



OpenShift Container Platform 4.13

Logging

OpenShift Logging installation, usage, and release notes

OpenShift Container Platform 4.13 Logging

OpenShift Logging installation, usage, and release notes

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, 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, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document provides instructions for installing, configuring, and using OpenShift Logging, which aggregates logs for a range of OpenShift Container Platform services.

Table of Contents

CHAPTER 1. RELEASE NOTES	15
1.1. LOGGING 5.7	15
1.1.1. Logging 5.7.6	15
1.1.1.1. Bug fixes	15
1.1.1.2. CVEs	16
1.1.2. Logging 5.7.4	16
1.1.2.1. Bug fixes	16
1.1.2.2. CVEs	16
1.1.3. Logging 5.7.3	17
1.1.3.1. Bug fixes	17
1.1.3.2. CVEs	18
1.1.4. Logging 5.7.2	18
1.1.4.1. Bug fixes	18
1.1.4.2. CVEs	20
1.1.5. Logging 5.7.1	22
1.1.5.1. Bug fixes	22
1.1.5.2. CVEs	22
1.1.6. Logging 5.7.0	23
1.1.6.1. Enhancements	23
1.1.6.2. Known Issues	23
1.1.6.3. Bug fixes	23
1.1.6.4. CVEs	23
1.2. LOGGING 5.6	23
1.2.1. Logging 5.6.11	23
1.2.1.1. Bug fixes	23
1.2.1.2. CVEs	24
1.2.2. Logging 5.6.9	24
1.2.2.1. Bug fixes	24
1.2.2.2. CVEs	24
1.2.3. Logging 5.6.8	24
1.2.3.1. Bug fixes	24
1.2.3.2. CVEs	25
1.2.4. Logging 5.6.5	25
1.2.4.1. Bug fixes	25
1.2.4.2. CVEs	26
1.2.5. Logging 5.6.4	26
1.2.5.1. Bug fixes	26
1.2.5.2. CVEs	27
1.2.6. Logging 5.6.3	27
1.2.6.1. Bug fixes	27
1.2.6.2. CVEs	28
1.2.7. Logging 5.6.2	28
1.2.7.1. Bug fixes	28
1.2.7.2. CVEs	28
1.2.8. Logging 5.6.1	29
1.2.8.1. Bug fixes	29
1.2.8.2. CVEs	29
1.2.9. Logging 5.6.0	30
1.2.9.1. Deprecation notice	30
1.2.9.2. Enhancements	30
1.2.9.3. Known Issues	30

1.2.9.4. Bug fixes	30
1.2.9.5. CVEs	32
CHAPTER 2. SUPPORT	33
2.1. UNSUPPORTED CONFIGURATIONS	33
2.2. SUPPORT POLICY FOR UNMANAGED OPERATORS	34
2.3. COLLECTING LOGGING DATA FOR RED HAT SUPPORT	35
2.3.1. About the must-gather tool	35
2.3.2. Collecting OpenShift Logging data	35
CHAPTER 3. LOGGING 5.7	37
3.1. CONFIGURING YOUR LOGGING DEPLOYMENT	37
3.1.1. Enabling stream-based retention with Loki	37
3.1.2. Enabling multi-line exception detection	39
3.1.2.1. Details	40
3.1.2.2. Troubleshooting	41
3.1.3. Enabling log based alerts with Loki	41
3.2. ADMINISTERING YOUR LOGGING DEPLOYMENT	44
3.2.1. Deploying Red Hat OpenShift Logging Operator using the web console	44
3.2.2. Deploying the Loki Operator using the web console	45
3.2.3. Installing from OperatorHub using the CLI	47
3.2.4. Deleting Operators from a cluster using the web console	50
3.2.5. Deleting Operators from a cluster using the CLI	51
CHAPTER 4. LOGGING 5.6	53
4.1. CONFIGURING YOUR LOGGING DEPLOYMENT	53
4.1.1. Enabling stream-based retention with Loki	53
4.1.2. Enabling multi-line exception detection	55
4.1.2.1. Details	56
4.1.2.2. Troubleshooting	57
4.2. ADMINISTERING YOUR LOGGING DEPLOYMENT	57
4.2.1. Deploying Red Hat OpenShift Logging Operator using the web console	57
4.2.2. Deploying the Loki Operator using the web console	59
4.2.3. Installing from OperatorHub using the CLI	61
4.2.4. Deleting Operators from a cluster using the web console	64
4.2.5. Deleting Operators from a cluster using the CLI	65
4.3. LOGGING REFERENCES	65
4.3.1. Collector features	65
4.3.2. Logging 5.6 API reference	69
4.3.2.1. ClusterLogForwarder	69
4.3.2.1.1. .spec	70
4.3.2.1.1.1. Description	70
4.3.2.1.1.1.1. Type	70
4.3.2.1.2. .spec.inputs[]	70
4.3.2.1.2.1. Description	70
4.3.2.1.2.1.1. Type	70
4.3.2.1.3. .spec.inputs[].application	71
4.3.2.1.3.1. Description	71
4.3.2.1.3.1.1. Type	71
4.3.2.1.4. .spec.inputs[].application.namespaces[]	71
4.3.2.1.4.1. Description	71
4.3.2.1.4.1.1. Type	71
4.3.2.1.5. .spec.inputs[].application.selector	71
4.3.2.1.5.1. Description	71

4.3.2.1.5.1.1. Type	71
4.3.2.1.6. .spec.inputs[].application.selector.matchLabels	72
4.3.2.1.6.1. Description	72
4.3.2.1.6.1.1. Type	72
4.3.2.1.7. .spec.outputDefaults	72
4.3.2.1.7.1. Description	72
4.3.2.1.7.1.1. Type	72
4.3.2.1.8. .spec.outputDefaults.elasticsearch	72
4.3.2.1.8.1. Description	72
4.3.2.1.8.1.1. Type	72
4.3.2.1.9. .spec.outputs[]	73
4.3.2.1.9.1. Description	73
4.3.2.1.9.1.1. Type	73
4.3.2.1.10. .spec.outputs[].secret	74
4.3.2.1.10.1. Description	74
4.3.2.1.10.1.1. Type	74
4.3.2.1.11. .spec.outputs[].tls	74
4.3.2.1.11.1. Description	74
4.3.2.1.11.1.1. Type	74
4.3.2.1.12. .spec.pipelines[]	74
4.3.2.1.12.1. Description	75
4.3.2.1.12.1.1. Type	75
4.3.2.1.13. .spec.pipelines[].inputRefs[]	75
4.3.2.1.13.1. Description	75
4.3.2.1.13.1.1. Type	75
4.3.2.1.14. .spec.pipelines[].labels	75
4.3.2.1.14.1. Description	75
4.3.2.1.14.1.1. Type	75
4.3.2.1.15. .spec.pipelines[].outputRefs[]	75
4.3.2.1.15.1. Description	76
4.3.2.1.15.1.1. Type	76
4.3.2.1.16. .status	76
4.3.2.1.16.1. Description	76
4.3.2.1.16.1.1. Type	76
4.3.2.1.17. .status.conditions	76
4.3.2.1.17.1. Description	76
4.3.2.1.17.1.1. Type	76
4.3.2.1.18. .status.inputs	76
4.3.2.1.18.1. Description	76
4.3.2.1.18.1.1. Type	76
4.3.2.1.19. .status.outputs	76
4.3.2.1.19.1. Description	76
4.3.2.1.19.1.1. Type	77
4.3.2.1.20. .status.pipelines	77
4.3.2.1.20.1. Description	77
4.3.2.1.20.1.1. Type	77
4.3.2.1.21. .spec	77
4.3.2.1.21.1. Description	77
4.3.2.1.21.1.1. Type	77
4.3.2.1.22. .spec.collection	78
4.3.2.1.22.1. Description	78
4.3.2.1.22.1.1. Type	78
4.3.2.1.23. .spec.collection.fluentd	78

4.3.2.1.23.1. Description	78
4.3.2.1.23.1.1. Type	79
4.3.2.1.24. .spec.collection.fluentd.buffer	79
4.3.2.1.24.1. Description	79
4.3.2.1.24.1.1. Type	79
4.3.2.1.25. .spec.collection.fluentd.inFile	80
4.3.2.1.25.1. Description	80
4.3.2.1.25.1.1. Type	80
4.3.2.1.26. .spec.collection.logs	80
4.3.2.1.26.1. Description	80
4.3.2.1.26.1.1. Type	81
4.3.2.1.27. .spec.collection.logs.fluentd	81
4.3.2.1.27.1. Description	81
4.3.2.1.27.1.1. Type	81
4.3.2.1.28. .spec.collection.logs.fluentd.nodeSelector	81
4.3.2.1.28.1. Description	81
4.3.2.1.28.1.1. Type	81
4.3.2.1.29. .spec.collection.logs.fluentd.resources	81
4.3.2.1.29.1. Description	81
4.3.2.1.29.1.1. Type	81
4.3.2.1.30. .spec.collection.logs.fluentd.resources.limits	82
4.3.2.1.30.1. Description	82
4.3.2.1.30.1.1. Type	82
4.3.2.1.31. .spec.collection.logs.fluentd.resources.requests	82
4.3.2.1.31.1. Description	82
4.3.2.1.31.1.1. Type	82
4.3.2.1.32. .spec.collection.logs.fluentd.tolerations[]	82
4.3.2.1.32.1. Description	82
4.3.2.1.32.1.1. Type	82
4.3.2.1.33. .spec.collection.logs.fluentd.tolerations[].tolerationSeconds	83
4.3.2.1.33.1. Description	83
4.3.2.1.33.1.1. Type	83
4.3.2.1.34. .spec.curation	83
4.3.2.1.34.1. Description	83
4.3.2.1.34.1.1. Type	83
4.3.2.1.35. .spec.curation.curator	83
4.3.2.1.35.1. Description	83
4.3.2.1.35.1.1. Type	83
4.3.2.1.36. .spec.curation.curator.nodeSelector	84
4.3.2.1.36.1. Description	84
4.3.2.1.36.1.1. Type	84
4.3.2.1.37. .spec.curation.curator.resources	84
4.3.2.1.37.1. Description	84
4.3.2.1.37.1.1. Type	84
4.3.2.1.38. .spec.curation.curator.resources.limits	84
4.3.2.1.38.1. Description	84
4.3.2.1.38.1.1. Type	84
4.3.2.1.39. .spec.curation.curator.resources.requests	84
4.3.2.1.39.1. Description	84
4.3.2.1.39.1.1. Type	85
4.3.2.1.40. .spec.curation.curator.tolerations[]	85
4.3.2.1.40.1. Description	85
4.3.2.1.40.1.1. Type	85

4.3.2.1.41. .spec.curation.curator.tolerations[].tolerationSeconds	85
4.3.2.1.41.1. Description	85
4.3.2.1.41.1.1. Type	85
4.3.2.1.42. .spec.forwarder	85
4.3.2.1.42.1. Description	85
4.3.2.1.42.1.1. Type	85
4.3.2.1.43. .spec.forwarder.fluentd	86
4.3.2.1.43.1. Description	86
4.3.2.1.43.1.1. Type	86
4.3.2.1.44. .spec.forwarder.fluentd.buffer	86
4.3.2.1.44.1. Description	86
4.3.2.1.44.1.1. Type	86
4.3.2.1.45. .spec.forwarder.fluentd.inFile	87
4.3.2.1.45.1. Description	87
4.3.2.1.45.1.1. Type	88
4.3.2.1.46. .spec.logStore	88
4.3.2.1.46.1. Description	88
4.3.2.1.46.1.1. Type	88
4.3.2.1.47. .spec.logStore.elasticsearch	88
4.3.2.1.47.1. Description	88
4.3.2.1.47.1.1. Type	88
4.3.2.1.48. .spec.logStore.elasticsearch.nodeSelector	89
4.3.2.1.48.1. Description	89
4.3.2.1.48.1.1. Type	89
4.3.2.1.49. .spec.logStore.elasticsearch.proxy	89
4.3.2.1.49.1. Description	89
4.3.2.1.49.1.1. Type	89
4.3.2.1.50. .spec.logStore.elasticsearch.proxy.resources	89
4.3.2.1.50.1. Description	89
4.3.2.1.50.1.1. Type	90
4.3.2.1.51. .spec.logStore.elasticsearch.proxy.resources.limits	90
4.3.2.1.51.1. Description	90
4.3.2.1.51.1.1. Type	90
4.3.2.1.52. .spec.logStore.elasticsearch.proxy.resources.requests	90
4.3.2.1.52.1. Description	90
4.3.2.1.52.1.1. Type	90
4.3.2.1.53. .spec.logStore.elasticsearch.resources	90
4.3.2.1.53.1. Description	90
4.3.2.1.53.1.1. Type	90
4.3.2.1.54. .spec.logStore.elasticsearch.resources.limits	90
4.3.2.1.54.1. Description	91
4.3.2.1.54.1.1. Type	91
4.3.2.1.55. .spec.logStore.elasticsearch.resources.requests	91
4.3.2.1.55.1. Description	91
4.3.2.1.55.1.1. Type	91
4.3.2.1.56. .spec.logStore.elasticsearch.storage	91
4.3.2.1.56.1. Description	91
4.3.2.1.56.1.1. Type	91
4.3.2.1.57. .spec.logStore.elasticsearch.storage.size	91
4.3.2.1.57.1. Description	91
4.3.2.1.57.1.1. Type	91
4.3.2.1.58. .spec.logStore.elasticsearch.storage.size.d	92
4.3.2.1.58.1. Description	92

4.3.2.1.58.1.1. Type	92
4.3.2.1.59. .spec.logStore.elasticsearch.storage.size.d.Dec	92
4.3.2.1.59.1. Description	92
4.3.2.1.59.1.1. Type	92
4.3.2.1.60. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled	92
4.3.2.1.60.1. Description	92
4.3.2.1.60.1.1. Type	92
4.3.2.1.61. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled.abs	93
4.3.2.1.61.1. Description	93
4.3.2.1.61.1.1. Type	93
4.3.2.1.62. .spec.logStore.elasticsearch.storage.size.i	93
4.3.2.1.62.1. Description	93
4.3.2.1.62.1.1. Type	93
4.3.2.1.63. .spec.logStore.elasticsearch.tolerations[]	93
4.3.2.1.63.1. Description	93
4.3.2.1.63.1.1. Type	93
4.3.2.1.64. .spec.logStore.elasticsearch.tolerations[].tolerationSeconds	94
4.3.2.1.64.1. Description	94
4.3.2.1.64.1.1. Type	94
4.3.2.1.65. .spec.logStore.lokiStack	94
4.3.2.1.65.1. Description	94
4.3.2.1.65.1.1. Type	94
4.3.2.1.66. .spec.logStore.retentionPolicy	94
4.3.2.1.66.1. Description	94
4.3.2.1.66.1.1. Type	94
4.3.2.1.67. .spec.logStore.retentionPolicy.application	95
4.3.2.1.67.1. Description	95
4.3.2.1.67.1.1. Type	95
4.3.2.1.68. .spec.logStore.retentionPolicy.application.namespaceSpec[]	95
4.3.2.1.68.1. Description	95
4.3.2.1.68.1.1. Type	95
4.3.2.1.69. .spec.logStore.retentionPolicy.audit	96
4.3.2.1.69.1. Description	96
4.3.2.1.69.1.1. Type	96
4.3.2.1.70. .spec.logStore.retentionPolicy.audit.namespaceSpec[]	96
4.3.2.1.70.1. Description	96
4.3.2.1.70.1.1. Type	96
4.3.2.1.71. .spec.logStore.retentionPolicy.infra	96
4.3.2.1.71.1. Description	96
4.3.2.1.71.1.1. Type	96
4.3.2.1.72. .spec.logStore.retentionPolicy.infra.namespaceSpec[]	97
4.3.2.1.72.1. Description	97
4.3.2.1.72.1.1. Type	97
4.3.2.1.73. .spec.visualization	97
4.3.2.1.73.1. Description	97
4.3.2.1.73.1.1. Type	97
4.3.2.1.74. .spec.visualization.kibana	98
4.3.2.1.74.1. Description	98
4.3.2.1.74.1.1. Type	98
4.3.2.1.75. .spec.visualization.kibana.nodeSelector	98
4.3.2.1.75.1. Description	98
4.3.2.1.75.1.1. Type	98
4.3.2.1.76. .spec.visualization.kibana.proxy	98

4.3.2.1.76.1. Description	98
4.3.2.1.76.1.1. Type	98
4.3.2.1.77. .spec.visualization.kibana.proxy.resources	99
4.3.2.1.77.1. Description	99
4.3.2.1.77.1.1. Type	99
4.3.2.1.78. .spec.visualization.kibana.proxy.resources.limits	99
4.3.2.1.78.1. Description	99
4.3.2.1.78.1.1. Type	99
4.3.2.1.79. .spec.visualization.kibana.proxy.resources.requests	99
4.3.2.1.79.1. Description	99
4.3.2.1.79.1.1. Type	99
4.3.2.1.80. .spec.visualization.kibana.replicas	99
4.3.2.1.80.1. Description	99
4.3.2.1.80.1.1. Type	99
4.3.2.1.81. .spec.visualization.kibana.resources	100
4.3.2.1.81.1. Description	100
4.3.2.1.81.1.1. Type	100
4.3.2.1.82. .spec.visualization.kibana.resources.limits	100
4.3.2.1.82.1. Description	100
4.3.2.1.82.1.1. Type	100
4.3.2.1.83. .spec.visualization.kibana.resources.requests	100
4.3.2.1.83.1. Description	100
4.3.2.1.83.1.1. Type	100
4.3.2.1.84. .spec.visualization.kibana.tolerations[]	100
4.3.2.1.84.1. Description	100
4.3.2.1.84.1.1. Type	100
4.3.2.1.85. .spec.visualization.kibana.tolerations[].tolerationSeconds	101
4.3.2.1.85.1. Description	101
4.3.2.1.85.1.1. Type	101
4.3.2.1.86. .status	101
4.3.2.1.86.1. Description	101
4.3.2.1.86.1.1. Type	101
4.3.2.1.87. .status.collection	101
4.3.2.1.87.1. Description	102
4.3.2.1.87.1.1. Type	102
4.3.2.1.88. .status.collection.logs	102
4.3.2.1.88.1. Description	102
4.3.2.1.88.1.1. Type	102
4.3.2.1.89. .status.collection.logs.fluentdStatus	102
4.3.2.1.89.1. Description	102
4.3.2.1.89.1.1. Type	102
4.3.2.1.90. .status.collection.logs.fluentdStatus.clusterCondition	102
4.3.2.1.90.1. Description	102
4.3.2.1.90.1.1. Type	103
4.3.2.1.91. .status.collection.logs.fluentdStatus.nodes	103
4.3.2.1.91.1. Description	103
4.3.2.1.91.1.1. Type	103
4.3.2.1.92. .status.conditions	103
4.3.2.1.92.1. Description	103
4.3.2.1.92.1.1. Type	103
4.3.2.1.93. .status.curation	103
4.3.2.1.93.1. Description	103
4.3.2.1.93.1.1. Type	103

4.3.2.1.94. .status.curation.curatorStatus[]	103
4.3.2.1.94.1. Description	103
4.3.2.1.94.1.1. Type	103
4.3.2.1.95. .status.curation.curatorStatus[].clusterCondition	104
4.3.2.1.95.1. Description	104
4.3.2.1.95.1.1. Type	104
4.3.2.1.96. .status.logStore	104
4.3.2.1.96.1. Description	104
4.3.2.1.96.1.1. Type	104
4.3.2.1.97. .status.logStore.elasticsearchStatus[]	104
4.3.2.1.97.1. Description	104
4.3.2.1.97.1.1. Type	104
4.3.2.1.98. .status.logStore.elasticsearchStatus[].cluster	105
4.3.2.1.98.1. Description	105
4.3.2.1.98.1.1. Type	105
4.3.2.1.99. .status.logStore.elasticsearchStatus[].clusterConditions	106
4.3.2.1.99.1. Description	106
4.3.2.1.99.1.1. Type	106
4.3.2.1.100. .status.logStore.elasticsearchStatus[].deployments[]	106
4.3.2.1.100.1. Description	106
4.3.2.1.100.1.1. Type	106
4.3.2.1.101. .status.logStore.elasticsearchStatus[].nodeConditions	106
4.3.2.1.101.1. Description	106
4.3.2.1.101.1.1. Type	106
4.3.2.1.102. .status.logStore.elasticsearchStatus[].pods	106
4.3.2.1.102.1. Description	106
4.3.2.1.102.1.1. Type	106
4.3.2.1.103. .status.logStore.elasticsearchStatus[].replicaSets[]	106
4.3.2.1.103.1. Description	106
4.3.2.1.103.1.1. Type	106
4.3.2.1.104. .status.logStore.elasticsearchStatus[].statefulSets[]	106
4.3.2.1.104.1. Description	107
4.3.2.1.104.1.1. Type	107
4.3.2.1.105. .status.visualization	107
4.3.2.1.105.1. Description	107
4.3.2.1.105.1.1. Type	107
4.3.2.1.106. .status.visualization.kibanaStatus[]	107
4.3.2.1.106.1. Description	107
4.3.2.1.106.1.1. Type	107
4.3.2.1.107. .status.visualization.kibanaStatus[].clusterCondition	107
4.3.2.1.107.1. Description	107
4.3.2.1.107.1.1. Type	107
4.3.2.1.108. .status.visualization.kibanaStatus[].replicaSets[]	108
4.3.2.1.108.1. Description	108
4.3.2.1.108.1.1. Type	108
CHAPTER 5. LOGGING 5.5	109
5.1. ADMINISTERING YOUR LOGGING DEPLOYMENT	109
5.1.1. Deploying Red Hat OpenShift Logging Operator using the web console	109
5.1.2. Deploying the Loki Operator using the web console	110
5.1.3. Installing from OperatorHub using the CLI	112
5.1.4. Deleting Operators from a cluster using the web console	115
5.1.5. Deleting Operators from a cluster using the CLI	116

CHAPTER 6. ABOUT LOGGING	118
6.1. LOGGING ARCHITECTURE	118
6.2. ABOUT DEPLOYING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT	119
6.2.1. About JSON OpenShift Container Platform Logging	119
6.2.2. About collecting and storing Kubernetes events	119
6.2.3. About updating OpenShift Container Platform Logging	119
6.2.4. About viewing the cluster dashboard	120
6.2.5. About troubleshooting OpenShift Container Platform Logging	120
6.2.6. About uninstalling OpenShift Container Platform Logging	120
6.2.7. About exporting fields	120
6.2.8. About logging subsystem components	120
6.2.9. About the logging collector	121
6.2.10. About the log store	121
6.2.11. About logging visualization	122
6.2.12. About event routing	122
6.3. ABOUT VECTOR	122
6.3.1. Enabling Vector	123
6.3.2. Collector features	123
CHAPTER 7. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT	128
7.1. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT USING THE WEB CONSOLE	128
7.2. POST-INSTALLATION TASKS	134
7.3. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT USING THE CLI	134
7.4. POST-INSTALLATION TASKS	142
7.4.1. Defining Kibana index patterns	142
7.4.2. Allowing traffic between projects when network isolation is enabled	143
CHAPTER 8. CONFIGURING YOUR LOGGING DEPLOYMENT	145
8.1. ABOUT THE CLUSTER LOGGING CUSTOM RESOURCE	145
8.1.1. About the ClusterLogging custom resource	145
8.2. CONFIGURING THE LOG STORE	146
8.2.1. Forwarding audit logs to the log store	146
8.2.2. Configuring log retention time	148
8.2.3. Configuring CPU and memory requests for the log store	150
8.2.4. Configuring replication policy for the log store	152
8.2.5. Scaling down Elasticsearch pods	153
8.2.6. Configuring persistent storage for the log store	153
8.2.7. Configuring the log store for emptyDir storage	154
8.2.8. Performing an Elasticsearch rolling cluster restart	154
8.2.9. Exposing the log store service as a route	158
8.2.10. Removing unused components if you do not use the default Elasticsearch log store	160
8.3. CONFIGURING THE LOG VISUALIZER	162
8.3.1. Configuring CPU and memory limits	162
8.3.2. Scaling redundancy for the log visualizer nodes	163
8.4. CONFIGURING LOGGING SUBSYSTEM STORAGE	163
8.4.1. Storage considerations for the logging subsystem for Red Hat OpenShift	164
8.4.2. Additional resources	164
8.5. CONFIGURING CPU AND MEMORY LIMITS FOR LOGGING SUBSYSTEM COMPONENTS	164
8.5.1. Configuring CPU and memory limits	164
8.6. USING TOLERATIONS TO CONTROL OPENSIFT LOGGING POD PLACEMENT	166
8.6.1. Using tolerations to control the log store pod placement	167
8.6.2. Using tolerations to control the log visualizer pod placement	168
8.6.3. Using tolerations to control the log collector pod placement	169

8.6.4. Additional resources	170
8.7. MOVING LOGGING SUBSYSTEM RESOURCES WITH NODE SELECTORS	170
8.7.1. Moving OpenShift Logging resources	171
8.8. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD	174
8.8.1. Configuring systemd-journald for OpenShift Logging	174
CHAPTER 9. LOGGING USING LOKISTACK	178
9.1. DEPLOYMENT SIZING	178
9.1.1. Supported API Custom Resource Definitions	178
9.2. DEPLOYING THE LOKISTACK	179
9.3. INSTALLING LOGGING OPERATORS USING THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE	181
9.4. ENABLING STREAM-BASED RETENTION WITH LOKI	183
9.5. FORWARDING LOGS TO LOKISTACK	185
9.5.1. Troubleshooting Loki rate limit errors	186
9.6. ADDITIONAL RESOURCES	187
CHAPTER 10. VIEWING LOGS FOR A RESOURCE	189
10.1. VIEWING RESOURCE LOGS	189
CHAPTER 11. VIEWING CLUSTER LOGS BY USING KIBANA	191
11.1. DEFINING KIBANA INDEX PATTERNS	191
11.2. VIEWING CLUSTER LOGS IN KIBANA	192
CHAPTER 12. LOG COLLECTION AND FORWARDING	195
12.1. ABOUT LOG COLLECTION AND FORWARDING	195
12.1.1. Sending audit logs to the internal log store	195
12.1.2. About forwarding logs to third-party systems	195
Fluentd log handling when the external log aggregator is unavailable	199
Supported Authorization Keys	199
12.1.2.1. Creating a Secret	200
12.1.3. Forwarding JSON logs from containers in the same pod to separate indices	200
12.1.4. Forwarding logs to an external Elasticsearch instance	201
12.1.5. Forwarding logs using the Fluentd forward protocol	204
12.1.5.1. Enabling nanosecond precision for Logstash to ingest data from fluentd	206
12.1.6. Forwarding logs using the syslog protocol	207
12.1.6.1. Adding log source information to message output	209
12.1.6.2. Syslog parameters	210
12.1.6.3. Additional RFC5424 syslog parameters	211
12.1.7. Forwarding logs to Amazon CloudWatch	211
12.1.7.1. Forwarding logs to Amazon CloudWatch from STS enabled clusters	217
12.1.7.2. Creating a secret for AWS CloudWatch with an existing AWS role	219
12.1.8. Forwarding logs to Loki	219
12.1.8.1. Troubleshooting Loki rate limit errors	221
12.1.9. Forwarding logs to Google Cloud Platform (GCP)	223
12.1.10. Forwarding logs to Splunk	224
12.1.11. Forwarding logs over HTTP	225
12.1.12. Forwarding application logs from specific projects	226
12.1.13. Forwarding application logs from specific pods	228
12.1.14. Troubleshooting log forwarding	230
12.2. LOG OUTPUT TYPES	230
12.2.1. Supported log data output types in OpenShift Logging 5.7	231
12.2.2. Supported log data output types in OpenShift Logging 5.6	232
12.2.3. Supported log data output types in OpenShift Logging 5.5	233

12.2.4. Supported log data output types in OpenShift Logging 5.4	233
12.2.5. Supported log data output types in OpenShift Logging 5.3	234
12.2.6. Supported log data output types in OpenShift Logging 5.2	234
12.2.7. Supported log data output types in OpenShift Logging 5.1	235
12.3. ENABLING JSON LOG FORWARDING	236
12.3.1. Parsing JSON logs	236
12.3.2. Configuring JSON log data for Elasticsearch	237
12.3.3. Forwarding JSON logs to the Elasticsearch log store	239
12.4. CONFIGURING THE LOGGING COLLECTOR	240
12.4.1. Viewing logging collector pods	240
12.4.2. Configure log collector CPU and memory limits	241
12.4.3. Advanced configuration for the Fluentd log forwarder	241
12.5. COLLECTING AND STORING KUBERNETES EVENTS	246
12.5.1. Deploying and configuring the Event Router	246
CHAPTER 13. UPDATING OPENSIFT LOGGING	250
13.1. SUPPORTED VERSIONS	250
13.2. UPDATING LOGGING TO THE CURRENT VERSION	250
CHAPTER 14. VIEWING CLUSTER DASHBOARDS	255
14.1. ACCESSING THE ELASTICSEARCH AND OPENSIFT LOGGING DASHBOARDS	255
14.2. ABOUT THE OPENSIFT LOGGING DASHBOARD	255
14.3. CHARTS ON THE LOGGING/ELASTICSEARCH NODES DASHBOARD	257
CHAPTER 15. TROUBLESHOOTING LOGGING	263
15.1. VIEWING OPENSIFT LOGGING STATUS	263
15.1.1. Viewing the status of the Red Hat OpenShift Logging Operator	263
15.1.1.1. Example condition messages	265
15.1.2. Viewing the status of logging subsystem components	267
15.2. VIEWING THE STATUS OF THE ELASTICSEARCH LOG STORE	268
15.2.1. Viewing the status of the log store	268
15.2.1.1. Example condition messages	270
15.2.2. Viewing the status of the log store components	272
15.2.3. Elasticsearch cluster status	276
15.3. UNDERSTANDING LOGGING SUBSYSTEM ALERTS	277
15.3.1. Viewing logging collector alerts	277
15.3.2. About logging collector alerts	277
15.3.3. About Elasticsearch alerting rules	278
15.4. TROUBLESHOOTING FOR CRITICAL ALERTS	279
15.4.1. Elasticsearch Cluster Health is Red	279
15.4.2. Elasticsearch Cluster Health is Yellow	281
15.4.3. Elasticsearch Node Disk Low Watermark Reached	282
15.4.4. Elasticsearch Node Disk High Watermark Reached	283
15.4.5. Elasticsearch Node Disk Flood Watermark Reached	284
15.4.6. Elasticsearch JVM Heap Use is High	286
15.4.7. Aggregated Logging System CPU is High	286
15.4.8. Elasticsearch Process CPU is High	286
15.4.9. Elasticsearch Disk Space is Running Low	286
15.4.10. Elasticsearch FileDescriptor Usage is high	287
CHAPTER 16. UNINSTALLING OPENSIFT LOGGING	288
16.1. UNINSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT	288
CHAPTER 17. LOG RECORD FIELDS	291

CHAPTER 18. MESSAGE	292
CHAPTER 19. STRUCTURED	293
CHAPTER 20. @TIMESTAMP	294
CHAPTER 21. HOSTNAME	295
CHAPTER 22. IPADDR4	296
CHAPTER 23. IPADDR6	297
CHAPTER 24. LEVEL	298
CHAPTER 25. PID	299
CHAPTER 26. SERVICE	300
CHAPTER 27. TAGS	301
CHAPTER 28. FILE	302
CHAPTER 29. OFFSET	303
CHAPTER 30. KUBERNETES	304
30.1. KUBERNETES.POD_NAME	304
30.2. KUBERNETES.POD_ID	304
30.3. KUBERNETES.NAMESPACE_NAME	304
30.4. KUBERNETES.NAMESPACE_ID	304
30.5. KUBERNETES.HOST	304
30.6. KUBERNETES.CONTAINER_NAME	304
30.7. KUBERNETES.ANNOTATIONS	305
30.8. KUBERNETES.LABELS	305
30.9. KUBERNETES.EVENT	305
30.9.1. kubernetes.event.verb	305
30.9.2. kubernetes.event.metadata	305
30.9.2.1. kubernetes.event.metadata.name	305
30.9.2.2. kubernetes.event.metadata.namespace	305
30.9.2.3. kubernetes.event.metadata.selfLink	306
30.9.2.4. kubernetes.event.metadata.uid	306
30.9.2.5. kubernetes.event.metadata.resourceVersion	306
30.9.3. kubernetes.event.involvedObject	306
30.9.3.1. kubernetes.event.involvedObject.kind	306
30.9.3.2. kubernetes.event.involvedObject.namespace	307
30.9.3.3. kubernetes.event.involvedObject.name	307
30.9.3.4. kubernetes.event.involvedObject.uid	307
30.9.3.5. kubernetes.event.involvedObject.apiVersion	307
30.9.3.6. kubernetes.event.involvedObject.resourceVersion	307
30.9.4. kubernetes.event.reason	308
30.9.5. kubernetes.event.source_component	308
30.9.6. kubernetes.event.firstTimestamp	308
30.9.7. kubernetes.event.count	308
30.9.8. kubernetes.event.type	308
CHAPTER 31. OPENSIFT	310
31.1. OPENSIFT.LABELS	310

CHAPTER 32. GLOSSARY	311
----------------------------	-----

CHAPTER 1. RELEASE NOTES

1.1. LOGGING 5.7



NOTE

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

1.1.1. Logging 5.7.6

This release includes [OpenShift Logging Bug Fix Release 5.7.6](#).

