"new image"}]}}'

# Update a container's image using a JSON patch with positional arrays
oc patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/image", "value":"new image"}]'

# Update a deployment's replicas through the 'scale' subresource using a merge patch
oc patch deployment nginx-deployment --subresource='scale' --type='merge' -p '{"spec":{"replicas":2}}'
```

2.6.1.98. oc plugin

Provides utilities for interacting with plugins

Example usage

```
# List all available plugins
oc plugin list

# List only binary names of available plugins without paths
oc plugin list --name-only
```

2.6.1.99. oc plugin list

List all visible plugin executables on a user's PATH

Example usage

```
# List all available plugins
oc plugin list

# List only binary names of available plugins without paths
oc plugin list --name-only
```

2.6.1.100. oc policy add-role-to-user

Add a role to users or service accounts for the current project

Example usage

```
# Add the 'view' role to user1 for the current project
oc policy add-role-to-user view user1

# Add the 'edit' role to serviceaccount1 for the current project
oc policy add-role-to-user edit -z serviceaccount1
```

2.6.1.101. oc policy scc-review

Check which service account can create a pod

Example usage

```
# Check whether service accounts sa1 and sa2 can admit a pod with a template pod spec specified in my_resource.yaml
# Service Account specified in myresource.yaml file is ignored
oc policy scc-review -z sa1,sa2 -f my_resource.yaml

# Check whether service accounts system:serviceaccount:bob:default can admit a pod with a template pod spec specified in my_resource.yaml
oc policy scc-review -z system:serviceaccount:bob:default -f my_resource.yaml

# Check whether the service account specified in my_resource_with_sa.yaml can admit the pod
oc policy scc-review -f my_resource_with_sa.yaml

# Check whether the default service account can admit the pod; default is taken since no service account is defined in myresource_with_no_sa.yaml
oc policy scc-review -f myresource_with_no_sa.yaml
```

2.6.1.102. oc policy scc-subject-review

Check whether a user or a service account can create a pod

Example usage

```
# Check whether user bob can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -f myresource.yaml

# Check whether user bob who belongs to projectAdmin group can create a pod specified in myresource.yaml
oc policy scc-subject-review -u bob -g projectAdmin -f myresource.yaml

# Check whether a service account specified in the pod template spec in myresourcewithsa.yaml can create the pod
oc policy scc-subject-review -f myresourcewithsa.yaml
```

2.6.1.103. oc port-forward

Forward one or more local ports to a pod

Example usage

```
# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod
oc port-forward pod/mypod 5000 6000

# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in a pod selected by the deployment
oc port-forward deployment/mydeployment 5000 6000

# Listen on port 8443 locally, forwarding to the targetPort of the service's port named "https" in a pod selected by the service
oc port-forward service/myservice 8443:https
```

```
# Listen on port 8888 locally, forwarding to 5000 in the pod
oc port-forward pod/mypod 8888:5000

# Listen on port 8888 on all addresses, forwarding to 5000 in the pod
oc port-forward --address 0.0.0.0 pod/mypod 8888:5000

# Listen on port 8888 on localhost and selected IP, forwarding to 5000 in the pod
oc port-forward --address localhost,10.19.21.23 pod/mypod 8888:5000

# Listen on a random port locally, forwarding to 5000 in the pod
oc port-forward pod/mypod :5000
```

2.6.1.104. oc process

Process a template into list of resources

Example usage

```
# Convert the template.json file into a resource list and pass to create
oc process -f template.json | oc create -f -

# Process a file locally instead of contacting the server
oc process -f template.json --local -o yaml

# Process template while passing a user-defined label
oc process -f template.json -l name=mytemplate

# Convert a stored template into a resource list
oc process foo

# Convert a stored template into a resource list by setting/overriding parameter values
oc process foo PARM1=VALUE1 PARM2=VALUE2

# Convert a template stored in different namespace into a resource list
oc process openshift//foo

# Convert template.json into a resource list
cat template.json | oc process -f -
```

2.6.1.105. oc project

Switch to another project

Example usage

```
# Switch to the 'myapp' project
oc project myapp

# Display the project currently in use
oc project
```

2.6.1.106. oc projects

Display existing projects

Example usage

```
# List all projects
oc projects
```

2.6.1.107. oc proxy

Run a proxy to the Kubernetes API server

Example usage

```
# To proxy all of the Kubernetes API and nothing else
oc proxy --api-prefix=/

# To proxy only part of the Kubernetes API and also some static files
# You can get pods info with 'curl localhost:8001/api/v1/pods'
oc proxy --www=/my/files --www-prefix=/static/ --api-prefix=/api/

# To proxy the entire Kubernetes API at a different root
# You can get pods info with 'curl localhost:8001/custom/api/v1/pods'
oc proxy --api-prefix=/custom/

# Run a proxy to the Kubernetes API server on port 8011, serving static content from ./local/www/
oc proxy --port=8011 --www=./local/www/

# Run a proxy to the Kubernetes API server on an arbitrary local port
# The chosen port for the server will be output to stdout
oc proxy --port=0

# Run a proxy to the Kubernetes API server, changing the API prefix to k8s-api
# This makes e.g. the pods API available at localhost:8001/k8s-api/v1/pods/
oc proxy --api-prefix=/k8s-api
```

2.6.1.108. oc registry login

Log in to the integrated registry

Example usage

```
# Log in to the integrated registry
oc registry login

# Log in to different registry using BASIC auth credentials
oc registry login --registry quay.io/myregistry --auth-basic=USER:PASS
```

2.6.1.109. oc replace

Replace a resource by file name or stdin

Example usage

```
# Replace a pod using the data in pod.json
oc replace -f ./pod.json
```

```
# Replace a pod based on the JSON passed into stdin
cat pod.json | oc replace -f -

# Update a single-container pod's image version (tag) to v4
oc get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | oc replace -f -

# Force replace, delete and then re-create the resource
oc replace --force -f ./pod.json
```

2.6.1.110. oc rollback

Revert part of an application back to a previous deployment

Example usage

```
# Perform a rollback to the last successfully completed deployment for a deployment config
oc rollback frontend

# See what a rollback to version 3 will look like, but do not perform the rollback
oc rollback frontend --to-version=3 --dry-run

# Perform a rollback to a specific deployment
oc rollback frontend-2

# Perform the rollback manually by piping the JSON of the new config back to oc
oc rollback frontend -o json | oc replace dc/frontend -f -

# Print the updated deployment configuration in JSON format instead of performing the rollback
oc rollback frontend -o json
```

2.6.1.111. oc rollout

Manage the rollout of a resource

Example usage

```
# Roll back to the previous deployment
oc rollout undo deployment/abc

# Check the rollout status of a daemonset
oc rollout status daemonset/foo

# Restart a deployment
oc rollout restart deployment/abc

# Restart deployments with the 'app=nginx' label
oc rollout restart deployment --selector=app=nginx
```

2.6.1.112. oc rollout cancel

Cancel the in-progress deployment

Example usage

```
# Cancel the in-progress deployment based on 'nginx'
oc rollout cancel dc/nginx
```

2.6.1.113. oc rollout history

View rollout history

Example usage

```
# View the rollout history of a deployment
oc rollout history deployment/abc

# View the details of daemonset revision 3
oc rollout history daemonset/abc --revision=3
```

2.6.1.114. oc rollout latest

Start a new rollout for a deployment config with the latest state from its triggers

Example usage

```
# Start a new rollout based on the latest images defined in the image change triggers
oc rollout latest dc/nginx

# Print the rolled out deployment config
oc rollout latest dc/nginx -o json
```

2.6.1.115. oc rollout pause

Mark the provided resource as paused

Example usage

```
# Mark the nginx deployment as paused
# Any current state of the deployment will continue its function; new updates
# to the deployment will not have an effect as long as the deployment is paused
oc rollout pause deployment/nginx
```

2.6.1.116. oc rollout restart

Restart a resource

Example usage

```
# Restart all deployments in the test-namespace namespace
oc rollout restart deployment -n test-namespace

# Restart a deployment
oc rollout restart deployment/nginx

# Restart a daemon set
oc rollout restart daemonset/abc
```

```
# Restart deployments with the app=nginx label
oc rollout restart deployment --selector=app=nginx
```

2.6.1.117. oc rollout resume

Resume a paused resource

Example usage

```
# Resume an already paused deployment
oc rollout resume deployment/nginx
```

2.6.1.118. oc rollout retry

Retry the latest failed rollout

Example usage

```
# Retry the latest failed deployment based on 'frontend'
# The deployer pod and any hook pods are deleted for the latest failed deployment
oc rollout retry dc/frontend
```

2.6.1.119. oc rollout status

Show the status of the rollout

Example usage

```
# Watch the rollout status of a deployment
oc rollout status deployment/nginx
```

2.6.1.120. oc rollout undo

Undo a previous rollout

Example usage

```
# Roll back to the previous deployment
oc rollout undo deployment/abc

# Roll back to daemonset revision 3
oc rollout undo daemonset/abc --to-revision=3

# Roll back to the previous deployment with dry-run
oc rollout undo --dry-run=server deployment/abc
```

2.6.1.121. oc rsh

Start a shell session in a container

Example usage

```
# Open a shell session on the first container in pod 'foo'
oc rsh foo

# Open a shell session on the first container in pod 'foo' and namespace 'bar'
# (Note that oc client specific arguments must come before the resource name and its arguments)
oc rsh -n bar foo

# Run the command 'cat /etc/resolv.conf' inside pod 'foo'
oc rsh foo cat /etc/resolv.conf

# See the configuration of your internal registry
oc rsh dc/docker-registry cat config.yml

# Open a shell session on the container named 'index' inside a pod of your job
oc rsh -c index job/scheduled
```

2.6.1.122. oc rsync

Copy files between a local file system and a pod

Example usage

```
# Synchronize a local directory with a pod directory
oc rsync ./local/dir/ POD:/remote/dir

# Synchronize a pod directory with a local directory
oc rsync POD:/remote/dir/ ./local/dir
```

2.6.1.123. oc run

Run a particular image on the cluster

Example usage

```
# Start a nginx pod
oc run nginx --image=nginx

# Start a hazelcast pod and let the container expose port 5701
oc run hazelcast --image=hazelcast/hazelcast --port=5701

# Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluster" and
"POD_NAMESPACE=default" in the container
oc run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster" --
env="POD_NAMESPACE=default"

# Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in the container
oc run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=prod"

# Dry run; print the corresponding API objects without creating them
oc run nginx --image=nginx --dry-run=client

# Start a nginx pod, but overload the spec with a partial set of values parsed from JSON
oc run nginx --image=nginx --overrides='{ "apiVersion": "v1", "spec": { ... } }'
```

```
# Start a busybox pod and keep it in the foreground, don't restart it if it exits
oc run -i -t busybox --image=busybox --restart=Never

# Start the nginx pod using the default command, but use custom arguments (arg1 .. argN) for that command
oc run nginx --image=nginx -- <arg1> <arg2> ... <argN>

# Start the nginx pod using a different command and custom arguments
oc run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

2.6.1.124. oc scale

Set a new size for a deployment, replica set, or replication controller

Example usage

```
# Scale a replica set named 'foo' to 3
oc scale --replicas=3 rs/foo

# Scale a resource identified by type and name specified in "foo.yaml" to 3
oc scale --replicas=3 -f foo.yaml

# If the deployment named mysql's current size is 2, scale mysql to 3
oc scale --current-replicas=2 --replicas=3 deployment/mysql

# Scale multiple replication controllers
oc scale --replicas=5 rc/example1 rc/example2 rc/example3

# Scale stateful set named 'web' to 3
oc scale --replicas=3 statefulset/web
```

2.6.1.125. oc secrets link

Link secrets to a service account

Example usage

```
# Add an image pull secret to a service account to automatically use it for pulling pod images
oc secrets link serviceaccount-name pull-secret --for=pull

# Add an image pull secret to a service account to automatically use it for both pulling and pushing build images
oc secrets link builder builder-image-secret --for=pull,mount
```

2.6.1.126. oc secrets unlink

Detach secrets from a service account

Example usage

```
# Unlink a secret currently associated with a service account
oc secrets unlink serviceaccount-name secret-name another-secret-name ...
```

2.6.1.127. oc set build-hook

Update a build hook on a build config

Example usage

```
# Clear post-commit hook on a build config
oc set build-hook bc/mybuild --post-commit --remove

# Set the post-commit hook to execute a test suite using a new entrypoint
oc set build-hook bc/mybuild --post-commit --command -- /bin/bash -c /var/lib/test-image.sh

# Set the post-commit hook to execute a shell script
oc set build-hook bc/mybuild --post-commit --script="/var/lib/test-image.sh param1 param2 &&
/var/lib/done.sh"
```

2.6.1.128. oc set build-secret

Update a build secret on a build config

Example usage

```
# Clear the push secret on a build config
oc set build-secret --push --remove bc/mybuild

# Set the pull secret on a build config
oc set build-secret --pull bc/mybuild mysecret

# Set the push and pull secret on a build config
oc set build-secret --push --pull bc/mybuild mysecret

# Set the source secret on a set of build configs matching a selector
oc set build-secret --source -l app=myapp gitsecret
```

2.6.1.129. oc set data

Update the data within a config map or secret

Example usage

```
# Set the 'password' key of a secret
oc set data secret/foo password=this_is_secret

# Remove the 'password' key from a secret
oc set data secret/foo password-

# Update the 'haproxy.conf' key of a config map from a file on disk
oc set data configmap/bar --from-file=../haproxy.conf

# Update a secret with the contents of a directory, one key per file
oc set data secret/foo --from-file=secret-dir
```

2.6.1.130. oc set deployment-hook

Update a deployment hook on a deployment config

Example usage

```
# Clear pre and post hooks on a deployment config
oc set deployment-hook dc/myapp --remove --pre --post

# Set the pre deployment hook to execute a db migration command for an application
# using the data volume from the application
oc set deployment-hook dc/myapp --pre --volumes=data -- /var/lib/migrate-db.sh

# Set a mid deployment hook along with additional environment variables
oc set deployment-hook dc/myapp --mid --volumes=data -e VAR1=value1 -e VAR2=value2 --
/var/lib/prepare-deploy.sh
```

2.6.1.131. oc set env

Update environment variables on a pod template

Example usage

```
# Update deployment config 'myapp' with a new environment variable
oc set env dc/myapp STORAGE_DIR=/local

# List the environment variables defined on a build config 'sample-build'
oc set env bc/sample-build --list

# List the environment variables defined on all pods
oc set env pods --all --list

# Output modified build config in YAML
oc set env bc/sample-build STORAGE_DIR=/data -o yaml

# Update all containers in all replication controllers in the project to have ENV=prod
oc set env rc --all ENV=prod

# Import environment from a secret
oc set env --from=secret/mysecret dc/myapp

# Import environment from a config map with a prefix
oc set env --from=configmap/myconfigmap --prefix=MYSQL_ dc/myapp

# Remove the environment variable ENV from container 'c1' in all deployment configs
oc set env dc --all --containers="c1" ENV-

# Remove the environment variable ENV from a deployment config definition on disk and
# update the deployment config on the server
oc set env -f dc.json ENV-

# Set some of the local shell environment into a deployment config on the server
oc set env | grep RAILS_ | oc env -e - dc/myapp
```

2.6.1.132. oc set image

Update the image of a pod template

Example usage

```
# Set a deployment config's nginx container image to 'nginx:1.9.1', and its busybox container image to 'busybox'.
oc set image dc/nginx busybox=busybox nginx=nginx:1.9.1

# Set a deployment config's app container image to the image referenced by the imagestream tag 'openshift/ruby:2.3'.
oc set image dc/myapp app=openshift/ruby:2.3 --source=imagestreamtag

# Update all deployments' and rc's nginx container's image to 'nginx:1.9.1'
oc set image deployments,rc nginx=nginx:1.9.1 --all

# Update image of all containers of daemonset abc to 'nginx:1.9.1'
oc set image daemonset abc *=nginx:1.9.1

# Print result (in YAML format) of updating nginx container image from local file, without hitting the server
oc set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

2.6.1.133. oc set image-lookup

Change how images are resolved when deploying applications

Example usage

```
# Print all of the image streams and whether they resolve local names
oc set image-lookup

# Use local name lookup on image stream mysql
oc set image-lookup mysql

# Force a deployment to use local name lookup
oc set image-lookup deploy/mysql

# Show the current status of the deployment lookup
oc set image-lookup deploy/mysql --list

# Disable local name lookup on image stream mysql
oc set image-lookup mysql --enabled=false

# Set local name lookup on all image streams
oc set image-lookup --all
```

2.6.1.134. oc set probe

Update a probe on a pod template

Example usage

```
# Clear both readiness and liveness probes off all containers
oc set probe dc/myapp --remove --readiness --liveness
```

```
# Set an exec action as a liveness probe to run 'echo ok'
oc set probe dc/myapp --liveness -- echo ok

# Set a readiness probe to try to open a TCP socket on 3306
oc set probe rc/mysql --readiness --open-tcp=3306

# Set an HTTP startup probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --startup --get-url=http://:8080/healthz

# Set an HTTP readiness probe for port 8080 and path /healthz over HTTP on the pod IP
oc set probe dc/webapp --readiness --get-url=http://:8080/healthz

# Set an HTTP readiness probe over HTTPS on 127.0.0.1 for a hostNetwork pod
oc set probe dc/router --readiness --get-url=https://127.0.0.1:1936/stats

# Set only the initial-delay-seconds field on all deployments
oc set probe dc --all --readiness --initial-delay-seconds=30
```

2.6.1.135. oc set resources

Update resource requests/limits on objects with pod templates

Example usage

```
# Set a deployments nginx container CPU limits to "200m and memory to 512Mi"
oc set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Set the resource request and limits for all containers in nginx
oc set resources deployment nginx --limits=cpu=200m,memory=512Mi --
requests=cpu=100m,memory=256Mi

# Remove the resource requests for resources on containers in nginx
oc set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=0,memory=0

# Print the result (in YAML format) of updating nginx container limits locally, without hitting the server
oc set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --local -o yaml
```

2.6.1.136. oc set route-backends

Update the backends for a route

Example usage

```
# Print the backends on the route 'web'
oc set route-backends web

# Set two backend services on route 'web' with 2/3rds of traffic going to 'a'
oc set route-backends web a=2 b=1

# Increase the traffic percentage going to b by 10%% relative to a
oc set route-backends web --adjust b=+10%%

# Set traffic percentage going to b to 10%% of the traffic going to a
oc set route-backends web --adjust b=10%%
```

```
# Set weight of b to 10
oc set route-backends web --adjust b=10

# Set the weight to all backends to zero
oc set route-backends web --zero
```

2.6.1.137. oc set selector

Set the selector on a resource

Example usage

```
# Set the labels and selector before creating a deployment/service pair.
oc create service clusterip my-svc --clusterip="None" -o yaml --dry-run | oc set selector --local -f -
'environment=qa' -o yaml | oc create -f -
oc create deployment my-dep -o yaml --dry-run | oc label --local -f - environment=qa -o yaml | oc
create -f -
```

2.6.1.138. oc set serviceaccount

Update the service account of a resource

Example usage

```
# Set deployment nginx-deployment's service account to serviceaccount1
oc set serviceaccount deployment nginx-deployment serviceaccount1

# Print the result (in YAML format) of updated nginx deployment with service account from a local
file, without hitting the API server
oc set sa -f nginx-deployment.yaml serviceaccount1 --local --dry-run -o yaml
```

2.6.1.139. oc set subject

Update the user, group, or service account in a role binding or cluster role binding

Example usage

```
# Update a cluster role binding for serviceaccount1
oc set subject clusterrolebinding admin --serviceaccount=namespace:serviceaccount1

# Update a role binding for user1, user2, and group1
oc set subject rolebinding admin --user=user1 --user=user2 --group=group1

# Print the result (in YAML format) of updating role binding subjects locally, without hitting the server
oc create rolebinding admin --role=admin --user=admin -o yaml --dry-run | oc set subject --local -f -
--user=foo -o yaml
```

2.6.1.140. oc set triggers

Update the triggers on one or more objects

Example usage

```
# Print the triggers on the deployment config 'myapp'
oc set triggers dc/myapp

# Set all triggers to manual
oc set triggers dc/myapp --manual

# Enable all automatic triggers
oc set triggers dc/myapp --auto

# Reset the GitHub webhook on a build to a new, generated secret
oc set triggers bc/webapp --from-github
oc set triggers bc/webapp --from-webhook

# Remove all triggers
oc set triggers bc/webapp --remove-all

# Stop triggering on config change
oc set triggers dc/myapp --from-config --remove

# Add an image trigger to a build config
oc set triggers bc/webapp --from-image=namespace1/image:latest

# Add an image trigger to a stateful set on the main container
oc set triggers statefulset/db --from-image=namespace1/image:latest -c main
```

2.6.1.141. oc set volumes

Update volumes on a pod template

Example usage

```
# List volumes defined on all deployment configs in the current project
oc set volume dc --all

# Add a new empty dir volume to deployment config (dc) 'myapp' mounted under
# /var/lib/myapp
oc set volume dc/myapp --add --mount-path=/var/lib/myapp

# Use an existing persistent volume claim (PVC) to overwrite an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-name=pvc1 --overwrite

# Remove volume 'v1' from deployment config 'myapp'
oc set volume dc/myapp --remove --name=v1

# Create a new persistent volume claim that overwrites an existing volume 'v1'
oc set volume dc/myapp --add --name=v1 -t pvc --claim-size=1G --overwrite

# Change the mount point for volume 'v1' to /data
oc set volume dc/myapp --add --name=v1 -m /data --overwrite

# Modify the deployment config by removing volume mount "v1" from container "c1"
# (and by removing the volume "v1" if no other containers have volume mounts that reference it)
oc set volume dc/myapp --remove --name=v1 --containers=c1
```

```
# Add new volume based on a more complex volume source (AWS EBS, GCE PD,
# Ceph, Gluster, NFS, ISCSI, ...)
oc set volume dc/myapp --add -m /data --source=<json-string>
```

2.6.1.142. oc start-build

Start a new build

Example usage

```
# Starts build from build config "hello-world"
oc start-build hello-world

# Starts build from a previous build "hello-world-1"
oc start-build --from-build=hello-world-1

# Use the contents of a directory as build input
oc start-build hello-world --from-dir=src/

# Send the contents of a Git repository to the server from tag 'v2'
oc start-build hello-world --from-repo=../hello-world --commit=v2

# Start a new build for build config "hello-world" and watch the logs until the build
# completes or fails
oc start-build hello-world --follow

# Start a new build for build config "hello-world" and wait until the build completes. It
# exits with a non-zero return code if the build fails
oc start-build hello-world --wait
```

2.6.1.143. oc status

Show an overview of the current project

Example usage

```
# See an overview of the current project
oc status

# Export the overview of the current project in an svg file
oc status -o dot | dot -T svg -o project.svg

# See an overview of the current project including details for any identified issues
oc status --suggest
```

2.6.1.144. oc tag

Tag existing images into image streams

Example usage

```
# Tag the current image for the image stream 'openshift/ruby' and tag '2.0' into the image stream
'yourproject/ruby with tag 'tip'
oc tag openshift/ruby:2.0 yourproject/ruby:tip
```

```
# Tag a specific image
oc tag
openshift/ruby@sha256:6b646fa6bf5e5e4c7fa41056c27910e679c03ebe7f93e361e6515a9da7e258cc
yourproject/ruby:tip

# Tag an external container image
oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip

# Tag an external container image and request pullthrough for it
oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --reference-
policy=local

# Tag an external container image and include the full manifest list
oc tag --source=docker openshift/origin-control-plane:latest yourproject/ruby:tip --import-
mode=PreserveOriginal

# Remove the specified spec tag from an image stream
oc tag openshift/origin-control-plane:latest -d
```

2.6.1.145. oc version

Print the client and server version information

Example usage

```
# Print the OpenShift client, kube-apiserver, and openshift-apiserver version information for the
current context
oc version

# Print the OpenShift client, kube-apiserver, and openshift-apiserver version numbers for the current
context in JSON format
oc version --output json

# Print the OpenShift client version information for the current context
oc version --client
```

2.6.1.146. oc wait

Experimental: Wait for a specific condition on one or many resources

Example usage

```
# Wait for the pod "busybox1" to contain the status condition of type "Ready"
oc wait --for=condition=Ready pod/busybox1

# The default value of status condition is true; you can wait for other targets after an equal delimiter
(compared after Unicode simple case folding, which is a more general form of case-insensitivity)
oc wait --for=condition=Ready=false pod/busybox1

# Wait for the pod "busybox1" to contain the status phase to be "Running"
oc wait --for=jsonpath='{.status.phase}'=Running pod/busybox1

# Wait for pod "busybox1" to be Ready
```

```
oc wait --for='jsonpath={.status.conditions[?(@.type=="Ready")].status}=True' pod/busybox1

# Wait for the service "loadbalancer" to have ingress
oc wait --for=jsonpath='{.status.loadBalancer.ingress}' service/loadbalancer

# Wait for the secret "busybox1" to be created, with a timeout of 30s
oc create secret generic busybox1
oc wait --for=create secret/busybox1 --timeout=30s

# Wait for the pod "busybox1" to be deleted, with a timeout of 60s, after having issued the "delete" command
oc delete pod/busybox1
oc wait --for=delete pod/busybox1 --timeout=60s
```

2.6.1.147. oc whoami

Return information about the current session

Example usage

```
# Display the currently authenticated user
oc whoami
```

2.7. OPENSHIFT CLI ADMINISTRATOR COMMAND REFERENCE

This reference provides descriptions and example commands for OpenShift CLI (**oc**) administrator commands. You must have **cluster-admin** or equivalent permissions to use these commands.

For developer commands, see the OpenShift CLI developer command reference.

Run **oc adm -h** to list all administrator commands or run **oc <command> --help** to get additional details for a specific command.

2.7.1. OpenShift CLI (oc) administrator commands

2.7.1.1. oc adm build-chain

Output the inputs and dependencies of your builds

Example usage

```
# Build the dependency tree for the 'latest' tag in <image-stream>
oc adm build-chain <image-stream>

# Build the dependency tree for the 'v2' tag in dot format and visualize it via the dot utility
oc adm build-chain <image-stream>:v2 -o dot | dot -T svg -o deps.svg

# Build the dependency tree across all namespaces for the specified image stream tag found in the 'test' namespace
oc adm build-chain <image-stream> -n test --all
```

2.7.1.2. oc adm catalog mirror

Mirror an operator-registry catalog

Example usage

```
# Mirror an operator-registry image and its contents to a registry
oc adm catalog mirror quay.io/my/image:latest myregistry.com

# Mirror an operator-registry image and its contents to a particular namespace in a registry
oc adm catalog mirror quay.io/my/image:latest myregistry.com/my-namespace

# Mirror to an airgapped registry by first mirroring to files
oc adm catalog mirror quay.io/my/image:latest file:///local/index
oc adm catalog mirror file:///local/index/my/image:latest my-airgapped-registry.com

# Configure a cluster to use a mirrored registry
oc apply -f manifests/imageDigestMirrorSet.yaml

# Edit the mirroring mappings and mirror with "oc image mirror" manually
oc adm catalog mirror --manifests-only quay.io/my/image:latest myregistry.com
oc image mirror -f manifests/mapping.txt

# Delete all ImageDigestMirrorSets generated by oc adm catalog mirror
oc delete imagedigestmirrorset -l operators.openshift.org/catalog=true
```

2.7.1.3. oc adm certificate approve

Approve a certificate signing request

Example usage

```
# Approve CSR 'csr-sqgzp'
oc adm certificate approve csr-sqgzp
```

2.7.1.4. oc adm certificate deny

Deny a certificate signing request

Example usage

```
# Deny CSR 'csr-sqgzp'
oc adm certificate deny csr-sqgzp
```

2.7.1.5. oc adm copy-to-node

Copy specified files to the node

Example usage

```
# Copy a new bootstrap kubeconfig file to node-0
oc adm copy-to-node --copy=new-bootstrap-kubeconfig=/etc/kubernetes/kubeconfig node/node-0
```

2.7.1.6. oc adm cordon

Mark node as unschedulable

Example usage

```
# Mark node "foo" as unschedulable
oc adm cordon foo
```

2.7.1.7. oc adm create-bootstrap-project-template

Create a bootstrap project template

Example usage

```
# Output a bootstrap project template in YAML format to stdout
oc adm create-bootstrap-project-template -o yaml
```

2.7.1.8. oc adm create-error-template

Create an error page template

Example usage

```
# Output a template for the error page to stdout
oc adm create-error-template
```

2.7.1.9. oc adm create-login-template

Create a login template

Example usage

```
# Output a template for the login page to stdout
oc adm create-login-template
```

2.7.1.10. oc adm create-provider-selection-template

Create a provider selection template

Example usage

```
# Output a template for the provider selection page to stdout
oc adm create-provider-selection-template
```

2.7.1.11. oc adm drain

Drain node in preparation for maintenance

Example usage

```
# Drain node "foo", even if there are pods not managed by a replication controller, replica set, job,
daemon set, or stateful set on it
```

```
oc adm drain foo --force

# As above, but abort if there are pods not managed by a replication controller, replica set, job, daemon set, or stateful set, and use a grace period of 15 minutes
oc adm drain foo --grace-period=900
```

2.7.1.12. oc adm groups add-users

Add users to a group

Example usage

```
# Add user1 and user2 to my-group
oc adm groups add-users my-group user1 user2
```

2.7.1.13. oc adm groups new

Create a new group

Example usage

```
# Add a group with no users
oc adm groups new my-group

# Add a group with two users
oc adm groups new my-group user1 user2

# Add a group with one user and shorter output
oc adm groups new my-group user1 -o name
```

2.7.1.14. oc adm groups prune

Remove old OpenShift groups referencing missing records from an external provider

Example usage

```
# Prune all orphaned groups
oc adm groups prune --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Prune all orphaned groups except the ones from the denylist file
oc adm groups prune --blacklist=/path/to/denylist.txt --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Prune all orphaned groups from a list of specific groups specified in an allowlist file
oc adm groups prune --whitelist=/path/to/allowlist.txt --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Prune all orphaned groups from a list of specific groups specified in a list
oc adm groups prune groups/group_name groups/other_name --sync-config=/path/to/ldap-sync-config.yaml --confirm
```

2.7.1.15. oc adm groups remove-users

Remove users from a group

Example usage

```
# Remove user1 and user2 from my-group
oc adm groups remove-users my-group user1 user2
```

2.7.1.16. oc adm groups sync

Sync OpenShift groups with records from an external provider

Example usage

```
# Sync all groups with an LDAP server
oc adm groups sync --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Sync all groups except the ones from the blacklist file with an LDAP server
oc adm groups sync --blacklist=/path/to/blacklist.txt --sync-config=/path/to/ldap-sync-config.yaml --
confirm

# Sync specific groups specified in an allowlist file with an LDAP server
oc adm groups sync --whitelist=/path/to/allowlist.txt --sync-config=/path/to/sync-config.yaml --
confirm

# Sync all OpenShift groups that have been synced previously with an LDAP server
oc adm groups sync --type=openshift --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Sync specific OpenShift groups if they have been synced previously with an LDAP server
oc adm groups sync groups/group1 groups/group2 groups/group3 --sync-config=/path/to/sync-
config.yaml --confirm
```

2.7.1.17. oc adm inspect

Collect debugging data for a given resource

Example usage

```
# Collect debugging data for the "openshift-apiserver" clusteroperator
oc adm inspect clusteroperator/openshift-apiserver

# Collect debugging data for the "openshift-apiserver" and "kube-apiserver" clusteroperators
oc adm inspect clusteroperator/openshift-apiserver clusteroperator/kube-apiserver

# Collect debugging data for all clusteroperators
oc adm inspect clusteroperator

# Collect debugging data for all clusteroperators and clusterversions
oc adm inspect clusteroperators,clusterversions
```

2.7.1.18. oc adm migrate icsp

Update imagecontentsourcepolicy file(s) to imagedigestmirrorset file(s)

Example usage

```
# Update the imagecontentsourcepolicy.yaml file to a new imagedigestmirrorset file under the mydir directory
oc adm migrate icsp imagecontentsourcepolicy.yaml --dest-dir mydir
```

2.7.1.19. oc adm migrate template-instances

Update template instances to point to the latest group-version-kinds

Example usage

```
# Perform a dry-run of updating all objects
oc adm migrate template-instances

# To actually perform the update, the confirm flag must be appended
oc adm migrate template-instances --confirm
```

2.7.1.20. oc adm must-gather

Launch a new instance of a pod for gathering debug information

Example usage

```
# Gather information using the default plug-in image and command, writing into ./must-gather.local.<rand>
oc adm must-gather

# Gather information with a specific local folder to copy to
oc adm must-gather --dest-dir=/local/directory

# Gather audit information
oc adm must-gather -- /usr/bin/gather_audit_logs

# Gather information using multiple plug-in images
oc adm must-gather --image=quay.io/kubevirt/must-gather --image=quay.io/openshift/origin-must-gather

# Gather information using a specific image stream plug-in
oc adm must-gather --image-stream=openshift/must-gather:latest