1.1.1.1. Bug fixes

- Before this update, the collector relied on the default configuration settings for reading the container log lines. As a result, the collector did not read the rotated files efficiently. With this update, there is an increase in the number of bytes read, which allows the collector to efficiently process rotated files. ([LOG-4501](#))
- Before this update, when users pasted a URL with predefined filters, some filters did not reflect. With this update, the UI reflects all the filters in the URL. ([LOG-4459](#))
- Before this update, forwarding to Loki using custom labels generated an error when switching from Fluentd to Vector. With this update, the Vector configuration sanitizes labels in the same way as Fluentd to ensure the collector starts and correctly processes labels. ([LOG-4460](#))
- Before this update, the Observability Logs console search field did not accept special characters that it should escape. With this update, it is escaping special characters properly in the query. ([LOG-4456](#))
- Before this update, the following warning message appeared while sending logs to Splunk: **Timestamp was not found.** With this update, the change overrides the name of the log field used to retrieve the Timestamp and sends it to Splunk without warning. ([LOG-4413](#))
- Before this update, the CPU and memory usage of Vector was increasing over time. With this update, the Vector configuration now contains the **expire_metrics_secs=60** setting to limit the lifetime of the metrics and cap the associated CPU usage and memory footprint. ([LOG-4171](#))
- Before this update, the LokiStack gateway cached authorized requests very broadly. As a result, this caused wrong authorization results. With this update, LokiStack gateway caches on a more fine-grained basis which resolves this issue. ([LOG-4393](#))
- Before this update, the Fluentd runtime image included builder tools which were unnecessary at runtime. With this update, the builder tools are removed, resolving the issue. ([LOG-4467](#))

1.1.1.2. CVEs

- [CVE-2023-3899](#)
- [CVE-2023-4456](#)
- [CVE-2023-32360](#)
- [CVE-2023-34969](#)

1.1.2. Logging 5.7.4

This release includes [OpenShift Logging Bug Fix Release 5.7.4](#).

1.1.2.1. Bug fixes

- Before this update, when forwarding logs to CloudWatch, a **namespaceUUID** value was not appended to the **logGroupName** field. With this update, the **namespaceUUID** value is included, so a **logGroupName** in CloudWatch appears as **logGroupName: vectorcw.b443fb9e-bd4c-4b6a-b9d3-c0097f9ed286**. ([LOG-2701](#))
- Before this update, when forwarding logs over HTTP to an off-cluster destination, the Vector collector was unable to authenticate to the cluster-wide HTTP proxy even though correct credentials were provided in the proxy URL. With this update, the Vector log collector can now authenticate to the cluster-wide HTTP proxy. ([LOG-3381](#))
- Before this update, the Operator would fail if the Fluentd collector was configured with Splunk as an output, due to this configuration being unsupported. With this update, configuration validation rejects unsupported outputs, resolving the issue. ([LOG-4237](#))
- Before this update, when the Vector collector was updated an **enabled = true** value in the TLS configuration for AWS Cloudwatch logs and the GCP Stackdriver caused a configuration error. With this update, **enabled = true** value will be removed for these outputs, resolving the issue. ([LOG-4242](#))
- Before this update, the Vector collector occasionally panicked with the following error message in its log: **thread 'vector-worker' panicked at 'all branches are disabled and there is no else branch', src/kubernetes/reflector.rs:26:9**. With this update, the error has been resolved. ([LOG-4275](#))
- Before this update, an issue in the Loki Operator caused the **alert-manager** configuration for the application tenant to disappear if the Operator was configured with additional options for that tenant. With this update, the generated Loki configuration now contains both the custom and the auto-generated configuration. ([LOG-4361](#))
- Before this update, when multiple roles were used to authenticate using STS with AWS Cloudwatch forwarding, a recent update caused the credentials to be non-unique. With this update, multiple combinations of STS roles and static credentials can once again be used to authenticate with AWS Cloudwatch. ([LOG-4368](#))
- Before this update, Loki filtered label values for active streams but did not remove duplicates, making Grafana's Label Browser unusable. With this update, Loki filters out duplicate label values for active streams, resolving the issue. ([LOG-4389](#))

1.1.2.2. CVEs

- [CVE-2022-25883](#)
- [CVE-2023-22796](#)

1.1.3. Logging 5.7.3

This release includes [OpenShift Logging Bug Fix Release 5.7.3](#).

1.1.3.1. Bug fixes

- Before this update, when viewing logs within the OpenShift Container Platform web console, cached files caused the data to not refresh. With this update the bootstrap files are not cached, resolving the issue. ([LOG-4100](#))
- Before this update, the Loki Operator reset errors in a way that made identifying configuration problems difficult to troubleshoot. With this update, errors persist until the configuration error is resolved. ([LOG-4156](#))
- Before this update, the LokiStack ruler did not restart after changes were made to the **RulerConfig** custom resource (CR). With this update, the Loki Operator restarts the ruler pods after the **RulerConfig** CR is updated. ([LOG-4161](#))
- Before this update, the vector collector terminated unexpectedly when input match label values contained a / character within the **ClusterLogForwarder**. This update resolves the issue by quoting the match label, enabling the collector to start and collect logs. ([LOG-4176](#))
- Before this update, the Loki Operator terminated unexpectedly when a **LokiStack** CR defined tenant limits, but not global limits. With this update, the Loki Operator can process **LokiStack** CRs without global limits, resolving the issue. ([LOG-4198](#))
- Before this update, Fluentd did not send logs to an Elasticsearch cluster when the private key provided was passphrase-protected. With this update, Fluentd properly handles passphrase-protected private keys when establishing a connection with Elasticsearch. ([LOG-4258](#))
- Before this update, clusters with more than 8,000 namespaces caused Elasticsearch to reject queries because the list of namespaces was larger than the **http.max_header_size** setting. With this update, the default value for header size has been increased, resolving the issue. ([LOG-4277](#))
- Before this update, label values containing a / character within the **ClusterLogForwarder** CR would cause the collector to terminate unexpectedly. With this update, slashes are replaced with underscores, resolving the issue. ([LOG-4095](#))
- Before this update, the Cluster Logging Operator terminated unexpectedly when set to an unmanaged state. With this update, a check to ensure that the **ClusterLogging** resource is in the correct Management state before initiating the reconciliation of the **ClusterLogForwarder** CR, resolving the issue. ([LOG-4177](#))
- Before this update, when viewing logs within the OpenShift Container Platform web console, selecting a time range by dragging over the histogram didn't work on the aggregated logs view inside the pod detail. With this update, the time range can be selected by dragging on the histogram in this view. ([LOG-4108](#))
- Before this update, when viewing logs within the OpenShift Container Platform web console, queries longer than 30 seconds timed out. With this update, the timeout value can be configured in the configmap/logging-view-plugin. ([LOG-3498](#))

- Before this update, when viewing logs within the OpenShift Container Platform web console, clicking the **more data available** option loaded more log entries only the first time it was clicked. With this update, more entries are loaded with each click. ([OU-188](#))
- Before this update, when viewing logs within the OpenShift Container Platform web console, clicking the **streaming** option would only display the **streaming logs** message without showing the actual logs. With this update, both the message and the log stream are displayed correctly. ([OU-166](#))

1.1.3.2. CVEs

- [CVE-2020-24736](#)
- [CVE-2022-48281](#)
- [CVE-2023-1667](#)
- [CVE-2023-2283](#)
- [CVE-2023-24329](#)
- [CVE-2023-26115](#)
- [CVE-2023-26136](#)
- [CVE-2023-26604](#)
- [CVE-2023-28466](#)

1.1.4. Logging 5.7.2

This release includes [OpenShift Logging Bug Fix Release 5.7.2](#).

1.1.4.1. Bug fixes

- Before this update, it was not possible to delete the **openshift-logging** namespace directly due to the presence of a pending finalizer. With this update, the finalizer is no longer utilized, enabling direct deletion of the namespace. ([LOG-3316](#))
- Before this update, the **run.sh** script would display an incorrect **chunk_limit_size** value if it was changed according to the OpenShift Container Platform documentation. However, when setting the **chunk_limit_size** via the environment variable **\$BUFFER_SIZE_LIMIT**, the script would show the correct value. With this update, the **run.sh** script now consistently displays the correct **chunk_limit_size** value in both scenarios. ([LOG-3330](#))
- Before this update, the OpenShift Container Platform web console's logging view plugin did not allow for custom node placement or tolerations. This update adds the ability to define node placement and tolerations for the logging view plugin. ([LOG-3749](#))
- Before this update, the Cluster Logging Operator encountered an Unsupported Media Type exception when trying to send logs to DataDog via the Fluentd HTTP Plugin. With this update, users can seamlessly assign the content type for log forwarding by configuring the HTTP header Content-Type. The value provided is automatically assigned to the **content_type** parameter within the plugin, ensuring successful log transmission. ([LOG-3784](#))
- Before this update, when the **detectMultilineErrors** field was set to **true** in the

ClusterLogForwarder custom resource (CR), PHP multi-line errors were recorded as separate log entries, causing the stack trace to be split across multiple messages. With this update, multi-line error detection for PHP is enabled, ensuring that the entire stack trace is included in a single log message. ([LOG-3878](#))

- Before this update, **ClusterLogForwarder** pipelines containing a space in their name caused the Vector collector pods to continuously crash. With this update, all spaces, dashes (-), and dots (.) in pipeline names are replaced with underscores (_). ([LOG-3945](#))
- Before this update, the **log_forwarder_output** metric did not include the **http** parameter. This update adds the missing parameter to the metric. ([LOG-3997](#))
- Before this update, Fluentd did not identify some multi-line JavaScript client exceptions when they ended with a colon. With this update, the Fluentd buffer name is prefixed with an underscore, resolving the issue. ([LOG-4019](#))
- Before this update, when configuring log forwarding to write to a Kafka output topic which matched a key in the payload, logs dropped due to an error. With this update, Fluentd's buffer name has been prefixed with an underscore, resolving the issue. ([LOG-4027](#))
- Before this update, the LokiStack gateway returned label values for namespaces without applying the access rights of a user. With this update, the LokiStack gateway applies permissions to label value requests, resolving the issue. ([LOG-4049](#))
- Before this update, the Cluster Logging Operator API required a certificate to be provided by a secret when the **tls.insecureSkipVerify** option was set to **true**. With this update, the Cluster Logging Operator API no longer requires a certificate to be provided by a secret in such cases. The following configuration has been added to the Operator's CR:

```
tls.verify_certificate = false
tls.verify_hostname = false
```

([LOG-3445](#))

- Before this update, the LokiStack route configuration caused queries running longer than 30 seconds to timeout. With this update, the LokiStack global and per-tenant **queryTimeout** settings affect the route timeout settings, resolving the issue. ([LOG-4052](#))
- Before this update, a prior fix to remove defaulting of the **collection.type** resulted in the Operator no longer honoring the deprecated specs for resource, node selections, and tolerations. This update modifies the Operator behavior to always prefer the **collection.logs** spec over those of **collection**. This varies from previous behavior that allowed using both the preferred fields and deprecated fields but would ignore the deprecated fields when **collection.type** was populated. ([LOG-4185](#))
- Before this update, the Vector log collector did not generate TLS configuration for forwarding logs to multiple Kafka brokers if the broker URLs were not specified in the output. With this update, TLS configuration is generated appropriately for multiple brokers. ([LOG-4163](#))
- Before this update, the option to enable passphrase for log forwarding to Kafka was unavailable. This limitation presented a security risk as it could potentially expose sensitive information. With this update, users now have a seamless option to enable passphrase for log forwarding to Kafka. ([LOG-3314](#))
- Before this update, Vector log collector did not honor the **tlsSecurityProfile** settings for outgoing TLS connections. After this update, Vector handles TLS connection settings appropriately. ([LOG-4011](#))

- Before this update, not all available output types were included in the **log_forwarder_output_info** metrics. With this update, metrics contain Splunk and Google Cloud Logging data which was missing previously. ([LOG-4098](#))
- Before this update, when **follow_inodes** was set to **true**, the Fluentd collector could crash on file rotation. With this update, the **follow_inodes** setting does not crash the collector. ([LOG-4151](#))
- Before this update, the Fluentd collector could incorrectly close files that should be watched because of how those files were tracked. With this update, the tracking parameters have been corrected. ([LOG-4149](#))
- Before this update, forwarding logs with the Vector collector and naming a pipeline in the **ClusterLogForwarder** instance **audit**, **application** or **infrastructure** resulted in collector pods staying in the **CrashLoopBackOff** state with the following error in the collector log:

```
ERROR vector::cli: Configuration error. error=redefinition of table transforms.audit for key transforms.audit
```

After this update, pipeline names no longer clash with reserved input names, and pipelines can be named **audit**, **application** or **infrastructure**. ([LOG-4218](#))

- Before this update, when forwarding logs to a syslog destination with the Vector collector and setting the **addLogSource** flag to **true**, the following extra empty fields were added to the forwarded messages: **namespace_name=**, **container_name=**, and **pod_name=**. With this update, these fields are no longer added to journal logs. ([LOG-4219](#))
- Before this update, when a **structuredTypeKey** was not found, and a **structuredTypeName** was not specified, log messages were still parsed into structured object. With this update, parsing of logs is as expected. ([LOG-4220](#))

1.1.4.2. CVEs

- [CVE-2021-26341](#)
- [CVE-2021-33655](#)
- [CVE-2021-33656](#)
- [CVE-2022-1462](#)
- [CVE-2022-1679](#)
- [CVE-2022-1789](#)
- [CVE-2022-2196](#)
- [CVE-2022-2663](#)
- [CVE-2022-3028](#)
- [CVE-2022-3239](#)
- [CVE-2022-3522](#)
- [CVE-2022-3524](#)

- [CVE-2022-3564](#)
- [CVE-2022-3566](#)
- [CVE-2022-3567](#)
- [CVE-2022-3619](#)
- [CVE-2022-3623](#)
- [CVE-2022-3625](#)
- [CVE-2022-3627](#)
- [CVE-2022-3628](#)
- [CVE-2022-3707](#)
- [CVE-2022-3970](#)
- [CVE-2022-4129](#)
- [CVE-2022-20141](#)
- [CVE-2022-25147](#)
- [CVE-2022-25265](#)
- [CVE-2022-30594](#)
- [CVE-2022-36227](#)
- [CVE-2022-39188](#)
- [CVE-2022-39189](#)
- [CVE-2022-41218](#)
- [CVE-2022-41674](#)
- [CVE-2022-42703](#)
- [CVE-2022-42720](#)
- [CVE-2022-42721](#)
- [CVE-2022-42722](#)
- [CVE-2022-43750](#)
- [CVE-2022-47929](#)
- [CVE-2023-0394](#)
- [CVE-2023-0461](#)
- [CVE-2023-1195](#)

- [CVE-2023-1582](#)
- [CVE-2023-2491](#)
- [CVE-2023-22490](#)
- [CVE-2023-23454](#)
- [CVE-2023-23946](#)
- [CVE-2023-25652](#)
- [CVE-2023-25815](#)
- [CVE-2023-27535](#)
- [CVE-2023-29007](#)

1.1.5. Logging 5.7.1

This release includes: [OpenShift Logging Bug Fix Release 5.7.1](#).

1.1.5.1. Bug fixes

- Before this update, the presence of numerous noisy messages within the Cluster Logging Operator pod logs caused reduced log readability, and increased difficulty in identifying important system events. With this update, the issue is resolved by significantly reducing the noisy messages within Cluster Logging Operator pod logs. ([LOG-3482](#))
- Before this update, the API server would reset the value for the **CollectorSpec.Type** field to **vector**, even when the custom resource used a different value. This update removes the default for the **CollectorSpec.Type** field to restore the previous behavior. ([LOG-4086](#))
- Before this update, a time range could not be selected in the OpenShift Container Platform web console by clicking and dragging over the logs histogram. With this update, clicking and dragging can be used to successfully select a time range. ([LOG-4501](#))
- Before this update, clicking on the **Show Resources** link in the OpenShift Container Platform web console did not produce any effect. With this update, the issue is resolved by fixing the functionality of the "Show Resources" link to toggle the display of resources for each log entry. ([LOG-3218](#))

1.1.5.2. CVEs

- [CVE-2023-21930](#)
- [CVE-2023-21937](#)
- [CVE-2023-21938](#)
- [CVE-2023-21939](#)
- [CVE-2023-21954](#)
- [CVE-2023-21967](#)

- [CVE-2023-21968](#)
- [CVE-2023-28617](#)

1.1.6. Logging 5.7.0

This release includes [OpenShift Logging Bug Fix Release 5.7.0](#).

1.1.6.1. Enhancements

With this update, you can enable logging to detect multi-line exceptions and reassemble them into a single log entry.

To enable logging to detect multi-line exceptions and reassemble them into a single log entry, ensure that the **ClusterLogForwarder** Custom Resource (CR) contains a **detectMultilineErrors** field, with a value of **true**.

1.1.6.2. Known Issues

None.

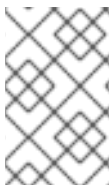
1.1.6.3. Bug fixes

- Before this update, the **nodeSelector** attribute for the Gateway component of the LokiStack did not impact node scheduling. With this update, the **nodeSelector** attribute works as expected. ([LOG-3713](#))

1.1.6.4. CVEs

- [CVE-2023-1999](#)
- [CVE-2023-28617](#)

1.2. LOGGING 5.6



NOTE

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

1.2.1. Logging 5.6.11

This release includes [OpenShift Logging Bug Fix Release 5.6.11](#).

1.2.1.1. Bug fixes

- Before this update, the LokiStack gateway cached authorized requests very broadly. As a result, this caused wrong authorization results. With this update, LokiStack gateway caches on a more fine-grained basis which resolves this issue. ([LOG-4435](#))

1.2.1.2. CVEs

- [CVE-2023-3899](#)
- [CVE-2023-32360](#)
- [CVE-2023-34969](#)

1.2.2. Logging 5.6.9

This release includes [OpenShift Logging Bug Fix Release 5.6.9](#) .

1.2.2.1. Bug fixes

- Before this update, when multiple roles were used to authenticate using STS with AWS Cloudwatch forwarding, a recent update caused the credentials to be non-unique. With this update, multiple combinations of STS roles and static credentials can once again be used to authenticate with AWS Cloudwatch. ([LOG-4084](#))
- Before this update, the Vector collector occasionally panicked with the following error message in its log: **thread 'vector-worker' panicked at 'all branches are disabled and there is no else branch', src/kubernetes/reflector.rs:26:9**. With this update, the error has been resolved. ([LOG-4276](#))
- Before this update, Loki filtered label values for active streams but did not remove duplicates, making Grafana's Label Browser unusable. With this update, Loki filters out duplicate label values for active streams, resolving the issue. ([LOG-4390](#))

1.2.2.2. CVEs

- [CVE-2020-24736](#)
- [CVE-2022-48281](#)
- [CVE-2023-1667](#)
- [CVE-2023-2283](#)
- [CVE-2023-24329](#)
- [CVE-2023-26604](#)
- [CVE-2023-28466](#)
- [CVE-2023-32233](#)

1.2.3. Logging 5.6.8

This release includes [OpenShift Logging Bug Fix Release 5.6.8](#) .

1.2.3.1. Bug fixes

- Before this update, the vector collector terminated unexpectedly when input match label values contained a / character within the **ClusterLogForwarder**. This update resolves the issue by quoting the match label, enabling the collector to start and collect logs. ([LOG-4091](#))
- Before this update, when viewing logs within the OpenShift Container Platform web console, clicking the **more data available** option loaded more log entries only the first time it was clicked. With this update, more entries are loaded with each click. ([OU-187](#))
- Before this update, when viewing logs within the OpenShift Container Platform web console, clicking the **streaming** option would only display the **streaming logs** message without showing the actual logs. With this update, both the message and the log stream are displayed correctly. ([OU-189](#))
- Before this update, the Loki Operator reset errors in a way that made identifying configuration problems difficult to troubleshoot. With this update, errors persist until the configuration error is resolved. ([LOG-4158](#))
- Before this update, clusters with more than 8,000 namespaces caused Elasticsearch to reject queries because the list of namespaces was larger than the **http.max_header_size** setting. With this update, the default value for header size has been increased, resolving the issue. ([LOG-4278](#))

1.2.3.2. CVEs

- [CVE-2020-24736](#)
- [CVE-2022-48281](#)
- [CVE-2023-1667](#)
- [CVE-2023-2283](#)
- [CVE-2023-24329](#)
- [CVE-2023-26604](#)
- [CVE-2023-28466](#)

1.2.4. Logging 5.6.5

This release includes [OpenShift Logging Bug Fix Release 5.6.5](#).

1.2.4.1. Bug fixes

- Before this update, the template definitions prevented Elasticsearch from indexing some labels and namespace_labels, causing issues with data ingestion. With this update, the fix replaces dots and slashes in labels to ensure proper ingestion, effectively resolving the issue. ([LOG-3419](#))
- Before this update, if the Logs page of the OpenShift Web Console failed to connect to the LokiStack, a generic error message was displayed, providing no additional context or troubleshooting suggestions. With this update, the error message has been enhanced to include more specific details and recommendations for troubleshooting. ([LOG-3750](#))
- Before this update, time range formats were not validated, leading to errors selecting a custom date range. With this update, time formats are now validated, enabling users to select a valid range. If an invalid time range format is selected, an error message is displayed to the user.

([LOG-3583](#))

- Before this update, when searching logs in Loki, even if the length of an expression did not exceed 5120 characters, the query would fail in many cases. With this update, query authorization label matchers have been optimized, resolving the issue. ([LOG-3480](#))
- Before this update, the Loki Operator failed to produce a memberlist configuration that was sufficient for locating all the components when using a memberlist for private IPs. With this update, the fix ensures that the generated configuration includes the advertised port, allowing for successful lookup of all components. ([LOG-4008](#))

1.2.4.2. CVEs

- [CVE-2022-4269](#)
- [CVE-2022-4378](#)
- [CVE-2023-0266](#)
- [CVE-2023-0361](#)
- [CVE-2023-0386](#)
- [CVE-2023-27539](#)
- [CVE-2023-28120](#)

1.2.5. Logging 5.6.4

This release includes [OpenShift Logging Bug Fix Release 5.6.4](#) .

1.2.5.1. Bug fixes

- Before this update, when LokiStack was deployed as the log store, the logs generated by Loki pods were collected and sent to LokiStack. With this update, the logs generated by Loki are excluded from collection and will not be stored. ([LOG-3280](#))
- Before this update, when the query editor on the Logs page of the OpenShift Web Console was empty, the drop-down menus did not populate. With this update, if an empty query is attempted, an error message is displayed and the drop-down menus now populate as expected. ([LOG-3454](#))
- Before this update, when the **tls.insecureSkipVerify** option was set to **true**, the Cluster Logging Operator would generate incorrect configuration. As a result, the operator would fail to send data to Elasticsearch when attempting to skip certificate validation. With this update, the Cluster Logging Operator generates the correct TLS configuration even when **tls.insecureSkipVerify** is enabled. As a result, data can be sent successfully to Elasticsearch even when attempting to skip certificate validation. ([LOG-3475](#))
- Before this update, when structured parsing was enabled and messages were forwarded to multiple destinations, they were not deep copied. This resulted in some of the received logs including the structured message, while others did not. With this update, the configuration generation has been modified to deep copy messages before JSON parsing. As a result, all received messages now have structured messages included, even when they are forwarded to multiple destinations. ([LOG-3640](#))

- Before this update, if the **collection** field contained **{}** it could result in the Operator crashing. With this update, the Operator will ignore this value, allowing the operator to continue running smoothly without interruption. ([LOG-3733](#))
- Before this update, the **nodeSelector** attribute for the Gateway component of LokiStack did not have any effect. With this update, the **nodeSelector** attribute functions as expected. ([LOG-3783](#))
- Before this update, the static LokiStack memberlist configuration relied solely on private IP networks. As a result, when the OpenShift Container Platform cluster pod network was configured with a public IP range, the LokiStack pods would crashloop. With this update, the LokiStack administrator now has the option to use the pod network for the memberlist configuration. This resolves the issue and prevents the LokiStack pods from entering a crashloop state when the OpenShift Container Platform cluster pod network is configured with a public IP range. ([LOG-3814](#))
- Before this update, if the **tls.insecureSkipVerify** field was set to **true**, the Cluster Logging Operator would generate an incorrect configuration. As a result, the Operator would fail to send data to Elasticsearch when attempting to skip certificate validation. With this update, the Operator generates the correct TLS configuration even when **tls.insecureSkipVerify** is enabled. As a result, data can be sent successfully to Elasticsearch even when attempting to skip certificate validation. ([LOG-3838](#))
- Before this update, if the Cluster Logging Operator (CLO) was installed without the Elasticsearch Operator, the CLO pod would continuously display an error message related to the deletion of Elasticsearch. With this update, the CLO now performs additional checks before displaying any error messages. As a result, error messages related to Elasticsearch deletion are no longer displayed in the absence of the Elasticsearch Operator. ([LOG-3763](#))

1.2.5.2. CVEs

- [CVE-2022-4304](#)
- [CVE-2022-4450](#)
- [CVE-2023-0215](#)
- [CVE-2023-0286](#)
- [CVE-2023-0767](#)
- [CVE-2023-23916](#)

1.2.6. Logging 5.6.3

This release includes [OpenShift Logging Bug Fix Release 5.6.3](#).

1.2.6.1. Bug fixes

- Before this update, the operator stored gateway tenant secret information in a config map. With this update, the operator stores this information in a secret. ([LOG-3717](#))
- Before this update, the Fluentd collector did not capture OAuth login events stored in **/var/log/auth-server/audit.log**. With this update, Fluentd captures these OAuth login events, resolving the issue. ([LOG-3729](#))

1.2.6.2. CVEs

- [CVE-2020-10735](#)
- [CVE-2021-28861](#)
- [CVE-2022-2873](#)
- [CVE-2022-4415](#)
- [CVE-2022-40897](#)
- [CVE-2022-41222](#)
- [CVE-2022-43945](#)
- [CVE-2022-45061](#)
- [CVE-2022-48303](#)

1.2.7. Logging 5.6.2

This release includes [OpenShift Logging Bug Fix Release 5.6.2](#) .

1.2.7.1. Bug fixes

- Before this update, the collector did not set **level** fields correctly based on priority for systemd logs. With this update, **level** fields are set correctly. ([LOG-3429](#))
- Before this update, the Operator incorrectly generated incompatibility warnings on OpenShift Container Platform 4.12 or later. With this update, the Operator max OpenShift Container Platform version value has been corrected, resolving the issue. ([LOG-3584](#))
- Before this update, creating a **ClusterLogForwarder** custom resource (CR) with an output value of **default** did not generate any errors. With this update, an error warning that this value is invalid generates appropriately. ([LOG-3437](#))
- Before this update, when the **ClusterLogForwarder** custom resource (CR) had multiple pipelines configured with one output set as **default**, the collector pods restarted. With this update, the logic for output validation has been corrected, resolving the issue. ([LOG-3559](#))
- Before this update, collector pods restarted after being created. With this update, the deployed collector does not restart on its own. ([LOG-3608](#))
- Before this update, patch releases removed previous versions of the Operators from the catalog. This made installing the old versions impossible. This update changes bundle configurations so that previous releases of the same minor version stay in the catalog. ([LOG-3635](#))

1.2.7.2. CVEs

- [CVE-2022-23521](#)
- [CVE-2022-40303](#)
- [CVE-2022-40304](#)

- [CVE-2022-41903](#)
- [CVE-2022-47629](#)
- [CVE-2023-21835](#)
- [CVE-2023-21843](#)

1.2.8. Logging 5.6.1

This release includes [OpenShift Logging Bug Fix Release 5.6.1](#).

1.2.8.1. Bug fixes

- Before this update, the compactor would report TLS certificate errors from communications with the querier when retention was active. With this update, the compactor and querier no longer communicate erroneously over HTTP. ([LOG-3494](#))
- Before this update, the Loki Operator would not retry setting the status of the **LokiStack** CR, which caused stale status information. With this update, the Operator retries status information updates on conflict. ([LOG-3496](#))
- Before this update, the Loki Operator Webhook server caused TLS errors when the **kube-apiserver-operator** Operator checked the webhook validity. With this update, the Loki Operator Webhook PKI is managed by the Operator Lifecycle Manager (OLM), resolving the issue. ([LOG-3510](#))
- Before this update, the LokiStack Gateway Labels Enforcer generated parsing errors for valid LogQL queries when using combined label filters with boolean expressions. With this update, the LokiStack LogQL implementation supports label filters with boolean expression and resolves the issue. ([LOG-3441](#)), ([LOG-3397](#))
- Before this update, records written to Elasticsearch would fail if multiple label keys had the same prefix and some keys included dots. With this update, underscores replace dots in label keys, resolving the issue. ([LOG-3463](#))
- Before this update, the **Red Hat OpenShift Logging** Operator was not available for OpenShift Container Platform 4.10 clusters because of an incompatibility between OpenShift Container Platform console and the logging-view-plugin. With this update, the plugin is properly integrated with the OpenShift Container Platform 4.10 admin console. ([LOG-3447](#))
- Before this update the reconciliation of the **ClusterLogForwarder** custom resource would incorrectly report a degraded status of pipelines that reference the default logstore. With this update, the pipeline validates properly. ([LOG-3477](#))

1.2.8.2. CVEs

- [CVE-2021-46848](#)
- [CVE-2022-3821](#)
- [CVE-2022-35737](#)
- [CVE-2022-42010](#)
- [CVE-2022-42011](#)

- [CVE-2022-42012](#)
- [CVE-2022-42898](#)
- [CVE-2022-43680](#)
- [CVE-2021-35065](#)
- [CVE-2022-46175](#)

1.2.9. Logging 5.6.0

This release includes [OpenShift Logging Release 5.6](#).