# Gather information using a specific image, command, and pod directory
oc adm must-gather --image=my/image:tag --source-dir=/pod/directory -- myspecial-command.sh
```

2.7.1.21. oc adm new-project

Create a new project

Example usage

```
# Create a new project using a node selector
oc adm new-project myproject --node-selector='type=user-node,region=east'
```

2.7.1.22. oc adm node-image create

Create an ISO image for booting the nodes to be added to the target cluster

Example usage

```
# Create the ISO image and download it in the current folder
oc adm node-image create

# Use a different assets folder
oc adm node-image create --dir=/tmp/assets

# Specify a custom image name
oc adm node-image create -o=my-node.iso

# In place of an ISO, creates files that can be used for PXE boot
oc adm node-image create --pxe

# Create an ISO to add a single node without using the configuration file
oc adm node-image create --mac-address=00:d8:e7:c7:4b:bb

# Create an ISO to add a single node with a root device hint and without
# using the configuration file
oc adm node-image create --mac-address=00:d8:e7:c7:4b:bb --root-device-
hint=deviceName:/dev/sda
```

2.7.1.23. oc adm node-image monitor

Monitor new nodes being added to an OpenShift cluster

Example usage

```
# Monitor a single node being added to a cluster
oc adm node-image monitor --ip-addresses 192.168.111.83

# Monitor multiple nodes being added to a cluster by separating each
# IP address with a comma
oc adm node-image monitor --ip-addresses 192.168.111.83,192.168.111.84
```

2.7.1.24. oc adm node-logs

Display and filter node logs

Example usage

```
# Show kubelet logs from all control plane nodes
oc adm node-logs --role master -u kubelet

# See what logs are available in control plane nodes in /var/log
oc adm node-logs --role master --path=/

# Display cron log file from all control plane nodes
oc adm node-logs --role master --path=cron
```

2.7.1.25. oc adm ocp-certificates monitor-certificates

Watch platform certificates

Example usage

```
# Watch platform certificates
oc adm ocp-certificates monitor-certificates
```

2.7.1.26. oc adm ocp-certificates regenerate-leaf

Regenerate client and serving certificates of an OpenShift cluster

Example usage

```
# Regenerate a leaf certificate contained in a particular secret
oc adm ocp-certificates regenerate-leaf -n openshift-config-managed secret/kube-controller-manager-client-cert-key
```

2.7.1.27. oc adm ocp-certificates regenerate-machine-config-server-serving-cert

Regenerate the machine config operator certificates in an OpenShift cluster

Example usage

```
# Regenerate the MCO certs without modifying user-data secrets
oc adm ocp-certificates regenerate-machine-config-server-serving-cert --update-ignition=false

# Update the user-data secrets to use new MCS certs
oc adm ocp-certificates update-ignition-ca-bundle-for-machine-config-server
```

2.7.1.28. oc adm ocp-certificates regenerate-top-level

Regenerate the top level certificates in an OpenShift cluster

Example usage

```
# Regenerate the signing certificate contained in a particular secret
oc adm ocp-certificates regenerate-top-level -n openshift-kube-apiserver-operator secret/loadbalancer-serving-signer-key
```

2.7.1.29. oc adm ocp-certificates remove-old-trust

Remove old CAs from ConfigMaps representing platform trust bundles in an OpenShift cluster

Example usage

```
# Remove a trust bundled contained in a particular config map
oc adm ocp-certificates remove-old-trust -n openshift-config-managed configmaps/kube-apiserver-aggregator-client-ca --created-before 2023-06-05T14:44:06Z
```

```
# Remove only CA certificates created before a certain date from all trust bundles
oc adm ocp-certificates remove-old-trust configmaps -A --all --created-before 2023-06-05T14:44:06Z
```

2.7.1.30. oc adm ocp-certificates update-ignition-ca-bundle-for-machine-config-server

Update user-data secrets in an OpenShift cluster to use updated MCO certfs

Example usage

```
# Regenerate the MCO certs without modifying user-data secrets
oc adm ocp-certificates regenerate-machine-config-server-serving-cert --update-ignition=false

# Update the user-data secrets to use new MCS certs
oc adm ocp-certificates update-ignition-ca-bundle-for-machine-config-server
```

2.7.1.31. oc adm policy add-cluster-role-to-group

Add a role to groups for all projects in the cluster

Example usage

```
# Add the 'cluster-admin' cluster role to the 'cluster-admins' group
oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
```

2.7.1.32. oc adm policy add-cluster-role-to-user

Add a role to users for all projects in the cluster

Example usage

```
# Add the 'system:build-strategy-docker' cluster role to the 'devuser' user
oc adm policy add-cluster-role-to-user system:build-strategy-docker devuser
```

2.7.1.33. oc adm policy add-role-to-user

Add a role to users or service accounts for the current project

Example usage

```
# Add the 'view' role to user1 for the current project
oc adm policy add-role-to-user view user1

# Add the 'edit' role to serviceaccount1 for the current project
oc adm policy add-role-to-user edit -z serviceaccount1
```

2.7.1.34. oc adm policy add-scc-to-group

Add a security context constraint to groups

Example usage

```
# Add the 'restricted' security context constraint to group1 and group2
oc adm policy add-scc-to-group restricted group1 group2
```

2.7.1.35. oc adm policy add-scc-to-user

Add a security context constraint to users or a service account

Example usage

```
# Add the 'restricted' security context constraint to user1 and user2
oc adm policy add-scc-to-user restricted user1 user2

# Add the 'privileged' security context constraint to serviceaccount1 in the current namespace
oc adm policy add-scc-to-user privileged -z serviceaccount1
```

2.7.1.36. oc adm policy remove-cluster-role-from-group

Remove a role from groups for all projects in the cluster

Example usage

```
# Remove the 'cluster-admin' cluster role from the 'cluster-admins' group
oc adm policy remove-cluster-role-from-group cluster-admin cluster-admins
```

2.7.1.37. oc adm policy remove-cluster-role-from-user

Remove a role from users for all projects in the cluster

Example usage

```
# Remove the 'system:build-strategy-docker' cluster role from the 'devuser' user
oc adm policy remove-cluster-role-from-user system:build-strategy-docker devuser
```

2.7.1.38. oc adm policy scc-review

Check which service account can create a pod

Example usage

```
  # Check whether service accounts sa1 and sa2 can admit a pod with a template pod spec specified
in my_resource.yaml
  # Service Account specified in myresource.yaml file is ignored
  oc adm policy scc-review -z sa1,sa2 -f my_resource.yaml

  # Check whether service accounts system:serviceaccount:bob:default can admit a pod with a
template pod spec specified in my_resource.yaml
  oc adm policy scc-review -z system:serviceaccount:bob:default -f my_resource.yaml

  # Check whether the service account specified in my_resource_with_sa.yaml can admit the pod
  oc adm policy scc-review -f my_resource_with_sa.yaml
```

```
# Check whether the default service account can admit the pod; default is taken since no service account is defined in myresource_with_no_sa.yaml
oc adm policy scc-review -f myresource_with_no_sa.yaml
```

2.7.1.39. oc adm policy scc-subject-review

Check whether a user or a service account can create a pod

Example usage

```
# Check whether user bob can create a pod specified in myresource.yaml
oc adm policy scc-subject-review -u bob -f myresource.yaml

# Check whether user bob who belongs to projectAdmin group can create a pod specified in myresource.yaml
oc adm policy scc-subject-review -u bob -g projectAdmin -f myresource.yaml

# Check whether a service account specified in the pod template spec in myresourcewithsa.yaml can create the pod
oc adm policy scc-subject-review -f myresourcewithsa.yaml
```

2.7.1.40. oc adm prune builds

Remove old completed and failed builds

Example usage

```
# Dry run deleting older completed and failed builds and also including
# all builds whose associated build config no longer exists
oc adm prune builds --orphans

# To actually perform the prune operation, the confirm flag must be appended
oc adm prune builds --orphans --confirm
```

2.7.1.41. oc adm prune deployments

Remove old completed and failed deployment configs

Example usage

```
# Dry run deleting all but the last complete deployment for every deployment config
oc adm prune deployments --keep-complete=1

# To actually perform the prune operation, the confirm flag must be appended
oc adm prune deployments --keep-complete=1 --confirm
```

2.7.1.42. oc adm prune groups

Remove old OpenShift groups referencing missing records from an external provider

Example usage

```
# Prune all orphaned groups
```

```
oc adm prune groups --sync-config=/path/to/ldap-sync-config.yaml --confirm

# Prune all orphaned groups except the ones from the denylist file
oc adm prune groups --blacklist=/path/to/denylist.txt --sync-config=/path/to/ldap-sync-config.yaml --
confirm

# Prune all orphaned groups from a list of specific groups specified in an allowlist file
oc adm prune groups --whitelist=/path/to/allowlist.txt --sync-config=/path/to/ldap-sync-config.yaml --
confirm

# Prune all orphaned groups from a list of specific groups specified in a list
oc adm prune groups groups/group_name groups/other_name --sync-config=/path/to/ldap-sync-
config.yaml --confirm
```

2.7.1.43. oc adm prune images

Remove unreferenced images

Example usage

```
# See what the prune command would delete if only images and their referrers were more than an
hour old
# and obsoleted by 3 newer revisions under the same tag were considered
oc adm prune images --keep-tag-revisions=3 --keep-younger-than=60m

# To actually perform the prune operation, the confirm flag must be appended
oc adm prune images --keep-tag-revisions=3 --keep-younger-than=60m --confirm

# See what the prune command would delete if we are interested in removing images
# exceeding currently set limit ranges ('openshift.io/Image')
oc adm prune images --prune-over-size-limit

# To actually perform the prune operation, the confirm flag must be appended
oc adm prune images --prune-over-size-limit --confirm

# Force the insecure HTTP protocol with the particular registry host name
oc adm prune images --registry-url=http://registry.example.org --confirm

# Force a secure connection with a custom certificate authority to the particular registry host name
oc adm prune images --registry-url=registry.example.org --certificate-
authority=/path/to/custom/ca.crt --confirm
```

2.7.1.44. oc adm prune renderedmachineconfigs

Prunes rendered MachineConfigs in an OpenShift cluster

Example usage

```
# See what the prune command would delete if run with no options
oc adm prune renderedmachineconfigs

# To actually perform the prune operation, the confirm flag must be appended
oc adm prune renderedmachineconfigs --confirm

# See what the prune command would delete if run on the worker MachineConfigPool
```

```
oc adm prune renderedmachineconfigs --pool-name=worker

# Prunes 10 oldest rendered MachineConfigs in the cluster
oc adm prune renderedmachineconfigs --count=10 --confirm

# Prunes 10 oldest rendered MachineConfigs in the cluster for the worker MachineConfigPool
oc adm prune renderedmachineconfigs --count=10 --pool-name=worker --confirm
```

2.7.1.45. oc adm prune renderedmachineconfigs list

Lists rendered MachineConfigs in an OpenShift cluster

Example usage

```
# List all rendered MachineConfigs for the worker MachineConfigPool in the cluster
oc adm prune renderedmachineconfigs list --pool-name=worker

# List all rendered MachineConfigs in use by the cluster's MachineConfigPools
oc adm prune renderedmachineconfigs list --in-use
```

2.7.1.46. oc adm reboot-machine-config-pool

Initiate reboot of the specified MachineConfigPool

Example usage

```
# Reboot all MachineConfigPools
oc adm reboot-machine-config-pool mcp/worker mcp/master

# Reboot all MachineConfigPools that inherit from worker.  This include all custom
MachineConfigPools and infra.
oc adm reboot-machine-config-pool mcp/worker

# Reboot masters
oc adm reboot-machine-config-pool mcp/master
```

2.7.1.47. oc adm release extract

Extract the contents of an update payload to disk

Example usage

```
# Use git to check out the source code for the current cluster release to DIR
oc adm release extract --git=DIR

# Extract cloud credential requests for AWS
oc adm release extract --credentials-requests --cloud=aws

# Use git to check out the source code for the current cluster release to DIR from linux/s390x image
# Note: Wildcard filter is not supported; pass a single os/arch to extract
oc adm release extract --git=DIR quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-
os=linux/s390x
```

2.7.1.48. oc adm release info

Display information about a release

Example usage

```
# Show information about the cluster's current release
oc adm release info

# Show the source code that comprises a release
oc adm release info 4.11.2 --commit-urls

# Show the source code difference between two releases
oc adm release info 4.11.0 4.11.2 --commits

# Show where the images referenced by the release are located
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --pullspecs

# Show information about linux/s390x image
# Note: Wildcard filter is not supported; pass a single os/arch to extract
oc adm release info quay.io/openshift-release-dev/ocp-release:4.11.2 --filter-by-os=linux/s390x
```

2.7.1.49. oc adm release mirror

Mirror a release to a different image registry location

Example usage

```
# Perform a dry run showing what would be mirrored, including the mirror objects
oc adm release mirror 4.11.0 --to myregistry.local/openshift/release \
--release-image-signature-to-dir /tmp/releases --dry-run

# Mirror a release into the current directory
oc adm release mirror 4.11.0 --to file://openshift/release \
--release-image-signature-to-dir /tmp/releases

# Mirror a release to another directory in the default location
oc adm release mirror 4.11.0 --to-dir /tmp/releases

# Upload a release from the current directory to another server
oc adm release mirror --from file://openshift/release --to myregistry.com/openshift/release \
--release-image-signature-to-dir /tmp/releases

# Mirror the 4.11.0 release to repository registry.example.com and apply signatures to connected
cluster
oc adm release mirror --from=quay.io/openshift-release-dev/ocp-release:4.11.0-x86_64 \
--to=registry.example.com/your/repository --apply-release-image-signature
```

2.7.1.50. oc adm release new

Create a new OpenShift release

Example usage

```
# Create a release from the latest origin images and push to a DockerHub repository
oc adm release new --from-image-stream=4.11 -n origin --to-image
docker.io/mycompany/myrepo:latest

# Create a new release with updated metadata from a previous release
oc adm release new --from-release registry.ci.openshift.org/origin/release:v4.11 --name 4.11.1 \
--previous 4.11.0 --metadata ... --to-image docker.io/mycompany/myrepo:latest

# Create a new release and override a single image
oc adm release new --from-release registry.ci.openshift.org/origin/release:v4.11 \
cli=docker.io/mycompany/cli:latest --to-image docker.io/mycompany/myrepo:latest

# Run a verification pass to ensure the release can be reproduced
oc adm release new --from-release registry.ci.openshift.org/origin/release:v4.11
```

2.7.1.51. oc adm restart-kubelet

Restart kubelet on the specified nodes

Example usage

```
# Restart all the nodes, 10% at a time
oc adm restart-kubelet nodes --all --directive=RemoveKubeletKubeconfig

# Restart all the nodes, 20 nodes at a time
oc adm restart-kubelet nodes --all --parallelism=20 --directive=RemoveKubeletKubeconfig

# Restart all the nodes, 15% at a time
oc adm restart-kubelet nodes --all --parallelism=15% --directive=RemoveKubeletKubeconfig

# Restart all the masters at the same time
oc adm restart-kubelet nodes -l node-role.kubernetes.io/master --parallelism=100% --
directive=RemoveKubeletKubeconfig
```

2.7.1.52. oc adm taint

Update the taints on one or more nodes

Example usage

```
# Update node 'foo' with a taint with key 'dedicated' and value 'special-user' and effect 'NoSchedule'
# If a taint with that key and effect already exists, its value is replaced as specified
oc adm taint nodes foo dedicated=special-user:NoSchedule

# Remove from node 'foo' the taint with key 'dedicated' and effect 'NoSchedule' if one exists
oc adm taint nodes foo dedicated:NoSchedule-

# Remove from node 'foo' all the taints with key 'dedicated'
oc adm taint nodes foo dedicated-

# Add a taint with key 'dedicated' on nodes having label myLabel=X
oc adm taint node -l myLabel=X  dedicated=foo:PreferNoSchedule
```

```
# Add to node 'foo' a taint with key 'bar' and no value
oc adm taint nodes foo bar:NoSchedule
```

2.7.1.53. oc adm top images

Show usage statistics for images

Example usage

```
# Show usage statistics for images
oc adm top images
```

2.7.1.54. oc adm top imagestreams

Show usage statistics for image streams

Example usage

```
# Show usage statistics for image streams
oc adm top imagestreams
```

2.7.1.55. oc adm top node

Display resource (CPU/memory) usage of nodes

Example usage

```
# Show metrics for all nodes
oc adm top node

# Show metrics for a given node
oc adm top node NODE_NAME
```

2.7.1.56. oc adm top persistentvolumeclaims

Experimental: Show usage statistics for bound persistentvolumeclaims

Example usage

```
# Show usage statistics for all the bound persistentvolumeclaims across the cluster
oc adm top persistentvolumeclaims -A

# Show usage statistics for all the bound persistentvolumeclaims in a specific namespace
oc adm top persistentvolumeclaims -n default

# Show usage statistics for specific bound persistentvolumeclaims
oc adm top persistentvolumeclaims database-pvc app-pvc -n default
```

2.7.1.57. oc adm top pod

Display resource (CPU/memory) usage of pods

Example usage

```
# Show metrics for all pods in the default namespace
oc adm top pod

# Show metrics for all pods in the given namespace
oc adm top pod --namespace=NAMESPACE

# Show metrics for a given pod and its containers
oc adm top pod POD_NAME --containers

# Show metrics for the pods defined by label name=myLabel
oc adm top pod -l name=myLabel
```

2.7.1.58. oc adm uncordon

Mark node as schedulable

Example usage

```
# Mark node "foo" as schedulable
oc adm uncordon foo
```

2.7.1.59. oc adm upgrade

Upgrade a cluster or adjust the upgrade channel

Example usage

```
# View the update status and available cluster updates
oc adm upgrade

# Update to the latest version
oc adm upgrade --to-latest=true
```

2.7.1.60. oc adm verify-image-signature

Verify the image identity contained in the image signature

Example usage

```
# Verify the image signature and identity using the local GPG keychain
oc adm verify-image-signature
sha256:c841e9b64e4579bd56c794bdd7c36e1c257110fd2404bebbb8b613e4935228c4 \
--expected-identity=registry.local:5000/foo/bar:v1

# Verify the image signature and identity using the local GPG keychain and save the status
oc adm verify-image-signature
sha256:c841e9b64e4579bd56c794bdd7c36e1c257110fd2404bebbb8b613e4935228c4 \
--expected-identity=registry.local:5000/foo/bar:v1 --save

# Verify the image signature and identity via exposed registry route
oc adm verify-image-signature
```

```
sha256:c841e9b64e4579bd56c794bdd7c36e1c257110fd2404bebbb8b613e4935228c4 \
  --expected-identity=registry.local:5000/foo/bar:v1 \
  --registry-url=docker-registry.foo.com

  # Remove all signature verifications from the image
  oc adm verify-image-signature
sha256:c841e9b64e4579bd56c794bdd7c36e1c257110fd2404bebbb8b613e4935228c4 --remove-all
```

2.7.1.61. oc adm wait-for-node-reboot

Wait for nodes to reboot after running **oc adm reboot-machine-config-pool**

Example usage

```
  # Wait for all nodes to complete a requested reboot from 'oc adm reboot-machine-config-pool
mcp/worker mcp/master'
  oc adm wait-for-node-reboot nodes --all

  # Wait for masters to complete a requested reboot from 'oc adm reboot-machine-config-pool
mcp/master'
  oc adm wait-for-node-reboot nodes -l node-role.kubernetes.io/master

  # Wait for masters to complete a specific reboot
  oc adm wait-for-node-reboot nodes -l node-role.kubernetes.io/master --reboot-number=4
```

2.7.1.62. oc adm wait-for-stable-cluster

Wait for the platform operators to become stable

Example usage

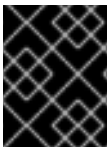
```
  # Wait for all cluster operators to become stable
  oc adm wait-for-stable-cluster

  # Consider operators to be stable if they report as such for 5 minutes straight
  oc adm wait-for-stable-cluster --minimum-stable-period 5m
```

2.7.2. Additional resources

- OpenShift CLI developer command reference

CHAPTER 3. IMPORTANT UPDATE ON ODO

Red Hat does not provide information about **odo** on the Red Hat OpenShift Service on AWS documentation site. See the documentation maintained by Red Hat and the upstream community for documentation information related to **odo**.



IMPORTANT

For the materials maintained by the upstream community, Red Hat provides support under Cooperative Community Support.

CHAPTER 4. KNATIVE CLI FOR USE WITH OPENSHIFT SERVERLESS

The Knative (**kn**) CLI enables simple interaction with Knative components on Red Hat OpenShift Service on AWS.

4.1. KEY FEATURES

The Knative (**kn**) CLI is designed to make serverless computing tasks simple and concise. Key features of the Knative CLI include:

- Deploy serverless applications from the command line.
- Manage features of Knative Serving, such as services, revisions, and traffic-splitting.
- Create and manage Knative Eventing components, such as event sources and triggers.
- Create sink bindings to connect existing Kubernetes applications and Knative services.
- Extend the Knative CLI with flexible plugin architecture, similar to the **kubectl** CLI.
- Configure autoscaling parameters for Knative services.
- Scripted usage, such as waiting for the results of an operation, or deploying custom rollout and rollback strategies.

4.2. INSTALLING THE KNATIVE CLI

See Installing the Knative CLI.

CHAPTER 5. PIPELINES CLI (TKN)

5.1. INSTALLING TKN

Use the CLI tool to manage Red Hat OpenShift Pipelines from a terminal. The following section describes how to install the CLI tool on different platforms.

You can also find the URL to the latest binaries from the Red Hat OpenShift Service on AWS web console by clicking the **?** icon in the upper-right corner and selecting **Command Line Tools**.

IMPORTANT

Running Red Hat OpenShift Pipelines on ARM hardware is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

NOTE

Both the archives and the RPMs contain the following executables:

- **tkn**
- **tkn-pac**
- **opc**

IMPORTANT

Running Red Hat OpenShift Pipelines with the **opc** CLI tool is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview Features Support Scope .

5.1.1. Installing the Red Hat OpenShift Pipelines CLI on Linux

For Linux distributions, you can download the CLI as a **tar.gz** archive.

Procedure

1. Download the relevant CLI tool.
   - Linux (x86_64, amd64)

- Linux on IBM Z® and IBM® LinuxONE (s390x)
- Linux on IBM Power® (ppc64le)
- Linux on ARM (aarch64, arm64)

1. Unpack the archive:

```
$ tar xvzf <file>
```

2. Add the location of your **tkn** and **tkn-pac** files to your **PATH** environment variable.
3. To check your **PATH**, run the following command:

```
$ echo $PATH
```

5.1.2. Installing the Red Hat OpenShift Pipelines CLI on Linux using an RPM

For Red Hat Enterprise Linux (RHEL) version 8, you can install the Red Hat OpenShift Pipelines CLI as an RPM.

Prerequisites

- You have an active Red Hat OpenShift Service on AWS subscription on your Red Hat account.
- You have root or sudo privileges on your local system.

Procedure

1. Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

2. Pull the latest subscription data:

```
# subscription-manager refresh
```

3. List the available subscriptions:

```
# subscription-manager list --available --matches '*pipelines*'
```

4. In the output for the previous command, find the pool ID for your Red Hat OpenShift Service on AWS subscription and attach the subscription to the registered system:

```
# subscription-manager attach --pool=<pool_id>
```

5. Enable the repositories required by Red Hat OpenShift Pipelines:
   - Linux (x86_64, amd64)

     ```
     # subscription-manager repos --enable="pipelines-1.18-for-rhel-8-x86_64-rpms"
     ```

   - Linux on IBM Z® and IBM® LinuxONE (s390x)

```
# subscription-manager repos --enable="pipelines-1.18-for-rhel-8-s390x-rpms"
```

- Linux on IBM Power® (ppc64le)

```
# subscription-manager repos --enable="pipelines-1.18-for-rhel-8-ppc64le-rpms"
```

- Linux on ARM (aarch64, arm64)

```
# subscription-manager repos --enable="pipelines-1.18-for-rhel-8-aarch64-rpms"
```

6. Install the **openshift-pipelines-client** package:

```
# yum install openshift-pipelines-client
```

After you install the CLI, it is available using the **tkn** command:

```
$ tkn version
```

5.1.3. Installing the Red Hat OpenShift Pipelines CLI on Windows

For Windows, you can download the CLI as a **zip** archive.

Procedure

1. Download the CLI tool.
2. Extract the archive with a ZIP program.
3. Add the location of your **tkn** and **tkn-pac** files to your **PATH** environment variable.
4. To check your **PATH**, run the following command:

```
C:\> path
```

5.1.4. Installing the Red Hat OpenShift Pipelines CLI on macOS

For macOS, you can download the CLI as a **tar.gz** archive.

Procedure

1. Download the relevant CLI tool.
   - macOS
   - macOS on ARM
2. Unpack and extract the archive.
3. Add the location of your **tkn** and **tkn-pac** and files to your **PATH** environment variable.
4. To check your **PATH**, run the following command:

```
$ echo $PATH
```

5.2. CONFIGURING THE OPENSHIFT PIPELINES TKN CLI

Configure the Red Hat OpenShift Pipelines **tkn** CLI to enable tab completion.

5.2.1. Enabling tab completion

After you install the **tkn** CLI, you can enable tab completion to automatically complete **tkn** commands or suggest options when you press Tab.

Prerequisites

- You must have the **tkn** CLI tool installed.
- You must have **bash-completion** installed on your local system.

Procedure

The following procedure enables tab completion for Bash.

1. Save the Bash completion code to a file:

   ```
   $ tkn completion bash > tkn_bash_completion
   ```

2. Copy the file to **/etc/bash_completion.d/**:

   ```
   $ sudo cp tkn_bash_completion /etc/bash_completion.d/
   ```

   Alternatively, you can save the file to a local directory and source it from your **.bashrc** file instead.

Tab completion is enabled when you open a new terminal.

5.3. OPENSHIFT PIPELINES TKN REFERENCE

This section lists the basic **tkn** CLI commands.

5.3.1. Basic syntax

tkn [command or options] [arguments…]

5.3.2. Global options

--help, -h

5.3.3. Utility commands

5.3.3.1. tkn

Parent command for **tkn** CLI.

Example: Display all options

```
$ tkn
```

5.3.3.2. completion [shell]

Print shell completion code which must be evaluated to provide interactive completion. Supported shells are **bash** and **zsh**.

Example: Completion code for bash shell

```
$ tkn completion bash
```

5.3.3.3. version

Print version information of the **tkn** CLI.

Example: Check the tkn version

```
$ tkn version
```

5.3.4. Pipelines management commands

5.3.4.1. pipeline

Manage pipelines.

Example: Display help

```
$ tkn pipeline --help
```

5.3.4.2. pipeline delete

Delete a pipeline.

Example: Delete the mypipeline pipeline from a namespace

```
$ tkn pipeline delete mypipeline -n myspace
```

5.3.4.3. pipeline describe

Describe a pipeline.

Example: Describe the mypipeline pipeline

```
$ tkn pipeline describe mypipeline
```

5.3.4.4. pipeline list

Display a list of pipelines.

Example: Display a list of pipelines

```
$ tkn pipeline list
```

5.3.4.5. pipeline logs

Display the logs for a specific pipeline.

Example: Stream the live logs for the `mypipeline` pipeline

```
$ tkn pipeline logs -f mypipeline
```

5.3.4.6. pipeline start

Start a pipeline.

Example: Start the `mypipeline` pipeline

```
$ tkn pipeline start mypipeline
```

5.3.5. Pipeline run commands

5.3.5.1. pipelinerun

Manage pipeline runs.

Example: Display help

```
$ tkn pipelinerun -h
```

5.3.5.2. pipelinerun cancel

Cancel a pipeline run.

Example: Cancel the `mypipelinerun` pipeline run from a namespace

```
$ tkn pipelinerun cancel mypipelinerun -n myspace
```

5.3.5.3. pipelinerun delete

Delete a pipeline run.

Example: Delete pipeline runs from a namespace

```
$ tkn pipelinerun delete mypipelinerun1 mypipelinerun2 -n myspace
```

Example: Delete all pipeline runs from a namespace, except the five most recently executed pipeline runs

```
$ tkn pipelinerun delete -n myspace --keep 5 (1)
```

(1) Replace **5** with the number of most recently executed pipeline runs you want to retain.

Example: Delete all pipelines

```
$ tkn pipelinerun delete --all
```

NOTE

Starting with Red Hat OpenShift Pipelines 1.6, the **tkn pipelinerun delete --all** command does not delete any resources that are in the running state.

5.3.5.4. pipelinerun describe

Describe a pipeline run.

Example: Describe the mypipelinerun pipeline run in a namespace

```
$ tkn pipelinerun describe mypipelinerun -n myspace
```

5.3.5.5. pipelinerun list

List pipeline runs.

Example: Display a list of pipeline runs in a namespace

```
$ tkn pipelinerun list -n myspace
```

5.3.5.6. pipelinerun logs

Display the logs of a pipeline run.

Example: Display the logs of the mypipelinerun pipeline run with all tasks and steps in a namespace

```
$ tkn pipelinerun logs mypipelinerun -a -n myspace
```

5.3.6. Task management commands

5.3.6.1. task

Manage tasks.

Example: Display help

```
$ tkn task -h
```

5.3.6.2. task delete

Delete a task.

Example: Delete mytask1 and mytask2 tasks from a namespace

```
$ tkn task delete mytask1 mytask2 -n myspace
```

5.3.6.3. task describe

Describe a task.

Example: Describe the mytask task in a namespace

```
$ tkn task describe mytask -n myspace
```

5.3.6.4. task list

List tasks.

Example: List all the tasks in a namespace

```
$ tkn task list -n myspace
```

5.3.6.5. task logs

Display task logs.

Example: Display logs for the mytaskrun task run of the mytask task

```
$ tkn task logs mytask mytaskrun -n myspace
```

5.3.6.6. task start

Start a task.

Example: Start the mytask task in a namespace

```
$ tkn task start mytask -s <ServiceAccountName> -n myspace
```

5.3.7. Task run commands

5.3.7.1. taskrun

Manage task runs.

Example: Display help

```
$ tkn taskrun -h
```

5.3.7.2. taskrun cancel

Cancel a task run.

Example: Cancel the mytaskrun task run from a namespace

```
$ tkn taskrun cancel mytaskrun -n myspace
```

5.3.7.3. taskrun delete

Delete a TaskRun.

Example: Delete the mytaskrun1 and mytaskrun2 task runs from a namespace

```
$ tkn taskrun delete mytaskrun1 mytaskrun2 -n myspace
```

Example: Delete all but the five most recently executed task runs from a namespace

```
$ tkn taskrun delete -n myspace --keep 5 (1)
```

(1) Replace **5** with the number of most recently executed task runs you want to retain.

5.3.7.4. taskrun describe

Describe a task run.

Example: Describe the mytaskrun task run in a namespace

```
$ tkn taskrun describe mytaskrun -n myspace
```

5.3.7.5. taskrun list

List task runs.

Example: List all the task runs in a namespace

```
$ tkn taskrun list -n myspace
```

5.3.7.6. taskrun logs

Display task run logs.

Example: Display live logs for the mytaskrun task run in a namespace

```
$ tkn taskrun logs -f mytaskrun -n myspace
```

5.3.8. Condition management commands

5.3.8.1. condition

Manage Conditions.

Example: Display help

```
$ tkn condition --help
```

5.3.8.2. condition delete

Delete a Condition.

Example: Delete the mycondition1 Condition from a namespace

```
$ tkn condition delete mycondition1 -n myspace
```

5.3.8.3. condition describe

Describe a Condition.

Example: Describe the mycondition1 Condition in a namespace

```
$ tkn condition describe mycondition1 -n myspace
```

5.3.8.4. condition list

List Conditions.

Example: List Conditions in a namespace

```
$ tkn condition list -n myspace
```

5.3.9. Pipeline Resource management commands

5.3.9.1. resource

Manage Pipeline Resources.

Example: Display help

```
$ tkn resource -h
```

5.3.9.2. resource create

Create a Pipeline Resource.

Example: Create a Pipeline Resource in a namespace

```
$ tkn resource create -n myspace
```

This is an interactive command that asks for input on the name of the Resource, type of the Resource, and the values based on the type of the Resource.

5.3.9.3. resource delete

Delete a Pipeline Resource.

Example: Delete the myresource Pipeline Resource from a namespace

```
$ tkn resource delete myresource -n myspace
```

5.3.9.4. resource describe

Describe a Pipeline Resource.

Example: Describe the myresource Pipeline Resource

```
$ tkn resource describe myresource -n myspace
```

5.3.9.5. resource list

List Pipeline Resources.

Example: List all Pipeline Resources in a namespace

```
$ tkn resource list -n myspace
```

5.3.10. ClusterTask management commands

IMPORTANT

In Red Hat OpenShift Pipelines 1.10, ClusterTask functionality of the **tkn** command-line utility is deprecated and is planned to be removed in a future release.

5.3.10.1. clustertask

Manage ClusterTasks.

Example: Display help