1.2.9.1. Deprecation notice

In logging version 5.6, Fluentd is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to Fluentd, you can use Vector instead.

1.2.9.2. Enhancements

- With this update, Logging is compliant with OpenShift Container Platform cluster-wide cryptographic policies. ([LOG-895](#))
- With this update, you can declare per-tenant, per-stream, and global policies retention policies through the LokiStack custom resource, ordered by priority. ([LOG-2695](#))
- With this update, Splunk is an available output option for log forwarding. ([LOG-2913](#))
- With this update, Vector replaces Fluentd as the default Collector. ([LOG-2222](#))
- With this update, the **Developer** role can access the per-project workload logs they are assigned to within the Log Console Plugin on clusters running OpenShift Container Platform 4.11 and higher. ([LOG-3388](#))
- With this update, logs from any source contain a field **openshift.cluster_id**, the unique identifier of the cluster in which the Operator is deployed. You can view the **clusterID** value by using the following command:

```
$ oc get clusterversion/version -o jsonpath='{.spec.clusterID}'{"\n"}
```

([LOG-2715](#))

1.2.9.3. Known Issues

- Before this update, Elasticsearch would reject logs if multiple label keys had the same prefix and some keys included the `.` character. This fixes the limitation of Elasticsearch by replacing `.` in the label keys with `_`. As a workaround for this issue, remove the labels that cause errors, or add a namespace to the label. ([LOG-3463](#))

1.2.9.4. Bug fixes

- Before this update, if you deleted the Kibana Custom Resource, the OpenShift Container Platform web console continued displaying a link to Kibana. With this update, removing the Kibana Custom Resource also removes that link. ([LOG-2993](#))
- Before this update, a user was not able to view the application logs of namespaces they have access to. With this update, the Loki Operator automatically creates a cluster role and cluster role binding allowing users to read application logs. ([LOG-3072](#))
- Before this update, the Operator removed any custom outputs defined in the **ClusterLogForwarder** custom resource when using LokiStack as the default log storage. With this update, the Operator merges custom outputs with the default outputs when processing the **ClusterLogForwarder** custom resource. ([LOG-3090](#))
- Before this update, the CA key was used as the volume name for mounting the CA into Loki, causing error states when the CA Key included non-conforming characters, such as dots. With this update, the volume name is standardized to an internal string which resolves the issue. ([LOG-3331](#))
- Before this update, a default value set within the LokiStack Custom Resource Definition, caused an inability to create a LokiStack instance without a **ReplicationFactor** of **1**. With this update, the operator sets the actual value for the size used. ([LOG-3296](#))
- Before this update, Vector parsed the message field when JSON parsing was enabled without also defining **structuredTypeKey** or **structuredTypeName** values. With this update, a value is required for either **structuredTypeKey** or **structuredTypeName** when writing structured logs to Elasticsearch. ([LOG-3195](#))
- Before this update, the secret creation component of the Elasticsearch Operator modified internal secrets constantly. With this update, the existing secret is properly handled. ([LOG-3161](#))
- Before this update, the Operator could enter a loop of removing and recreating the collector daemonset while the Elasticsearch or Kibana deployments changed their status. With this update, a fix in the status handling of the Operator resolves the issue. ([LOG-3157](#))
- Before this update, Kibana had a fixed **24h** OAuth cookie expiration time, which resulted in 401 errors in Kibana whenever the **accessTokenInactivityTimeout** field was set to a value lower than **24h**. With this update, Kibana's OAuth cookie expiration time synchronizes to the **accessTokenInactivityTimeout**, with a default value of **24h**. ([LOG-3129](#))
- Before this update, the Operators general pattern for reconciling resources was to try and create before attempting to get or update which would lead to constant HTTP 409 responses after creation. With this update, Operators first attempt to retrieve an object and only create or update it if it is either missing or not as specified. ([LOG-2919](#))
- Before this update, the **.level** and **`structure.level`** fields in Fluentd could contain different values. With this update, the values are the same for each field. ([LOG-2819](#))
- Before this update, the Operator did not wait for the population of the trusted CA bundle and deployed the collector a second time once the bundle updated. With this update, the Operator waits briefly to see if the bundle has been populated before it continues the collector deployment. ([LOG-2789](#))
- Before this update, logging telemetry info appeared twice when reviewing metrics. With this update, logging telemetry info displays as expected. ([LOG-2315](#))
- Before this update, Fluentd pod logs contained a warning message after enabling the JSON parsing addition. With this update, that warning message does not appear. ([LOG-1806](#))

- Before this update, the **must-gather** script did not complete because **oc** needs a folder with write permission to build its cache. With this update, **oc** has write permissions to a folder, and the **must-gather** script completes successfully. ([LOG-3446](#))
- Before this update the log collector SCC could be superseded by other SCCs on the cluster, rendering the collector unusable. This update sets the priority of the log collector SCC so that it takes precedence over the others. ([LOG-3235](#))
- Before this update, Vector was missing the field **sequence**, which was added to fluentd as a way to deal with a lack of actual nanoseconds precision. With this update, the field **openshift.sequence** has been added to the event logs. ([LOG-3106](#))

1.2.9.5. CVEs

- [CVE-2020-36518](#)
- [CVE-2021-46848](#)
- [CVE-2022-2879](#)
- [CVE-2022-2880](#)
- [CVE-2022-27664](#)
- [CVE-2022-32190](#)
- [CVE-2022-35737](#)
- [CVE-2022-37601](#)
- [CVE-2022-41715](#)
- [CVE-2022-42003](#)
- [CVE-2022-42004](#)
- [CVE-2022-42010](#)
- [CVE-2022-42011](#)
- [CVE-2022-42012](#)
- [CVE-2022-42898](#)
- [CVE-2022-43680](#)

CHAPTER 2. SUPPORT

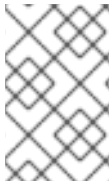
Only the configuration options described in this documentation are supported for the logging subsystem.

Do not use any other configuration options, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will be overwritten, because Operators are designed to reconcile any differences.



NOTE

If you must perform configurations not described in the OpenShift Container Platform documentation, you must set your Red Hat OpenShift Logging Operator to **Unmanaged**. An unmanaged logging subsystem for Red Hat OpenShift is not supported and does not receive updates until you return its status to **Managed**.



NOTE

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.

The logging subsystem for Red Hat OpenShift is an opinionated collector and normalizer of application, infrastructure, and audit logs. It is intended to be used for forwarding logs to various supported systems.

The logging subsystem for Red Hat OpenShift is not:

- A high scale log collection system
- Security Information and Event Monitoring (SIEM) compliant
- Historical or long term log retention or storage
- A guaranteed log sink
- Secure storage - audit logs are not stored by default

2.1. UNSUPPORTED CONFIGURATIONS

You must set the Red Hat OpenShift Logging Operator to the **Unmanaged** state to modify the following components:

- The **Elasticsearch** custom resource (CR)
- The Kibana deployment
- The **fluent.conf** file
- The Fluentd daemon set

You must set the OpenShift Elasticsearch Operator to the **Unmanaged** state to modify the Elasticsearch deployment files.

Explicitly unsupported cases include:

- **Configuring default log rotation** You cannot modify the default log rotation configuration.
- **Configuring the collected log location** You cannot change the location of the log collector output file, which by default is `/var/log/fluentd/fluentd.log`.
- **Throttling log collection.** You cannot throttle down the rate at which the logs are read in by the log collector.
- **Configuring the logging collector using environment variables** You cannot use environment variables to modify the log collector.
- **Configuring how the log collector normalizes logs** You cannot modify default log normalization.

2.2. SUPPORT POLICY FOR UNMANAGED OPERATORS

The *management state* of an Operator determines whether an Operator is actively managing the resources for its related component in the cluster as designed. If an Operator is set to an *unmanaged* state, it does not respond to changes in configuration nor does it receive updates.

While this can be helpful in non-production clusters or during debugging, Operators in an unmanaged state are unsupported and the cluster administrator assumes full control of the individual component configurations and upgrades.

An Operator can be set to an unmanaged state using the following methods:

- **Individual Operator configuration**
Individual Operators have a **managementState** parameter in their configuration. This can be accessed in different ways, depending on the Operator. For example, the Red Hat OpenShift Logging Operator accomplishes this by modifying a custom resource (CR) that it manages, while the Cluster Samples Operator uses a cluster-wide configuration resource.

Changing the **managementState** parameter to **Unmanaged** means that the Operator is not actively managing its resources and will take no action related to the related component. Some Operators might not support this management state as it might damage the cluster and require manual recovery.



WARNING

Changing individual Operators to the **Unmanaged** state renders that particular component and functionality unsupported. Reported issues must be reproduced in **Managed** state for support to proceed.

- **Cluster Version Operator (CVO) overrides**
The **spec.overrides** parameter can be added to the CVO's configuration to allow administrators to provide a list of overrides to the CVO's behavior for a component. Setting the **spec.overrides[].unmanaged** parameter to **true** for a component blocks cluster upgrades and alerts the administrator after a CVO override has been set:

■

Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.



WARNING

Setting a CVO override puts the entire cluster in an unsupported state. Reported issues must be reproduced after removing any overrides for support to proceed.

2.3. COLLECTING LOGGING DATA FOR RED HAT SUPPORT

When opening a support case, it is helpful to provide debugging information about your cluster to Red Hat Support.

You can use the [must-gather tool](#) to collect diagnostic information for project-level resources, cluster-level resources, and each of the logging subsystem components.

For prompt support, supply diagnostic information for both OpenShift Container Platform and the logging subsystem for Red Hat OpenShift.



NOTE

Do not use the **hack/logging-dump.sh** script. The script is no longer supported and does not collect data.

2.3.1. About the must-gather tool

The **oc adm must-gather** CLI command collects the information from your cluster that is most likely needed for debugging issues.

For your logging subsystem, **must-gather** collects the following information:

- Project-level resources, including pods, configuration maps, service accounts, roles, role bindings, and events at the project level
- Cluster-level resources, including nodes, roles, and role bindings at the cluster level
- OpenShift Logging resources in the **openshift-logging** and **openshift-operators-redhat** namespaces, including health status for the log collector, the log store, and the log visualizer

When you run **oc adm must-gather**, a new pod is created on the cluster. The data is collected on that pod and saved in a new directory that starts with **must-gather.local**. This directory is created in the current working directory.

2.3.2. Collecting OpenShift Logging data

You can use the **oc adm must-gather** CLI command to collect information about your logging subsystem.

Procedure

To collect logging subsystem information with **must-gather**:

1. Navigate to the directory where you want to store the **must-gather** information.
2. Run the **oc adm must-gather** command against the OpenShift Logging image:

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

The **must-gather** tool creates a new directory that starts with **must-gather.local** within the current directory. For example: **must-gather.local.4157245944708210408**.

3. Create a compressed file from the **must-gather** directory that was just created. For example, on a computer that uses a Linux operating system, run the following command:

```
$ tar -cvaf must-gather.tar.gz must-gather.local.4157245944708210408
```

4. Attach the compressed file to your support case on the [Red Hat Customer Portal](#).

CHAPTER 3. LOGGING 5.7

3.1. CONFIGURING YOUR LOGGING DEPLOYMENT

You can configure your logging subsystem deployment with Custom Resource (CR) YAML files implemented by each Operator.

Red Hat Openshift Logging Operator:

- **ClusterLogging** (CL) - Deploys the collector and forwarder which currently are both implemented by a daemonset running on each node.
- **ClusterLogForwarder** (CLF) - Generates collector configuration to forward logs per user configuration.

Loki Operator:

- **LokiStack** - Controls the Loki cluster as log store and the web proxy with OpenShift Container Platform authentication integration to enforce multi-tenancy.

OpenShift Elasticsearch Operator:



NOTE

These CRs are generated and managed by the **ClusterLogging** Operator, manual changes cannot be made without being overwritten by the Operator.

- **ElasticSearch** - Configure and deploy an Elasticsearch instance as the default log store.
- **Kibana** - Configure and deploy Kibana instance to search, query and view logs.

Only the configuration options described in this documentation are supported for the logging subsystem.

Do not use any other configuration options, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will be overwritten, because Operators are designed to reconcile any differences.



NOTE

If you must perform configurations not described in the OpenShift Container Platform documentation, you must set your Red Hat OpenShift Logging Operator to **Unmanaged**. An unmanaged logging subsystem for Red Hat OpenShift is not supported and does not receive updates until you return its status to **Managed**.

3.1.1. Enabling stream-based retention with Loki

With Logging version 5.6 and higher, you can configure retention policies based on log streams. Rules for these may be set globally, per tenant, or both. If you configure both, tenant rules apply before global rules.

1. To enable stream-based retention, create a **LokiStack** custom resource (CR):

Example global stream-based retention

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global: ❶
    retention: ❷
      days: 20
      streams:
        - days: 4
          priority: 1
          selector: '{kubernetes_namespace_name=~"test.+"}' ❸
        - days: 1
          priority: 1
          selector: '{log_type="infrastructure"}'
  managementState: Managed
  replicationFactor: 1
  size: 1x.small
  storage:
    schemas:
      - effectiveDate: "2020-10-11"
        version: v11
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: standard
  tenants:
    mode: openshift-logging

```

- ❶ Sets retention policy for all log streams. **Note: This field does not impact the retention period for stored logs in object storage.**
- ❷ Retention is enabled in the cluster when this block is added to the CR.
- ❸ Contains the [LogQL query](#) used to define the log stream.

Example per-tenant stream-based retention

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      retention:
        days: 20
    tenants: ❶
      application:

```

```

retention:
  days: 1
  streams:
    - days: 4
      selector: '{kubernetes_namespace_name=~"test.+"}' 2
infrastructure:
  retention:
    days: 5
    streams:
      - days: 1
        selector: '{kubernetes_namespace_name=~"openshift-cluster.+"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
    - effectiveDate: "2020-10-11"
      version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: standard
tenants:
  mode: openshift-logging

```

- 1 Sets retention policy by tenant. Valid tenant types are **application**, **audit**, and **infrastructure**.
- 2 Contains the [LogQL query](#) used to define the log stream.

2. Apply the **LokiStack** CR:

```
$ oc apply -f <filename>.yaml
```



NOTE

This is not for managing the retention for stored logs. Global retention periods for stored logs to a supported maximum of 30 days is configured with your object storage.

3.1.2. Enabling multi-line exception detection

Enables multi-line error detection of container logs.



WARNING

Enabling this feature could have performance implications and may require additional computing resources or alternate logging solutions.

Log parsers often incorrectly identify separate lines of the same exception as separate exceptions. This leads to extra log entries and an incomplete or inaccurate view of the traced information.

Example java exception

```
java.lang.NullPointerException: Cannot invoke "String.toString()" because "<param1>" is null
    at testjava.Main.handle(Main.java:47)
    at testjava.Main.printMe(Main.java:19)
    at testjava.Main.main(Main.java:10)
```

- To enable logging to detect multi-line exceptions and reassemble them into a single log entry, ensure that the **ClusterLogForwarder** Custom Resource (CR) contains a **detectMultilineErrors** field, with a value of **true**.

Example ClusterLogForwarder CR

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines:
    - name: my-app-logs
      inputRefs:
        - application
      outputRefs:
        - default
      detectMultilineErrors: true
```

3.1.2.1. Details

When log messages appear as a consecutive sequence forming an exception stack trace, they are combined into a single, unified log record. The first log message's content is replaced with the concatenated content of all the message fields in the sequence.

Table 3.1. Supported languages per collector:

Language	Fluentd	Vector
Java	✓	✓
JS	✓	✓
Ruby	✓	✓
Python	✓	✓
Golang	✓	✓
PHP	✓	

Language	Fluentd	Vector
Dart	✓	✓

3.1.2.2. Troubleshooting

When enabled, the collector configuration will include a new section with type: **detect_exceptions**

Example vector configuration section

```
[transforms.detect_exceptions_app-logs]
  type = "detect_exceptions"
  inputs = ["application"]
  languages = ["All"]
  group_by = ["kubernetes.namespace_name","kubernetes.pod_name","kubernetes.container_name"]
  expire_after_ms = 2000
  multiline_flush_interval_ms = 1000
```

Example fluentd config section

```
<label @MULTILINE_APP_LOGS>
  <match kubernetes.**>
    @type detect_exceptions
    remove_tag_prefix 'kubernetes'
    message message
    force_line_breaks true
    multiline_flush_interval .2
  </match>
</label>
```

3.1.3. Enabling log based alerts with Loki

Loki alerting rules use [LogQL](#) and follow [Prometheus formatting](#). You can set log based alerts by creating an **AlertingRule** custom resource (CR). **AlertingRule** CRs may be created for **application**, **audit**, or **infrastructure** tenants.

Tenant type	Valid namespaces
application	
audit	openshift-logging
infrastructure	openshift-/*, kube-/*, default

Application, Audit, and Infrastructure alerts are sent to the Cluster Monitoring Operator (CMO) Alertmanager in the **openshift-monitoring** namespace by default unless you have disabled the local **Alertmanager** instance.

Application alerts are not sent to the CMO Alertmanager in the **openshift-user-workload-monitoring** namespace by default unless you have enabled a separate **Alertmanager** instance.

The **AlertingRule** CR contains a set of specifications and webhook validation definitions to declare groups of alerting rules for a single LokiStack instance. In addition, the webhook validation definition provides support for rule validation conditions:

- If an **AlertingRule** CR includes an invalid **interval** period, it is an invalid alerting rule
- If an **AlertingRule** CR includes an invalid **for** period, it is an invalid alerting rule.
- If an **AlertingRule** CR includes an invalid LogQL **expr**, it is an invalid alerting rule.
- If an **AlertingRule** CR includes two groups with the same name, it is an invalid alerting rule.
- If none of above applies, an **AlertingRule** is considered a valid alerting rule.

Prerequisites

- Logging subsystem for Red Hat OpenShift Operator 5.7 and later
- OpenShift Container Platform 4.13 and later

Procedure

1. Create an **AlertingRule** custom resource (CR):

Example infrastructure AlertingRule CR

```
apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: loki-operator-alerts
  namespace: openshift-operators-redhat 1
  labels: 2
    openshift.io/cluster-monitoring: "true"
spec:
  tenantID: "infrastructure" 3
  groups:
    - name: LokiOperatorHighReconciliationError
      rules:
        - alert: HighPercentageError
          expr: | 4
            sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
            kubernetes_pod_name=~"loki-operator-controller-manager.*"} |= "error" [1m])) by (job)
            /
            sum(rate({kubernetes_namespace_name="openshift-operators-redhat",
            kubernetes_pod_name=~"loki-operator-controller-manager.*"}[1m])) by (job)
            > 0.01
          for: 10s
          labels:
            severity: critical 5
          annotations:
            summary: High Loki Operator Reconciliation Errors 6
            description: High Loki Operator Reconciliation Errors 7
```

- 1** The namespace where this **AlertingRule** CR is created must have a label matching the LokiStack **spec.rules.namespaceSelector** definition.

- 2 The **labels** block must match the LokiStack **spec.rules.selector** definition.
- 3 **AlertingRule** CRs for **infrastructure** tenants are only supported in the **openshift-***, **kube-***, or **default** namespaces.
- 4 The value for **kubernetes_namespace_name**: must match the value for **metadata.namespace**.
- 5 The value of this mandatory field must be **critical**, **warning**, or **info**.
- 6 This field is mandatory.
- 7 This field is mandatory.

Example application AlertingRule CR

```

apiVersion: loki.grafana.com/v1
kind: AlertingRule
metadata:
  name: app-user-workload
  namespace: app-ns 1
  labels: 2
    openshift.io/cluster-monitoring: "true"
spec:
  tenantID: "application"
  groups:
    - name: AppUserWorkloadHighError
      rules:
        - alert:
            expr: | 3
              sum(rate({kubernetes_namespace_name="app-ns",
kubernetes_pod_name=~"podName.*"} |= "error" [1m])) by (job)
            for: 10s
            labels:
              severity: critical 4
            annotations:
              summary: 5
              description: 6

```

- 1 The namespace where this **AlertingRule** CR is created must have a label matching the LokiStack **spec.rules.namespaceSelector** definition.
- 2 The **labels** block must match the LokiStack **spec.rules.selector** definition.
- 3 Value for **kubernetes_namespace_name**: must match the value for **metadata.namespace**.
- 4 The value of this mandatory field must be **critical**, **warning**, or **info**.
- 5 The value of this mandatory field is a summary of the rule.
- 6 The value of this mandatory field is a detailed description of the rule.

2. Apply the **AlertingRule** CR:

```
$ oc apply -f <filename>.yaml
```

3.2. ADMINISTERING YOUR LOGGING DEPLOYMENT

3.2.1. Deploying Red Hat OpenShift Logging Operator using the web console

You can use the OpenShift Container Platform web console to deploy the Red Hat OpenShift Logging Operator.



PREREQUISITES

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.

Procedure

To deploy the Red Hat OpenShift Logging Operator using the OpenShift Container Platform web console:

1. Install the Red Hat OpenShift Logging Operator:
 - a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
 - b. Type **Logging** in the **Filter by keyword** field.
 - c. Choose **Red Hat OpenShift Logging** from the list of available Operators, and click **Install**.
 - d. Select **stable** or **stable-5.y** as the **Update Channel**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

- e. Ensure that **A specific namespace on the cluster** is selected under **Installation Mode**.
- f. Ensure that **Operator recommended namespace** is **openshift-logging** under **Installed Namespace**.
- g. Select **Enable Operator recommended cluster monitoring on this Namespace**
- h. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** option requires a user with appropriate credentials to approve the Operator update.
- i. Select **Enable** or **Disable** for the Console plugin.

- j. Click **Install**.
2. Verify that the **Red Hat OpenShift Logging Operator** is installed by switching to the **Operators** → **Installed Operators** page.
 - a. Ensure that **Red Hat OpenShift Logging** is listed in the **openshift-logging** project with a **Status** of **Succeeded**.
3. Create a **ClusterLogging** instance.

**NOTE**

The form view of the web console does not include all available options. The **YAML view** is recommended for completing your setup.

- a. In the **collection** section, select a Collector Implementation.

**NOTE**

As of logging version 5.6 Fluentd is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to Fluentd, you can use Vector instead.

- b. In the **logStore** section, select a type.

**NOTE**

As of logging version 5.4.3 the OpenShift Elasticsearch Operator is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to using the OpenShift Elasticsearch Operator to manage the default log storage, you can use the Loki Operator.

- c. Click **Create**.

3.2.2. Deploying the Loki Operator using the web console

You can use the OpenShift Container Platform web console to install the Loki Operator.

Prerequisites

- Supported Log Store (AWS S3, Google Cloud Storage, Azure, Swift, Minio, OpenShift Data Foundation)

Procedure

To install the Loki Operator using the OpenShift Container Platform web console:

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Type **Loki** in the **Filter by keyword** field.

- a. Choose **Loki Operator** from the list of available Operators, and click **Install**.
3. Select **stable** or **stable-5.y** as the **Update Channel**.

**NOTE**

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

4. Ensure that **All namespaces on the cluster** is selected under **Installation Mode**.
5. Ensure that **openshift-operators-redhat** is selected under **Installed Namespace**.
6. Select **Enable Operator recommended cluster monitoring on this Namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.
7. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** option requires a user with appropriate credentials to approve the Operator update.
8. Click **Install**.
9. Verify that the **LokiOperator** installed by switching to the **Operators → Installed Operators** page.
 - a. Ensure that **LokiOperator** is listed with **Status** as **Succeeded** in all the projects.
10. Create a **Secret** YAML file that uses the **access_key_id** and **access_key_secret** fields to specify your credentials and **bucketnames**, **endpoint**, and **region** to define the object storage location. AWS is used in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

11. Select **Create instance** under LokiStack on the **Details** tab. Then select **YAML view**. Paste in the following template, substituting values where appropriate.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
```

```

metadata:
  name: logging-loki ❶
  namespace: openshift-logging
spec:
  size: 1x.small ❷
  storage:
    schemas:
      - version: v12
        effectiveDate: '2022-06-01'
    secret:
      name: logging-loki-s3 ❸
      type: s3 ❹
  storageClassName: <storage_class_name> ❺
  tenants:
    mode: openshift-logging

```

- ❶ Name should be **logging-loki**.
- ❷ Select your Loki deployment size.
- ❸ Define the secret used for your log storage.
- ❹ Define corresponding storage type.
- ❺ Enter the name of an existing storage class for temporary storage. For best performance, specify a storage class that allocates block storage. Available storage classes for your cluster can be listed using **oc get storageclasses**.

a. Apply the configuration:

```
oc apply -f logging-loki.yaml
```

12. Create or edit a **ClusterLogging** CR:

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
    collection:
      type: vector

```

a. Apply the configuration:

```
oc apply -f cr-lokistack.yaml
```

3.2.3. Installing from OperatorHub using the CLI

Instead of using the OpenShift Container Platform web console, you can install an Operator from OperatorHub by using the CLI. Use the **oc** command to create or update a **Subscription** object.

Prerequisites

- Access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. View the list of Operators available to the cluster from OperatorHub:

```
$ oc get packagemanifests -n openshift-marketplace
```

Example output

NAME	CATALOG	AGE
3scale-operator	Red Hat Operators	91m
advanced-cluster-management	Red Hat Operators	91m
amq7-cert-manager	Red Hat Operators	91m
...		
couchbase-enterprise-certified	Certified Operators	91m
crunchy-postgres-operator	Certified Operators	91m
mongodb-enterprise	Certified Operators	91m
...		
etcd	Community Operators	91m
jaeger	Community Operators	91m
kubefed	Community Operators	91m
...		

Note the catalog for your desired Operator.

2. Inspect your desired Operator to verify its supported install modes and available channels:

```
$ oc describe packagemanifests <operator_name> -n openshift-marketplace
```

3. An Operator group, defined by an **OperatorGroup** object, selects target namespaces in which to generate required RBAC access for all Operators in the same namespace as the Operator group.

The namespace to which you subscribe the Operator must have an Operator group that matches the install mode of the Operator, either the **AllNamespaces** or **SingleNamespace** mode. If the Operator you intend to install uses the **AllNamespaces**, then the **openshift-operators** namespace already has an appropriate Operator group in place.

However, if the Operator uses the **SingleNamespace** mode and you do not already have an appropriate Operator group in place, you must create one.



NOTE

The web console version of this procedure handles the creation of the **OperatorGroup** and **Subscription** objects automatically behind the scenes for you when choosing **SingleNamespace** mode.

- a. Create an **OperatorGroup** object YAML file, for example **operatorgroup.yaml**:

Example OperatorGroup object

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <namespace>
spec:
  targetNamespaces:
    - <namespace>
```

- b. Create the **OperatorGroup** object:

```
$ oc apply -f operatorgroup.yaml
```

4. Create a **Subscription** object YAML file to subscribe a namespace to an Operator, for example **sub.yaml**:

Example Subscription object

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: <subscription_name>
  namespace: openshift-operators 1
spec:
  channel: <channel_name> 2
  name: <operator_name> 3
  source: redhat-operators 4
  sourceNamespace: openshift-marketplace 5
  config:
    env: 6
    - name: ARGS
      value: "-v=10"
    envFrom: 7
    - secretRef:
        name: license-secret
  volumes: 8
  - name: <volume_name>
    configMap:
      name: <configmap_name>
  volumeMounts: 9
  - mountPath: <directory_name>
    name: <volume_name>
  tolerations: 10
  - operator: "Exists"
  resources: 11
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
```

```
memory: "128Mi"
cpu: "500m"
nodeSelector: 12
foo: bar
```

- 1 For default **AllNamespaces** install mode usage, specify the **openshift-operators** namespace. Alternatively, you can specify a custom global namespace, if you have created one. Otherwise, specify the relevant single namespace for **SingleNamespace** install mode usage.
- 2 Name of the channel to subscribe to.
- 3 Name of the Operator to subscribe to.
- 4 Name of the catalog source that provides the Operator.
- 5 Namespace of the catalog source. Use **openshift-marketplace** for the default OperatorHub catalog sources.
- 6 The **env** parameter defines a list of Environment Variables that must exist in all containers in the pod created by OLM.
- 7 The **envFrom** parameter defines a list of sources to populate Environment Variables in the container.
- 8 The **volumes** parameter defines a list of Volumes that must exist on the pod created by OLM.
- 9 The **volumeMounts** parameter defines a list of VolumeMounts that must exist in all containers in the pod created by OLM. If a **volumeMount** references a **volume** that does not exist, OLM fails to deploy the Operator.
- 10 The **tolerations** parameter defines a list of Tolerations for the pod created by OLM.
- 11 The **resources** parameter defines resource constraints for all the containers in the pod created by OLM.
- 12 The **nodeSelector** parameter defines a **NodeSelector** for the pod created by OLM.

5. Create the **Subscription** object:

```
$ oc apply -f sub.yaml
```

At this point, OLM is now aware of the selected Operator. A cluster service version (CSV) for the Operator should appear in the target namespace, and APIs provided by the Operator should be available for creation.

3.2.4. Deleting Operators from a cluster using the web console

Cluster administrators can delete installed Operators from a selected namespace by using the web console.

Prerequisites

- You have access to an OpenShift Container Platform cluster web console using an account with **cluster-admin** permissions.

Procedure

1. Navigate to the **Operators → Installed Operators** page.
2. Scroll or enter a keyword into the **Filter by name** field to find the Operator that you want to remove. Then, click on it.
3. On the right side of the **Operator Details** page, select **Uninstall Operator** from the **Actions** list. An **Uninstall Operator?** dialog box is displayed.
4. Select **Uninstall** to remove the Operator, Operator deployments, and pods. Following this action, the Operator stops running and no longer receives updates.



NOTE

This action does not remove resources managed by the Operator, including custom resource definitions (CRDs) and custom resources (CRs). Dashboards and navigation items enabled by the web console and off-cluster resources that continue to run might need manual clean up. To remove these after uninstalling the Operator, you might need to manually delete the Operator CRDs.

3.2.5. Deleting Operators from a cluster using the CLI

Cluster administrators can delete installed Operators from a selected namespace by using the CLI.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- The OpenShift CLI (**oc**) is installed on your workstation.

Procedure

1. Check the current version of the subscribed Operator (for example, **jaeger**) in the **currentCSV** field:

```
$ oc get subscription jaeger -n openshift-operators -o yaml | grep currentCSV
```

Example output

```
currentCSV: jaeger-operator.v1.8.2
```

2. Delete the subscription (for example, **jaeger**):

```
$ oc delete subscription jaeger -n openshift-operators
```

Example output

```
subscription.operators.coreos.com "jaeger" deleted
```

3. Delete the CSV for the Operator in the target namespace using the **currentCSV** value from the previous step:

```
$ oc delete clusterserviceversion jaeger-operator.v1.8.2 -n openshift-operators
```

Example output

```
clusterserviceversion.operators.coreos.com "jaeger-operator.v1.8.2" deleted
```


CHAPTER 4. LOGGING 5.6

4.1. CONFIGURING YOUR LOGGING DEPLOYMENT

You can configure your logging subsystem deployment with Custom Resource (CR) YAML files implemented by each Operator.

Red Hat Openshift Logging Operator:

- **ClusterLogging** (CL) - Deploys the collector and forwarder which currently are both implemented by a daemonset running on each node.
- **ClusterLogForwarder** (CLF) - Generates collector configuration to forward logs per user configuration.

Loki Operator:

- **LokiStack** - Controls the Loki cluster as log store and the web proxy with OpenShift Container Platform authentication integration to enforce multi-tenancy.

OpenShift Elasticsearch Operator:



NOTE

These CRs are generated and managed by the **ClusterLogging** Operator, manual changes cannot be made without being overwritten by the Operator.

- **ElasticSearch** - Configure and deploy an Elasticsearch instance as the default log store.
- **Kibana** - Configure and deploy Kibana instance to search, query and view logs.

Only the configuration options described in this documentation are supported for the logging subsystem.

Do not use any other configuration options, as they are unsupported. Configuration paradigms might change across OpenShift Container Platform releases, and such cases can only be handled gracefully if all configuration possibilities are controlled. If you use configurations other than those described in this documentation, your changes will be overwritten, because Operators are designed to reconcile any differences.



NOTE

If you must perform configurations not described in the OpenShift Container Platform documentation, you must set your Red Hat OpenShift Logging Operator to **Unmanaged**. An unmanaged logging subsystem for Red Hat OpenShift is not supported and does not receive updates until you return its status to **Managed**.

4.1.1. Enabling stream-based retention with Loki

With Logging version 5.6 and higher, you can configure retention policies based on log streams. Rules for these may be set globally, per tenant, or both. If you configure both, tenant rules apply before global rules.

1. To enable stream-based retention, create a **LokiStack** custom resource (CR):

Example global stream-based retention

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global: ❶
    retention: ❷
      days: 20
      streams:
        - days: 4
          priority: 1
          selector: '{kubernetes_namespace_name=~"test.+"}' ❸
        - days: 1
          priority: 1
          selector: '{log_type="infrastructure"}'
  managementState: Managed
  replicationFactor: 1
  size: 1x.small
  storage:
    schemas:
      - effectiveDate: "2020-10-11"
        version: v11
    secret:
      name: logging-loki-s3
      type: aws
  storageClassName: standard
  tenants:
    mode: openshift-logging

```

- ❶ Sets retention policy for all log streams. **Note: This field does not impact the retention period for stored logs in object storage.**
- ❷ Retention is enabled in the cluster when this block is added to the CR.
- ❸ Contains the [LogQL query](#) used to define the log stream.