```
$ tkn clustertask --help
```

5.3.10.2. clustertask delete

Delete a ClusterTask resource in a cluster.

Example: Delete mytask1 and mytask2 ClusterTasks

```
$ tkn clustertask delete mytask1 mytask2
```

5.3.10.3. clustertask describe

Describe a ClusterTask.

Example: Describe the mytask ClusterTask

```
$ tkn clustertask describe mytask1
```

5.3.10.4. clustertask list

List ClusterTasks.

Example: List ClusterTasks

```
$ tkn clustertask list
```

5.3.10.5. clustertask start

Start ClusterTasks.

Example: Start the mytask ClusterTask

```
$ tkn clustertask start mytask
```

5.3.11. Trigger management commands

5.3.11.1. eventlistener

Manage EventListeners.

Example: Display help

```
$ tkn eventlistener -h
```

5.3.11.2. eventlistener delete

Delete an EventListener.

Example: Delete mylistener1 and mylistener2 EventListeners in a namespace

```
$ tkn eventlistener delete mylistener1 mylistener2 -n myspace
```

5.3.11.3. eventlistener describe

Describe an EventListener.

Example: Describe the mylistener EventListener in a namespace

```
$ tkn eventlistener describe mylistener -n myspace
```

5.3.11.4. eventlistener list

List EventListeners.

Example: List all the EventListeners in a namespace

```
$ tkn eventlistener list -n myspace
```

5.3.11.5. eventlistener logs

Display logs of an EventListener.

Example: Display the logs of the mylistener EventListener in a namespace

```
$ tkn eventlistener logs mylistener -n myspace
```

5.3.11.6. triggerbinding

Manage TriggerBindings.

Example: Display TriggerBindings help

```
$ tkn triggerbinding -h
```

5.3.11.7. triggerbinding delete

Delete a TriggerBinding.

Example: Delete mybinding1 and mybinding2 TriggerBindings in a namespace

```
$ tkn triggerbinding delete mybinding1 mybinding2 -n myspace
```

5.3.11.8. triggerbinding describe

Describe a TriggerBinding.

Example: Describe the mybinding TriggerBinding in a namespace

```
$ tkn triggerbinding describe mybinding -n myspace
```

5.3.11.9. triggerbinding list

List TriggerBindings.

Example: List all the TriggerBindings in a namespace

```
$ tkn triggerbinding list -n myspace
```

5.3.11.10. triggertemplate

Manage TriggerTemplates.

Example: Display TriggerTemplate help

```
$ tkn triggertemplate -h
```

5.3.11.11. triggertemplate delete

Delete a TriggerTemplate.

Example: Delete mytemplate1 and mytemplate2 TriggerTemplates in a namespace

```
$ tkn triggertemplate delete mytemplate1 mytemplate2 -n `myspace`
```

5.3.11.12. triggertemplate describe

Describe a TriggerTemplate.

Example: Describe the mytemplate TriggerTemplate in a namespace

```
$ tkn triggertemplate describe mytemplate -n `myspace`
```

5.3.11.13. triggertemplate list

List TriggerTemplates.

Example: List all the TriggerTemplates in a namespace

```
$ tkn triggertemplate list -n myspace
```

5.3.11.14. clustertriggerbinding

Manage ClusterTriggerBindings.

Example: Display ClusterTriggerBindings help

```
$ tkn clustertriggerbinding -h
```

5.3.11.15. clustertriggerbinding delete

Delete a ClusterTriggerBinding.

Example: Delete myclusterbinding1 and myclusterbinding2 ClusterTriggerBindings

```
$ tkn clustertriggerbinding delete myclusterbinding1 myclusterbinding2
```

5.3.11.16. clustertriggerbinding describe

Describe a ClusterTriggerBinding.

Example: Describe the myclusterbinding ClusterTriggerBinding

```
$ tkn clustertriggerbinding describe myclusterbinding
```

5.3.11.17. clustertriggerbinding list

List ClusterTriggerBindings.

Example: List all ClusterTriggerBindings

```
$ tkn clustertriggerbinding list
```

5.3.12. Hub interaction commands

Interact with Tekton Hub for resources such as tasks and pipelines.

5.3.12.1. hub

Interact with hub.

Example: Display help

```
$ tkn hub -h
```

Example: Interact with a hub API server

```
$ tkn hub --api-server https://api.hub.tekton.dev
```

> **NOTE**
>
> For each example, to get the corresponding sub-commands and flags, run **tkn hub <command> --help**.

5.3.12.2. hub downgrade

Downgrade an installed resource.

Example: Downgrade the mytask task in the mynamespace namespace to its older version

```
$ tkn hub downgrade task mytask --to version -n mynamespace
```

5.3.12.3. hub get

Get a resource manifest by its name, kind, catalog, and version.

Example: Get the manifest for a specific version of the myresource pipeline or task from the tekton catalog

```
$ tkn hub get [pipeline | task] myresource --from tekton --version version
```

5.3.12.4. hub info

Display information about a resource by its name, kind, catalog, and version.

Example: Display information about a specific version of the `mytask` task from the `tekton` catalog

```
$ tkn hub info task mytask --from tekton --version version
```

5.3.12.5. hub install

Install a resource from a catalog by its kind, name, and version.

Example: Install a specific version of the `mytask` task from the `tekton` catalog in the `mynamespace` namespace

```
$ tkn hub install task mytask --from tekton --version version -n mynamespace
```

5.3.12.6. hub reinstall

Reinstall a resource by its kind and name.

Example: Reinstall a specific version of the `mytask` task from the `tekton` catalog in the `mynamespace` namespace

```
$ tkn hub reinstall task mytask --from tekton --version version -n mynamespace
```

5.3.12.7. hub search

Search a resource by a combination of name, kind, and tags.

Example: Search a resource with a tag `cli`

```
$ tkn hub search --tags cli
```

5.3.12.8. hub upgrade

Upgrade an installed resource.

Example: Upgrade the installed `mytask` task in the `mynamespace` namespace to a new version

```
$ tkn hub upgrade task mytask --to version -n mynamespace
```

CHAPTER 6. OPM CLI

6.1. INSTALLING THE OPM CLI

6.1.1. About the opm CLI

The **opm** CLI tool is provided by the Operator Framework for use with the Operator bundle format. This tool allows you to create and maintain catalogs of Operators from a list of Operator bundles that are similar to software repositories. The result is a container image which can be stored in a container registry and then installed on a cluster.

A catalog contains a database of pointers to Operator manifest content that can be queried through an included API that is served when the container image is run. On Red Hat OpenShift Service on AWS, Operator Lifecycle Manager (OLM) can reference the image in a catalog source, defined by a **CatalogSource** object, which polls the image at regular intervals to enable frequent updates to installed Operators on the cluster.

6.1.2. Installing the opm CLI

You can install the **opm** CLI tool on your Linux, macOS, or Windows workstation.

Prerequisites

- For Red Hat Enterprise Linux (RHEL) 9.0 and later, you must provide the following packages:
  - **podman** version 1.9.3+ (version 2.0+ recommended)
  - **glibc** version 2.28+

Procedure

1. Navigate to the OpenShift mirror site and download the latest version of the tarball that matches your operating system.
2. Unpack the archive.
   - For Linux or macOS:

     ```
     $ tar xvf <file>
     ```

   - For Windows, unzip the archive with a ZIP program.
3. Place the file anywhere in your **PATH**.
   - For Linux or macOS:
     a. Check your **PATH**:

     ```
     $ echo $PATH
     ```

     b. Move the file. For example:

     ```
     $ sudo mv ./opm /usr/local/bin/
     ```

- For Windows:
  a. Check your **PATH**:

```
C:\> path
```

  b. Move the file:

```
C:\> move opm.exe <directory>
```

Verification

- After you install the **opm** CLI, verify that it is available:

```
$ opm version
```

6.2. OPM CLI REFERENCE

The **opm** command-line interface (CLI) is a tool for creating and maintaining Operator catalogs.

opm CLI syntax

```
$ opm <command> [<subcommand>] [<argument>] [<flags>]
```

> **WARNING**
>
> The **opm** CLI is not forward compatible. The version of the **opm** CLI used to generate catalog content must be earlier than or equal to the version used to serve the content on a cluster.

Table 6.1. Global flags

Flag	Description
-skip-tls-verify	Skip TLS certificate verification for container image registries while pulling bundles or indexes.
--use-http	When you pull bundles, use plain HTTP for container image registries.

IMPORTANT

The SQLite-based catalog format, including the related CLI commands, is a deprecated feature. Deprecated functionality is still included in Red Hat OpenShift Service on AWS and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments.

For the most recent list of major functionality that has been deprecated or removed within Red Hat OpenShift Service on AWS, refer to the *Deprecated and removed features* section of the Red Hat OpenShift Service on AWS release notes.

6.2.1. generate

Generate various artifacts for declarative config indexes.

Command syntax

```
$ opm generate <subcommand> [<flags>]
```

Table 6.2. **generate** subcommands

Subcommand	Description
dockerfile	Generate a Dockerfile for a declarative config index.

Table 6.3. **generate** flags

Flags	Description
-h, **--help**	Help for generate.

6.2.1.1. dockerfile

Generate a Dockerfile for a declarative config index.

IMPORTANT

This command creates a Dockerfile in the same directory as the **<dcRootDir>** (named **<dcDirName>.Dockerfile**) that is used to build the index. If a Dockerfile with the same name already exists, this command fails.

When specifying extra labels, if duplicate keys exist, only the last value of each duplicate key gets added to the generated Dockerfile.

Command syntax

```
$ opm generate dockerfile <dcRootDir> [<flags>]
```

Table 6.4. **generate dockerfile** flags

Flag	Description
-i, --binary-image (string)	Image in which to build catalog. The default value is **quay.io/operator-framework/opm:latest**.
-l, --extra-labels (string)	Extra labels to include in the generated Dockerfile. Labels have the form **key=value**.
-h, --help	Help for Dockerfile.

NOTE

To build with the official Red Hat image, use the **registry.redhat.io/openshift4/ose-operator-registry-rhel9:v4** value with the **-i** flag.

6.2.2. index

Generate Operator index for SQLite database format container images from pre-existing Operator bundles.

IMPORTANT

As of Red Hat OpenShift Service on AWS 4.11, the default Red Hat-provided Operator catalog releases in the file-based catalog format. The default Red Hat-provided Operator catalogs for Red Hat OpenShift Service on AWS 4.6 through 4.10 released in the deprecated SQLite database format.

The **opm** subcommands, flags, and functionality related to the SQLite database format are also deprecated and will be removed in a future release. The features are still supported and must be used for catalogs that use the deprecated SQLite database format.

Many of the **opm** subcommands and flags for working with the SQLite database format, such as **opm index prune**, do not work with the file-based catalog format.

Command syntax

```
$ opm index <subcommand> [<flags>]
```

Table 6.5. **index** subcommands

Subcommand	Description
add	Add Operator bundles to an index.
prune	Prune an index of all but specified packages.
prune-stranded	Prune an index of stranded bundles, which are bundles that are not associated with a particular image.

Subcommand	Description
rm	Delete an entire Operator from an index.

6.2.2.1. add

Add Operator bundles to an index.

Command syntax

```
$ opm index add [<flags>]
```

Table 6.6. **index add** flags

Flag	Description
-i, **--binary-image**	Container image for on-image **opm** command
-u, **--build-tool** (string)	Tool to build container images: **podman** (the default value) or **docker**. Overrides part of the **--container-tool** flag.
-b, **--bundles** (strings)	Comma-separated list of bundles to add.
-c, **--container-tool** (string)	Tool to interact with container images, such as for saving and building: **docker** or **podman**.
-f, **--from-index** (string)	Previous index to add to.
--generate	If enabled, only creates the Dockerfile and saves it to local disk.
--mode (string)	Graph update mode that defines how channel graphs are updated: **replaces** (the default value), **semver**, or **semver-skippatch**.
-d, **--out-dockerfile** (string)	Optional: If generating the Dockerfile, specify a file name.
--permissive	Allow registry load errors.
-p, **--pull-tool** (string)	Tool to pull container images: **none** (the default value), **docker**, or **podman**. Overrides part of the **--container-tool** flag.
-t, **--tag** (string)	Custom tag for container image being built.

6.2.2.2. prune

Prune an index of all but specified packages.

Command syntax

```
$ opm index prune [<flags>]
```

Table 6.7. **index prune** flags

Flag	Description
-i, **--binary-image**	Container image for on-image **opm** command
-c, **--container-tool** (string)	Tool to interact with container images, such as for saving and building: **docker** or **podman**.
-f, **--from-index** (string)	Index to prune.
--generate	If enabled, only creates the Dockerfile and saves it to local disk.
-d, **--out-dockerfile** (string)	Optional: If generating the Dockerfile, specify a file name.
-p, **--packages** (strings)	Comma-separated list of packages to keep.
--permissive	Allow registry load errors.
-t, **--tag** (string)	Custom tag for container image being built.

6.2.2.3. prune-stranded

Prune an index of stranded bundles, which are bundles that are not associated with a particular image.

Command syntax

```
$ opm index prune-stranded [<flags>]
```

Table 6.8. **index prune-stranded** flags

Flag	Description
-i, **--binary-image**	Container image for on-image **opm** command
-c, **--container-tool** (string)	Tool to interact with container images, such as for saving and building: **docker** or **podman**.
-f, **--from-index** (string)	Index to prune.

Flag	Description
--generate	If enabled, only creates the Dockerfile and saves it to local disk.
-d, **--out-dockerfile** (string)	Optional: If generating the Dockerfile, specify a file name.
-p, **--packages** (strings)	Comma-separated list of packages to keep.
--permissive	Allow registry load errors.
-t, **--tag** (string)	Custom tag for container image being built.

6.2.2.4. rm

Delete an entire Operator from an index.

Command syntax

```
$ opm index rm [<flags>]
```

Table 6.9. **index rm** flags

Flag	Description
-i, **--binary-image**	Container image for on-image **opm** command
-u, **--build-tool** (string)	Tool to build container images: **podman** (the default value) or **docker**. Overrides part of the **--container-tool** flag.
-c, **--container-tool** (string)	Tool to interact with container images, such as for saving and building: **docker** or **podman**.
-f, **--from-index** (string)	Previous index to delete from.
--generate	If enabled, only creates the Dockerfile and saves it to local disk.
-o, **--operators** (strings)	Comma-separated list of Operators to delete.
-d, **--out-dockerfile** (string)	Optional: If generating the Dockerfile, specify a file name.

Flag	Description
-p, **--packages** (strings)	Comma-separated list of packages to keep.
--permissive	Allow registry load errors.
-p, **--pull-tool** (string)	Tool to pull container images: **none** (the default value), **docker**, or **podman**. Overrides part of the **--container-tool** flag.
-t, **--tag** (string)	Custom tag for container image being built.

6.2.3. init

Generate an **olm.package** declarative config blob.

Command syntax

```
$ opm init <package_name> [<flags>]
```

Table 6.10. **init** flags

Flag	Description
-c, **--default-channel** (string)	The channel that subscriptions will default to if unspecified.
-d, **--description** (string)	Path to the Operator's **README.md** or other documentation.
-i, **--icon** (string)	Path to package's icon.
-o, **--output** (string)	Output format: **json** (the default value) or **yaml**.

6.2.4. migrate

Migrate a SQLite database format index image or database file to a file-based catalog.

IMPORTANT

The SQLite-based catalog format, including the related CLI commands, is a deprecated feature. Deprecated functionality is still included in Red Hat OpenShift Service on AWS and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments.

For the most recent list of major functionality that has been deprecated or removed within Red Hat OpenShift Service on AWS, refer to the *Deprecated and removed features* section of the Red Hat OpenShift Service on AWS release notes.

Command syntax

```
$ opm migrate <index_ref> <output_dir> [<flags>]
```

Table 6.11. **migrate** flags

Flag	Description
-o, **--output** (string)	Output format: **json** (the default value) or **yaml**.

6.2.5. render

Generate a declarative config blob from the provided index images, bundle images, and SQLite database files.

Command syntax

```
$ opm render <index_image | bundle_image | sqlite_file> [<flags>]
```

Table 6.12. **render** flags

Flag	Description
-o, **--output** (string)	Output format: **json** (the default value) or **yaml**.

6.2.6. serve

Serve declarative configs via a GRPC server.

NOTE

The declarative config directory is loaded by the **serve** command at startup. Changes made to the declarative config after this command starts are not reflected in the served content.

Command syntax

```
$ opm serve <source_path> [<flags>]
```

Table 6.13. **serve** flags

Flag	Description
--cache-dir (string)	If this flag is set, it syncs and persists the server cache directory.
--cache-enforce-integrity	Exits with an error if the cache is not present or is invalidated. The default value is **true** when the **--cache-dir** flag is set and the **--cache-only** flag is **false**. Otherwise, the default is **false**.

Flag	Description
--cache-only	Syncs the serve cache and exits without serving.
--debug	Enables debug logging.
h, **--help**	Help for serve.
-p, **--port** (string)	The port number for the service. The default value is **50051**.
--pprof-addr (string)	The address of the startup profiling endpoint. The format is **Addr:Port**.
-t, **--termination-log** (string)	The path to a container termination log file. The default value is **/dev/termination-log**.

6.2.7. validate

Validate the declarative config JSON file(s) in a given directory.

Command syntax

```
$ opm validate <directory> [<flags>]
```

CHAPTER 7. ROSA CLI

7.1. GETTING STARTED WITH THE ROSA CLI

7.1.1. About the ROSA CLI

Use the ROSA command-line interface (CLI) (**rosa**) to create, update, manage, and delete Red Hat OpenShift Service on AWS clusters and resources.

7.1.2. Setting up the ROSA CLI

Use the following steps to install and configure the ROSA CLI (**rosa**) on your installation host.

Procedure

1. Install and configure the latest AWS CLI (**aws**).

   a. Follow the AWS Command Line Interface documentation to install and configure the AWS CLI for your operating system.
   Specify your **aws_access_key_id**, **aws_secret_access_key**, and **region** in the **.aws/credentials** file. See AWS Configuration basics in the AWS documentation.

   > **NOTE**
   >
   > You can optionally use the **AWS_DEFAULT_REGION** environment variable to set the default AWS region.

   b. Query the AWS API to verify if the AWS CLI is installed and configured correctly:

   ```
   $ aws sts get-caller-identity  --output text
   ```

   Example output

   ```
   <aws_account_id>    arn:aws:iam::<aws_account_id>:user/<username>  <aws_user_id>
   ```

2. Download the latest version of the ROSA CLI (**rosa**) for your operating system from the **Downloads** page on OpenShift Cluster Manager.

3. Extract the **rosa** binary file from the downloaded archive. The following example extracts the binary from a Linux tar archive:

   ```
   $ tar xvf rosa-linux.tar.gz
   ```

4. Add **rosa** to your path. In the following example, the **/usr/local/bin** directory is included in the path of the user:

   ```
   $ sudo mv rosa /usr/local/bin/rosa
   ```

5. Verify if the ROSA CLI is installed correctly by querying the **rosa** version:

   ```
   $ rosa version
   ```

Example output

```
1.2.15
Your ROSA CLI is up to date.
```

6. Optional: Enable tab completion for the ROSA CLI. With tab completion enabled, you can press the **Tab** key twice to automatically complete subcommands and receive command suggestions:

   - To enable persistent tab completion for Bash on a Linux host:

     a. Generate a **rosa** tab completion configuration file for Bash and save it to your **/etc/bash_completion.d/** directory:

     ```
     # rosa completion bash > /etc/bash_completion.d/rosa
     ```

     b. Open a new terminal to activate the configuration.

   - To enable persistent tab completion for Bash on a macOS host:

     a. Generate a **rosa** tab completion configuration file for Bash and save it to your **/usr/local/etc/bash_completion.d/** directory:

     ```
     $ rosa completion bash > /usr/local/etc/bash_completion.d/rosa
     ```

     b. Open a new terminal to activate the configuration.

   - To enable persistent tab completion for Zsh:

     a. If tab completion is not enabled for your Zsh environment, enable it by running the following command:

     ```
     $ echo "autoload -U compinit; compinit" >> ~/.zshrc
     ```

     b. Generate a **rosa** tab completion configuration file for Zsh and save it to the first directory in your functions path:

     ```
     $ rosa completion zsh > "${fpath[1]}/_rosa"
     ```

     c. Open a new terminal to activate the configuration.

   - To enable persistent tab completion for fish:

     a. Generate a **rosa** tab completion configuration file for fish and save it to your **~/.config/fish/completions/** directory:

     ```
     $ rosa completion fish > ~/.config/fish/completions/rosa.fish
     ```

     b. Open a new terminal to activate the configuration.

   - To enable persistent tab completion for PowerShell:

     a. Generate a **rosa** tab completion configuration file for PowerShell and save it to a file named **rosa.ps1**:

```
PS> rosa completion powershell | Out-String | Invoke-Expression
```

b. Source the **rosa.ps1** file from your PowerShell profile.

NOTE

For more information about configuring **rosa** tab completion, see the help menu by running the **rosa completion --help** command.

7.1.3. Configuring the ROSA CLI

Use the following commands to configure the ROSA command-line interface (CLI) (**rosa**).

7.1.3.1. login

There are several methods you can use to log in to your Red Hat account using the ROSA command-line interface (CLI) (**rosa**). These methods are described in detail below.

7.1.3.1.1. Authenticating the ROSA CLI with Red Hat single sign-on

You can log in to the ROSA CLI (**rosa**) with Red Hat single sign-on. Red Hat recommends using the **rosa** command line tool with Red Hat single sign-on, instead of using an offline authentication token.

An offline authentication token is long-lived, stored on your operating system, and cannot be revoked. These factors increase overall security risks and the likelihood of unauthorized access to your account.

Alternatively, authenticating with the Red Hat single sign-on method automatically sends your **rosa** instance a refresh token that is valid for 10 hours. This unique, temporary authorization code enhances security and reduces the risk of unauthorized access.

IMPORTANT

The method of authenticating using Red Hat single sign-on does not break any existing automations that rely on offline tokens. Red Hat recommends using services accounts for automation purposes. If you still need to use offline tokens for automation or other purposes, you can download the OpenShift Cluster Manager API token from the OpenShift Cluster Manager API Token page.

Use one of the following methods of authentication:

- If your system has a web browser, see the "Authenticating the ROSA CLI with a single sign-on authorization code" section to authenticate with Red Hat single sign-on.
- If you are working with containers, remote hosts, or other environments without a web browser, see the "Authenticating the ROSA CLI with a single sign-on device code" section to authenticate with Red Hat single sign-on.
- To authenticate the ROSA CLI using an offline token, see the "Authenticating the ROSA CLI with an offline token" section.

NOTE

Single sign-on authorization is supported with ROSA CLI (**rosa**) version 1.2.36 or later.

7.1.3.1.2. Authenticating the ROSA CLI with a single sign-on authorization code

- To log in to the ROSA CLI (**rosa**) with a Red Hat single sign-on authorization code, run the following command:

  Syntax

  ```
  $ rosa login --use-auth-code
  ```

  Running this command redirects you to the Red Hat single sign-on login. Log in with your Red Hat login or email.

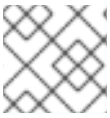
  Table 7.1. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

  To switch accounts, logout from https://sso.redhat.com and run the **rosa logout** command in your terminal before attempting to login again.

7.1.3.1.3. Authenticating the ROSA CLI with a single sign-on device code

If you are working with containers, remote hosts, and other environments without a web browser, you can use a Red Hat single sign-on device code for secure authentication. To do this, you must use a second device that has a web browser to approve the login.



NOTE

Single sign-on authorization is supported with ROSA CLI (**rosa**) version 1.2.36 or later.

- To log in to the ROSA CLI (**rosa**) with a Red Hat single sign-on device code, run the following command:

  Syntax

  ```
  $ rosa login --use-device-code
  ```

  Running this command will redirect you to the Red Hat SSO login and provide a log in code.

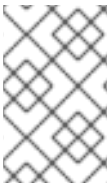
  Table 7.2. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

To switch accounts, logout from https://sso.redhat.com and run the **rosa logout** command in your terminal before attempting to login again.

7.1.3.1.4. Authenticating the ROSA CLI with an offline token

Log in to your Red Hat account, saving the credentials to the **rosa** configuration file.



NOTE

To use offline tokens for automation purposes, you can download the OpenShift Cluster Manager API token from the OpenShift Cluster Manager API Token page. To use service accounts for automation purposes, see the Service Accounts page.

IMPORTANT

Red Hat recommends using service accounts for automation purposes.

- To log in to ROSA CLI (**rosa**) with a Red Hat offline token, run the following command:

Syntax

```
$ rosa login [arguments]
```

Table 7.3. Arguments

Option	Definition
--client-id	The OpenID client identifier (string). Default: **cloud-services**
--client-secret	The OpenID client secret (string).
--insecure	Enables insecure communication with the server. This disables verification of TLS certificates and host names.
--scope	The OpenID scope (string). If this option is used, it replaces the default scopes. This can be repeated multiple times to specify multiple scopes. Default: **openid**
--token	Accesses or refreshes the token (string).
--token-url	The OpenID token URL (string). Default: **https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/token**

Table 7.4. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.

Option	Definition
--debug	Enables debug mode.
--profile	Specifies an AWS profile (string) from your credentials file.

7.1.3.2. logout

Log out of **rosa**. Logging out also removes the **rosa** configuration file.

Syntax

```
$ rosa logout [arguments]
```

Table 7.5. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.
--profile	Specifies an AWS profile (string) from your credentials file.

7.1.3.3. verify permissions

Verify that the AWS permissions required to create a Red Hat OpenShift Service on AWS cluster are configured correctly:

Syntax

```
$ rosa verify permissions [arguments]
```

NOTE

This command verifies permissions only for clusters that do not use the AWS Security Token Service (STS).

Table 7.6. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

Option	Definition
--region	The AWS region (string) in which to run the command. This value overrides the **AWS_REGION** environment variable.
--profile	Specifies an AWS profile (string) from your credentials file.

Examples

Verify that the AWS permissions are configured correctly:

```
$ rosa verify permissions
```

Verify that the AWS permissions are configured correctly in a specific region:

```
$ rosa verify permissions --region=us-west-2
```

7.1.3.4. verify quota

Verifies that AWS quotas are configured correctly for your default region.

Syntax

```
$ rosa verify quota [arguments]
```

Table 7.7. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.
--region	The AWS region (string) in which to run the command. This value overrides the **AWS_REGION** environment variable.
--profile	Specifies an AWS profile (string) from your credentials file.

Examples

Verify that the AWS quotas are configured correctly for the default region:

```
$ rosa verify quota
```

Verify that the AWS quotas are configured correctly in a specific region:

```
$ rosa verify quota --region=us-west-2
```

7.1.3.5. download rosa

Download the latest compatible version of the **rosa** CLI.

After you download **rosa**, extract the contents of the archive and add it to your path.

Syntax

```
$ rosa download rosa [arguments]
```

Table 7.8. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

7.1.3.6. download oc

Download the latest compatible version of the OpenShift Container Platform CLI (**oc**).

After you download **oc**, you must extract the contents of the archive and add it to your path.

Syntax

```
$ rosa download oc [arguments]
```

Table 7.9. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

Example

Download **oc** client tools:

```
$ rosa download oc
```

7.1.3.7. verify oc

Verifies that the OpenShift Container Platform CLI (**oc**) is installed correctly.

Syntax

```
$ rosa verify oc [arguments]
```

Table 7.10. Optional arguments inherited from parent commands

Option	Definition
--help	Shows help for this command.
--debug	Enables debug mode.

Example

Verify **oc** client tools:

```
$ rosa verify oc
```

Additional resources

- Setting up the ROSA CLI
- Getting started with the OpenShift CLI

7.1.4. Updating the ROSA CLI

Update to the latest compatible version of the ROSA CLI (**rosa**).

Procedure

1. Confirm that a new version of the ROSA CLI (**rosa**) is available:

   ```
   $ rosa version
   ```

   Example output

   ```
   1.2.12
   There is a newer release version '1.2.15', please consider updating:
   https://mirror.openshift.com/pub/openshift-v4/clients/rosa/latest/
   ```

2. Download the latest compatible version of the ROSA CLI:

   ```
   $ rosa download rosa
   ```

   This command downloads an archive called **rosa-*.tar.gz** into the current directory. The exact name of the file depends on your operating system and system architecture.

3. Extract the contents of the archive:

   ```
   $ tar -xzf rosa-linux.tar.gz
   ```

4. Install the new version of the ROSA CLI by moving the extracted file into your path. In the following example, the **/usr/local/bin** directory is included in the path of the user:

   ```
   $ sudo mv rosa /usr/local/bin/rosa
   ```

Verification

- Verify that the new version of the ROSA CLI is installed.

```
$ rosa version
```

Example output

```
1.2.15
Your ROSA CLI is up to date.
```

7.2. ROSA CLI COMMAND REFERENCE

This reference provides descriptions and example commands for ROSA CLI (**rosa**) commands.

Run **rosa -h** to list all commands or run **rosa <command> --help** to get additional details for a specific command.

7.2.1. ROSA CLI commands

This reference provides descriptions and example commands for ROSA CLI (**rosa**) commands.

Run **rosa -h** to list all commands or run **rosa <command> --help** to get additional details for a specific command.

7.2.1.1. rosa create account-roles

Create account-wide IAM roles before creating your cluster.

Example usage

```
# Create default account roles for ROSA clusters using STS
  rosa create account-roles

  # Create account roles with a specific permissions boundary
  rosa create account-roles --permissions-boundary arn:aws:iam::123456789012:policy/perm-
boundary
```

7.2.1.2. rosa create admin

Creates an admin user to login to the cluster

Example usage

```
# Create an admin user to login to the cluster
  rosa create admin -c mycluster -p MasterKey123
```

7.2.1.3. rosa create autoscaler

Create an autoscaler for a cluster

Example usage

```
# Interactively create an autoscaler to a cluster named "mycluster"
  rosa create autoscaler --cluster=mycluster --interactive

  # Create a cluster-autoscaler where it should skip nodes with local storage
  rosa create autoscaler --cluster=mycluster --skip-nodes-with-local-storage

  # Create a cluster-autoscaler with log verbosity of '3'
  rosa create autoscaler --cluster=mycluster --log-verbosity 3

  # Create a cluster-autoscaler with total CPU constraints
  rosa create autoscaler --cluster=mycluster --min-cores 10 --max-cores 100
```

7.2.1.4. rosa create break-glass-credential

Create a break glass credential for a cluster.

Example usage

```
# Interactively create a break glass credential to a cluster named "mycluster"
  rosa create break-glass-credential --cluster=mycluster --interactive
```

7.2.1.5. rosa create cluster

Create cluster

Example usage

```
# Create a cluster named "mycluster"
  rosa create cluster --cluster-name=mycluster

  # Create a cluster in the us-east-2 region
  rosa create cluster --cluster-name=mycluster --region=us-east-2
```

7.2.1.6. rosa create decision

Create a decision for an Access Request

Example usage

```
# Create a decision for an Access Request to approve it
  rosa create decision --access-request <access_request_id> --decision Approved
```

7.2.1.7. rosa create dns-domain

Create DNS Domain.

Example usage

```
# Create DNS Domain
 rosa create dns-domain
```

7.2.1.8. rosa create external-auth-provider

Create an external authentication provider for a cluster.

Example usage

```
# Interactively create an external authentication provider to a cluster named "mycluster"
  rosa create external-auth-provider --cluster=mycluster --interactive
```

7.2.1.9. rosa create iamserviceaccount

Create IAM role for Kubernetes service account

Example usage

```
# Create an IAM role for a service account
  rosa create iamserviceaccount --cluster my-cluster --name my-app --namespace default
```

7.2.1.10. rosa create idp

Add IDP for cluster

Example usage

```
# Add a GitHub identity provider to a cluster named "mycluster"
  rosa create idp --type=github --cluster=mycluster

  # Add an identity provider following interactive prompts
  rosa create idp --cluster=mycluster --interactive
```

7.2.1.11. rosa create image-mirror

Create image mirror for a cluster

Example usage

```
# Create an image mirror for cluster "mycluster"
  rosa create image-mirror --cluster=mycluster \
    --source=registry.example.com/team \
    --mirrors=mirror.corp.com/team,backup.corp.com/team

  # Create with a specific type (digest is default and only supported type)
  rosa create image-mirror --cluster=mycluster \
    --type=digest --source=docker.io/library \
    --mirrors=internal-registry.company.com/dockerhub
```

7.2.1.12. rosa create kubeletconfig

Create a custom kubeletconfig for a cluster

Example usage

```
# Create a custom kubeletconfig with a pod-pids-limit of 5000
  rosa create kubeletconfig --cluster=mycluster --pod-pids-limit=5000
```

7.2.1.13. rosa create log-forwarder

Create a log forwarder for a Hosted Control Plane cluster

Example usage

```
# Create a log forwarder using a config file
  rosa create log-forwarder -c mycluster-hcp --log-fwd-config=s3.yml

  # Create a log forwarder interactively
  rosa create log-forwarder -c mycluster-hcp --interactive
```

7.2.1.14. rosa create machinepool

Add machine pool to cluster

Example usage

```
# Interactively add a machine pool to a cluster named "mycluster"
  rosa create machinepool --cluster=mycluster --interactive
  # Add a machine pool mp-1 with 3 replicas of m5.xlarge to a cluster
  rosa create machinepool --cluster=mycluster --name=mp-1 --replicas=3 --instance-type=m5.xlarge
  # Add a machine pool mp-1 with autoscaling enabled and 3 to 6 replicas of m5.xlarge to a cluster
  rosa create machinepool --cluster=mycluster --name=mp-1 --enable-autoscaling \
 --min-replicas=3 --max-replicas=6 --instance-type=m5.xlarge
  # Add a machine pool with labels to a cluster
  rosa create machinepool -c mycluster --name=mp-1 --replicas=2 --instance-type=r5.2xlarge --
labels=foo=bar,bar=baz,
  # Add a machine pool with spot instances to a cluster
  rosa create machinepool -c mycluster --name=mp-1 --replicas=2 --instance-type=r5.2xlarge --use-
spot-instances \
    --spot-max-price=0.5
  # Add a machine pool to a cluster and set the node drain grace period
  rosa create machinepool -c mycluster --name=mp-1 --node-drain-grace-period="90 minutes"
```

7.2.1.15. rosa create network

Network AWS cloudformation stack

Example usage

```
# Create a AWS cloudformation stack
  rosa create network <template-name> --param Param1=Value1 --param Param2=Value2

  # ROSA quick start HCP VPC example with one availability zone
  rosa create network rosa-quickstart-default-vpc --param Region=us-west-2 --param
Name=quickstart-stack --param AvailabilityZoneCount=1 --param VpcCidr=10.0.0.0/16

  # ROSA quick start HCP VPC example with two explicit availability zones
  rosa create network rosa-quickstart-default-vpc --param Region=us-west-2 --param
```

```
Name=quickstart-stack --param AZ1=us-west-2b --param AZ2=us-west-2d --param
VpcCidr=10.0.0.0/16

  # To delete the AWS cloudformation stack
  aws cloudformation delete-stack --stack-name <name> --region <region>

# TEMPLATE_NAME:
Specifies the name of the template to use. This should match the name of a directory
under the path specified by '--template-dir' or the 'OCM_TEMPLATE_DIR' environment variable.
The directory should contain a YAML file defining the custom template structure.