Example per-tenant stream-based retention

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      retention:
        days: 20
    tenants: ❶
      application:

```

```

retention:
  days: 1
  streams:
    - days: 4
      selector: '{kubernetes_namespace_name=~"test.+"}' 2
infrastructure:
  retention:
    days: 5
    streams:
      - days: 1
        selector: '{kubernetes_namespace_name=~"openshift-cluster.+"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
    - effectiveDate: "2020-10-11"
      version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: standard
tenants:
  mode: openshift-logging

```

- 1 Sets retention policy by tenant. Valid tenant types are **application**, **audit**, and **infrastructure**.
- 2 Contains the [LogQL query](#) used to define the log stream.

2. Apply the **LokiStack** CR:

```
$ oc apply -f <filename>.yaml
```



NOTE

This is not for managing the retention for stored logs. Global retention periods for stored logs to a supported maximum of 30 days is configured with your object storage.

4.1.2. Enabling multi-line exception detection

Enables multi-line error detection of container logs.



WARNING

Enabling this feature could have performance implications and may require additional computing resources or alternate logging solutions.

Log parsers often incorrectly identify separate lines of the same exception as separate exceptions. This leads to extra log entries and an incomplete or inaccurate view of the traced information.

Example java exception

```
java.lang.NullPointerException: Cannot invoke "String.toString()" because "<param1>" is null
    at testjava.Main.handle(Main.java:47)
    at testjava.Main.printMe(Main.java:19)
    at testjava.Main.main(Main.java:10)
```

- To enable logging to detect multi-line exceptions and reassemble them into a single log entry, ensure that the **ClusterLogForwarder** Custom Resource (CR) contains a **detectMultilineErrors** field, with a value of **true**.

Example ClusterLogForwarder CR

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines:
    - name: my-app-logs
      inputRefs:
        - application
      outputRefs:
        - default
      detectMultilineErrors: true
```

4.1.2.1. Details

When log messages appear as a consecutive sequence forming an exception stack trace, they are combined into a single, unified log record. The first log message's content is replaced with the concatenated content of all the message fields in the sequence.

Table 4.1. Supported languages per collector:

Language	Fluentd	Vector
Java	✓	✓
JS	✓	✓
Ruby	✓	✓
Python	✓	✓
Golang	✓	✓
PHP	✓	

Language	Fluentd	Vector
Dart	✓	✓

4.1.2.2. Troubleshooting

When enabled, the collector configuration will include a new section with type: **detect_exceptions**

Example vector configuration section

```
[transforms.detect_exceptions_app-logs]
  type = "detect_exceptions"
  inputs = ["application"]
  languages = ["All"]
  group_by = ["kubernetes.namespace_name","kubernetes.pod_name","kubernetes.container_name"]
  expire_after_ms = 2000
  multiline_flush_interval_ms = 1000
```

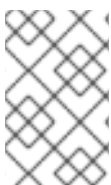
Example fluentd config section

```
<label @MULTILINE_APP_LOGS>
  <match kubernetes.**>
    @type detect_exceptions
    remove_tag_prefix 'kubernetes'
    message message
    force_line_breaks true
    multiline_flush_interval .2
  </match>
</label>
```

4.2. ADMINISTERING YOUR LOGGING DEPLOYMENT

4.2.1. Deploying Red Hat OpenShift Logging Operator using the web console

You can use the OpenShift Container Platform web console to deploy the Red Hat OpenShift Logging Operator.



PREREQUISITES

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.

Procedure

To deploy the Red Hat OpenShift Logging Operator using the OpenShift Container Platform web console:

1. Install the Red Hat OpenShift Logging Operator:
 - a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.

- b. Type **Logging** in the **Filter by keyword** field.
- c. Choose **Red Hat OpenShift Logging** from the list of available Operators, and click **Install**.
- d. Select **stable** or **stable-5.y** as the **Update Channel**.

**NOTE**

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

- e. Ensure that **A specific namespace on the cluster** is selected under **Installation Mode**.
 - f. Ensure that **Operator recommended namespace** is **openshift-logging** under **Installed Namespace**.
 - g. Select **Enable Operator recommended cluster monitoring on this Namespace**
 - h. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** option requires a user with appropriate credentials to approve the Operator update.
 - i. Select **Enable** or **Disable** for the Console plugin.
 - j. Click **Install**.
2. Verify that the **Red Hat OpenShift Logging Operator** is installed by switching to the **Operators** → **Installed Operators** page.
 - a. Ensure that **Red Hat OpenShift Logging** is listed in the **openshift-logging** project with a **Status** of **Succeeded**.
 3. Create a **ClusterLogging** instance.

**NOTE**

The form view of the web console does not include all available options. The **YAML view** is recommended for completing your setup.

- a. In the **collection** section, select a Collector Implementation.

**NOTE**

As of logging version 5.6 Fluentd is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to Fluentd, you can use Vector instead.

- b. In the **logStore** section, select a type.



NOTE

As of logging version 5.4.3 the OpenShift Elasticsearch Operator is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to using the OpenShift Elasticsearch Operator to manage the default log storage, you can use the Loki Operator.

- c. Click **Create**.

4.2.2. Deploying the Loki Operator using the web console

You can use the OpenShift Container Platform web console to install the Loki Operator.

Prerequisites

- Supported Log Store (AWS S3, Google Cloud Storage, Azure, Swift, Minio, OpenShift Data Foundation)

Procedure

To install the Loki Operator using the OpenShift Container Platform web console:

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Type **Loki** in the **Filter by keyword** field.
 - a. Choose **Loki Operator** from the list of available Operators, and click **Install**.
3. Select **stable** or **stable-5.y** as the **Update Channel**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

4. Ensure that **All namespaces on the cluster** is selected under **Installation Mode**.
5. Ensure that **openshift-operators-redhat** is selected under **Installed Namespace**.
6. Select **Enable Operator recommended cluster monitoring on this Namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.
7. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.

- The **Manual** option requires a user with appropriate credentials to approve the Operator update.
8. Click **Install**.
 9. Verify that the **LokiOperator** installed by switching to the **Operators → Installed Operators** page.
 - a. Ensure that **LokiOperator** is listed with **Status** as **Succeeded** in all the projects.
 10. Create a **Secret** YAML file that uses the **access_key_id** and **access_key_secret** fields to specify your credentials and **bucketnames**, **endpoint**, and **region** to define the object storage location. AWS is used in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

11. Select **Create instance** under LokiStack on the **Details** tab. Then select **YAML view**. Paste in the following template, substituting values where appropriate.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki ❶
  namespace: openshift-logging
spec:
  size: 1x.small ❷
  storage:
    schemas:
      - version: v12
        effectiveDate: '2022-06-01'
    secret:
      name: logging-loki-s3 ❸
      type: s3 ❹
  storageClassName: <storage_class_name> ❺
  tenants:
    mode: openshift-logging
```

- ❶ Name should be **logging-loki**.
- ❷ Select your Loki deployment size.
- ❸ Define the secret used for your log storage.
- ❹ Define corresponding storage type.

- 5 Enter the name of an existing storage class for temporary storage. For best performance, specify a storage class that allocates block storage. Available storage classes for your

- a. Apply the configuration:

```
oc apply -f logging-loki.yaml
```

12. Create or edit a **ClusterLogging** CR:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
  lokistack:
    name: logging-loki
  collection:
    type: vector
```

- a. Apply the configuration:

```
oc apply -f cr-lokistack.yaml
```

4.2.3. Installing from OperatorHub using the CLI

Instead of using the OpenShift Container Platform web console, you can install an Operator from OperatorHub by using the CLI. Use the **oc** command to create or update a **Subscription** object.

Prerequisites

- Access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. View the list of Operators available to the cluster from OperatorHub:

```
$ oc get packagemanifests -n openshift-marketplace
```

Example output

```
NAME                  CATALOG          AGE
3scale-operator       Red Hat Operators 91m
advanced-cluster-management Red Hat Operators 91m
amq7-cert-manager     Red Hat Operators 91m
...
couchbase-enterprise-certified Certified Operators 91m
```

crunchy-postgres-operator	Certified Operators	91m
mongodb-enterprise	Certified Operators	91m
...		
etcd	Community Operators	91m
jaeger	Community Operators	91m
kubefed	Community Operators	91m
...		

Note the catalog for your desired Operator.

- Inspect your desired Operator to verify its supported install modes and available channels:

```
$ oc describe packagemanifests <operator_name> -n openshift-marketplace
```

- An Operator group, defined by an **OperatorGroup** object, selects target namespaces in which to generate required RBAC access for all Operators in the same namespace as the Operator group.

The namespace to which you subscribe the Operator must have an Operator group that matches the install mode of the Operator, either the **AllNamespaces** or **SingleNamespace** mode. If the Operator you intend to install uses the **AllNamespaces**, then the **openshift-operators** namespace already has an appropriate Operator group in place.

However, if the Operator uses the **SingleNamespace** mode and you do not already have an appropriate Operator group in place, you must create one.



NOTE

The web console version of this procedure handles the creation of the **OperatorGroup** and **Subscription** objects automatically behind the scenes for you when choosing **SingleNamespace** mode.

- Create an **OperatorGroup** object YAML file, for example **operatorgroup.yaml**:

Example OperatorGroup object

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <namespace>
spec:
  targetNamespaces:
    - <namespace>
```

- Create the **OperatorGroup** object:

```
$ oc apply -f operatorgroup.yaml
```

- Create a **Subscription** object YAML file to subscribe a namespace to an Operator, for example **sub.yaml**:

Example Subscription object

```

apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: <subscription_name>
  namespace: openshift-operators 1
spec:
  channel: <channel_name> 2
  name: <operator_name> 3
  source: redhat-operators 4
  sourceNamespace: openshift-marketplace 5
  config:
    env: 6
    - name: ARGS
      value: "-v=10"
    envFrom: 7
    - secretRef:
        name: license-secret
  volumes: 8
  - name: <volume_name>
    configMap:
      name: <configmap_name>
  volumeMounts: 9
  - mountPath: <directory_name>
    name: <volume_name>
  tolerations: 10
  - operator: "Exists"
  resources: 11
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
  nodeSelector: 12
  foo: bar

```

- 1** For default **AllNamespaces** install mode usage, specify the **openshift-operators** namespace. Alternatively, you can specify a custom global namespace, if you have created one. Otherwise, specify the relevant single namespace for **SingleNamespace** install mode usage.
- 2** Name of the channel to subscribe to.
- 3** Name of the Operator to subscribe to.
- 4** Name of the catalog source that provides the Operator.
- 5** Namespace of the catalog source. Use **openshift-marketplace** for the default OperatorHub catalog sources.
- 6** The **env** parameter defines a list of Environment Variables that must exist in all containers in the pod created by OLM.
- 7** The **envFrom** parameter defines a list of sources to populate Environment Variables in the container.

- 8 The **volumes** parameter defines a list of Volumes that must exist on the pod created by OLM.
- 9 The **volumeMounts** parameter defines a list of VolumeMounts that must exist in all containers in the pod created by OLM. If a **volumeMount** references a **volume** that does not exist, OLM fails to deploy the Operator.
- 10 The **tolerations** parameter defines a list of Tolerations for the pod created by OLM.
- 11 The **resources** parameter defines resource constraints for all the containers in the pod created by OLM.
- 12 The **nodeSelector** parameter defines a **NodeSelector** for the pod created by OLM.

5. Create the **Subscription** object:

```
$ oc apply -f sub.yaml
```

At this point, OLM is now aware of the selected Operator. A cluster service version (CSV) for the Operator should appear in the target namespace, and APIs provided by the Operator should be available for creation.

4.2.4. Deleting Operators from a cluster using the web console

Cluster administrators can delete installed Operators from a selected namespace by using the web console.

Prerequisites

- You have access to an OpenShift Container Platform cluster web console using an account with **cluster-admin** permissions.

Procedure

- Navigate to the **Operators → Installed Operators** page.
- Scroll or enter a keyword into the **Filter by name** field to find the Operator that you want to remove. Then, click on it.
- On the right side of the **Operator Details** page, select **Uninstall Operator** from the **Actions** list. An **Uninstall Operator?** dialog box is displayed.
- Select **Uninstall** to remove the Operator, Operator deployments, and pods. Following this action, the Operator stops running and no longer receives updates.



NOTE

This action does not remove resources managed by the Operator, including custom resource definitions (CRDs) and custom resources (CRs). Dashboards and navigation items enabled by the web console and off-cluster resources that continue to run might need manual clean up. To remove these after uninstalling the Operator, you might need to manually delete the Operator CRDs.

4.2.5. Deleting Operators from a cluster using the CLI

Cluster administrators can delete installed Operators from a selected namespace by using the CLI.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- The OpenShift CLI (**oc**) is installed on your workstation.

Procedure

1. Check the current version of the subscribed Operator (for example, **jaeger**) in the **currentCSV** field:

```
$ oc get subscription jaeger -n openshift-operators -o yaml | grep currentCSV
```

Example output

```
currentCSV: jaeger-operator.v1.8.2
```

2. Delete the subscription (for example, **jaeger**):

```
$ oc delete subscription jaeger -n openshift-operators
```

Example output

```
subscription.operators.coreos.com "jaeger" deleted
```

3. Delete the CSV for the Operator in the target namespace using the **currentCSV** value from the previous step:

```
$ oc delete clusterserviceversion jaeger-operator.v1.8.2 -n openshift-operators
```

Example output

```
clusterserviceversion.operators.coreos.com "jaeger-operator.v1.8.2" deleted
```

4.3. LOGGING REFERENCES

4.3.1. Collector features

Output	Protocol	Tested with	Fluentd	Vector
Cloudwatch	REST over HTTP(S)		✓	✓
Elasticsearch v6		v6.8.1	✓	✓

Output	Protocol	Tested with	Fluentd	Vector
Elasticsearch v7		v7.12.2, 7.17.7	✓	✓
Elasticsearch v8		v8.4.3		✓
Fluent Forward	Fluentd forward v1	Fluentd 1.14.6, Logstash 7.10.1	✓	
Google Cloud Logging				✓
HTTP	HTTP 1.1	Fluentd 1.14.6, Vector 0.21		
Kafka	Kafka 0.11	Kafka 2.4.1, 2.7.0, 3.3.1	✓	✓
Loki	REST over HTTP(S)	Loki 2.3.0, 2.7	✓	✓
Splunk	HEC	v8.2.9, 9.0.0		✓
Syslog	RFC3164, RFC5424	Rsyslog 8.37.0- 9.e17	✓	

Table 4.2. Log Sources

Feature	Fluentd	Vector
App container logs	✓	✓
App-specific routing	✓	✓
App-specific routing by namespace	✓	✓
Infra container logs	✓	✓
Infra journal logs	✓	✓
Kube API audit logs	✓	✓
OpenShift API audit logs	✓	✓
Open Virtual Network (OVN) audit logs	✓	✓

Table 4.3. Authorization and Authentication

Feature	Fluentd	Vector
Elasticsearch certificates	✓	✓
Elasticsearch username / password	✓	✓
Cloudwatch keys	✓	✓
Cloudwatch STS	✓	✓
Kafka certificates	✓	✓
Kafka username / password	✓	✓
Kafka SASL	✓	✓
Loki bearer token	✓	✓

Table 4.4. Normalizations and Transformations

Feature	Fluentd	Vector
Viaq data model - app	✓	✓
Viaq data model - infra	✓	✓
Viaq data model - infra(journal)	✓	✓
Viaq data model - Linux audit	✓	✓
Viaq data model - kube-apiserver audit	✓	✓
Viaq data model - OpenShift API audit	✓	✓
Viaq data model - OVN	✓	✓
Loglevel Normalization	✓	✓
JSON parsing	✓	✓
Structured Index	✓	✓

Feature	Fluentd	Vector
Multiline error detection	✓	
Multicontainer / split indices	✓	✓
Flatten labels	✓	✓
CLF static labels	✓	✓

Table 4.5. Tuning

Feature	Fluentd	Vector
Fluentd readlinelimit	✓	
Fluentd buffer	✓	
- chunklimitsize	✓	
- totallimitsize	✓	
- overflowaction	✓	
- flushthreadcount	✓	
- flushmode	✓	
- flushinterval	✓	
- retrywait	✓	
- retrytype	✓	
- retrymaxinterval	✓	
- retrytimeout	✓	

Table 4.6. Visibility

Feature	Fluentd	Vector
Metrics	✓	✓
Dashboard	✓	✓

Feature	Fluentd	Vector
Alerts	✓	

Table 4.7. Miscellaneous

Feature	Fluentd	Vector
Global proxy support	✓	✓
x86 support	✓	✓
ARM support	✓	✓
IBM Power support	✓	✓
IBM Z support	✓	✓
IPv6 support	✓	✓
Log event buffering	✓	
Disconnected Cluster	✓	✓

Additional resources

- [Vector Documentation](#)

4.3.2. Logging 5.6 API reference

4.3.2.1. ClusterLogForwarder

ClusterLogForwarder is an API to configure forwarding logs.

You configure forwarding by specifying a list of **pipelines**, which forward from a set of named inputs to a set of named outputs.

There are built-in input names for common log categories, and you can define custom inputs to do additional filtering.

There is a built-in output name for the default openshift log store, but you can define your own outputs with a URL and other connection information to forward logs to other stores or processors, inside or outside the cluster.

For more details see the documentation on the API fields.

Property	Type	Description
spec	object	Specification of the desired behavior of ClusterLogForwarder
status	object	Status of the ClusterLogForwarder

4.3.2.1.1. .spec

4.3.2.1.1.1. Description

ClusterLogForwarderSpec defines how logs should be forwarded to remote targets.

4.3.2.1.1.1.1. Type

- object

Property	Type	Description
inputs	array	(optional) Inputs are named filters for log messages to be forwarded.
outputDefaults	object	(optional) DEPRECATED OutputDefaults specify forwarder config explicitly for the default store.
outputs	array	(optional) Outputs are named destinations for log messages.
pipelines	array	Pipelines forward the messages selected by a set of inputs to a set of outputs.

4.3.2.1.2. .spec.inputs[]

4.3.2.1.2.1. Description

InputSpec defines a selector of log messages.

4.3.2.1.2.1.1. Type

- array

Property	Type	Description
application	object	(optional) Application, if present, enables named set of application logs that
name	string	Name used to refer to the input of a pipeline .

4.3.2.1.3. .spec.inputs[].application

4.3.2.1.3.1. Description

Application log selector. All conditions in the selector must be satisfied (logical AND) to select logs.

4.3.2.1.3.1.1. Type

- object

Property	Type	Description
namespaces	array	(optional) Namespaces from which to collect application logs.
selector	object	(optional) Selector for logs from pods with matching labels.

4.3.2.1.4. .spec.inputs[].application.namespaces[]

4.3.2.1.4.1. Description

4.3.2.1.4.1.1. Type

- array

4.3.2.1.5. .spec.inputs[].application.selector

4.3.2.1.5.1. Description

A label selector is a label query over a set of resources.

4.3.2.1.5.1.1. Type

- object

Property	Type	Description
matchLabels	object	(optional) matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels

4.3.2.1.6. .spec.inputs[].application.selector.matchLabels

4.3.2.1.6.1. Description

4.3.2.1.6.1.1. Type

- object

4.3.2.1.7. .spec.outputDefaults

4.3.2.1.7.1. Description

4.3.2.1.7.1.1. Type

- object

Property	Type	Description
elasticsearch	object	(optional) Elasticsearch OutputSpec default values

4.3.2.1.8. .spec.outputDefaults.elasticsearch

4.3.2.1.8.1. Description

ElasticsearchStructuredSpec is spec related to structured log changes to determine the elasticsearch index

4.3.2.1.8.1.1. Type

- object

Property	Type	Description
enableStructuredContainerLogs	bool	(optional) EnableStructuredContainerLogs enables multi-container structured logs to allow

Property	Type	Description
structuredTypeKey	string	(optional) StructuredTypeKey specifies the metadata key to be used as name of elasticsearch index
structuredTypeName	string	(optional) StructuredTypeName specifies the name of elasticsearch schema

4.3.2.1.9. .spec.outputs[]

4.3.2.1.9.1. Description

Output defines a destination for log messages.

4.3.2.1.9.1.1. Type

- array

Property	Type	Description
syslog	object	(optional)
fluentdForward	object	(optional)
elasticsearch	object	(optional)
kafka	object	(optional)
cloudwatch	object	(optional)
loki	object	(optional)
googleCloudLogging	object	(optional)
splunk	object	(optional)
name	string	Name used to refer to the output from a pipeline .
secret	object	(optional) Secret for authentication.

Property	Type	Description
tls	object	TLS contains settings for controlling options on TLS client connections.
type	string	Type of output plugin.
url	string	(optional) URL to send log records to.

4.3.2.1.10. .spec.outputs[].secret

4.3.2.1.10.1. Description

OutputSecretSpec is a secret reference containing name only, no namespace.

4.3.2.1.10.1.1. Type

- object

Property	Type	Description
name	string	Name of a secret in the namespace configured for log forwarder secrets.

4.3.2.1.11. .spec.outputs[].tls

4.3.2.1.11.1. Description

OutputTLSSpec contains options for TLS connections that are agnostic to the output type.

4.3.2.1.11.1.1. Type

- object

Property	Type	Description
insecureSkipVerify	bool	If InsecureSkipVerify is true, then the TLS client will be configured to ignore errors with certificates.

4.3.2.1.12. .spec.pipelines[]

4.3.2.1.12.1. Description

PipelinesSpec link a set of inputs to a set of outputs.

4.3.2.1.12.1.1. Type

- array

Property	Type	Description
detectMultilineErrors	bool	(optional) DetectMultilineErrors enables multiline error detection of container logs
inputRefs	array	InputRefs lists the names (input.name) of inputs to this pipeline.
labels	object	(optional) Labels applied to log records passing through this pipeline.
name	string	(optional) Name is optional, but must be unique in the pipelines list if provided.
outputRefs	array	OutputRefs lists the names (output.name) of outputs from this pipeline.
parse	string	(optional) Parse enables parsing of log entries into structured logs

4.3.2.1.13. .spec.pipelines[].inputRefs[]

4.3.2.1.13.1. Description

4.3.2.1.13.1.1. Type

- array

4.3.2.1.14. .spec.pipelines[].labels

4.3.2.1.14.1. Description

4.3.2.1.14.1.1. Type

- object

4.3.2.1.15. .spec.pipelines[].outputRefs[]

4.3.2.1.15.1. Description

4.3.2.1.15.1.1. Type

- array

4.3.2.1.16. .status

4.3.2.1.16.1. Description

ClusterLogForwarderStatus defines the observed state of ClusterLogForwarder

4.3.2.1.16.1.1. Type

- object

Property	Type	Description
conditions	object	Conditions of the log forwarder.
inputs	Conditions	Inputs maps input name to condition of the input.
outputs	Conditions	Outputs maps output name to condition of the output.
pipelines	Conditions	Pipelines maps pipeline name to condition of the pipeline.

4.3.2.1.17. .status.conditions

4.3.2.1.17.1. Description

4.3.2.1.17.1.1. Type

- object

4.3.2.1.18. .status.inputs

4.3.2.1.18.1. Description

4.3.2.1.18.1.1. Type

- Conditions

4.3.2.1.19. .status.outputs

4.3.2.1.19.1. Description

4.3.2.1.19.1.1. Type

- Conditions

4.3.2.1.20. .status.pipelines

4.3.2.1.20.1. Description

4.3.2.1.20.1.1. Type

- Conditions== ClusterLogging A Red Hat OpenShift Logging instance. ClusterLogging is the Schema for the clusterloggings API

Property	Type	Description
spec	object	Specification of the desired behavior of ClusterLogging
status	object	Status defines the observed state of ClusterLogging

4.3.2.1.21. .spec

4.3.2.1.21.1. Description

ClusterLoggingSpec defines the desired state of ClusterLogging

4.3.2.1.21.1.1. Type

- object

Property	Type	Description
collection	object	Specification of the Collection component for the cluster
curation	object	(DEPRECATED) (optional) Deprecated. Specification of the Curation component for the cluster
forwarder	object	(DEPRECATED) (optional) Deprecated. Specification for Forwarder component for the cluster
logStore	object	(optional) Specification of the Log Storage component for the cluster

Property	Type	Description
managementState	string	(optional) Indicator if the resource is 'Managed' or 'Unmanaged' by the operator
visualization	object	(optional) Specification of the Visualization component for the cluster

4.3.2.1.22. .spec.collection

4.3.2.1.22.1. Description

This is the struct that will contain information pertinent to Log and event collection

4.3.2.1.22.1.1. Type

- object

Property	Type	Description
resources	object	(optional) The resource requirements for the collector
nodeSelector	object	(optional) Define which Nodes the Pods are scheduled on.
tolerations	array	(optional) Define the tolerations the Pods will accept
fluentd	object	(optional) Fluentd represents the configuration for forwarders of type fluentd.
logs	object	(DEPRECATED) (optional) Deprecated. Specification of Log Collection for the cluster
type	string	(optional) The type of Log Collection to configure

4.3.2.1.23. .spec.collection.fluentd

4.3.2.1.23.1. Description

FluentdForwarderSpec represents the configuration for forwarders of type fluentd.

4.3.2.1.23.1.1. Type

- object

Property	Type	Description
buffer	object	
inFile	object	

4.3.2.1.24. .spec.collection.fluentd.buffer

4.3.2.1.24.1. Description

FluentdBufferSpec represents a subset of fluentd buffer parameters to tune the buffer configuration for all fluentd outputs. It supports a subset of parameters to configure buffer and queue sizing, flush operations and retry flushing.

For general parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#buffering-parameters>

For flush parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#flushing-parameters>

For retry parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#retries-parameters>

4.3.2.1.24.1.1. Type

- object

Property	Type	Description
chunkLimitSize	string	(optional) ChunkLimitSize represents the maximum size of each chunk. Events will be
flushInterval	string	(optional) FlushInterval represents the time duration to wait between two consecutive flush
flushMode	string	(optional) FlushMode represents the mode of the flushing thread to write chunks. The mode
flushThreadCount	int	(optional) FlushThreadCount represents the number of threads used by the fluentd buffer

Property	Type	Description
overflowAction	string	(optional) OverflowAction represents the action for the fluentd buffer plugin to
retryMaxInterval	string	(optional) RetryMaxInterval represents the maximum time interval for exponential backoff
retryTimeout	string	(optional) RetryTimeout represents the maximum time interval to attempt retries before giving up
retryType	string	(optional) RetryType represents the type of retrying flush operations. Flush operations can
retryWait	string	(optional) RetryWait represents the time duration between two consecutive retries to flush
totalLimitSize	string	(optional) TotalLimitSize represents the threshold of node space allowed per fluentd

4.3.2.1.25. .spec.collection.fluentd.inFile

4.3.2.1.25.1. Description

FluentdInFileSpec represents a subset of fluentd in-tail plugin parameters to tune the configuration for all fluentd in-tail inputs.

For general parameters refer to: <https://docs.fluentd.org/input/tail#parameters>

4.3.2.1.25.1.1. Type

- object

Property	Type	Description
readLinesLimit	int	(optional) ReadLinesLimit represents the number of lines to read with each I/O operation

4.3.2.1.26. .spec.collection.logs

4.3.2.1.26.1. Description

4.3.2.126.1.1. Type

- object

Property	Type	Description
fluentd	object	Specification of the Fluentd Log Collection component
type	string	The type of Log Collection to configure

4.3.2.127. .spec.collection.logs.fluentd

4.3.2.127.1. Description

CollectorSpec is spec to define scheduling and resources for a collector

4.3.2.127.1.1. Type

- object

Property	Type	Description
nodeSelector	object	(optional) Define which Nodes the Pods are scheduled on.
resources	object	(optional) The resource requirements for the collector
tolerations	array	(optional) Define the tolerations the Pods will accept

4.3.2.128. .spec.collection.logs.fluentd.nodeSelector

4.3.2.128.1. Description

4.3.2.128.1.1. Type

- object

4.3.2.129. .spec.collection.logs.fluentd.resources

4.3.2.129.1. Description

4.3.2.129.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.1.30. .spec.collection.logs.fluentd.resources.limits

4.3.2.1.30.1. Description

4.3.2.1.30.1.1. Type

- object

4.3.2.1.31. .spec.collection.logs.fluentd.resources.requests

4.3.2.1.31.1. Description

4.3.2.1.31.1.1. Type

- object

4.3.2.1.32. .spec.collection.logs.fluentd.tolerations[]

4.3.2.1.32.1. Description

4.3.2.1.32.1.1. Type

- array

Property	Type	Description
effect	string	(optional) Effect indicates the taint effect to match. Empty means match all taint effects.
key	string	(optional) Key is the taint key that the toleration applies to. Empty means match all taint keys.
operator	string	(optional) Operator represents a key's relationship to the value.

Property	Type	Description
tolerationSeconds	int	(optional) TolerationSeconds represents the period of time the toleration (which must be
value	string	(optional) Value is the taint value the toleration matches to.

4.3.2.1.33. .spec.collection.logs.fluentd.tolerations[].tolerationSeconds

4.3.2.1.33.1. Description

4.3.2.1.33.1.1. Type

- int

4.3.2.1.34. .spec.curation

4.3.2.1.34.1. Description

This is the struct that will contain information pertinent to Log curation (Curator)

4.3.2.1.34.1.1. Type

- object

Property	Type	Description
curator	object	The specification of curation to configure
type	string	The kind of curation to configure

4.3.2.1.35. .spec.curation.curator

4.3.2.1.35.1. Description

4.3.2.1.35.1.1. Type

- object

Property	Type	Description
nodeSelector	object	Define which Nodes the Pods are scheduled on.

Property	Type	Description
resources	object	(optional) The resource requirements for Curator
schedule	string	The cron schedule that the Curator job is run. Defaults to "30 3 * * *"
tolerations	array	

4.3.2.1.36. .spec.curation.curator.nodeSelector

4.3.2.1.36.1. Description

4.3.2.1.36.1.1. Type

- object

4.3.2.1.37. .spec.curation.curator.resources

4.3.2.1.37.1. Description

4.3.2.1.37.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.1.38. .spec.curation.curator.resources.limits

4.3.2.1.38.1. Description

4.3.2.1.38.1.1. Type

- object

4.3.2.1.39. .spec.curation.curator.resources.requests

4.3.2.1.39.1. Description

4.3.2.1.39.1.1. Type

- object

4.3.2.1.40. .spec.curation.curator.tolerations[]

4.3.2.1.40.1. Description

4.3.2.1.40.1.1. Type

- array

Property	Type	Description
effect	string	(optional) Effect indicates the taint effect to match. Empty means match all taint effects.
key	string	(optional) Key is the taint key that the toleration applies to. Empty means match all taint keys.
operator	string	(optional) Operator represents a key's relationship to the value.
tolerationSeconds	int	(optional) TolerationSeconds represents the period of time the toleration (which must be
value	string	(optional) Value is the taint value the toleration matches to.

4.3.2.1.41. .spec.curation.curator.tolerations[].tolerationSeconds

4.3.2.1.41.1. Description

4.3.2.1.41.1.1. Type

- int

4.3.2.1.42. .spec.forwarder

4.3.2.1.42.1. Description

ForwarderSpec contains global tuning parameters for specific forwarder implementations. This field is not required for general use, it allows performance tuning by users familiar with the underlying forwarder technology. Currently supported: **fluentd**.

4.3.2.1.42.1.1. Type

- object

Property	Type	Description
fluentd	object	

4.3.2.1.43. .spec.forwarder.fluentd

4.3.2.1.43.1. Description

FluentdForwarderSpec represents the configuration for forwarders of type fluentd.

4.3.2.1.43.1.1. Type

- object

Property	Type	Description
buffer	object	
inFile	object	

4.3.2.1.44. .spec.forwarder.fluentd.buffer

4.3.2.1.44.1. Description

FluentdBufferSpec represents a subset of fluentd buffer parameters to tune the buffer configuration for all fluentd outputs. It supports a subset of parameters to configure buffer and queue sizing, flush operations and retry flushing.

For general parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#buffering-parameters>

For flush parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#flushing-parameters>

For retry parameters refer to: <https://docs.fluentd.org/configuration/buffer-section#retries-parameters>

4.3.2.1.44.1.1. Type

- object

Property	Type	Description
chunkLimitSize	string	(optional) ChunkLimitSize represents the maximum size of each chunk. Events will be

Property	Type	Description
flushInterval	string	(optional) FlushInterval represents the time duration to wait between two consecutive flush
flushMode	string	(optional) FlushMode represents the mode of the flushing thread to write chunks. The mode
flushThreadCount	int	(optional) FlushThreadCount represents the number of threads used by the fluentd buffer
overflowAction	string	(optional) OverflowAction represents the action for the fluentd buffer plugin to
retryMaxInterval	string	(optional) RetryMaxInterval represents the maximum time interval for exponential backoff
retryTimeout	string	(optional) RetryTimeout represents the maximum time interval to attempt retries before giving up
retryType	string	(optional) RetryType represents the type of retrying flush operations. Flush operations can
retryWait	string	(optional) RetryWait represents the time duration between two consecutive retries to flush
totalLimitSize	string	(optional) TotalLimitSize represents the threshold of node space allowed per fluentd

4.3.2.1.45. .spec.forwarder.fluentd.inFile

4.3.2.1.45.1. Description

FluentdInFileSpec represents a subset of fluentd in-tail plugin parameters to tune the configuration for all fluentd in-tail inputs.

For general parameters refer to: <https://docs.fluentd.org/input/tail#parameters>

4.3.2.145.1.1. Type

- object

Property	Type	Description
readLinesLimit	int	(optional) ReadLinesLimit represents the number of lines to read with each I/O operation

4.3.2.146. .spec.logStore

4.3.2.146.1. Description

The LogStoreSpec contains information about how logs are stored.

4.3.2.146.1.1. Type

- object

Property	Type	Description
elasticsearch	object	Specification of the Elasticsearch Log Store component
lokistack	object	LokiStack contains information about which LokiStack to use for log storage if Type is set to LogStoreTypeLokiStack.
retentionPolicy	object	(optional) Retention policy defines the maximum age for an index after which it should be deleted
type	string	The Type of Log Storage to configure. The operator currently supports either using ElasticSearch

4.3.2.147. .spec.logStore.elasticsearch

4.3.2.147.1. Description

4.3.2.147.1.1. Type

- object

Property	Type	Description
nodeCount	int	Number of nodes to deploy for Elasticsearch
nodeSelector	object	Define which Nodes the Pods are scheduled on.
proxy	object	Specification of the Elasticsearch Proxy component
redundancyPolicy	string	(optional)
resources	object	(optional) The resource requirements for Elasticsearch
storage	object	(optional) The storage specification for Elasticsearch data nodes
tolerations	array	

4.3.2.1.48. .spec.logStore.elasticsearch.nodeSelector

4.3.2.1.48.1. Description

4.3.2.1.48.1.1. Type

- object

4.3.2.1.49. .spec.logStore.elasticsearch.proxy

4.3.2.1.49.1. Description

4.3.2.1.49.1.1. Type

- object

Property	Type	Description
resources	object	

4.3.2.1.50. .spec.logStore.elasticsearch.proxy.resources

4.3.2.1.50.1. Description

4.3.2.150.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.151. .spec.logStore.elasticsearch.proxy.resources.limits

4.3.2.151.1. Description

4.3.2.151.1.1. Type

- object

4.3.2.152. .spec.logStore.elasticsearch.proxy.resources.requests

4.3.2.152.1. Description

4.3.2.152.1.1. Type

- object

4.3.2.153. .spec.logStore.elasticsearch.resources

4.3.2.153.1. Description

4.3.2.153.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.154. .spec.logStore.elasticsearch.resources.limits

4.3.2.154.1. Description

4.3.2.154.1.1. Type

- object

4.3.2.155. .spec.logStore.elasticsearch.resources.requests

4.3.2.155.1. Description

4.3.2.155.1.1. Type

- object

4.3.2.156. .spec.logStore.elasticsearch.storage

4.3.2.156.1. Description

4.3.2.156.1.1. Type

- object

Property	Type	Description
size	object	The max storage capacity for the node to provision.
storageClassName	string	(optional) The name of the storage class to use with creating the node's PVC.

4.3.2.157. .spec.logStore.elasticsearch.storage.size

4.3.2.157.1. Description

4.3.2.157.1.1. Type

- object

Property	Type	Description
Format	string	Change Format at will. See the comment for Canonicalize for
d	object	d is the quantity in inf.Dec form if d.Dec != nil

Property	Type	Description
i	int	i is the quantity in int64 scaled form, if d.Dec == nil
s	string	s is the generated value of this quantity to avoid recalculation

4.3.2.158. .spec.logStore.elasticsearch.storage.size.d

4.3.2.158.1. Description

4.3.2.158.1.1. Type

- object

Property	Type	Description
Dec	object	

4.3.2.159. .spec.logStore.elasticsearch.storage.size.d.Dec

4.3.2.159.1. Description

4.3.2.159.1.1. Type

- object

Property	Type	Description
scale	int	
unscaled	object	

4.3.2.160. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled

4.3.2.160.1. Description

4.3.2.160.1.1. Type

- object

Property	Type	Description
abs	Word	sign

Property	Type	Description
neg	bool	

4.3.2.1.61. .spec.logStore.elasticsearch.storage.size.d.Dec.unscaled.abs

4.3.2.1.61.1. Description

4.3.2.1.61.1.1. Type

- Word

4.3.2.1.62. .spec.logStore.elasticsearch.storage.size.i

4.3.2.1.62.1. Description

4.3.2.1.62.1.1. Type

- int

Property	Type	Description
scale	int	
value	int	

4.3.2.1.63. .spec.logStore.elasticsearch.tolerations[]

4.3.2.1.63.1. Description

4.3.2.1.63.1.1. Type

- array

Property	Type	Description
effect	string	(optional) Effect indicates the taint effect to match. Empty means match all taint effects.
key	string	(optional) Key is the taint key that the toleration applies to. Empty means match all taint keys.
operator	string	(optional) Operator represents a key's relationship to the value.

Property	Type	Description
tolerationSeconds	int	(optional) TolerationSeconds represents the period of time the toleration (which must be
value	string	(optional) Value is the taint value the toleration matches to.

4.3.2.1.64. .spec.logStore.elasticsearch.tolerations[].tolerationSeconds

4.3.2.1.64.1. Description

4.3.2.1.64.1.1. Type

- int

4.3.2.1.65. .spec.logStore.lokiStack

4.3.2.1.65.1. Description

LokiStackStoreSpec is used to set up cluster-logging to use a LokiStack as logging storage. It points to an existing LokiStack in the same namespace.

4.3.2.1.65.1.1. Type

- object

Property	Type	Description
name	string	Name of the LokiStack resource.

4.3.2.1.66. .spec.logStore.retentionPolicy

4.3.2.1.66.1. Description

4.3.2.1.66.1.1. Type

- object

Property	Type	Description
application	object	
audit	object	

Property	Type	Description
infra	object	

4.3.2.1.67. .spec.logStore.retentionPolicy.application

4.3.2.1.67.1. Description

4.3.2.1.67.1.1. Type

- object

Property	Type	Description
diskThresholdPercent	int	(optional) The threshold percentage of ES disk usage that when reached, old indices should be deleted (e.g. 75)
maxAge	string	(optional)
namespaceSpec	array	(optional) The per namespace specification to delete documents older than a given minimum age
pruneNamespacesInterval	string	(optional) How often to run a new prune-namespaces job

4.3.2.1.68. .spec.logStore.retentionPolicy.application.namespaceSpec[]

4.3.2.1.68.1. Description

4.3.2.1.68.1.1. Type

- array

Property	Type	Description
minAge	string	(optional) Delete the records matching the namespaces which are older than this MinAge (e.g. 1d)
namespace	string	Target Namespace to delete logs older than MinAge (defaults to 7d)

4.3.2.1.69. .spec.logStore.retentionPolicy.audit

4.3.2.1.69.1. Description

4.3.2.1.69.1.1. Type

- object

Property	Type	Description
diskThresholdPercent	int	(optional) The threshold percentage of ES disk usage that when reached, old indices should be deleted (e.g. 75)
maxAge	string	(optional)
namespaceSpec	array	(optional) The per namespace specification to delete documents older than a given minimum age
pruneNamespacesInterval	string	(optional) How often to run a new prune-namespaces job

4.3.2.1.70. .spec.logStore.retentionPolicy.audit.namespaceSpec[]

4.3.2.1.70.1. Description

4.3.2.1.70.1.1. Type

- array

Property	Type	Description
minAge	string	(optional) Delete the records matching the namespaces which are older than this MinAge (e.g. 1d)
namespace	string	Target Namespace to delete logs older than MinAge (defaults to 7d)

4.3.2.1.71. .spec.logStore.retentionPolicy.infra

4.3.2.1.71.1. Description

4.3.2.1.71.1.1. Type

- object

Property	Type	Description
diskThresholdPercent	int	(optional) The threshold percentage of ES disk usage that when reached, old indices should be deleted (e.g. 75)
maxAge	string	(optional)
namespaceSpec	array	(optional) The per namespace specification to delete documents older than a given minimum age
pruneNamespacesInterval	string	(optional) How often to run a new prune-namespaces job

4.3.2.1.72. .spec.logStore.retentionPolicy.infra.namespaceSpec[]

4.3.2.1.72.1. Description

4.3.2.1.72.1.1. Type

- array

Property	Type	Description
minAge	string	(optional) Delete the records matching the namespaces which are older than this MinAge (e.g. 1d)
namespace	string	Target Namespace to delete logs older than MinAge (defaults to 7d)

4.3.2.1.73. .spec.visualization

4.3.2.1.73.1. Description

This is the struct that will contain information pertinent to Log visualization (Kibana)

4.3.2.1.73.1.1. Type

- object

Property	Type	Description
kibana	object	Specification of the Kibana Visualization component
type	string	The type of Visualization to configure

4.3.2.174. .spec.visualization.kibana

4.3.2.174.1. Description

4.3.2.174.1.1. Type

- object

Property	Type	Description
nodeSelector	object	Define which Nodes the Pods are scheduled on.
proxy	object	Specification of the Kibana Proxy component
replicas	int	Number of instances to deploy for a Kibana deployment
resources	object	(optional) The resource requirements for Kibana
tolerations	array	

4.3.2.175. .spec.visualization.kibana.nodeSelector

4.3.2.175.1. Description

4.3.2.175.1.1. Type

- object

4.3.2.176. .spec.visualization.kibana.proxy

4.3.2.176.1. Description

4.3.2.176.1.1. Type

- object

Property	Type	Description
resources	object	

4.3.2.1.77. .spec.visualization.kibana.proxy.resources

4.3.2.1.77.1. Description

4.3.2.1.77.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.1.78. .spec.visualization.kibana.proxy.resources.limits

4.3.2.1.78.1. Description

4.3.2.1.78.1.1. Type

- object

4.3.2.1.79. .spec.visualization.kibana.proxy.resources.requests

4.3.2.1.79.1. Description

4.3.2.1.79.1.1. Type

- object

4.3.2.1.80. .spec.visualization.kibana.replicas

4.3.2.1.80.1. Description

4.3.2.1.80.1.1. Type

- int

4.3.2.1.81. `.spec.visualization.kibana.resources`

4.3.2.1.81.1. Description

4.3.2.1.81.1.1. Type

- object

Property	Type	Description
limits	object	(optional) Limits describes the maximum amount of compute resources allowed.
requests	object	(optional) Requests describes the minimum amount of compute resources required.

4.3.2.1.82. `.spec.visualization.kibana.resources.limits`

4.3.2.1.82.1. Description

4.3.2.1.82.1.1. Type

- object

4.3.2.1.83. `.spec.visualization.kibana.resources.requests`

4.3.2.1.83.1. Description

4.3.2.1.83.1.1. Type

- object

4.3.2.1.84. `.spec.visualization.kibana.tolerations[]`

4.3.2.1.84.1. Description

4.3.2.1.84.1.1. Type

- array

Property	Type	Description
effect	string	(optional) Effect indicates the taint effect to match. Empty means match all taint effects.

Property	Type	Description
key	string	(optional) Key is the taint key that the toleration applies to. Empty means match all taint keys.
operator	string	(optional) Operator represents a key's relationship to the value.
tolerationSeconds	int	(optional) TolerationSeconds represents the period of time the toleration (which must be
value	string	(optional) Value is the taint value the toleration matches to.

4.3.2.1.85. `.spec.visualization.kibana.tolerations[].tolerationSeconds`

4.3.2.1.85.1. Description

4.3.2.1.85.1.1. Type

- int

4.3.2.1.86. `.status`

4.3.2.1.86.1. Description

ClusterLoggingStatus defines the observed state of ClusterLogging

4.3.2.1.86.1.1. Type

- object

Property	Type	Description
collection	object	(optional)
conditions	object	(optional)
curation	object	(optional)
logStore	object	(optional)
visualization	object	(optional)

4.3.2.1.87. `.status.collection`

4.3.2.1.87.1. Description

4.3.2.1.87.1.1. Type

- object

Property	Type	Description
logs	object	(optional)

4.3.2.1.88. .status.collection.logs

4.3.2.1.88.1. Description

4.3.2.1.88.1.1. Type

- object

Property	Type	Description
fluentdStatus	object	(optional)

4.3.2.1.89. .status.collection.logs.fluentdStatus

4.3.2.1.89.1. Description

4.3.2.1.89.1.1. Type

- object

Property	Type	Description
clusterCondition	object	(optional)
daemonSet	string	(optional)
nodes	object	(optional)
Pods	string	(optional)

4.3.2.1.90. .status.collection.logs.fluentdStatus.clusterCondition

4.3.2.1.90.1. Description

operator-sdk generate crds does not allow map-of-slice, must use a named type.

4.3.2.1.90.1.1. Type

- object

4.3.2.1.91. .status.collection.logs.fluentdStatus.nodes

4.3.2.1.91.1. Description

4.3.2.1.91.1.1. Type

- object

4.3.2.1.92. .status.conditions

4.3.2.1.92.1. Description

4.3.2.1.92.1.1. Type

- object

4.3.2.1.93. .status.curation

4.3.2.1.93.1. Description

4.3.2.1.93.1.1. Type

- object

Property	Type	Description
curatorStatus	array	(optional)

4.3.2.1.94. .status.curation.curatorStatus[]

4.3.2.1.94.1. Description

4.3.2.1.94.1.1. Type

- array

Property	Type	Description
clusterCondition	object	(optional)
cronJobs	string	(optional)
schedules	string	(optional)

Property	Type	Description
suspended	bool	(optional)

4.3.2.1.95. `.status.curation.curatorStatus[].clusterCondition`

4.3.2.1.95.1. Description

operator-sdk generate crds does not allow map-of-slice, must use a named type.

4.3.2.1.95.1.1. Type

- object

4.3.2.1.96. `.status.logStore`

4.3.2.1.96.1. Description

4.3.2.1.96.1.1. Type

- object

Property	Type	Description
elasticsearchStatus	array	(optional)

4.3.2.1.97. `.status.logStore.elasticsearchStatus[]`

4.3.2.1.97.1. Description

4.3.2.1.97.1.1. Type

- array

Property	Type	Description
cluster	object	(optional)
clusterConditions	object	(optional)
clusterHealth	string	(optional)
clusterName	string	(optional)
deployments	array	(optional)

Property	Type	Description
nodeConditions	object	(optional)
nodeCount	int	(optional)
Pods	object	(optional)
replicaSets	array	(optional)
shardAllocationEnabled	string	(optional)
statefulSets	array	(optional)

4.3.2.1.98. .status.logStore.elasticsearchStatus[].cluster

4.3.2.1.98.1. Description

4.3.2.1.98.1.1. Type

- object

Property	Type	Description
activePrimaryShards	int	The number of Active Primary Shards for the Elasticsearch Cluster
activeShards	int	The number of Active Shards for the Elasticsearch Cluster
initializingShards	int	The number of Initializing Shards for the Elasticsearch Cluster
numDataNodes	int	The number of Data Nodes for the Elasticsearch Cluster
numNodes	int	The number of Nodes for the Elasticsearch Cluster
pendingTasks	int	
relocatingShards	int	The number of Relocating Shards for the Elasticsearch Cluster
status	string	The current Status of the Elasticsearch Cluster

Property	Type	Description
unassignedShards	int	The number of Unassigned Shards for the Elasticsearch Cluster

4.3.2.1.99. `.status.logStore.elasticsearchStatus[].clusterConditions`

4.3.2.1.99.1. Description

4.3.2.1.99.1.1. Type

- object

4.3.2.1.100. `.status.logStore.elasticsearchStatus[].deployments[]`

4.3.2.1.100.1. Description

4.3.2.1.100.1.1. Type

- array

4.3.2.1.101. `.status.logStore.elasticsearchStatus[].nodeConditions`

4.3.2.1.101.1. Description

4.3.2.1.101.1.1. Type

- object

4.3.2.1.102. `.status.logStore.elasticsearchStatus[].pods`

4.3.2.1.102.1. Description

4.3.2.1.102.1.1. Type

- object

4.3.2.1.103. `.status.logStore.elasticsearchStatus[].replicaSets[]`

4.3.2.1.103.1. Description

4.3.2.1.103.1.1. Type

- array

4.3.2.1.104. `.status.logStore.elasticsearchStatus[].statefulSets[]`

4.3.2.1.104.1. Description

4.3.2.1.104.1.1. Type

- array

4.3.2.1.105. .status.visualization

4.3.2.1.105.1. Description

4.3.2.1.105.1.1. Type

- object

Property	Type	Description
kibanaStatus	array	(optional)

4.3.2.1.106. .status.visualization.kibanaStatus[]

4.3.2.1.106.1. Description

4.3.2.1.106.1.1. Type

- array

Property	Type	Description
clusterCondition	object	(optional)
deployment	string	(optional)
Pods	string	(optional) The status for each of the Kibana pods for the Visualization component
replicaSets	array	(optional)
replicas	int	(optional)

4.3.2.1.107. .status.visualization.kibanaStatus[].clusterCondition

4.3.2.1.107.1. Description

4.3.2.1.107.1.1. Type

- object

4.3.2.1.108. `.status.visualization.kibanaStatus[].replicaSets[]`

4.3.2.1.108.1. Description

4.3.2.1.108.1.1. Type

- array

CHAPTER 5. LOGGING 5.5

5.1. ADMINISTERING YOUR LOGGING DEPLOYMENT

5.1.1. Deploying Red Hat OpenShift Logging Operator using the web console

You can use the OpenShift Container Platform web console to deploy the Red Hat OpenShift Logging Operator.



PREREQUISITES

Logging is provided as an installable component, with a distinct release cycle from the core OpenShift Container Platform. The [Red Hat OpenShift Container Platform Life Cycle Policy](#) outlines release compatibility.

Procedure

To deploy the Red Hat OpenShift Logging Operator using the OpenShift Container Platform web console:

1. Install the Red Hat OpenShift Logging Operator:
 - a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
 - b. Type **Logging** in the **Filter by keyword** field.
 - c. Choose **Red Hat OpenShift Logging** from the list of available Operators, and click **Install**.
 - d. Select **stable** or **stable-5.y** as the **Update Channel**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

- e. Ensure that **A specific namespace on the cluster** is selected under **Installation Mode**.
- f. Ensure that **Operator recommended namespace** is **openshift-logging** under **Installed Namespace**.
- g. Select **Enable Operator recommended cluster monitoring on this Namespace**
- h. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** option requires a user with appropriate credentials to approve the Operator update.
- i. Select **Enable** or **Disable** for the Console plugin.

- j. Click **Install**.
2. Verify that the **Red Hat OpenShift Logging Operator** is installed by switching to the **Operators** → **Installed Operators** page.
 - a. Ensure that **Red Hat OpenShift Logging** is listed in the **openshift-logging** project with a **Status** of **Succeeded**.
3. Create a **ClusterLogging** instance.

**NOTE**

The form view of the web console does not include all available options. The **YAML view** is recommended for completing your setup.

- a. In the **collection** section, select a Collector Implementation.

**NOTE**

As of logging version 5.6 Fluentd is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to Fluentd, you can use Vector instead.

- b. In the **logStore** section, select a type.

**NOTE**

As of logging version 5.4.3 the OpenShift Elasticsearch Operator is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to using the OpenShift Elasticsearch Operator to manage the default log storage, you can use the Loki Operator.

- c. Click **Create**.

5.1.2. Deploying the Loki Operator using the web console

You can use the OpenShift Container Platform web console to install the Loki Operator.

Prerequisites

- Supported Log Store (AWS S3, Google Cloud Storage, Azure, Swift, Minio, OpenShift Data Foundation)

Procedure

To install the Loki Operator using the OpenShift Container Platform web console:

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Type **Loki** in the **Filter by keyword** field.

- a. Choose **Loki Operator** from the list of available Operators, and click **Install**.
3. Select **stable** or **stable-5.y** as the **Update Channel**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

4. Ensure that **All namespaces on the cluster** is selected under **Installation Mode**.
5. Ensure that **openshift-operators-redhat** is selected under **Installed Namespace**.
6. Select **Enable Operator recommended cluster monitoring on this Namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.
7. Select an option for **Update approval**.
 - The **Automatic** option allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** option requires a user with appropriate credentials to approve the Operator update.
8. Click **Install**.
9. Verify that the **LokiOperator** installed by switching to the **Operators → Installed Operators** page.
 - a. Ensure that **LokiOperator** is listed with **Status** as **Succeeded** in all the projects.
10. Create a **Secret** YAML file that uses the **access_key_id** and **access_key_secret** fields to specify your credentials and **bucketnames**, **endpoint**, and **region** to define the object storage location. AWS is used in the following example:

```
apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

11. Select **Create instance** under **LokiStack** on the **Details** tab. Then select **YAML view**. Paste in the following template, substituting values where appropriate.

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
```

```

metadata:
  name: logging-loki 1
  namespace: openshift-logging
spec:
  size: 1x.small 2
  storage:
    schemas:
      - version: v12
        effectiveDate: '2022-06-01'
    secret:
      name: logging-loki-s3 3
      type: s3 4
  storageClassName: <storage_class_name> 5
  tenants:
    mode: openshift-logging

```

- 1** Name should be **logging-loki**.
- 2** Select your Loki deployment size.
- 3** Define the secret used for your log storage.
- 4** Define corresponding storage type.
- 5** Enter the name of an existing storage class for temporary storage. For best performance, specify a storage class that allocates block storage. Available storage classes for your cluster can be listed using **oc get storageclasses**.

a. Apply the configuration:

```
oc apply -f logging-loki.yaml
```

12. Create or edit a **ClusterLogging** CR:

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
    collection:
      type: vector

```

a. Apply the configuration:

```
oc apply -f cr-lokistack.yaml
```

5.1.3. Installing from OperatorHub using the CLI

Instead of using the OpenShift Container Platform web console, you can install an Operator from OperatorHub by using the CLI. Use the **oc** command to create or update a **Subscription** object.

Prerequisites

- Access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. View the list of Operators available to the cluster from OperatorHub:

```
$ oc get packagemanifests -n openshift-marketplace
```

Example output

NAME	CATALOG	AGE
3scale-operator	Red Hat Operators	91m
advanced-cluster-management	Red Hat Operators	91m
amq7-cert-manager	Red Hat Operators	91m
...		
couchbase-enterprise-certified	Certified Operators	91m
crunchy-postgres-operator	Certified Operators	91m
mongodb-enterprise	Certified Operators	91m
...		
etcd	Community Operators	91m
jaeger	Community Operators	91m
kubefed	Community Operators	91m
...		

Note the catalog for your desired Operator.

2. Inspect your desired Operator to verify its supported install modes and available channels:

```
$ oc describe packagemanifests <operator_name> -n openshift-marketplace
```

3. An Operator group, defined by an **OperatorGroup** object, selects target namespaces in which to generate required RBAC access for all Operators in the same namespace as the Operator group.

The namespace to which you subscribe the Operator must have an Operator group that matches the install mode of the Operator, either the **AllNamespaces** or **SingleNamespace** mode. If the Operator you intend to install uses the **AllNamespaces**, then the **openshift-operators** namespace already has an appropriate Operator group in place.

However, if the Operator uses the **SingleNamespace** mode and you do not already have an appropriate Operator group in place, you must create one.



NOTE

The web console version of this procedure handles the creation of the **OperatorGroup** and **Subscription** objects automatically behind the scenes for you when choosing **SingleNamespace** mode.

- a. Create an **OperatorGroup** object YAML file, for example **operatorgroup.yaml**:

Example OperatorGroup object

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <operatorgroup_name>
  namespace: <namespace>
spec:
  targetNamespaces:
    - <namespace>
```

- b. Create the **OperatorGroup** object:

```
$ oc apply -f operatorgroup.yaml
```

4. Create a **Subscription** object YAML file to subscribe a namespace to an Operator, for example **sub.yaml**:

Example Subscription object

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: <subscription_name>
  namespace: openshift-operators ❶
spec:
  channel: <channel_name> ❷
  name: <operator_name> ❸
  source: redhat-operators ❹
  sourceNamespace: openshift-marketplace ❺
  config:
    env: ❻
    - name: ARGS
      value: "-v=10"
    envFrom: ❼
    - secretRef:
        name: license-secret
  volumes: ❽
  - name: <volume_name>
    configMap:
      name: <configmap_name>
  volumeMounts: ❾
  - mountPath: <directory_name>
    name: <volume_name>
  tolerations: ❿
  - operator: "Exists"
  resources: ⓫
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
```

```
memory: "128Mi"
cpu: "500m"
nodeSelector: 12
foo: bar
```

- 1 For default **AllNamespaces** install mode usage, specify the **openshift-operators** namespace. Alternatively, you can specify a custom global namespace, if you have created one. Otherwise, specify the relevant single namespace for **SingleNamespace** install mode usage.
- 2 Name of the channel to subscribe to.
- 3 Name of the Operator to subscribe to.
- 4 Name of the catalog source that provides the Operator.
- 5 Namespace of the catalog source. Use **openshift-marketplace** for the default OperatorHub catalog sources.
- 6 The **env** parameter defines a list of Environment Variables that must exist in all containers in the pod created by OLM.
- 7 The **envFrom** parameter defines a list of sources to populate Environment Variables in the container.
- 8 The **volumes** parameter defines a list of Volumes that must exist on the pod created by OLM.
- 9 The **volumeMounts** parameter defines a list of VolumeMounts that must exist in all containers in the pod created by OLM. If a **volumeMount** references a **volume** that does not exist, OLM fails to deploy the Operator.
- 10 The **tolerations** parameter defines a list of Tolerations for the pod created by OLM.
- 11 The **resources** parameter defines resource constraints for all the containers in the pod created by OLM.
- 12 The **nodeSelector** parameter defines a **NodeSelector** for the pod created by OLM.

5. Create the **Subscription** object:

```
$ oc apply -f sub.yaml
```

At this point, OLM is now aware of the selected Operator. A cluster service version (CSV) for the Operator should appear in the target namespace, and APIs provided by the Operator should be available for creation.

5.1.4. Deleting Operators from a cluster using the web console

Cluster administrators can delete installed Operators from a selected namespace by using the web console.

Prerequisites

- You have access to an OpenShift Container Platform cluster web console using an account with **cluster-admin** permissions.

Procedure

1. Navigate to the **Operators → Installed Operators** page.
2. Scroll or enter a keyword into the **Filter by name** field to find the Operator that you want to remove. Then, click on it.
3. On the right side of the **Operator Details** page, select **Uninstall Operator** from the **Actions** list. An **Uninstall Operator?** dialog box is displayed.
4. Select **Uninstall** to remove the Operator, Operator deployments, and pods. Following this action, the Operator stops running and no longer receives updates.



NOTE

This action does not remove resources managed by the Operator, including custom resource definitions (CRDs) and custom resources (CRs). Dashboards and navigation items enabled by the web console and off-cluster resources that continue to run might need manual clean up. To remove these after uninstalling the Operator, you might need to manually delete the Operator CRDs.

5.1.5. Deleting Operators from a cluster using the CLI

Cluster administrators can delete installed Operators from a selected namespace by using the CLI.

Prerequisites

- You have access to an OpenShift Container Platform cluster using an account with **cluster-admin** permissions.
- The OpenShift CLI (**oc**) is installed on your workstation.

Procedure

1. Check the current version of the subscribed Operator (for example, **jaeger**) in the **currentCSV** field:

```
$ oc get subscription jaeger -n openshift-operators -o yaml | grep currentCSV
```

Example output

```
currentCSV: jaeger-operator.v1.8.2
```

2. Delete the subscription (for example, **jaeger**):

```
$ oc delete subscription jaeger -n openshift-operators
```

Example output

```
subscription.operators.coreos.com "jaeger" deleted
```


3. Delete the CSV for the Operator in the target namespace using the **currentCSV** value from the previous step:

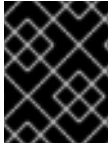
```
$ oc delete clusterserviceversion jaeger-operator.v1.8.2 -n openshift-operators
```

Example output

```
clusterserviceversion.operators.coreos.com "jaeger-operator.v1.8.2" deleted
```

CHAPTER 6. ABOUT LOGGING

As a cluster administrator, you can deploy logging subsystem on an OpenShift Container Platform cluster, and use it to collect and aggregate node system audit logs, application container logs, and infrastructure logs. You can forward logs to your chosen log outputs, including on-cluster, Red Hat managed log storage. You can also visualize your log data in the OpenShift Container Platform web console, or [the Kibana web console](#), depending on your deployed log storage solution.