If no TEMPLATE_NAME is provided, or if no matching directory is found, the default
built-in template 'rosa-quickstart-default-vpc' will be used.
```

7.2.1.16. rosa create ocm-role

Create role used by OCM

Example usage

```
# Create default ocm role for ROSA clusters using STS
  rosa create ocm-role

  # Create ocm role with a specific permissions boundary
  rosa create ocm-role --permissions-boundary arn:aws:iam::123456789012:policy/perm-boundary
```

7.2.1.17. rosa create oidc-config

Create OIDC config compliant with OIDC protocol.

Example usage

```
# Create OIDC config
 rosa create oidc-config
```

7.2.1.18. rosa create oidc-provider

Create OIDC provider for an STS cluster.

Example usage

```
# Create OIDC provider for cluster named "mycluster"
  rosa create oidc-provider --cluster=mycluster
```

7.2.1.19. rosa create operator-roles

Create operator IAM roles for a cluster.

Example usage

```
# Create default operator roles for cluster named "mycluster"
  rosa create operator-roles --cluster=mycluster
```

```
# Create operator roles with a specific permissions boundary
rosa create operator-roles -c mycluster --permissions-boundary
arn:aws:iam::123456789012:policy/perm-boundary
```

7.2.1.20. rosa create tuning-configs

Add tuning config

Example usage

```
# Add a tuning config with name "tuned1" and spec from a file "file1" to a cluster named "mycluster"
rosa create tuning-config --name=tuned1 --spec-path=file1 --cluster=mycluster"
```

7.2.1.21. rosa create user-role

Create user role to verify account association

Example usage

```
# Create user roles
rosa create user-role

# Create user role with a specific permissions boundary
rosa create user-role --permissions-boundary arn:aws:iam::123456789012:policy/perm-boundary
```

7.2.1.22. rosa delete account-roles

Delete Account Roles

Example usage

```
# Delete Account roles"
rosa delete account-roles -p prefix
```

7.2.1.23. rosa delete admin

Deletes the admin user

Example usage

```
# Delete the admin user
rosa delete admin --cluster=mycluster
```

7.2.1.24. rosa delete autoscaler

Delete autoscaler for cluster

Example usage

```
# Delete the autoscaler config for cluster named "mycluster"
rosa delete autoscaler --cluster=mycluster
```

7.2.1.25. rosa delete cluster

Delete cluster

Example usage

```
# Delete a cluster named "mycluster"
  rosa delete cluster --cluster=mycluster
```

7.2.1.26. rosa delete dns-domain

Delete DNS domain

Example usage

```
# Delete a DNS domain with ID github-1
  rosa delete dns-domain github-1
```

7.2.1.27. rosa delete external-auth-provider

Delete external authentication provider

Example usage

```
# Delete an external authentication provider named exauth-1
  rosa delete external-auth-provider exauth-1  --cluster=mycluster
```

7.2.1.28. rosa delete iamserviceaccount

Delete IAM role for Kubernetes service account

Example usage

```
# Delete IAM role for service account
  rosa delete iamserviceaccount --cluster my-cluster \
    --name my-app \
    --namespace default
```

7.2.1.29. rosa delete idp

Delete cluster IDPs

Example usage

```
# Delete an identity provider named github-1
  rosa delete idp github-1 --cluster=mycluster
```

7.2.1.30. rosa delete image-mirror

Delete image mirror from a cluster

Example usage

```
# Delete image mirror with ID "abc123" from cluster "mycluster"
  rosa delete image-mirror --cluster=mycluster abc123

  # Delete without confirmation prompt
  rosa delete image-mirror --cluster=mycluster abc123 --yes

  # Alternative: using the --id flag
  rosa delete image-mirror --cluster=mycluster --id=abc123
```

7.2.1.31. rosa delete ingress

Delete cluster ingress

Example usage

```
# Delete ingress with ID a1b2 from a cluster named 'mycluster'
  rosa delete ingress --cluster=mycluster a1b2

  # Delete secondary ingress using the sub-domain name
  rosa delete ingress --cluster=mycluster apps2
```

7.2.1.32. rosa delete kubeletconfig

Delete a kubeletconfig from a cluster

Example usage

```
# Delete the KubeletConfig for ROSA Classic cluster 'foo'
  rosa delete kubeletconfig --cluster foo
  # Delete the KubeletConfig named 'bar' from cluster 'foo'
  rosa delete kubeletconfig --cluster foo --name bar
```

7.2.1.33. rosa delete log-forwarder

Delete log forwarder

Example usage

```
# Delete log forwarder with ID 'example-id' from a cluster named 'mycluster-hcp'
  rosa delete log-forwarder --cluster=mycluster-hcp example-id
```

7.2.1.34. rosa delete machinepool

Delete machine pool

Example usage

```
# Delete machine pool with ID mp-1 from a cluster named 'mycluster'
  rosa delete machinepool --cluster=mycluster mp-1
```

7.2.1.35. rosa delete ocm-role

Delete OCM role

Example usage

```
# Delete OCM role
rosa delete ocm-role --role-arn arn:aws:iam::123456789012:role/xxx-OCM-Role-1223456778
```

7.2.1.36. rosa delete oidc-config

Delete OIDC Config

Example usage

```
# Delete OIDC config based on registered OIDC Config ID that has been supplied
 rosa delete oidc-config --oidc-config-id <oidc_config_id>
```

7.2.1.37. rosa delete oidc-provider

Delete OIDC Provider

Example usage

```
# Delete OIDC provider for cluster named "mycluster"
  rosa delete oidc-provider --cluster=mycluster
```

7.2.1.38. rosa delete operator-roles

Delete Operator Roles

Example usage

```
# Delete Operator roles for cluster named "mycluster"
  rosa delete operator-roles --cluster=mycluster
```

7.2.1.39. rosa delete tuning-configs

Delete tuning config

Example usage

```
# Delete tuning config with name tuned1 from a cluster named 'mycluster'
  rosa delete tuning-config --cluster=mycluster tuned1
```

7.2.1.40. rosa delete user-role

Delete user role

Example usage

```
# Delete user role
rosa delete user-role --role-arn {prefix}-User-{username}-Role
```

7.2.1.41. rosa describe access-request

Show details of an Access Request

Example usage

```
# Describe an Access Request wit id <access_request_id>
  rosa describe access-request --id <access_request_id>
```

7.2.1.42. rosa describe addon

Show details of an add-on

Example usage

```
# Describe an add-on named "codeready-workspaces"
  rosa describe addon codeready-workspaces
```

7.2.1.43. rosa describe addon-installation

Show details of an add-on installation

Example usage

```
# Describe the 'bar' add-on installation on cluster 'foo'
  rosa describe addon-installation --cluster foo --addon bar
```

7.2.1.44. rosa describe admin

Show details of the cluster-admin user

Example usage

```
# Describe cluster-admin user of a cluster named mycluster
  rosa describe admin -c mycluster
```

7.2.1.45. rosa describe autoscaler

Show details of the autoscaler for a cluster

Example usage

```
# Describe the autoscaler for cluster 'foo'
rosa describe autoscaler --cluster foo
```

7.2.1.46. rosa describe break-glass-credential

Show details of a break glass credential on a cluster

Example usage

```
# Show details of a break glass credential with ID "12345" on a cluster named "mycluster"
  rosa describe break-glass-credential 12345 --cluster=mycluster
```

7.2.1.47. rosa describe cluster

Show details of a cluster

Example usage

```
# Describe a cluster named "mycluster"
  rosa describe cluster --cluster=mycluster
```

7.2.1.48. rosa describe external-auth-provider

Show details of an external authentication provider on a cluster

Example usage

```
# Show details of an external authentication provider named "exauth" on a cluster named "mycluster"
  rosa describe external-auth-provider exauth --cluster=mycluster
```

7.2.1.49. rosa describe iamserviceaccount

Describe IAM role for Kubernetes service account

Example usage

```
# Describe IAM role for service account
  rosa describe iamserviceaccount --cluster my-cluster \
    --name my-app \
    --namespace default
```

7.2.1.50. rosa describe ingress

Show details of the specified ingress within cluster

Example usage

```
rosa describe ingress <ingress_id> -c mycluster
```

7.2.1.51. rosa describe kubeletconfig

Show details of a kubeletconfig for a cluster

Example usage

```
# Describe the custom kubeletconfig for ROSA Classic cluster 'foo'
  rosa describe kubeletconfig --cluster foo
  # Describe the custom kubeletconfig named 'bar' for cluster 'foo'
  rosa describe kubeletconfig --cluster foo --name bar
```

7.2.1.52. rosa describe log-forwarder

Show details of a specific log forwarder used by a cluster

Example usage

```
rosa describe log-forwarder <log_fwd_id> -c mycluster-hcp
```

7.2.1.53. rosa describe machinepool

Show details of a machine pool on a cluster

Example usage

```
# Show details of a machine pool named "mymachinepool" on a cluster named "mycluster"
  rosa describe machinepool --cluster=mycluster --machinepool=mymachinepool
```

7.2.1.54. rosa describe tuning-configs

Show details of tuning config

Example usage

```
# Describe the 'tuned1' tuned config on cluster 'foo'
  rosa describe tuning-config --cluster foo tuned1
```

7.2.1.55. rosa describe upgrade

Show details of an upgrade

Example usage

```
# Describe an upgrade-policy"
  rosa describe upgrade
```

7.2.1.56. rosa download openshift-client

Download OpenShift client tools

Example usage

```
# Download oc client tools
  rosa download oc
```

7.2.1.57. rosa download rosa-client

Download ROSA client tools

Example usage

```
# Download rosa client tools
  rosa download rosa
```

7.2.1.58. rosa edit addon

Edit add-on installation parameters on cluster

Example usage

```
# Edit the parameters of the Red Hat OpenShift logging operator add-on installation
  rosa edit addon --cluster=mycluster cluster-logging-operator
```

7.2.1.59. rosa edit autoscaler

Edit the autoscaler of a cluster

Example usage

```
# Interactively edit an autoscaler to a cluster named "mycluster"
  rosa edit autoscaler --cluster=mycluster --interactive

  # Edit a cluster-autoscaler to skip nodes with local storage
  rosa edit autoscaler --cluster=mycluster --skip-nodes-with-local-storage

  # Edit a cluster-autoscaler with log verbosity of '3'
  rosa edit autoscaler --cluster=mycluster --log-verbosity 3

  # Edit a cluster-autoscaler with total CPU constraints
  rosa edit autoscaler --cluster=mycluster --min-cores 10 --max-cores 100
```

7.2.1.60. rosa edit cluster

Edit cluster

Example usage

```
# Edit a cluster named "mycluster" to make it private
  rosa edit cluster -c mycluster --private

  # Edit all options interactively
  rosa edit cluster -c mycluster --interactive
```

7.2.1.61. rosa edit image-mirror

Edit image mirror for a cluster

Example usage

```
# Update mirrors for image mirror with ID "abc123" on cluster "mycluster"
  rosa edit image-mirror --cluster=mycluster abc123 \
    --mirrors=mirror.corp.com/team,backup.corp.com/team,new-mirror.corp.com/team
```

```
# Alternative: using the --id flag
rosa edit image-mirror --cluster=mycluster --id=abc123 \
  --mirrors=mirror.corp.com/team,backup.corp.com/team,new-mirror.corp.com/team
```

7.2.1.62. rosa edit ingress

Edit a cluster ingress (load balancer)

Example usage

```
# Make additional ingress with ID 'a1b2' private on a cluster named 'mycluster'
  rosa edit ingress --private --cluster=mycluster a1b2

  # Update the router selectors for the additional ingress with ID 'a1b2'
  rosa edit ingress --label-match=foo=bar --cluster=mycluster a1b2

  # Update the default ingress using the sub-domain identifier
  rosa edit ingress --private=false --cluster=mycluster apps

  # Update the load balancer type of the apps2 ingress
  rosa edit ingress --lb-type=nlb --cluster=mycluster apps2
```

7.2.1.63. rosa edit kubeletconfig

Edit a kubeletconfig for a cluster

Example usage

```
# Edit a KubeletConfig to have a pod-pids-limit of 10000
  rosa edit kubeletconfig --cluster=mycluster --pod-pids-limit=10000
  # Edit a KubeletConfig named 'bar' to have a pod-pids-limit of 10000
  rosa edit kubeletconfig --cluster=mycluster --name=bar --pod-pids-limit=10000
```

7.2.1.64. rosa edit machinepool

Edit machine pool

Example usage

```
# Set 4 replicas on machine pool 'mp1' on cluster 'mycluster'
 rosa edit machinepool --replicas=4 --cluster=mycluster mp1
 # Enable autoscaling and Set 3-5 replicas on machine pool 'mp1' on cluster 'mycluster'
 rosa edit machinepool --enable-autoscaling --min-replicas=3 --max-replicas=5 --cluster=mycluster
mp1
 # Set the node drain grace period to 1 hour on machine pool 'mp1' on cluster 'mycluster'
 rosa edit machinepool --node-drain-grace-period="1 hour" --cluster=mycluster mp1
```

7.2.1.65. rosa edit tuning-configs

Edit tuning config

Example usage

```
# Update the tuning config with name 'tuning-1' with the spec defined in file1
  rosa edit tuning-config --cluster=mycluster tuning-1 --spec-path file1
```

7.2.1.66. rosa grant user

Grant user access to cluster

Example usage

```
# Add cluster-admin role to a user
  rosa grant user cluster-admin --user=myusername --cluster=mycluster

  # Grant dedicated-admins role to a user
  rosa grant user dedicated-admin --user=myusername --cluster=mycluster
```

7.2.1.67. rosa init

Applies templates to support Red Hat OpenShift Service on AWS

Example usage

```
# Configure your AWS account to allow IAM (non-STS) ROSA clusters
  rosa init

  # Configure a new AWS account using pre-existing OCM credentials
  rosa init --token=$OFFLINE_ACCESS_TOKEN
```

7.2.1.68. rosa install addon

Install add-ons on cluster

Example usage

```
# Add the CodeReady Workspaces add-on installation to the cluster
  rosa install addon --cluster=mycluster codeready-workspaces
```

7.2.1.69. rosa link ocm-role

Link OCM role to specific OCM organization.

Example usage

```
# Link OCM role
  rosa link ocm-role --role-arn arn:aws:iam::123456789012:role/ManagedOpenshift-OCM-Role
```

7.2.1.70. rosa link user-role

Link user role to specific OCM account.

Example usage