IMPORTANT

The Kibana web console is now deprecated and will be removed in a future logging release.

OpenShift Container Platform cluster administrators can deploy the logging subsystem by using Operators. For information, see [Installing the logging subsystem for Red Hat OpenShift](#).

The Operators are responsible for deploying, upgrading, and maintaining the logging subsystem. After the Operators are installed, you can create a **ClusterLogging** custom resource (CR) to schedule logging subsystem pods and other resources necessary to support the logging subsystem. You can also create a **ClusterLogForwarder** CR to specify which logs are collected, how they are transformed, and where they are forwarded to.



NOTE

Because the internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs, audit logs are not stored in the internal Elasticsearch instance by default. If you want to send the audit logs to the default internal Elasticsearch log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API as described in [Forward audit logs to the log store](#).

6.1. LOGGING ARCHITECTURE

The logging subsystem consists of these logical components:

- **Collector** - Reads container log data from each node and forwards log data to configured outputs.
- **Store** - Stores log data for analysis; the default output for the forwarder.
- **Visualization** - Graphical interface for searching, querying, and viewing stored logs.

These components are managed by Operators and Custom Resource (CR) YAML files.

The logging subsystem for Red Hat OpenShift collects container logs and node logs. These are categorized into types:

- **application** - Container logs generated by user applications running in the cluster, except infrastructure container applications.
- **infrastructure** - Logs generated by infrastructure components running in the cluster and OpenShift Container Platform nodes, such as journal logs. Infrastructure components are pods that run in the **openshift***, **kube***, or **default** projects.

- **audit** - Logs generated by auditd, the node audit system, which are stored in the `/var/log/audit/audit.log` file, and logs from the **auditd**, **kube-apiserver**, **openshift-apiserver** projects, as well as the **ovn** project if enabled.

The logging collector is a daemonset that deploys pods to each OpenShift Container Platform node. System and infrastructure logs are generated by journald log messages from the operating system, the container runtime, and OpenShift Container Platform.

Container logs are generated by containers running in pods running on the cluster. Each container generates a separate log stream. The collector collects the logs from these sources and forwards them internally or externally as configured in the **ClusterLogForwarder** custom resource.

6.2. ABOUT DEPLOYING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT

OpenShift Container Platform cluster administrators can deploy the logging subsystem using the OpenShift Container Platform web console or CLI to install the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator. When the Operators are installed, you create a **ClusterLogging** custom resource (CR) to schedule logging subsystem pods and other resources necessary to support the logging subsystem. The Operators are responsible for deploying, upgrading, and maintaining the logging subsystem.

The **ClusterLogging** CR defines a complete logging subsystem environment that includes all the components of the logging stack to collect, store and visualize logs. The Red Hat OpenShift Logging Operator watches the logging subsystem CR and adjusts the logging deployment accordingly.

Administrators and application developers can view the logs of the projects for which they have view access.

6.2.1. About JSON OpenShift Container Platform Logging

You can use JSON logging to configure the Log Forwarding API to parse JSON strings into a structured object. You can perform the following tasks:

- Parse JSON logs
- Configure JSON log data for Elasticsearch
- Forward JSON logs to the Elasticsearch log store

6.2.2. About collecting and storing Kubernetes events

The OpenShift Container Platform Event Router is a pod that watches Kubernetes events and logs them for collection by OpenShift Container Platform Logging. You must manually deploy the Event Router.

For information, see [About collecting and storing Kubernetes events](#).

6.2.3. About updating OpenShift Container Platform Logging

OpenShift Container Platform allows you to update OpenShift Container Platform logging. You must update the following operators while updating OpenShift Container Platform Logging:

- Elasticsearch Operator

- Cluster Logging Operator

For information, see [Updating OpenShift Logging](#).

6.2.4. About viewing the cluster dashboard

The OpenShift Container Platform Logging dashboard contains charts that show details about your Elasticsearch instance at the cluster level. These charts help you diagnose and anticipate problems.

For information, see [About viewing the cluster dashboard](#).

6.2.5. About troubleshooting OpenShift Container Platform Logging

You can troubleshoot the logging issues by performing the following tasks:

- Viewing logging status
- Viewing the status of the log store
- Understanding logging alerts
- Collecting logging data for Red Hat Support
- Troubleshooting for critical alerts

6.2.6. About uninstalling OpenShift Container Platform Logging

You can stop log aggregation by deleting the ClusterLogging custom resource (CR). After deleting the CR, there are other cluster logging components that remain, which you can optionally remove.

For information, see [Uninstalling OpenShift Logging](#).

6.2.7. About exporting fields

The logging system exports fields. Exported fields are present in the log records and are available for searching from Elasticsearch and Kibana.

For information, see [About exporting fields](#).

6.2.8. About logging subsystem components

The logging subsystem components include a collector deployed to each node in the OpenShift Container Platform cluster that collects all node and container logs and writes them to a log store. You can use a centralized web UI to create rich visualizations and dashboards with the aggregated data.

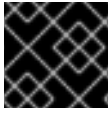
The major components of the logging subsystem are:

- collection - This is the component that collects logs from the cluster, formats them, and forwards them to the log store. The current implementation is Fluentd.
- log store - This is where the logs are stored. The default implementation is Elasticsearch. You can use the default Elasticsearch log store or forward logs to external log stores. The default log store is optimized and tested for short-term storage.

- visualization – This is the UI component you can use to view logs, graphs, charts, and so forth. The current implementation is Kibana.

6.2.9. About the logging collector

The Cluster Logging Operator deploys a collector based on the **ClusterLogForwarder** resource specification. There are two collector options supported by this Operator: the Fluentd collector, and the Vector collector.



IMPORTANT

The Fluentd collector is now deprecated and will be removed in a future logging release.

The logging subsystem for Red Hat OpenShift collects container and node logs.

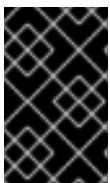
By default, the log collector uses the following sources:

- `journald` for all system logs
- `/var/log/containers/*.log` for all container logs

If you configure the log collector to collect audit logs, it gets them from `/var/log/audit/audit.log`.

The logging collector is a daemon set that deploys pods to each OpenShift Container Platform node. System and infrastructure logs are generated by `journald` log messages from the operating system, the container runtime, and OpenShift Container Platform. Application logs are generated by the CRI-O container engine. Fluentd collects the logs from these sources and forwards them internally or externally as you configure in OpenShift Container Platform.

The container runtimes provide minimal information to identify the source of log messages: project, pod name, and container ID. This information is not sufficient to uniquely identify the source of the logs. If a pod with a given name and project is deleted before the log collector begins processing its logs, information from the API server, such as labels and annotations, might not be available. There might not be a way to distinguish the log messages from a similarly named pod and project or trace the logs to their source. This limitation means that log collection and normalization are considered **best effort**.



IMPORTANT

The available container runtimes provide minimal information to identify the source of log messages and do not guarantee unique individual log messages or that these messages can be traced to their source.

For information, see [Configuring the logging collector](#).

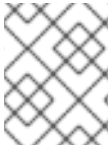
6.2.10. About the log store

By default, OpenShift Container Platform uses [Elasticsearch \(ES\)](#) to store log data. Optionally you can use the Log Forwarder API to forward logs to an external store. Several types of store are supported, including `fluentd`, `rsyslog`, `kafka` and others.

The logging subsystem Elasticsearch instance is optimized and tested for short term storage, approximately seven days. If you want to retain your logs over a longer term, it is recommended you move the data to a third-party storage system.

Elasticsearch organizes the log data from Fluentd into datastores, or *indices*, then subdivides each index

into multiple pieces called *shards*, which it spreads across a set of Elasticsearch nodes in an Elasticsearch cluster. You can configure Elasticsearch to make copies of the shards, called *replicas*, which Elasticsearch also spreads across the Elasticsearch nodes. The **ClusterLogging** custom resource (CR) allows you to specify how the shards are replicated to provide data redundancy and resilience to failure. You can also specify how long the different types of logs are retained using a retention policy in the **ClusterLogging** CR.

**NOTE**

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

The Red Hat OpenShift Logging Operator and companion OpenShift Elasticsearch Operator ensure that each Elasticsearch node is deployed using a unique deployment that includes its own storage volume. You can use a **ClusterLogging** custom resource (CR) to increase the number of Elasticsearch nodes, as needed. See the [Elasticsearch documentation](#) for considerations involved in configuring storage.

**NOTE**

A highly-available Elasticsearch environment requires at least three Elasticsearch nodes, each on a different host.

Role-based access control (RBAC) applied on the Elasticsearch indices enables the controlled access of the logs to the developers. Administrators can access all logs and developers can access only the logs in their projects.

For information, see [Configuring the log store](#).

6.2.11. About logging visualization

OpenShift Container Platform uses Kibana to display the log data collected by Fluentd and indexed by Elasticsearch.

Kibana is a browser-based console interface to query, discover, and visualize your Elasticsearch data through histograms, line graphs, pie charts, and other visualizations.

For information, see [Configuring the log visualizer](#).

6.2.12. About event routing

The Event Router is a pod that watches OpenShift Container Platform events so they can be collected by the logging subsystem for Red Hat OpenShift. The Event Router collects events from all projects and writes them to **STDOUT**. Fluentd collects those events and forwards them into the OpenShift Container Platform Elasticsearch instance. Elasticsearch indexes the events to the **infra** index.

You must manually deploy the Event Router.

For information, see [Collecting and storing Kubernetes events](#).

6.3. ABOUT VECTOR

Vector is a log collector offered as an alternative to Fluentd for the logging subsystem.

The following outputs are supported:

- **elasticsearch**. An external Elasticsearch instance. The **elasticsearch** output can use a TLS connection.
- **kafka**. A Kafka broker. The **kafka** output can use an unsecured or TLS connection.
- **loki**. Loki, a horizontally scalable, highly available, multitenant log aggregation system.

6.3.1. Enabling Vector

Vector is not enabled by default. Use the following steps to enable Vector on your OpenShift Container Platform cluster.

Prerequisites

- OpenShift Container Platform: 4.13
- Logging subsystem for Red Hat OpenShift: 5.4

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc -n openshift-logging edit ClusterLogging instance
```

2. Add a **logging.openshift.io/preview-vector-collector: enabled** annotation to the **ClusterLogging** custom resource (CR).
3. Add **vector** as a collection type to the **ClusterLogging** custom resource (CR).

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
  annotations:
    logging.openshift.io/preview-vector-collector: enabled
spec:
  collection:
    logs:
      type: "vector"
      vector: {}
```

Additional resources

- [Vector Documentation](#)

6.3.2. Collector features

Table 6.1. Log Sources

Feature	Fluentd	Vector
App container logs	✓	✓
App-specific routing	✓	✓
App-specific routing by namespace	✓	✓
Infra container logs	✓	✓
Infra journal logs	✓	✓
Kube API audit logs	✓	✓
OpenShift API audit logs	✓	✓
Open Virtual Network (OVN) audit logs	✓	✓

Table 6.2. Outputs

Feature	Fluentd	Vector
Elasticsearch v5-v7	✓	✓
Fluent forward	✓	
Syslog RFC3164	✓	✓ (Logging 5.7+)
Syslog RFC5424	✓	✓ (Logging 5.7+)
Kafka	✓	✓
Cloudwatch	✓	✓
Loki	✓	✓
HTTP	✓	✓ (Logging 5.7+)

Table 6.3. Authorization and Authentication

Feature	Fluentd	Vector
Elasticsearch certificates	✓	✓

Feature	Fluentd	Vector
Elasticsearch username / password	✓	✓
Cloudwatch keys	✓	✓
Cloudwatch STS	✓	✓
Kafka certificates	✓	✓
Kafka username / password	✓	✓
Kafka SASL	✓	✓
Loki bearer token	✓	✓

Table 6.4. Normalizations and Transformations

Feature	Fluentd	Vector
Viaq data model - app	✓	✓
Viaq data model - infra	✓	✓
Viaq data model - infra(journal)	✓	✓
Viaq data model - Linux audit	✓	✓
Viaq data model - kube-apiserver audit	✓	✓
Viaq data model - OpenShift API audit	✓	✓
Viaq data model - OVN	✓	✓
Loglevel Normalization	✓	✓
JSON parsing	✓	✓
Structured Index	✓	✓
Multiline error detection	✓	✓
Multicontainer / split indices	✓	✓

Feature	Fluentd	Vector
Flatten labels	✓	✓
CLF static labels	✓	✓

Table 6.5. Tuning

Feature	Fluentd	Vector
Fluentd readlinelimit	✓	
Fluentd buffer	✓	
- chunklimitsize	✓	
- totallimitsize	✓	
- overflowaction	✓	
- flushthreadcount	✓	
- flushmode	✓	
- flushinterval	✓	
- retrywait	✓	
- retrytype	✓	
- retrymaxinterval	✓	
- retrytimeout	✓	

Table 6.6. Visibility

Feature	Fluentd	Vector
Metrics	✓	✓
Dashboard	✓	✓
Alerts	✓	

Table 6.7. Miscellaneous

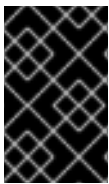
Feature	Fluentd	Vector
Global proxy support	✓	✓
x86 support	✓	✓
ARM support	✓	✓
IBM Power support	✓	✓
IBM Z support	✓	✓
IPv6 support	✓	✓
Log event buffering	✓	
Disconnected Cluster	✓	✓

CHAPTER 7. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT

You can install the logging subsystem for Red Hat OpenShift by deploying the OpenShift Elasticsearch and Red Hat OpenShift Logging Operators. The OpenShift Elasticsearch Operator creates and manages the Elasticsearch cluster used by OpenShift Logging. The logging subsystem Operator creates and manages the components of the logging stack.

The process for deploying the logging subsystem to OpenShift Container Platform involves:

- Reviewing the [Logging subsystem storage considerations](#).
- Installing the logging subsystem for OpenShift Container Platform using [the web console](#) or [the CLI](#).



IMPORTANT

For new installations, Vector and LokiStack are recommended. Documentation for logging is in the process of being updated to reflect these underlying component changes.



NOTE

As of logging version 5.4.3 the OpenShift Elasticsearch Operator is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to using the OpenShift Elasticsearch Operator to manage the default log storage, you can use the Loki Operator.



NOTE

As of logging version 5.6 Fluentd is deprecated and is planned to be removed in a future release. Red Hat will provide bug fixes and support for this feature during the current release lifecycle, but this feature will no longer receive enhancements and will be removed. As an alternative to Fluentd, you can use Vector instead.

7.1. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT USING THE WEB CONSOLE

You can use the OpenShift Container Platform web console to install the OpenShift Elasticsearch and Red Hat OpenShift Logging Operators.



NOTE

If you do not want to use the default Elasticsearch log store, you can remove the internal Elasticsearch **logStore** and Kibana **visualization** components from the **ClusterLogging** custom resource (CR). Removing these components is optional but saves resources. For more information, see the additional resources of this section.

Prerequisites

- Ensure that you have the necessary persistent storage for Elasticsearch. Note that each Elasticsearch node requires its own storage volume.



NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

Elasticsearch is a memory-intensive application. By default, OpenShift Container Platform installs three Elasticsearch nodes with memory requests and limits of 16 GB. This initial set of three OpenShift Container Platform nodes might not have enough memory to run Elasticsearch within your cluster. If you experience memory issues that are related to Elasticsearch, add more Elasticsearch nodes to your cluster rather than increasing the memory on existing nodes.

Procedure

To install the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator by using the OpenShift Container Platform web console:

1. Install the OpenShift Elasticsearch Operator:
 - a. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
 - b. Choose **OpenShift Elasticsearch Operator** from the list of available Operators, and click **Install**.
 - c. Ensure that the **All namespaces on the cluster** is selected under **Installation Mode**.
 - d. Ensure that **openshift-operators-redhat** is selected under **Installed Namespace**.
You must specify the **openshift-operators-redhat** namespace. The **openshift-operators** namespace might contain Community Operators, which are untrusted and could publish a metric with the same name as metric, which would cause conflicts.
 - e. Select **Enable operator recommended cluster monitoring on this namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.
 - f. Select **stable-5.x** as the **Update Channel**.
 - g. Select an **Approval Strategy**.
 - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.
 - h. Click **Install**.
 - i. Verify that the OpenShift Elasticsearch Operator installed by switching to the **Operators → Installed Operators** page.
 - j. Ensure that **OpenShift Elasticsearch Operator** is listed in all projects with a **Status** of **Succeeded**.

2. Install the Red Hat OpenShift Logging Operator:

- a. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
- b. Choose **Red Hat OpenShift Logging** from the list of available Operators, and click **Install**.
- c. Ensure that **A specific namespace on the cluster** is selected under **Installation Mode**.
- d. Ensure that **Operator recommended namespace** is **openshift-logging** under **Installed Namespace**.
- e. Select **Enable operator recommended cluster monitoring on this namespace**
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-logging** namespace.
- f. Select **stable-5.x** as the **Update Channel**.
- g. Select an **Approval Strategy**.
 - The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.
- h. Click **Install**.
- i. Verify that the Red Hat OpenShift Logging Operator installed by switching to the **Operators → Installed Operators** page.
- j. Ensure that **Red Hat OpenShift Logging** is listed in the **openshift-logging** project with a **Status** of **Succeeded**.
If the Operator does not appear as installed, to troubleshoot further:
 - Switch to the **Operators → Installed Operators** page and inspect the **Status** column for any errors or failures.
 - Switch to the **Workloads → Pods** page and check the logs in any pods in the **openshift-logging** project that are reporting issues.

3. Create an OpenShift Logging instance:

- a. Switch to the **Administration → Custom Resource Definitions** page.
- b. On the **Custom Resource Definitions** page, click **ClusterLogging**.
- c. On the **Custom Resource Definition details** page, select **View Instances** from the **Actions** menu.
- d. On the **ClusterLoggings** page, click **Create ClusterLogging**.
You might have to refresh the page to load the data.
- e. In the YAML field, replace the code with the following:



NOTE

This default OpenShift Logging configuration should support a wide array of environments. Review the topics on tuning and configuring logging subsystem components for information on modifications you can make to your OpenShift Logging cluster.

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" ❶
  namespace: "openshift-logging"
spec:
  managementState: "Managed" ❷
  logStore:
    type: "elasticsearch" ❸
    retentionPolicy: ❹
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3 ❺
      storage:
        storageClassName: "<storage_class_name>" ❻
        size: 200G
      resources: ❼
        limits:
          memory: "16Gi"
        requests:
          memory: "16Gi"
      proxy: ❸
        resources:
          limits:
            memory: 256Mi
          requests:
            memory: 256Mi
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" ❾
    kibana:
      replicas: 1
  collection:
    logs:
      type: "fluentd" ❿
      fluentd: {}

```

- ❶ The name must be **instance**.
- ❷ The OpenShift Logging management state. In some cases, if you change the OpenShift Logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until OpenShift Logging is placed

back into a managed state.

- 3 Settings for configuring Elasticsearch. Using the CR, you can configure shard replication policy and persistent storage.
- 4 Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **7d** for seven days. Logs older than the **maxAge** are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.
- 5 Specify the number of Elasticsearch nodes. See the note that follows this list.
- 6 Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Logging uses ephemeral storage.
- 7 Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **16Gi** for the memory request and **1** for the CPU request.
- 8 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.
- 9 Settings for configuring Kibana. Using the CR, you can scale Kibana for redundancy and configure the CPU and memory for your Kibana nodes. For more information, see **Configuring the log visualizer**.
- 10 Settings for configuring Fluentd. Using the CR, you can configure Fluentd CPU and memory limits. For more information, see **Configuring Fluentd**.

NOTE

The maximum number of Elasticsearch control plane nodes is three. If you specify a **nodeCount** greater than **3**, OpenShift Container Platform creates three Elasticsearch nodes that are Master-eligible nodes, with the master, client, and data roles. The additional Elasticsearch nodes are created as Data-only nodes, using client and data roles. Control plane nodes perform cluster-wide actions such as creating or deleting an index, shard allocation, and tracking nodes. Data nodes hold the shards and perform data-related operations such as CRUD, search, and aggregations. Data-related operations are I/O-, memory-, and CPU-intensive. It is important to monitor these resources and to add more Data nodes if the current nodes are overloaded.

For example, if **nodeCount=4**, the following nodes are created:

```
$ oc get deployment
```

Example output

```
cluster-logging-operator    1/1    1        1        18h
elasticsearch-cd-x6kdekli-1 0/1    1        0        6m54s
elasticsearch-cdm-x6kdekli-1 1/1    1        1        18h
elasticsearch-cdm-x6kdekli-2 0/1    1        0        6m49s
elasticsearch-cdm-x6kdekli-3 0/1    1        0        6m44s
```

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

- f. Click **Create**. This creates the logging subsystem components, the **Elasticsearch** custom resource and components, and the Kibana interface.
4. Verify the install:
 - a. Switch to the **Workloads → Pods** page.
 - b. Select the **openshift-logging** project.
You should see several pods for OpenShift Logging, Elasticsearch, Fluentd, and Kibana similar to the following list:
 - cluster-logging-operator-cb795f8dc-xkckc
 - collector-pb2f8
 - elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz
 - elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv
 - elasticsearch-cdm-b3nqzchd-3-588c65-clg7g
 - fluentd-2c7dg
 - fluentd-9z7kk
 - fluentd-br7r2
 - fluentd-fn2sb

- fluentd-zqgqx
- kibana-7fb4fd4cc9-bvt4p

Troubleshooting

- If Alertmanager logs alerts such as **Prometheus could not scrape fluentd for more than 10m**, make sure that **openshift.io/cluster-monitoring** is set to **"true"** for the OpenShift Elasticsearch Operator and OpenShift Logging Operator. See the Red Hat KnowledgeBase for more information: [Prometheus could not scrape fluentd for more than 10m alert in Alertmanager](#)

Additional resources

- [Installing Operators from the OperatorHub](#)
- [Removing unused components if you do not use the default Elasticsearch log store](#)

7.2. POST-INSTALLATION TASKS

If you plan to use Kibana, you must [manually create your Kibana index patterns and visualizations](#) to explore and visualize data in Kibana.

If your network plugin enforces network isolation, [allow network traffic between the projects that contain the logging subsystem Operators](#).

7.3. INSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT USING THE CLI

You can use the OpenShift Container Platform CLI to install the OpenShift Elasticsearch and Red Hat OpenShift Logging Operators.

Prerequisites

- Ensure that you have the necessary persistent storage for Elasticsearch. Note that each Elasticsearch node requires its own storage volume.



NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

Elasticsearch is a memory-intensive application. By default, OpenShift Container Platform installs three Elasticsearch nodes with memory requests and limits of 16 GB. This initial set of three OpenShift Container Platform nodes might not have enough memory to run Elasticsearch within your cluster. If you experience memory issues that are related to Elasticsearch, add more Elasticsearch nodes to your cluster rather than increasing the memory on existing nodes.

Procedure

To install the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator using the CLI:

1. Create a namespace for the OpenShift Elasticsearch Operator.
 - a. Create a namespace object YAML file (for example, **eo-namespace.yaml**) for the OpenShift Elasticsearch Operator:

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-operators-redhat 1
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true" 2
```

- 1** You must specify the **openshift-operators-redhat** namespace. To prevent possible conflicts with metrics, you should configure the Prometheus Cluster Monitoring stack to scrape metrics from the **openshift-operators-redhat** namespace and not the **openshift-operators** namespace. The **openshift-operators** namespace might contain community Operators, which are untrusted and could publish a metric with the same name as metric, which would cause conflicts.
- 2** String. You must specify this label as shown to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.

- b. Create the namespace:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f eo-namespace.yaml
```

2. Create a namespace for the Red Hat OpenShift Logging Operator:
 - a. Create a namespace object YAML file (for example, **olo-namespace.yaml**) for the Red Hat OpenShift Logging Operator:

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-monitoring: "true"
```

- b. Create the namespace:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f olo-namespace.yaml
```

3. Install the OpenShift Elasticsearch Operator by creating the following objects:

- a. Create an Operator Group object YAML file (for example, **eo-og.yaml**) for the OpenShift Elasticsearch Operator:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-operators-redhat
  namespace: openshift-operators-redhat 1
spec: {}
```

- 1** You must specify the **openshift-operators-redhat** namespace.

- b. Create an Operator Group object:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f eo-og.yaml
```

- c. Create a Subscription object YAML file (for example, **eo-sub.yaml**) to subscribe a namespace to the OpenShift Elasticsearch Operator.

Example Subscription

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: "elasticsearch-operator"
  namespace: "openshift-operators-redhat" 1
spec:
  channel: "stable-5.5" 2
  installPlanApproval: "Automatic" 3
  source: "redhat-operators" 4
  sourceNamespace: "openshift-marketplace"
  name: "elasticsearch-operator"
```

- 1** You must specify the **openshift-operators-redhat** namespace.
- 2** Specify **stable**, or **stable-5.<x>** as the channel. See the following note.
- 3** **Automatic** allows the Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available. **Manual** requires a user with appropriate credentials to approve the Operator update.
- 4** Specify **redhat-operators**. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the CatalogSource object created when you configured the Operator Lifecycle Manager (OLM).

**NOTE**

Specifying **stable** installs the current version of the latest stable release. Using **stable** with **installPlanApproval: "Automatic"**, will automatically upgrade your operators to the latest stable major and minor release.

Specifying **stable-5.<x>** installs the current minor version of a specific major release. Using **stable-5.<x>** with **installPlanApproval: "Automatic"**, will automatically upgrade your operators to the latest stable minor release within the major release you specify with **x**.

- d. Create the Subscription object:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f eo-sub.yaml
```

The OpenShift Elasticsearch Operator is installed to the **openshift-operators-redhat** namespace and copied to each project in the cluster.

- e. Verify the Operator installation:

```
$ oc get csv --all-namespaces
```

Example output

NAMESPACE	VERSION	REPLACES	PHASE	NAME	DISPLAY
default				elasticsearch-operator.5.1.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
kube-node-lease				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
kube-public				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
kube-system				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
openshift-apiserver-operator				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
openshift-apiserver				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
openshift-authentication-operator				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
openshift-authentication				elasticsearch-operator.5.5.0-202007012112.p0	
OpenShift Elasticsearch Operator	5.5.0-202007012112.p0		Succeeded		
...					

There should be an OpenShift Elasticsearch Operator in each namespace. The version number might be different than shown.

4. Install the Red Hat OpenShift Logging Operator by creating the following objects:

- a. Create an Operator Group object YAML file (for example, **olo-og.yaml**) for the Red Hat OpenShift Logging Operator:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: cluster-logging
  namespace: openshift-logging ❶
spec:
  targetNamespaces:
    - openshift-logging ❷
```

❶ ❷ You must specify the **openshift-logging** namespace.

- b. Create the OperatorGroup object:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f olo-og.yaml
```

- c. Create a Subscription object YAML file (for example, **olo-sub.yaml**) to subscribe a namespace to the Red Hat OpenShift Logging Operator.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: cluster-logging
  namespace: openshift-logging ❶
spec:
  channel: "stable" ❷
  name: cluster-logging
  source: redhat-operators ❸
  sourceNamespace: openshift-marketplace
```

❶ You must specify the **openshift-logging** namespace.

❷ Specify **stable**, or **stable-5.<x>** as the channel.

❸ Specify **redhat-operators**. If your OpenShift Container Platform cluster is installed on a restricted network, also known as a disconnected cluster, specify the name of the CatalogSource object you created when you configured the Operator Lifecycle Manager (OLM).

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f olo-sub.yaml
```

The Red Hat OpenShift Logging Operator is installed to the **openshift-logging** namespace.

- d. Verify the Operator installation.

There should be a Red Hat OpenShift Logging Operator in the **openshift-logging** namespace. The Version number might be different than shown.

```
$ oc get csv -n openshift-logging
```

Example output

NAMESPACE	NAME	DISPLAY
VERSION	REPLACES	PHASE
...		
openshift-logging	clusterlogging.5.1.0-202007012112.p0	
OpenShift Logging	5.1.0-202007012112.p0	Succeeded
...		

5. Create an OpenShift Logging instance:

- a. Create an instance object YAML file (for example, **olo-instance.yaml**) for the Red Hat OpenShift Logging Operator:



NOTE

This default OpenShift Logging configuration should support a wide array of environments. Review the topics on tuning and configuring logging subsystem components for information about modifications you can make to your OpenShift Logging cluster.

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" 1
  namespace: "openshift-logging"
spec:
  managementState: "Managed" 2
  logStore:
    type: "elasticsearch" 3
    retentionPolicy: 4
    application:
      maxAge: 1d
    infra:
      maxAge: 7d
    audit:
      maxAge: 7d
    elasticsearch:
      nodeCount: 3 5
      storage:
        storageClassName: "<storage-class-name>" 6
        size: 200G
      resources: 7
        limits:
          memory: "16Gi"
```

```

    requests:
      memory: "16Gi"
    proxy: 8
    resources:
      limits:
        memory: 256Mi
      requests:
        memory: 256Mi
    redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana" 9
    kibana:
      replicas: 1
  collection:
    logs:
      type: "fluentd" 10
      fluentd: {}

```

- 1 The name must be **instance**.
- 2 The OpenShift Logging management state. In some cases, if you change the OpenShift Logging defaults, you must set this to **Unmanaged**. However, an unmanaged deployment does not receive updates until OpenShift Logging is placed back into a managed state. Placing a deployment back into a managed state might revert any modifications you made.
- 3 Settings for configuring Elasticsearch. Using the custom resource (CR), you can configure shard replication policy and persistent storage.
- 4 Specify the length of time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **7d** for seven days. Logs older than the **maxAge** are deleted. You must specify a retention policy for each log source or the Elasticsearch indices will not be created for that source.
- 5 Specify the number of Elasticsearch nodes. See the note that follows this list.
- 6 Enter the name of an existing storage class for Elasticsearch storage. For best performance, specify a storage class that allocates block storage. If you do not specify a storage class, OpenShift Container Platform deploys OpenShift Logging with ephemeral storage only.
- 7 Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that are sufficient for most deployments. The default values are **16Gi** for the memory request and **1** for the CPU request.
- 8 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.
- 9 Settings for configuring Kibana. Using the CR, you can scale Kibana for redundancy and configure the CPU and memory for your Kibana pods. For more information, see **Configuring the log visualizer**.

- 10** Settings for configuring Fluentd. Using the CR, you can configure Fluentd CPU and memory limits. For more information, see **Configuring Fluentd**.

NOTE

The maximum number of Elasticsearch control plane nodes is three. If you specify a **nodeCount** greater than **3**, OpenShift Container Platform creates three Elasticsearch nodes that are Master-eligible nodes, with the master, client, and data roles. The additional Elasticsearch nodes are created as Data-only nodes, using client and data roles. Control plane nodes perform cluster-wide actions such as creating or deleting an index, shard allocation, and tracking nodes. Data nodes hold the shards and perform data-related operations such as CRUD, search, and aggregations. Data-related operations are I/O-, memory-, and CPU-intensive. It is important to monitor these resources and to add more Data nodes if the current nodes are overloaded.

For example, if **nodeCount=4**, the following nodes are created:

```
$ oc get deployment
```

Example output

```
cluster-logging-operator      1/1      1          1      18h
elasticsearch-cd-x6kdekli-1   1/1      1          0      6m54s
elasticsearch-cdm-x6kdekli-1  1/1      1          1      18h
elasticsearch-cdm-x6kdekli-2  1/1      1          0      6m49s
elasticsearch-cdm-x6kdekli-3  1/1      1          0      6m44s
```

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

- b. Create the instance:

```
$ oc create -f <file-name>.yaml
```

For example:

```
$ oc create -f olo-instance.yaml
```

This creates the logging subsystem components, the **Elasticsearch** custom resource and components, and the Kibana interface.

6. Verify the installation by listing the pods in the **openshift-logging** project. You should see several pods for components of the Logging subsystem, similar to the following list:

```
$ oc get pods -n openshift-logging
```

Example output

```
NAME                                READY STATUS RESTARTS AGE
cluster-logging-operator-66f77fccb-ppzbg  1/1 Running 0      7m
```

elasticsearch-cdm-ftuhduuw-1-ffc4b9566-q6bhp	2/2	Running	0	2m40s
elasticsearch-cdm-ftuhduuw-2-7b4994dbfc-rd2gc	2/2	Running	0	2m36s
elasticsearch-cdm-ftuhduuw-3-84b5ff7ff8-gqnm2	2/2	Running	0	2m4s
collector-587vb	1/1	Running	0	2m26s
collector-7mpb9	1/1	Running	0	2m30s
collector-flm6j	1/1	Running	0	2m33s
collector-gn4rn	1/1	Running	0	2m26s
collector-nlgb6	1/1	Running	0	2m30s
collector-snpkt	1/1	Running	0	2m28s
kibana-d6d5668c5-rppqm	2/2	Running	0	2m39s

7.4. POST-INSTALLATION TASKS

If you plan to use Kibana, you must [manually create your Kibana index patterns and visualizations](#) to explore and visualize data in Kibana.

If your network plugin enforces network isolation, [allow network traffic between the projects that contain the logging subsystem Operators](#).

7.4.1. Defining Kibana index patterns

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

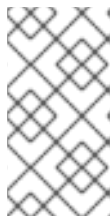
- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.

If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

```
$ oc auth can-i get pods --subresource log -n <project>
```

Example output

```
yes
```




NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher  and select **Logging**.
2. Create your Kibana index patterns by clicking **Management** → **Index Patterns** → **Create index pattern**:
 - Each user must manually create index patterns when logging into Kibana the first time to see logs for their projects. Users must create an index pattern named **app** and use the **@timestamp** time field to view their container logs.
 - Each admin user must create index patterns when logged into Kibana the first time for the **app**, **infra**, and **audit** indices using the **@timestamp** time field.
3. Create Kibana Visualizations from the new index patterns.

7.4.2. Allowing traffic between projects when network isolation is enabled

Your cluster network plugin might enforce network isolation. If so, you must allow network traffic between the projects that contain the operators deployed by OpenShift Logging.

Network isolation blocks network traffic between pods or services that are in different projects. The logging subsystem installs the *OpenShift Elasticsearch Operator* in the **openshift-operators-redhat** project and the *Red Hat OpenShift Logging Operator* in the **openshift-logging** project. Therefore, you must allow traffic between these two projects.

OpenShift Container Platform offers two supported choices for the network plugin, OpenShift SDN and OVN-Kubernetes. These two providers implement various network isolation policies.

OpenShift SDN has three modes:

network policy

This is the default mode. If no policy is defined, it allows all traffic. However, if a user defines a policy, they typically start by denying all traffic and then adding exceptions. This process might break applications that are running in different projects. Therefore, explicitly configure the policy to allow traffic to egress from one logging-related project to the other.

multitenant

This mode enforces network isolation. You must join the two logging-related projects to allow traffic between them.

subnet

This mode allows all traffic. It does not enforce network isolation. No action is needed.

OVN-Kubernetes always uses a **network policy**. Therefore, as with OpenShift SDN, you must configure the policy to allow traffic to egress from one logging-related project to the other.

Procedure

- If you are using OpenShift SDN in **multitenant** mode, join the two projects. For example:

```
$ oc adm pod-network join-projects --to=openshift-operators-redhat openshift-logging
```

- Otherwise, for OpenShift SDN in **network policy** mode and OVN-Kubernetes, perform the following actions:
 - a. Set a label on the **openshift-operators-redhat** namespace. For example:

```
$ oc label namespace openshift-operators-redhat project=openshift-operators-redhat
```

- b. Create a network policy object in the **openshift-logging** namespace that allows ingress from the **openshift-operators-redhat**, **openshift-monitoring** and **openshift-ingress** projects to the openshift-logging project. For example:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-monitoring-ingress-operators-redhat
spec:
  ingress:
    - from:
      - podSelector: {}
    - from:
      - namespaceSelector:
          matchLabels:
            project: "openshift-operators-redhat"
    - from:
      - namespaceSelector:
          matchLabels:
            name: "openshift-monitoring"
    - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: ingress
  podSelector: {}
  policyTypes:
    - Ingress
```

Additional resources

- [About network policy](#)
- [About the OpenShift SDN default CNI network provider](#)
- [About the OVN-Kubernetes default Container Network Interface \(CNI\) network provider](#)

CHAPTER 8. CONFIGURING YOUR LOGGING DEPLOYMENT

8.1. ABOUT THE CLUSTER LOGGING CUSTOM RESOURCE

To configure logging subsystem for Red Hat OpenShift you customize the **ClusterLogging** custom resource (CR).

8.1.1. About the ClusterLogging custom resource

To make changes to your logging subsystem environment, create and modify the **ClusterLogging** custom resource (CR).

Instructions for creating or modifying a CR are provided in this documentation as appropriate.

The following example shows a typical custom resource for the logging subsystem.

Sample ClusterLogging custom resource (CR)

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance" ❶
  namespace: "openshift-logging" ❷
spec:
  managementState: "Managed" ❸
  logStore:
    type: "elasticsearch" ❹
    retentionPolicy:
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          memory: 16Gi
        requests:
          cpu: 500m
          memory: 16Gi
      storage:
        storageClassName: "gp2"
        size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization: ❺
    type: "kibana"
    kibana:
      resources:
        limits:
          memory: 736Mi
        requests:
          cpu: 100m
```

```

    memory: 736Mi
    replicas: 1
collection: 6
logs:
  type: "fluentd"
  fluentd:
    resources:
      limits:
        memory: 736Mi
      requests:
        cpu: 100m
        memory: 736Mi

```

- 1 The CR name must be **instance**.
- 2 The CR must be installed to the **openshift-logging** namespace.
- 3 The Red Hat OpenShift Logging Operator management state. When set to **unmanaged** the operator is in an unsupported state and will not get updates.
- 4 Settings for the log store, including retention policy, the number of nodes, the resource requests and limits, and the storage class.
- 5 Settings for the visualizer, including the resource requests and limits, and the number of pod replicas.
- 6 Settings for the log collector, including the resource requests and limits.

8.2. CONFIGURING THE LOG STORE

Logging subsystem for Red Hat OpenShift uses Elasticsearch 6 (ES) to store and organize the log data.

You can make modifications to your log store, including:

- storage for your Elasticsearch cluster
- shard replication across data nodes in the cluster, from full replication to no replication
- external access to Elasticsearch data

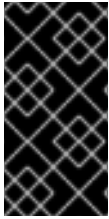
Elasticsearch is a memory-intensive application. Each Elasticsearch node needs at least 16G of memory for both memory requests and limits, unless you specify otherwise in the **ClusterLogging** custom resource. The initial set of OpenShift Container Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory, up to a maximum of 64G for each Elasticsearch node.

Each Elasticsearch node can operate with a lower memory setting, though this is not recommended for production environments.

8.2.1. Forwarding audit logs to the log store

By default, OpenShift Logging does not store audit logs in the internal OpenShift Container Platform Elasticsearch log store. You can send audit logs to this log store so, for example, you can view them in Kibana.

To send the audit logs to the default internal Elasticsearch log store, for example to view the audit logs in Kibana, you must use the Log Forwarding API.



IMPORTANT

The internal OpenShift Container Platform Elasticsearch log store does not provide secure storage for audit logs. Verify that the system to which you forward audit logs complies with your organizational and governmental regulations and is properly secured. The logging subsystem for Red Hat OpenShift does not comply with those regulations.

Procedure

To use the Log Forward API to forward audit logs to the internal Elasticsearch instance:

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:
 - Create a CR to send all log types to the internal Elasticsearch instance. You can use the following example without making any changes:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines: 1
    - name: all-to-default
      inputRefs:
        - infrastructure
        - application
        - audit
      outputRefs:
        - default
```

- 1** A pipeline defines the type of logs to forward using the specified output. The default output forwards logs to the internal Elasticsearch instance.



NOTE

You must specify all three types of logs in the pipeline: application, infrastructure, and audit. If you do not specify a log type, those logs are not stored and will be lost.

- If you have an existing **ClusterLogForwarder** CR, add a pipeline to the default output for the audit logs. You do not need to define the default output. For example:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
    - name: elasticsearch-insecure
```

```

type: "elasticsearch"
url: http://elasticsearch-insecure.messaging.svc.cluster.local
insecure: true
- name: elasticsearch-secure
type: "elasticsearch"
url: https://elasticsearch-secure.messaging.svc.cluster.local
secret:
  name: es-audit
- name: secureforward-offcluster
type: "fluentdForward"
url: https://secureforward.offcluster.com:24224
secret:
  name: secureforward
pipelines:
- name: container-logs
inputRefs:
- application
outputRefs:
- secureforward-offcluster
- name: infra-logs
inputRefs:
- infrastructure
outputRefs:
- elasticsearch-insecure
- name: audit-logs
inputRefs:
- audit
outputRefs:
- elasticsearch-secure
- default 1

```

- 1 This pipeline sends the audit logs to the internal Elasticsearch instance in addition to an external instance.

Additional resources

- For more information on the Log Forwarding API, see [Forwarding logs using the Log Forwarding API](#).

8.2.2. Configuring log retention time

You can configure a *retention policy* that specifies how long the default Elasticsearch log store keeps indices for each of the three log sources: infrastructure logs, application logs, and audit logs.

To configure the retention policy, you set a **maxAge** parameter for each log source in the **ClusterLogging** custom resource (CR). The CR applies these values to the Elasticsearch rollover schedule, which determines when Elasticsearch deletes the rolled-over indices.

Elasticsearch rolls over an index, moving the current index and creating a new index, when an index matches any of the following conditions:

- The index is older than the **rollover.maxAge** value in the **Elasticsearch** CR.
- The index size is greater than 40 GB × the number of primary shards.

- The index doc count is greater than 40960 KB × the number of primary shards.

Elasticsearch deletes the rolled-over indices based on the retention policy you configure. If you do not create a retention policy for any log sources, logs are deleted after seven days by default.

Prerequisites

- The logging subsystem for Red Hat OpenShift and the OpenShift Elasticsearch Operator must be installed.

Procedure

To configure the log retention time:

1. Edit the **ClusterLogging** CR to add or modify the **retentionPolicy** parameter:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    retentionPolicy: 1
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
  ...
```

- 1 Specify the time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, **1d** for one day. Logs older than the **maxAge** are deleted. By default, logs are retained for seven days.

2. You can verify the settings in the **Elasticsearch** custom resource (CR).
For example, the Red Hat OpenShift Logging Operator updated the following **Elasticsearch** CR to configure a retention policy that includes settings to roll over active indices for the infrastructure logs every eight hours and the rolled-over indices are deleted seven days after rollover. OpenShift Container Platform checks every 15 minutes to determine if the indices need to be rolled over.

```
apiVersion: "logging.openshift.io/v1"
kind: "Elasticsearch"
metadata:
  name: "elasticsearch"
spec:
  ...
  indexManagement:
    policies: 1
      - name: infra-policy
```

```

phases:
  delete:
    minAge: 7d 2
  hot:
    actions:
      rollover:
        maxAge: 8h 3
    pollInterval: 15m 4
...

```

- 1 For each log source, the retention policy indicates when to delete and roll over logs for that source.
- 2 When OpenShift Container Platform deletes the rolled-over indices. This setting is the **maxAge** you set in the **ClusterLogging** CR.
- 3 The index age for OpenShift Container Platform to consider when rolling over the indices. This value is determined from the **maxAge** you set in the **ClusterLogging** CR.
- 4 When OpenShift Container Platform checks if the indices should be rolled over. This setting is the default and cannot be changed.



NOTE

Modifying the **Elasticsearch** CR is not supported. All changes to the retention policies must be made in the **ClusterLogging** CR.

The OpenShift Elasticsearch Operator deploys a cron job to roll over indices for each mapping using the defined policy, scheduled using the **pollInterval**.

```
$ oc get cronjob
```

Example output

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
elasticsearch-im-app	*/15 * * * *	False	0	<none>	4s
elasticsearch-im-audit	*/15 * * * *	False	0	<none>	4s
elasticsearch-im-infra	*/15 * * * *	False	0	<none>	4s

8.2.3. Configuring CPU and memory requests for the log store

Each component specification allows for adjustments to both the CPU and memory requests. You should not have to manually adjust these values as the OpenShift Elasticsearch Operator sets values sufficient for your environment.



NOTE

In large-scale clusters, the default memory limit for the Elasticsearch proxy container might not be sufficient, causing the proxy container to be OOMKilled. If you experience this issue, increase the memory requests and limits for the Elasticsearch proxy.

Each Elasticsearch node can operate with a lower memory setting though this is **not** recommended for production deployments. For production use, you should have no less than the default 16Gi allocated to each pod. Preferably you should allocate as much as possible, up to 64Gi per pod.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch: 1
    resources:
      limits: 2
      memory: "32Gi"
      requests: 3
      cpu: "1"
      memory: "16Gi"
    proxy: 4
    resources:
      limits:
        memory: 100Mi
      requests:
        memory: 100Mi
```

- 1 Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are **16Gi** for the memory request and **1** for the CPU request.
- 2 The maximum amount of resources a pod can use.
- 3 The minimum resources required to schedule a pod.
- 4 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that are sufficient for most deployments. The default values are **256Mi** for the memory request and **100m** for the CPU request.

When adjusting the amount of Elasticsearch memory, the same value should be used for both **requests** and **limits**.

For example:

```
resources:
  limits: ❶
    memory: "32Gi"
  requests: ❷
    cpu: "8"
    memory: "32Gi"
```

- ❶ The maximum amount of the resource.
- ❷ The minimum amount required.

Kubernetes generally adheres the node configuration and does not allow Elasticsearch to use the specified limits. Setting the same value for the **requests** and **limits** ensures that Elasticsearch can use the memory you want, assuming the node has the memory available.

8.2.4. Configuring replication policy for the log store

You can define how Elasticsearch shards are replicated across data nodes in the cluster.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

- Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit clusterlogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy" ❶
```

- ❶ Specify a redundancy policy for the shards. The change is applied upon saving the changes.
 - FullRedundancy.** Elasticsearch fully replicates the primary shards for each index to every data node. This provides the highest safety, but at the cost of the highest amount of disk required and the poorest performance.
 - MultipleRedundancy.** Elasticsearch fully replicates the primary shards for each index to half of the data nodes. This provides a good tradeoff between safety and performance.

- **SingleRedundancy.** Elasticsearch makes one copy of the primary shards for each index. Logs are always available and recoverable as long as at least two data nodes exist. Better performance than MultipleRedundancy, when using 5 or more nodes. You cannot apply this policy on deployments of single Elasticsearch node.
- **ZeroRedundancy.** Elasticsearch does not make copies of the primary shards. Logs might be unavailable or lost in the event a node is down or fails. Use this mode when you are more concerned with performance than safety, or have implemented your own disk/PVC backup/restore strategy.



NOTE

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

8.2.5. Scaling down Elasticsearch pods

Reducing the number of Elasticsearch pods in your cluster can result in data loss or Elasticsearch performance degradation.

If you scale down, you should scale down by one pod at a time and allow the cluster to re-balance the shards and replicas. After the Elasticsearch health status returns to **green**, you can scale down by another pod.



NOTE

If your Elasticsearch cluster is set to **ZeroRedundancy**, you should not scale down your Elasticsearch pods.

8.2.6. Configuring persistent storage for the log store

Elasticsearch requires persistent storage. The faster the storage, the faster the Elasticsearch performance.



WARNING

Using NFS storage as a volume or a persistent volume (or via NAS such as Gluster) is not supported for Elasticsearch storage, as Lucene relies on file system behavior that NFS does not supply. Data corruption and other problems can occur.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Edit the **ClusterLogging** CR to specify that each data node in the cluster is bound to a Persistent Volume Claim.

■

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
# ...
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: "gp2"
        size: "200G"

```

This example specifies each data node in the cluster is bound to a Persistent Volume Claim that requests "200G" of AWS General Purpose SSD (gp2) storage.



NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

8.2.7. Configuring the log store for emptyDir storage

You can use emptyDir with your log store, which creates an ephemeral deployment in which all of a pod's data is lost upon restart.



NOTE

When using emptyDir, if log storage is restarted or redeployed, you will lose data.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Edit the **ClusterLogging** CR to specify emptyDir:

```

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage: {}

```

8.2.8. Performing an Elasticsearch rolling cluster restart

Perform a rolling restart when you change the **elasticsearch** config map or any of the **elasticsearch-*** deployment configurations.

Also, a rolling restart is recommended if the nodes on which an Elasticsearch pod runs requires a reboot.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

To perform a rolling cluster restart:

1. Change to the **openshift-logging** project:

```
$ oc project openshift-logging
```

2. Get the names of the Elasticsearch pods:

```
$ oc get pods -l component=elasticsearch-
```

3. Scale down the collector pods so they stop sending new logs to Elasticsearch:

```
$ oc -n openshift-logging patch daemonset/collector -p '{"spec":{"template":{"spec":{"nodeSelector":{"logging-infra-collector": "false"}}}}}'
```

4. Perform a shard synced flush using the OpenShift Container Platform [es_util](#) tool to ensure there are no pending operations waiting to be written to disk prior to shutting down:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_flush/synced" -XPOST
```

For example:

```
$ oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_flush/synced" -XPOST
```

Example output

```
{"_shards":{"total":4,"successful":4,"failed":0},".security":{"total":2,"successful":2,"failed":0},".kibana_1":{"total":2,"successful":2,"failed":0}}
```

5. Prevent shard balancing when purposely bringing down nodes using the OpenShift Container Platform [es_util](#) tool:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{"persistent": { "cluster.routing.allocation.enable" : "primaries" } }'
```

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{"persistent": { "cluster.routing.allocation.enable" : "primaries" } }'
```

Example output

```
{
  "acknowledged": true,
  "persistent": {
    "cluster": {
      "routing": {
        "allocation": {
          "enable": "primaries"
        }
      }
    }
  },
  "transient": {}
}
```

6. After the command is complete, for each deployment you have for an ES cluster:

- a. By default, the OpenShift Container Platform Elasticsearch cluster blocks rollouts to their nodes. Use the following command to allow rollouts and allow the pod to pick up the changes:

```
$ oc rollout resume deployment/<deployment-name>
```

For example:

```
$ oc rollout resume deployment/elasticsearch-cdm-0-1
```

Example output

```
deployment.extensions/elasticsearch-cdm-0-1 resumed
```

A new pod is deployed. After the pod has a ready container, you can move on to the next deployment.

```
$ oc get pods -l component=elasticsearch-
```

Example output

NAME	READY	STATUS	RESTARTS	AGE
elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k	2/2	Running	0	22h
elasticsearch-cdm-5ceex6ts-2-f799564cb-l9mj7	2/2	Running	0	22h
elasticsearch-cdm-5ceex6ts-3-585968dc68-k7kjr	2/2	Running	0	22h

- b. After the deployments are complete, reset the pod to disallow rollouts:

```
$ oc rollout pause deployment/<deployment-name>
```

For example:

```
$ oc rollout pause deployment/elasticsearch-cdm-0-1
```

Example output

```
deployment.extensions/elasticsearch-cdm-0-1 paused
```

- c. Check that the Elasticsearch cluster is in a **green** or **yellow** state:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query=_cluster/health?pretty=true
```


**NOTE**

If you performed a rollout on the Elasticsearch pod you used in the previous commands, the pod no longer exists and you need a new pod name here.

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query=_cluster/health?pretty=true
```

```
{
  "cluster_name" : "elasticsearch",
  "status" : "yellow", ❶
  "timed_out" : false,
  "number_of_nodes" : 3,
  "number_of_data_nodes" : 3,
  "active_primary_shards" : 8,
  "active_shards" : 16,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 1,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

❶ Make sure this parameter value is **green** or **yellow** before proceeding.

7. If you changed the Elasticsearch configuration map, repeat these steps for each Elasticsearch pod.
8. After all the deployments for the cluster have been rolled out, re-enable shard balancing:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --
query="_cluster/settings" -XPUT -d '{ "persistent": { "cluster.routing.allocation.enable" : "all" }
}'
```

Example output

```
{
  "acknowledged" : true,
  "persistent" : { },
  "transient" : {
    "cluster" : {
      "routing" : {
```

```

        "allocation" : {
          "enable" : "all"
        }
      }
    }
  }
}

```

9. Scale up the collector pods so they send new logs to Elasticsearch.

```

$ oc -n openshift-logging patch daemonset/collector -p '{"spec":{"template":{"spec":{"nodeSelector":{"logging-infra-collector": "true"}}}}}'

```

8.2.9. Exposing the log store service as a route

By default, the log store that is deployed with the logging subsystem for Red Hat OpenShift is not accessible from outside the logging cluster. You can enable a route with re-encryption termination for external access to the log store service for those tools that access its data.

Externally, you can access the log store by creating a reencrypt route, your OpenShift Container Platform token and the installed log store CA certificate. Then, access a node that hosts the log store service with a cURL request that contains:

- The **Authorization: Bearer \${token}**
- The Elasticsearch reencrypt route and an [Elasticsearch API request](#).

Internally, you can access the log store service using the log store cluster IP, which you can get by using either of the following commands:

```

$ oc get service elasticsearch -o jsonpath={.spec.clusterIP} -n openshift-logging

```

Example output

```

172.30.183.229

```

```

$ oc get service elasticsearch -n openshift-logging

```

Example output

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
elasticsearch	ClusterIP	172.30.183.229	<none>	9200/TCP	22h

You can check the cluster IP address with a command similar to the following:

```

$ oc exec elasticsearch-cdm-oplnhinv-1-5746475887-fj2f8 -n openshift-logging -- curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}" "https://172.30.183.229:9200/_cat/health"

```

Example output

% Total	% Received	% Xferd	Average Speed		Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left	Speed
100	29	100	29	0	0	108	0	--:--:-- --:--:-- --:--:-- 108

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.
- You must have access to the project to be able to access to the logs.

Procedure

To expose the log store externally:

1. Change to the **openshift-logging** project:

```
$ oc project openshift-logging
```

2. Extract the CA certificate from the log store and write to the **admin-ca** file:

```
$ oc extract secret/elasticsearch --to=. --keys=admin-ca
```

Example output

```
admin-ca
```

3. Create the route for the log store service as a YAML file:

- a. Create a YAML file with the following:

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: elasticsearch
  namespace: openshift-logging
spec:
  host:
  to:
    kind: Service
    name: elasticsearch
  tls:
    termination: reencrypt
    destinationCACertificate: | 1
```

1

Add the log store CA certificate or use the command in the next step. You do not have to set the **spec.tls.key**, **spec.tls.certificate**, and **spec.tls.caCertificate** parameters required by some reencrypt routes.

- b. Run the following command to add the log store CA certificate to the route YAML you created in the previous step:

```
$ cat ./admin-ca | sed -e "s/^ / /" >> <file-name>.yaml
```

- c. Create the route:

```
$ oc create -f <file-name>.yaml
```

Example output

```
route.route.openshift.io/elasticsearch created
```

4. Check that the Elasticsearch service is exposed:

- a. Get the token of this service account to be used in the request:

```
$ token=$(oc whoami -t)
```

- b. Set the **elasticsearch** route you created as an environment variable.

```
$ routeES=$(oc get route elasticsearch -o jsonpath={.spec.host})
```

- c. To verify the route was successfully created, run the following command that accesses Elasticsearch through the exposed route:

```
curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}" "https://${routeES}"
```

The response appears similar to the following:

Example output

```
{
  "name" : "elasticsearch-cdm-i40ktba0-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "0eY-tJzcR3KOdpgeMJo-MQ",
  "version" : {
    "number" : "6.8.1",
    "build_flavor" : "oss",
    "build_type" : "zip",
    "build_hash" : "Unknown",
    "build_date" : "Unknown",
    "build_snapshot" : true,
    "lucene_version" : "7.7.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "<tagline>" : "<for search>"
}
```

8.2.10. Removing unused components if you do not use the default Elasticsearch log store

As an administrator, in the rare case that you forward logs to a third-party log store and do not use the default Elasticsearch log store, you can remove several unused components from your logging cluster.

In other words, if you do not use the default Elasticsearch log store, you can remove the internal Elasticsearch **logStore** and Kibana **visualization** components from the **ClusterLogging** custom resource (CR). Removing these components is optional but saves resources.

Prerequisites

- Verify that your log forwarder does not send log data to the default internal Elasticsearch cluster. Inspect the **ClusterLogForwarder** CR YAML file that you used to configure log forwarding. Verify that it *does not* have an **outputRefs** element that specifies **default**. For example:

```
outputRefs:
- default
```



WARNING

Suppose the **ClusterLogForwarder** CR forwards log data to the internal Elasticsearch cluster, and you remove the **logStore** component from the **ClusterLogging** CR. In that case, the internal Elasticsearch cluster will not be present to store the log data. This absence can cause data loss.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance
```

2. If they are present, remove the **logStore** and **visualization** stanzas from the **ClusterLogging** CR.
3. Preserve the **collection** stanza of the **ClusterLogging** CR. The result should look similar to the following example:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

4. Verify that the collector pods are redeployed:

```
$ oc get pods -l component=collector -n openshift-logging
```

8.3. CONFIGURING THE LOG VISUALIZER

OpenShift Container Platform uses Kibana to display the log data collected by the logging subsystem.

You can scale Kibana for redundancy and configure the CPU and memory for your Kibana nodes.

8.3.1. Configuring CPU and memory limits

The logging subsystem components allow for adjustments to both the CPU and memory limits.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc -n openshift-logging edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: ①
      limits:
        memory: 16Gi
      requests:
        cpu: 200m
        memory: 16Gi
    storage:
      storageClassName: "gp2"
      size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: ②
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy:
      resources: ③
      limits:
        memory: 100Mi
      requests:
```

```

      cpu: 100m
      memory: 100Mi
    replicas: 2
  collection:
    logs:
      type: "fluentd"
    fluentd:
      resources: 4
      limits:
        memory: 736Mi
      requests:
        cpu: 200m
        memory: 736Mi

```

- 1 Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.
- 2 3 Specify the CPU and memory limits and requests for the log visualizer as needed.
- 4 Specify the CPU and memory limits and requests for the log collector as needed.

8.3.2. Scaling redundancy for the log visualizer nodes

You can scale the pod that hosts the log visualizer for redundancy.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```

$ oc edit ClusterLogging instance

$ oc edit ClusterLogging instance

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....

spec:
  visualization:
    type: "kibana"
    kibana:
      replicas: 1 1

```

- 1 Specify the number of Kibana nodes.

8.4. CONFIGURING LOGGING SUBSYSTEM STORAGE

Elasticsearch is a memory-intensive application. The default logging subsystem installation deploys 16G of memory for both memory requests and memory limits. The initial set of OpenShift Container

Platform nodes might not be large enough to support the Elasticsearch cluster. You must add additional nodes to the OpenShift Container Platform cluster to run with the recommended or higher memory. Each Elasticsearch node can operate with a lower memory setting, though this is not recommended for production environments.

8.4.1. Storage considerations for the logging subsystem for Red Hat OpenShift

A persistent volume is required for each Elasticsearch deployment configuration. On OpenShift Container Platform this is achieved using persistent volume claims.



NOTE

If you use a local volume for persistent storage, do not use a raw block volume, which is described with **volumeMode: block** in the **LocalVolume** object. Elasticsearch cannot use raw block volumes.

The OpenShift Elasticsearch Operator names the PVCs using the Elasticsearch resource name.

Fluentd ships any logs from **systemd journal** and **/var/log/containers/** to Elasticsearch.

Elasticsearch requires sufficient memory to perform large merge operations. If it does not have enough memory, it becomes unresponsive. To avoid this problem, evaluate how much application log data you need, and allocate approximately double that amount of free storage capacity.

By default, when storage capacity is 85% full, Elasticsearch stops allocating new data to the node. At 90%, Elasticsearch attempts to relocate existing shards from that node to other nodes if possible. But if no nodes have a free capacity below 85%, Elasticsearch effectively rejects creating new indices and becomes RED.



NOTE

These low and high watermark values are Elasticsearch defaults in the current release. You can modify these default values. Although the alerts use the same default values, you cannot change these values in the alerts.

8.4.2. Additional resources

- [Configuring persistent storage for the log store](#)

8.5. CONFIGURING CPU AND MEMORY LIMITS FOR LOGGING SUBSYSTEM COMPONENTS

You can configure both the CPU and memory limits for each of the logging subsystem components as needed.

8.5.1. Configuring CPU and memory limits

The logging subsystem components allow for adjustments to both the CPU and memory limits.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:


```
$ oc -n openshift-logging edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources: 1
      limits:
        memory: 16Gi
      requests:
        cpu: 200m
        memory: 16Gi
    storage:
      storageClassName: "gp2"
      size: "200G"
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      resources: 2
      limits:
        memory: 1Gi
      requests:
        cpu: 500m
        memory: 1Gi
    proxy:
      resources: 3
      limits:
        memory: 100Mi
      requests:
        cpu: 100m
        memory: 100Mi
    replicas: 2
  collection:
    logs:
      type: "fluentd"
      fluentd:
        resources: 4
        limits:
          memory: 736Mi
        requests:
          cpu: 200m
          memory: 736Mi
```

- 1 Specify the CPU and memory limits and requests for the log store as needed. For Elasticsearch, you must adjust both the request value and the limit value.
- 2 3 Specify the CPU and memory limits and requests for the log visualizer as needed.
- 4 Specify the CPU and memory limits and requests for the log collector as needed.

8.6. USING TOLERATIONS TO CONTROL OPENSIFT LOGGING POD PLACEMENT

You can use taints and tolerations to ensure that logging subsystem pods run on specific nodes and that no other workload can run on those nodes.

Taints and tolerations are simple **key:value** pair. A taint on a node instructs the node to repel all pods that do not tolerate the taint.

The **key** is any string, up to 253 characters and the **value** is any string up to 63 characters. The string must begin with a letter or number, and may contain letters, numbers, hyphens, dots, and underscores.

Sample logging subsystem CR with tolerations

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...

spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      tolerations: 1
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
  resources:
    limits:
      memory: 16Gi
    requests:
      cpu: 200m
      memory: 16Gi
    storage: {}
    redundancyPolicy: "ZeroRedundancy"
  visualization:
    type: "kibana"
    kibana:
      tolerations: 2
      - key: "logging"
        operator: "Exists"
```

```

    effect: "NoExecute"
    tolerationSeconds: 6000
  resources:
    limits:
      memory: 2Gi
    requests:
      cpu: 100m
      memory: 1Gi
  replicas: 1
collection:
  logs:
    type: "fluentd"
    fluentd:
      tolerations: ❸
      - key: "logging"
        operator: "Exists"
        effect: "NoExecute"
        tolerationSeconds: 6000
      resources:
        limits:
          memory: 2Gi
        requests:
          cpu: 100m
          memory: 1Gi

```

- ❶ This toleration is added to the Elasticsearch pods.
- ❷ This toleration is added to the Kibana pod.
- ❸ This toleration is added to the logging collector pods.