```
# Link user roles
  rosa link user-role --role-arn arn:aws:iam::{accountid}:role/{prefix}-User-{username}-Role
```

7.2.1.71. rosa list access-request

List Access Requests

Example usage

```
# List all Access Requests for cluster 'foo'
  rosa list access-request --cluster foo
```

7.2.1.72. rosa list account-roles

List account roles and policies

Example usage

```
# List all account roles
  rosa list account-roles
```

7.2.1.73. rosa list addons

List add-on installations

Example usage

```
# List all add-on installations on a cluster named "mycluster"
  rosa list addons --cluster=mycluster
```

7.2.1.74. rosa list break-glass-credentials

List break glass credential

Example usage

```
# List all break glass credentials for a cluster named 'mycluster'"
  rosa list break-glass-credentials -c mycluster
```

7.2.1.75. rosa list clusters

List clusters

Example usage

```
# List all clusters
  rosa list clusters
```

7.2.1.76. rosa list dns-domain

List DNS Domains

Example usage

```
# List all DNS Domains tied to your organization ID"
  rosa list dns-domain
```

7.2.1.77. rosa list external-auth-providers

List external authentication provider

Example usage

```
# List all external authentication providers for a cluster named 'mycluster'"
  rosa list external-auth-provider -c mycluster
```

7.2.1.78. rosa list gates

List available OCP Gates

Example usage

```
# List all OCP gates for OCP version
  rosa list gates --version 4.9

  # List all STS gates for OCP version
  rosa list gates --gate sts --version 4.9

  # List all OCP gates for OCP version
  rosa list gates --gate ocp --version 4.9

  # List available gates for cluster upgrade version
  rosa list gates -c <cluster_id> --version 4.9.15
```

7.2.1.79. rosa list iamserviceaccounts

List IAM roles for Kubernetes service accounts

Example usage

```
# List IAM roles for service accounts
  rosa list iamserviceaccounts --cluster my-cluster
```

7.2.1.80. rosa list idps

List cluster IDPs

Example usage

```
# List all identity providers on a cluster named "mycluster"
  rosa list idps --cluster=mycluster
```

7.2.1.81. rosa list image-mirrors

List cluster image mirrors

Example usage

```
# List all image mirrors on a cluster named "mycluster"
  rosa list image-mirrors --cluster=mycluster
```

7.2.1.82. rosa list ingresses

List cluster Ingresses

Example usage

```
# List all routes on a cluster named "mycluster"
  rosa list ingresses --cluster=mycluster
```

7.2.1.83. rosa list instance-types

List Instance types

Example usage

```
# List all instance types
 rosa list instance-types
```

7.2.1.84. rosa list kubeletconfigs

List kubeletconfigs

Example usage

```
# List the kubeletconfigs for cluster 'foo'
rosa list kubeletconfig --cluster foo
```

7.2.1.85. rosa list log-forwarders

List cluster log forwarders

Example usage

```
# List all log forwarders on a cluster named "mycluster": rosa list log-forwarders --cluster=mycluster
```

7.2.1.86. rosa list machinepools

List cluster machine pools

Example usage

```
# List all machine pools on a cluster named "mycluster"
  rosa list machinepools --cluster=mycluster
```

```
# List machine pools showing all information
rosa list machinepools --cluster=mycluster --all
```

7.2.1.87. rosa list ocm-roles

List ocm roles

Example usage

```
# List all ocm roles
rosa list ocm-roles
```

7.2.1.88. rosa list oidc-config

List OIDC Configuration resources

Example usage

```
# List all OIDC Configurations tied to your organization ID"
  rosa list oidc-config
```

7.2.1.89. rosa list oidc-providers

List OIDC providers

Example usage

```
# List all oidc providers
  rosa list oidc-providers
```

7.2.1.90. rosa list operator-roles

List operator roles and policies

Example usage

```
# List all operator roles
  rosa list operator-roles
```

7.2.1.91. rosa list regions

List available regions

Example usage

```
# List all available regions
  rosa list regions
```

7.2.1.92. rosa list tuning-configs

List tuning configs

Example usage

```
# List all tuning configuration for a cluster named 'mycluster'"
  rosa list tuning-configs -c mycluster
```

7.2.1.93. rosa list user-roles

List user roles

Example usage

```
# List all user roles
rosa list user-roles
```

7.2.1.94. rosa list users

List cluster users

Example usage

```
# List all users on a cluster named "mycluster"
  rosa list users --cluster=mycluster
```

7.2.1.95. rosa list versions

List available versions

Example usage

```
# List all OpenShift versions
  rosa list versions
```

7.2.1.96. rosa login

Log in to your Red Hat account

Example usage

```
# Login to the OpenShift API with an existing token generated from
https://console.redhat.com/openshift/token/rosa
  rosa login --token=$OFFLINE_ACCESS_TOKEN
```

7.2.1.97. rosa logs

Show installation or uninstallation logs for a cluster

Example usage

```
# Show install logs for a cluster named 'mycluster'
  rosa logs install --cluster=mycluster
```

```
# Show uninstall logs for a cluster named 'mycluster'
rosa logs uninstall --cluster=mycluster
```

7.2.1.98. rosa logs install

Show cluster installation logs

Example usage

```
# Show last 100 install log lines for a cluster named "mycluster"
rosa logs install mycluster --tail=100

# Show install logs for a cluster using the --cluster flag
rosa logs install --cluster=mycluster
```

7.2.1.99. rosa logs uninstall

Show cluster uninstallation logs

Example usage

```
# Show last 100 uninstall log lines for a cluster named "mycluster"
rosa logs uninstall mycluster --tail=100

# Show uninstall logs for a cluster using the --cluster flag
rosa logs uninstall --cluster=mycluster
```

7.2.1.100. rosa register oidc-config

Registers unmanaged OIDC config with Openshift Clusters Manager.

Example usage

```
# Register OIDC config
rosa register oidc-config
```

7.2.1.101. rosa revoke break-glass-credentials

Revoke break glass credentials

Example usage

```
# Revoke all break glass credentials
rosa revoke break-glass-credentials --cluster=mycluster
```

7.2.1.102. rosa revoke user

Revoke role from users

Example usage

```
# Revoke cluster-admin role from a user
```

```
rosa revoke user cluster-admins --user=myusername --cluster=mycluster

# Revoke dedicated-admin role from a user
rosa revoke user dedicated-admins --user=myusername --cluster=mycluster
```

7.2.1.103. rosa uninstall addon

Uninstall add-on from cluster

Example usage

```
# Remove the CodeReady Workspaces add-on installation from the cluster
rosa uninstall addon --cluster=mycluster codeready-workspaces
```

7.2.1.104. rosa unlink ocm-role

Unlink ocm role from a specific OCM organization

Example usage

```
#Unlink ocm role
rosa unlink ocm-role --role-arn arn:aws:iam::123456789012:role/ManagedOpenshift-OCM-Role
```

7.2.1.105. rosa unlink user-role

Unlink user role from a specific OCM account

Example usage

```
# Unlink user role
rosa unlink user-role --role-arn arn:aws:iam::{accountid}:role/{prefix}-User-{username}-Role
```

7.2.1.106. rosa upgrade account-roles

Upgrade account-wide IAM roles to the latest version.

Example usage

```
# Upgrade account roles for ROSA STS clusters
rosa upgrade account-roles
```

7.2.1.107. rosa upgrade cluster

Upgrade cluster

Example usage

```
# Interactively schedule an upgrade on the cluster named "mycluster"
rosa upgrade cluster --cluster=mycluster --interactive

# Schedule a cluster upgrade within the hour
rosa upgrade cluster -c mycluster --version 4.12.20
```

```
# Check if any gates need to be acknowledged prior to attempting an upgrading
rosa upgrade cluster -c mycluster --version 4.12.20 --dry-run
```

7.2.1.108. rosa upgrade machinepool

Upgrade machinepool

Example usage

```
# Interactively schedule an upgrade on the cluster named "mycluster"" for a machinepool named "np1"
rosa upgrade machinepool np1 --cluster=mycluster --interactive

# Schedule a machinepool upgrade within the hour
rosa upgrade machinepool np1 -c mycluster --version 4.12.20
```

7.2.1.109. rosa upgrade operator-roles

Upgrade operator IAM roles for a cluster.

Example usage

```
# Upgrade cluster-specific operator IAM roles
rosa upgrade operators-roles
```

7.2.1.110. rosa upgrade roles

Upgrade cluster-specific IAM roles to the latest version.

Example usage

```
# Upgrade cluster roles for ROSA STS clusters
rosa upgrade roles -c <cluster_key>
```

7.2.1.111. rosa verify network

Verify VPC subnets are configured correctly

Example usage

```
# Verify two subnets
rosa verify network --subnet-ids subnet-03046a9b92b5014fb,subnet-03046a9c92b5014fb
```

7.2.1.112. rosa verify openshift-client

Verify OpenShift client tools

Example usage

```
# Verify oc client tools
rosa verify oc
```

7.2.1.113. rosa verify permissions

Verify AWS permissions are ok for non-STS cluster install

Example usage

```
# Verify AWS permissions are configured correctly
rosa verify permissions

# Verify AWS permissions in a different region
rosa verify permissions --region=us-west-2
```

7.2.1.114. rosa verify quota

Verify AWS quota is ok for cluster install

Example usage

```
# Verify AWS quotas are configured correctly
rosa verify quota

# Verify AWS quotas in a different region
rosa verify quota --region=us-west-2
```

7.2.1.115. rosa verify rosa-client

Verify ROSA client tools

Example usage

```
# Verify rosa client tools
rosa verify rosa
```

7.2.1.116. rosa whoami

Displays user account information

Example usage

```
# Displays user information
rosa whoami
```

7.3. LEAST PRIVILEGE PERMISSIONS FOR ROSA CLI COMMANDS

You can create roles with permissions that adhere to the principal of least privilege, in which the users assigned the roles have no other permissions assigned to them outside the scope of the specific action they need to perform. These policies contain only the minimum required permissions needed to perform specific actions by using the ROSA command-line interface (CLI) (**rosa**).

IMPORTANT

Although the policies and commands presented in this topic will work in conjunction with one another, you might have other restrictions within your AWS environment that make the policies for these commands insufficient for your specific needs. Red Hat provides these examples as a baseline, assuming no other AWS Identity and Access Management (IAM) restrictions are present.

For more information about configuring permissions, policies, and roles in the AWS console, see AWS Identity and Access Management in the AWS documentation.

7.3.1. Least privilege permissions for common Red Hat OpenShift Service on AWS CLI commands

The following examples show the least privilege permissions needed for the most common ROSA CLI commands when building Red Hat OpenShift Service on AWS clusters.

7.3.1.1. Create a managed OpenID Connect (OIDC) provider

Run the following command with the specified permissions to create your managed OIDC provider by using **auto** mode.

Input

```
$ rosa create oidc-config --mode auto
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOidcConfig",
            "Effect": "Allow",
            "Action": [
                "iam:TagOpenIDConnectProvider",
                "iam:CreateOpenIDConnectProvider"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.2. Create an unmanaged OpenID Connect provider

Run the following command with the specified permissions to create your unmanaged OIDC provider by using **auto** mode.

Input

```
$ rosa create oidc-config --mode auto --managed=false
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:TagOpenIDConnectProvider",
                "iam:ListRoleTags",
                "iam:ListRoles",
                "iam:CreateOpenIDConnectProvider",
                "s3:CreateBucket",
                "s3:PutObject",
                "s3:PutBucketTagging",
                "s3:PutBucketPolicy",
                "s3:PutObjectTagging",
                "s3:PutBucketPublicAccessBlock",
                "secretsmanager:CreateSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.3. List your account roles

Run the following command with the specified permissions to list your account roles.

Input

```
$ rosa list account-roles
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAccountRoles",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoleTags",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.4. List your Operator roles

Run the following command with the specified permissions to list your Operator roles.

Input

```
$ rosa list operator-roles
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListOperatorRoles",
            "Effect": "Allow",
            "Action": [
                "iam:ListRoleTags",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:ListPolicyTags"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.5. List your OIDC providers

Run the following command with the specified permissions to list your OIDC providers.

Input

```
$ rosa list oidc-providers
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListOidcProviders",
            "Effect": "Allow",
            "Action": [
                "iam:ListOpenIDConnectProviders",
                "iam:ListOpenIDConnectProviderTags"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.6. Verify your quota

Run the following command with the specified permissions to verify your quota.

Input

```
$ rosa verify quota
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VerifyQuota",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeAccountLimits",
                "servicequotas:ListServiceQuotas"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.7. Delete your managed OIDC configuration

Run the following command with the specified permissions to delete your managed OIDC configuration by using **auto** mode.

Input

```
$ rosa delete oidc-config --mode auto
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeleteOidcConfig",
            "Effect": "Allow",
            "Action": [
                "iam:ListOpenIDConnectProviders",
                "iam:DeleteOpenIDConnectProvider"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.8. Delete your unmanaged OIDC configuration

Run the following command with the specified permissions to delete your unmanaged OIDC configuration by using **auto** mode.

Input

```
$ rosa delete oidc-config --mode auto
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:ListOpenIDConnectProviders",
                "iam:DeleteOpenIDConnectProvider",
                "secretsmanager:DeleteSecret",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:DeleteBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.9. Create a cluster

Run the following command with the specified permissions to create Red Hat OpenShift Service on AWS clusters.

Input

```
$ rosa create cluster --hosted-cp
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateCluster",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListRoleTags",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeAvailabilityZones"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.10. Create your account roles and Operator roles

Run the following command with the specified permissions to create account and Operator roles by using **auto** mode.

Input

```
$ rosa create account-roles --mode auto --hosted-cp
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAccountRoles",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:UpdateAssumeRolePolicy",
                "iam:ListRoleTags",
                "iam:GetPolicy",
                "iam:TagRole",
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:ListPolicyTags"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.11. Delete your account roles

Run the following command with the specified permissions to delete the account roles in **auto** mode.

Input

```
$ rosa delete account-roles —mode auto
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeleteAccountRoles",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListInstanceProfilesForRole",
                "iam:DetachRolePolicy",
                "iam:ListAttachedRolePolicies",
```

```
                "iam:ListRoles",
                "iam:DeleteRole",
                "iam:ListRolePolicies"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.1.12. Delete your Operator roles

Run the following command with the specified permissions to delete your Operator roles in **auto** mode.

Input

```
$ rosa delete operator-roles --mode auto
```

Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeleteOperatorRoles",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:DetachRolePolicy",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoles",
                "iam:DeleteRole"
            ],
            "Resource": "*"
        }
    ]
}
```

7.3.2. ROSA CLI commands with no required permissions

The following ROSA CLI commands do not require permissions or policies to run. Instead, they require an access key and configured secret key or an attached role.

Table 7.11. Commands

Command	Input
list cluster	**$ rosa list cluster**
list versions	**$ rosa list versions**
describe cluster	**$ rosa describe cluster -c <cluster name>**

Command	Input
create admin	**$ rosa create admin -c <cluster name>**
list users	**$ rosa list users -c <cluster-name>**
list upgrades	**$ rosa list upgrades**
list OIDC configuration	**$ rosa list oidc-config**
list identity providers	**$ rosa list idps -c <cluster-name>**
list ingresses	**$ rosa list ingresses -c <cluster-name>**

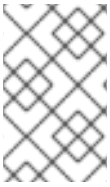
7.3.3. Additional resources

- IAM roles
- Policies and permissions in IAM

7.4. MANAGING BILLING ACCOUNTS FOR RED HAT OPENSHIFT SERVICE ON AWS CLUSTERS

You can use the ROSA CLI (**rosa**) to link your cluster to the desired AWS billing account after the cluster has been deployed.

This can be useful if you have accidentally linked to the wrong AWS billing account during cluster deployment, or if you simply want to update the billing account.



NOTE

You also have the option to update your billing account through the OpenShift Cluster Manager. For more information, see Updating billing accounts for Red Hat OpenShift Service on AWS clusters.

7.4.1. Update billing accounts for Red Hat OpenShift Service on AWS clusters

Prerequisites

- You must have more than one AWS billing account.
- The AWS billing account you want your cluster to link to must already be linked to the Red Hat organization where the cluster is deployed.

Procedure

1. Run the following command in your terminal window:

   Syntax

   ■

```
$ rosa edit cluster -c <cluster_ID> (1)
```

(1) Replace **<cluster_ID>** with the ID of the cluster that you want to update the AWS billing account.

> **NOTE**
>
> To locate the IDs of your active clusters, run the **$ rosa list clusters** command in your terminal window.

2. Skip to the **Billing Account** parameter within the interactive mode.
3. Select the desired AWS billing account from the list of available options and press "Enter". The AWS billing account for your cluster is now updated.