8.6.1. Using tolerations to control the log store pod placement

You can control which nodes the log store pods runs on and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log store pods through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all pods that do not tolerate the taint. Using a specific **key:value** pair that is not on other pods ensures only the log store pods can run on that node.

By default, the log store pods have the following toleration:

```

tolerations:
- effect: "NoExecute"
  key: "node.kubernetes.io/disk-pressure"
  operator: "Exists"

```

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the OpenShift Logging pods:

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

For example:

```
$ oc adm taint nodes node1 elasticsearch=node:NoExecute
```

This example places a taint on **node1** that has key **elasticsearch**, value **node**, and taint effect **NoExecute**. Nodes with the **NoExecute** effect schedule only pods that match the taint and remove existing pods that do not match.

2. Edit the **logstore** section of the **ClusterLogging** CR to configure a toleration for the Elasticsearch pods:

```
logStore:
  type: "elasticsearch"
  elasticsearch:
    nodeCount: 1
    tolerations:
      - key: "elasticsearch" 1
        operator: "Exists" 2
        effect: "NoExecute" 3
        tolerationSeconds: 6000 4
```

- 1 Specify the key that you added to the node.
- 2 Specify the **Exists** operator to require a taint with the key **elasticsearch** to be present on the Node.
- 3 Specify the **NoExecute** effect.
- 4 Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the **oc adm taint** command. A pod with this toleration could be scheduled onto **node1**.

8.6.2. Using tolerations to control the log visualizer pod placement

You can control the node where the log visualizer pod runs and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to the log visualizer pod through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. A taint on a node is a **key:value pair** that instructs the node to repel all pods that do not tolerate the taint. Using a specific **key:value pair** that is not on other pods ensures only the Kibana pod can run on that node.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Use the following command to add a taint to a node where you want to schedule the log visualizer pod:

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

For example:

```
$ oc adm taint nodes node1 kibana=node:NoExecute
```

This example places a taint on **node1** that has key **kibana**, value **node**, and taint effect **NoExecute**. You must use the **NoExecute** taint effect. **NoExecute** schedules only pods that match the taint and remove existing pods that do not match.

2. Edit the **visualization** section of the **ClusterLogging** CR to configure a toleration for the Kibana pod:

```
visualization:
  type: "kibana"
  kibana:
    tolerations:
      - key: "kibana" 1
        operator: "Exists" 2
        effect: "NoExecute" 3
        tolerationSeconds: 6000 4
```

- 1 Specify the key that you added to the node.
- 2 Specify the **Exists** operator to require the **key/value/effect** parameters to match.
- 3 Specify the **NoExecute** effect.
- 4 Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the **oc adm taint** command. A pod with this toleration would be able to schedule onto **node1**.

8.6.3. Using tolerations to control the log collector pod placement

You can ensure which nodes the logging collector pods run on and prevent other workloads from using those nodes by using tolerations on the pods.

You apply tolerations to logging collector pods through the **ClusterLogging** custom resource (CR) and apply taints to a node through the node specification. You can use taints and tolerations to ensure the pod does not get evicted for things like memory and CPU issues.

By default, the logging collector pods have the following toleration:

```
tolerations:
- key: "node-role.kubernetes.io/master"
  operator: "Exists"
  effect: "NoExecute"
```

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Use the following command to add a taint to a node where you want logging collector pods to schedule logging collector pods:

```
$ oc adm taint nodes <node-name> <key>=<value>:<effect>
```

For example:

```
$ oc adm taint nodes node1 collector=node:NoExecute
```

This example places a taint on **node1** that has key **collector**, value **node**, and taint effect **NoExecute**. You must use the **NoExecute** taint effect. **NoExecute** schedules only pods that match the taint and removes existing pods that do not match.

2. Edit the **collection** stanza of the **ClusterLogging** custom resource (CR) to configure a toleration for the logging collector pods:

```
collection:
  logs:
    type: "fluentd"
    fluentd:
      tolerations:
        - key: "collector" 1
          operator: "Exists" 2
          effect: "NoExecute" 3
          tolerationSeconds: 6000 4
```

- 1 Specify the key that you added to the node.
- 2 Specify the **Exists** operator to require the **key/value/effect** parameters to match.
- 3 Specify the **NoExecute** effect.
- 4 Optionally, specify the **tolerationSeconds** parameter to set how long a pod can remain bound to a node before being evicted.

This toleration matches the taint created by the **oc adm taint** command. A pod with this toleration would be able to schedule onto **node1**.

8.6.4. Additional resources

- [Controlling pod placement using node taints](#) .

8.7. MOVING LOGGING SUBSYSTEM RESOURCES WITH NODE SELECTORS

You can use node selectors to deploy the Elasticsearch and Kibana pods to different nodes.

8.7.1. Moving OpenShift Logging resources

You can configure the Cluster Logging Operator to deploy the pods for logging subsystem components, such as Elasticsearch and Kibana, to different nodes. You cannot move the Cluster Logging Operator pod from its installed location.

For example, you can move the Elasticsearch pods to a separate node because of high CPU, memory, and disk requirements.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed. These features are not installed by default.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
...
spec:
  collection:
    logs:
      fluentd:
        resources: null
        type: fluentd
  logStore:
    elasticsearch:
      nodeCount: 3
      nodeSelector: 1
        node-role.kubernetes.io/infra: "
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          value: reserved
        - effect: NoExecute
          key: node-role.kubernetes.io/infra
          value: reserved
      redundancyPolicy: SingleRedundancy
    resources:
      limits:
        cpu: 500m
        memory: 16Gi
      requests:
        cpu: 500m
        memory: 16Gi
    storage: {}
    type: elasticsearch
```

```

managementState: Managed
visualization:
  kibana:
    nodeSelector: 2
      node-role.kubernetes.io/infra: "
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        value: reserved
      - effect: NoExecute
        key: node-role.kubernetes.io/infra
        value: reserved
    proxy:
      resources: null
    replicas: 1
    resources: null
    type: kibana
...

```

- 1 2 Add a **nodeSelector** parameter with the appropriate value to the component you want to move. You can use a **nodeSelector** in the format shown or use **<key>: <value>** pairs, based on the value specified for the node. If you added a taint to the infrastructure node, also add a matching toleration.

Verification

To verify that a component has moved, you can use the **oc get pod -o wide** command.

For example:

- You want to move the Kibana pod from the **ip-10-0-147-79.us-east-2.compute.internal** node:

```
$ oc get pod kibana-5b8bdf44f9-ccpq9 -o wide
```

Example output

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE READINESS GATES						
kibana-5b8bdf44f9-ccpq9	2/2	Running	0	27s	10.129.2.18	ip-10-0-147-79.us-east-2.compute.internal
		<none>	<none>			

- You want to move the Kibana pod to the **ip-10-0-139-48.us-east-2.compute.internal** node, a dedicated infrastructure node:

```
$ oc get nodes
```

Example output

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-133-216.us-east-2.compute.internal	Ready	master	60m	v1.26.0
ip-10-0-139-146.us-east-2.compute.internal	Ready	master	60m	v1.26.0
ip-10-0-139-192.us-east-2.compute.internal	Ready	worker	51m	v1.26.0
ip-10-0-139-241.us-east-2.compute.internal	Ready	worker	51m	v1.26.0


```
ip-10-0-147-79.us-east-2.compute.internal Ready worker 51m v1.26.0
ip-10-0-152-241.us-east-2.compute.internal Ready master 60m v1.26.0
ip-10-0-139-48.us-east-2.compute.internal Ready infra 51m v1.26.0
```

Note that the node has a **node-role.kubernetes.io/infra: "** label:

```
$ oc get node ip-10-0-139-48.us-east-2.compute.internal -o yaml
```

Example output

```
kind: Node
apiVersion: v1
metadata:
  name: ip-10-0-139-48.us-east-2.compute.internal
  selfLink: /api/v1/nodes/ip-10-0-139-48.us-east-2.compute.internal
  uid: 62038aa9-661f-41d7-ba93-b5f1b6ef8751
  resourceVersion: '39083'
  creationTimestamp: '2020-04-13T19:07:55Z'
  labels:
    node-role.kubernetes.io/infra: "
...
```

- To move the Kibana pod, edit the **ClusterLogging** CR to add a node selector:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

...

spec:

...

visualization:
  kibana:
    nodeSelector: 1
    node-role.kubernetes.io/infra: "
    proxy:
      resources: null
    replicas: 1
    resources: null
    type: kibana
```

- 1 Add a node selector to match the label in the node specification.

- After you save the CR, the current Kibana pod is terminated and new pod is deployed:

```
$ oc get pods
```

Example output

```
NAME                                READY STATUS    RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7 1/1 Running 0      29m
```

```

elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg 2/2 Running 0 28m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj 2/2 Running 0 28m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78 2/2 Running 0 28m
fluentd-42dzz 1/1 Running 0 28m
fluentd-d74rq 1/1 Running 0 28m
fluentd-m5vr9 1/1 Running 0 28m
fluentd-nkx17 1/1 Running 0 28m
fluentd-pdvqb 1/1 Running 0 28m
fluentd-tflh6 1/1 Running 0 28m
kibana-5b8bdf44f9-ccpq9 2/2 Terminating 0 4m11s
kibana-7d85dcffc8-bfpfp 2/2 Running 0 33s

```

- The new pod is on the **ip-10-0-139-48.us-east-2.compute.internal** node:

```
$ oc get pod kibana-7d85dcffc8-bfpfp -o wide
```

Example output

```

NAME                                READY STATUS   RESTARTS AGE IP          NODE
NOMINATED NODE READINESS GATES
kibana-7d85dcffc8-bfpfp 2/2 Running    0      43s 10.131.0.22 ip-10-0-139-48.us-
east-2.compute.internal <none>    <none>

```

- After a few moments, the original Kibana pod is removed.

```
$ oc get pods
```

Example output

```

NAME                                READY STATUS   RESTARTS AGE
cluster-logging-operator-84d98649c4-zb9g7 1/1 Running 0 30m
elasticsearch-cdm-hwv01pf7-1-56588f554f-kpmlg 2/2 Running 0 29m
elasticsearch-cdm-hwv01pf7-2-84c877d75d-75wqj 2/2 Running 0 29m
elasticsearch-cdm-hwv01pf7-3-f5d95b87b-4nx78 2/2 Running 0 29m
fluentd-42dzz 1/1 Running 0 29m
fluentd-d74rq 1/1 Running 0 29m
fluentd-m5vr9 1/1 Running 0 29m
fluentd-nkx17 1/1 Running 0 29m
fluentd-pdvqb 1/1 Running 0 29m
fluentd-tflh6 1/1 Running 0 29m
kibana-7d85dcffc8-bfpfp 2/2 Running 0 62s

```

8.8. CONFIGURING SYSTEMD-JOURNALD AND FLUENTD

Because Fluentd reads from the journal, and the journal default settings are very low, journal entries can be lost because the journal cannot keep up with the logging rate from system services.

We recommend setting **RateLimitIntervalSec=30s** and **RateLimitBurst=10000** (or even higher if necessary) to prevent the journal from losing entries.

8.8.1. Configuring systemd-journald for OpenShift Logging

As you scale up your project, the default logging environment might need some adjustments.

For example, if you are missing logs, you might have to increase the rate limits for `journald`. You can adjust the number of messages to retain for a specified period of time to ensure that OpenShift Logging does not use excessive resources without dropping logs.

You can also determine if you want the logs compressed, how long to retain logs, how or if the logs are stored, and other settings.

Procedure

1. Create a Butane config file, **40-worker-custom-journald.bu**, that includes an **/etc/systemd/journald.conf** file with the required settings.



NOTE

See "Creating machine configs with Butane" for information about Butane.

```
variant: openshift
version: 4.13.0
metadata:
  name: 40-worker-custom-journald
labels:
  machineconfiguration.openshift.io/role: "worker"
storage:
  files:
  - path: /etc/systemd/journald.conf
    mode: 0644 1
    overwrite: true
    contents:
      inline: |
        Compress=yes 2
        ForwardToConsole=no 3
        ForwardToSyslog=no
        MaxRetentionSec=1month 4
        RateLimitBurst=10000 5
        RateLimitIntervalSec=30s
        Storage=persistent 6
        SyncIntervalSec=1s 7
        SystemMaxUse=8G 8
        SystemKeepFree=20% 9
        SystemMaxFileSize=10M 10
```

- 1 Set the permissions for the **journald.conf** file. It is recommended to set **0644** permissions.
- 2 Specify whether you want logs compressed before they are written to the file system. Specify **yes** to compress the message or **no** to not compress. The default is **yes**.
- 3 Configure whether to forward log messages. Defaults to **no** for each. Specify:
 - **ForwardToConsole** to forward logs to the system console.
 - **ForwardToKsmg** to forward logs to the kernel log buffer.
 - **ForwardToSyslog** to forward to a syslog daemon.

- **ForwardToWall** to forward messages as wall messages to all logged-in users.
- 4 Specify the maximum time to store journal entries. Enter a number to specify seconds. Or include a unit: "year", "month", "week", "day", "h" or "m". Enter **0** to disable. The default is **1month**.
 - 5 Configure rate limiting. If more logs are received than what is specified in **RateLimitBurst** during the time interval defined by **RateLimitIntervalSec**, all further messages within the interval are dropped until the interval is over. It is recommended to set **RateLimitIntervalSec=30s** and **RateLimitBurst=10000**, which are the defaults.
 - 6 Specify how logs are stored. The default is **persistent**:
 - **volatile** to store logs in memory in `/var/log/journal/`.
 - **persistent** to store logs to disk in `/var/log/journal/`. systemd creates the directory if it does not exist.
 - **auto** to store logs in `/var/log/journal/` if the directory exists. If it does not exist, systemd temporarily stores logs in `/run/systemd/journal`.
 - **none** to not store logs. systemd drops all logs.
 - 7 Specify the timeout before synchronizing journal files to disk for **ERR**, **WARNING**, **NOTICE**, **INFO**, and **DEBUG** logs. systemd immediately syncs after receiving a **CRIT**, **ALERT**, or **EMERG** log. The default is **1s**.
 - 8 Specify the maximum size the journal can use. The default is **8G**.
 - 9 Specify how much disk space systemd must leave free. The default is **20%**.
 - 10 Specify the maximum size for individual journal files stored persistently in `/var/log/journal`. The default is **10M**.

**NOTE**

If you are removing the rate limit, you might see increased CPU utilization on the system logging daemons as it processes any messages that would have previously been throttled.

For more information on systemd settings, see <https://www.freedesktop.org/software/systemd/man/journald.conf.html>. The default settings listed on that page might not apply to OpenShift Container Platform.

2. Use Butane to generate a **MachineConfig** object file, **40-worker-custom-journald.yaml**, containing the configuration to be delivered to the nodes:

```
$ butane 40-worker-custom-journald.bu -o 40-worker-custom-journald.yaml
```

3. Apply the machine config. For example:

```
$ oc apply -f 40-worker-custom-journald.yaml
```

The controller detects the new **MachineConfig** object and generates a new **rendered-worker-`<hash>`** version.

4. Monitor the status of the rollout of the new rendered configuration to each node:

```
$ oc describe machineconfigpool/worker
```

Example output

```
Name:      worker
Namespace:
Labels:     machineconfiguration.openshift.io/mco-built-in=
Annotations: <none>
API Version: machineconfiguration.openshift.io/v1
Kind:      MachineConfigPool

...

Conditions:
  Message:
  Reason:      All nodes are updating to rendered-worker-
913514517bcea7c93bd446f4830bc64e
```

CHAPTER 9. LOGGING USING LOKISTACK

In logging subsystem documentation, *LokiStack* refers to the logging subsystem supported combination of Loki and web proxy with OpenShift Container Platform authentication integration. LokiStack's proxy uses OpenShift Container Platform authentication to enforce multi-tenancy. *Loki* refers to the log store as either the individual component or an external store.

Loki is a horizontally scalable, highly available, multi-tenant log aggregation system currently offered as an alternative to Elasticsearch as a log store for the logging subsystem. Elasticsearch indexes incoming log records completely during ingestion. Loki only indexes a few fixed labels during ingestion and defers more complex parsing until after the logs have been stored. This means Loki can collect logs more quickly. You can query Loki by using the [LogQL log query language](#).

9.1. DEPLOYMENT SIZING

Sizing for Loki follows the format of **N<x>.<size>** where the value **<N>** is number of instances and **<size>** specifies performance capabilities.



NOTE

1x.extra-small is for demo purposes only, and is not supported.

Table 9.1. Loki Sizing

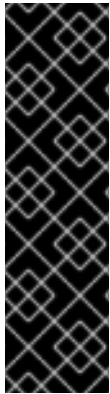
	1x.extra-small	1x.small	1x.medium
Data transfer	Demo use only.	500GB/day	2TB/day
Queries per second (QPS)	Demo use only.	25-50 QPS at 200ms	25-75 QPS at 200ms
Replication factor	None	2	3
Total CPU requests	5 vCPUs	36 vCPUs	54 vCPUs
Total Memory requests	7.5Gi	63Gi	139Gi
Total Disk requests	150Gi	300Gi	450Gi

9.1.1. Supported API Custom Resource Definitions

LokiStack development is ongoing, not all APIs are supported currently supported.

CustomResourceDefinition (CRD)	ApiVersion	Support state
LokiStack	lokistack.loki.grafana.com/v1	Supported in 5.5
RulerConfig	rulerconfig.loki.grafana.com/v1beta1	Technology Preview

CustomResourceDefinition (CRD)	ApiVersion	Support state
AlertingRule	alertingrule.loki.grafana/v1beta1	Technology Preview
RecordingRule	recordingrule.loki.grafana/v1beta1	Technology Preview



IMPORTANT

Usage of **RulerConfig**, **AlertingRule** and **RecordingRule** custom resource definitions (CRDs) 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](#).

9.2. DEPLOYING THE LOKISTACK

You can use the OpenShift Container Platform web console to deploy the LokiStack.

Prerequisites

- Logging subsystem for Red Hat OpenShift Operator 5.5 and later
- Supported Log Store (AWS S3, Google Cloud Storage, Azure, Swift, Minio, OpenShift Data Foundation)

Procedure

1. Install the **Loki Operator** Operator:
 - a. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
 - b. Choose **Loki Operator** from the list of available Operators, and click **Install**.
 - c. Under **Installation Mode**, select **All namespaces on the cluster**.
 - d. Under **Installed Namespace**, select **openshift-operators-redhat**.
You must specify the **openshift-operators-redhat** namespace. The **openshift-operators** namespace might contain Community Operators, which are untrusted and might publish a metric with the same name as an OpenShift Container Platform metric, which would cause conflicts.
 - e. Select **Enable operator recommended cluster monitoring on this namespace**.
This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.
 - f. Select an **Approval Strategy**.
 - The **Automatic** strategy allows OpenShift Lifecycle Manager (OLM) to automatically

- The **Automatic** strategy allows Operator Lifecycle Manager (OLM) to automatically update the Operator when a new version is available.
 - The **Manual** strategy requires a user with appropriate credentials to approve the Operator update.
- g. Click **Install**.
 - h. Verify that you installed the Loki Operator. Visit the **Operators → Installed Operators** page and look for **Loki Operator**.
 - i. Ensure that **Loki Operator** is listed with **Status** as **Succeeded** in all the projects.
2. Create a **Secret** YAML file that uses the **access_key_id** and **access_key_secret** fields to specify your AWS credentials and **bucketnames**, **endpoint** and **region** to define the object storage location. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: logging-loki-s3
  namespace: openshift-logging
stringData:
  access_key_id: AKIAIOSFODNN7EXAMPLE
  access_key_secret: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  bucketnames: s3-bucket-name
  endpoint: https://s3.eu-central-1.amazonaws.com
  region: eu-central-1
```

3. Create the **LokiStack** custom resource (CR):

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  size: 1x.small
  storage:
    schemas:
      - version: v12
        effectiveDate: "2022-06-01"
    secret:
      name: logging-loki-s3
      type: s3
  storageClassName: gp3-csi 1
  tenants:
    mode: openshift-logging
```

1 Or **gp2-csi**.

4. Apply the **LokiStack** CR:

```
$ oc apply -f logging-loki.yaml
```


5. Create a **ClusterLogging** custom resource (CR):

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
  lokistack:
    name: logging-loki
  collection:
    type: vector
```

6. Apply the **ClusterLogging** CR:

```
$ oc apply -f cr-lokistack.yaml
```

7. Enable the RedHat OpenShift Logging Console Plugin:

- a. In the OpenShift Container Platform web console, click **Operators** → **Installed Operators**.
- b. Select the **RedHat OpenShift Logging** Operator.
- c. Under Console plugin, click **Disabled**.
- d. Select **Enable** and then **Save**. This change restarts the **openshift-console** pods.
- e. After the pods restart, you will receive a notification that a web console update is available, prompting you to refresh.
- f. After refreshing the web console, click **Observe** from the left main menu. A new option for **Logs** is available.

9.3. INSTALLING LOGGING OPERATORS USING THE OPENSIFT CONTAINER PLATFORM WEB CONSOLE

To install and configure logging on your OpenShift Container Platform cluster, additional Operators must be installed. This can be done from the Operator Hub within the web console.

OpenShift Container Platform Operators use custom resources (CR) to manage applications and their components. High-level configuration and settings are provided by the user within a CR. The Operator translates high-level directives into low-level actions, based on best practices embedded within the Operator's logic. A custom resource definition (CRD) defines a CR and lists all the configurations available to users of the Operator. Installing an Operator creates the CRDs, which are then used to generate CRs.

Prerequisites

- Supported object store (AWS S3, Google Cloud Storage, Azure, Swift, Minio, OpenShift Data Foundation)

Procedure

1. Install the **Loki Operator**:

- a. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
- b. Type **Loki Operator** in the filter by keyword box. Choose **Loki Operator** from the list of available Operators and click **Install**.



NOTE

The Community Loki Operator is not supported by Red Hat.

- c. On the Install Operator page, for **Update Channel** select **stable**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

As the Loki Operator must be deployed to the global operator group namespace **openshift-operators-redhat**, **Installation mode** and **Installed Namespace** is already be selected. If this namespace does not already exist, it is created for you.

- a. Select **Enable operator-recommended cluster monitoring on this namespace**.

This option sets the **openshift.io/cluster-monitoring: "true"** label in the Namespace object. You must select this option to ensure that cluster monitoring scrapes the **openshift-operators-redhat** namespace.

- a. For **Update approval** select **Automatic**, then click **Install**.
If the approval strategy in the subscription is set to **Automatic**, the update process initiates as soon as a new Operator version is available in the selected channel. If the approval strategy is set to **Manual**, you must manually approve pending updates.

1. Install the **Red Hat OpenShift Logging Operator**:
- b. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
- c. Type **OpenShift Logging** in the filter by keyword box. Choose **Red Hat OpenShift Logging** from the list of available Operators and click **Install**.
- d. On the Install Operator page, under **Update channel** select **stable**.



NOTE

The **stable** channel only provides updates to the most recent release of logging. To continue receiving updates for prior releases, you must change your subscription channel to **stable-X** where **X** is the version of logging you have installed.

As the **Red Hat OpenShift Logging** Operator is only deployable to the **openshift-logging** namespace, **Installation mode** and **Installed Namespace** is already selected. If this namespace does not already exist, it is created for you.

- a. If you are creating the **openshift-logging** namespace, select the option to **Enable Operator recommended cluster monitoring on this Namespace**.



NOTE

If the **openshift-logging** namespace already exists, you must add the namespace label, **openshift.io/cluster-monitoring: "true"**, to enable metrics service discovery.

- b. Under **Update approval** select **Automatic**.
If the approval strategy in the subscription is set to **Automatic**, the update process initiates as soon as a new Operator version is available in the selected channel. If the approval strategy is set to **Manual**, you must manually approve pending updates.
- c. For **Console plugin** select **Enable**, then click **Install**.

The Operators should now be available to all users and projects that use this cluster.

1. Verify the operator installations:
 - a. Navigate to **Operators → Installed Operators**.
 - b. Make sure the **openshift-logging** project is selected.
 - c. In the **Status** column, verify that you see green checks with **InstallSucceeded** and the text **Up to date**, below.



NOTE

An Operator might display a **Failed** status before the installation finishes. If the Operator install completes with an **InstallSucceeded** message, refresh the page.

9.4. ENABLING STREAM-BASED RETENTION WITH LOKI

With Logging version 5.6 and higher, you can configure retention policies based on log streams. Rules for these may be set globally, per tenant, or both. If you configure both, tenant rules apply before global rules.

1. To enable stream-based retention, create a **LokiStack** custom resource (CR):

Example global stream-based retention

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global: 1
```

```

retention: 2
  days: 20
  streams:
    - days: 4
      priority: 1
      selector: '{kubernetes_namespace_name=~"test.+"}' 3
    - days: 1
      priority: 1
      selector: '{log_type="infrastructure"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
    - effectiveDate: "2020-10-11"
      version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: standard
tenants:
  mode: openshift-logging

```

- 1 Sets retention policy for all log streams. **Note: This field does not impact the retention period for stored logs in object storage.**
- 2 Retention is enabled in the cluster when this block is added to the CR.
- 3 Contains the [LogQL query](#) used to define the log stream.

Example per-tenant stream-based retention

```

apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      retention:
        days: 20
    tenants: 1
    application:
      retention:
        days: 1
      streams:
        - days: 4
          selector: '{kubernetes_namespace_name=~"test.+"}' 2
  infrastructure:
    retention:
      days: 5
    streams:
      - days: 1

```

```

      selector: '{kubernetes_namespace_name=~"openshift-cluster.+"}'
managementState: Managed
replicationFactor: 1
size: 1x.small
storage:
  schemas:
    - effectiveDate: "2020-10-11"
      version: v11
  secret:
    name: logging-loki-s3
    type: aws
storageClassName: standard
tenants:
  mode: openshift-logging

```

- 1 Sets retention policy by tenant. Valid tenant types are **application**, **audit**, and **infrastructure**.
- 2 Contains the [LogQL query](#) used to define the log stream.

2. Apply the **LokiStack** CR:

```
$ oc apply -f <filename>.yaml
```



NOTE

This is not for managing the retention for stored logs. Global retention periods for stored logs to a supported maximum of 30 days is configured with your object storage.

9.5. FORWARDING LOGS TO LOKISTACK

To configure log forwarding to the LokiStack gateway, you must create a **ClusterLogging** custom resource (CR).

Prerequisites

- The Logging subsystem for Red Hat OpenShift version 5.5 or newer is installed on your cluster.
- The Loki Operator is installed on your cluster.

Procedure

- Create a **ClusterLogging** custom resource (CR):

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack

```

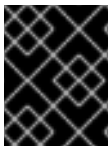
Downloaded from <http://ajph.org/> on November 10, 2014

9.5.1. Troubleshooting Loki rate limit errors

If the Log Forwarder API forwards a large block of messages that exceeds the rate limit to Loki, Loki generates rate limit (**429**) errors.

These errors can occur during normal operation. For example, when adding the logging subsystem to a cluster that already has some logs, rate limit errors might occur while the logging subsystem tries to ingest all of the existing log entries. In this case, if the rate of addition of new logs is less than the total rate limit, the historical data is eventually ingested, and the rate limit errors are resolved without requiring user intervention.

In cases where the rate limit errors continue to occur, you can fix the issue by modifying the **LokiStack** custom resource (CR).



IMPORTANT

The **LokiStack** CR is not available on Grafana-hosted Loki. This topic does not apply to Grafana-hosted Loki servers.

Conditions

- The Log Forwarder API is configured to forward logs to Loki.
- Your system sends a block of messages that is larger than 2 MB to Loki. For example:

100

- After you enter **oc logs -n openshift-logging -l component=collector**, the collector logs in your cluster show a line containing one of the following error messages:

1000

Example Vector error message

Downloaded from <http://ajphaphysocpharm.sagepub.com> at National Archive Publishing Co on June 11, 2015

Example Fluentd error message

```
2023-08-30 14:52:15 +0000 [warn]: [default_loki_infra] failed to flush the buffer. retry_times=2
next_retry_time=2023-08-30 14:52:19 +0000
chunk="604251225bf5378ed1567231a1c03b8b"
error_class=Fluent::Plugin::LokiOutput::LogPostError error="429 Too Many Requests
Ingestion rate limit exceeded for user infrastructure (limit: 4194304 bytes/sec) while
attempting to ingest '4082' lines totaling '7820025' bytes, reduce log volume or contact your
Loki administrator to see if the limit can be increased\n"
```

The error is also visible on the receiving end. For example, in the LokiStack ingester pod:

Example Loki ingester error message

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream"
```

Procedure

- Update the **ingestionBurstSize** and **ingestionRate** fields in the **LokiStack** CR:

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 1
        ingestionRate: 8 2
# ...
```

- 1** The **ingestionBurstSize** field defines the maximum local rate-limited sample size per distributor replica in MB. This value is a hard limit. Set this value to at least the maximum logs size expected in a single push request. Single requests that are larger than the **ingestionBurstSize** value are not permitted.
- 2** The **ingestionRate** field is a soft limit on the maximum amount of ingested samples per second in MB. Rate limit errors occur if the rate of logs exceeds the limit, but the collector retries sending the logs. As long as the total average is lower than the limit, the system recovers and errors are resolved without user intervention.

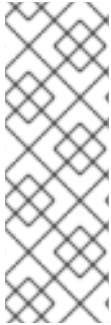
9.6. ADDITIONAL RESOURCES

- [Loki components documentation](#)
- [Loki Query Language \(LogQL\) documentation](#)
- [Grafana Dashboard documentation](#)
- [Loki Object Storage documentation](#)

- [Loki Operator **IngestionLimitSpec** documentation](#)
- [Loki Storage Schema documentation](#)

CHAPTER 10. VIEWING LOGS FOR A RESOURCE

You can view the logs for various resources, such as builds, deployments, and pods by using the OpenShift CLI (oc) and the web console.



NOTE

Resource logs are a default feature that provides limited log viewing capability. To enhance your log retrieving and viewing experience, it is recommended that you install [OpenShift Logging](#). The logging subsystem aggregates all the logs from your OpenShift Container Platform cluster, such as node system audit logs, application container logs, and infrastructure logs, into a dedicated log store. You can then query, discover, and visualize your log data through the [Kibana interface](#). Resource logs do not access the logging subsystem log store.

10.1. VIEWING RESOURCE LOGS

You can view the log for various resources in the OpenShift CLI (oc) and web console. Logs read from the tail, or end, of the log.

Prerequisites

- Access to the OpenShift CLI (oc).

Procedure (UI)

1. In the OpenShift Container Platform console, navigate to **Workloads** → **Pods** or navigate to the pod through the resource you want to investigate.



NOTE

Some resources, such as builds, do not have pods to query directly. In such instances, you can locate the **Logs** link on the **Details** page for the resource.

2. Select a project from the drop-down menu.
3. Click the name of the pod you want to investigate.
4. Click **Logs**.

Procedure (CLI)

- View the log for a specific pod:

```
$ oc logs -f <pod_name> -c <container_name>
```

where:

-f

Optional: Specifies that the output follows what is being written into the logs.

<pod_name>

Specifies the name of the pod.

<container_name>

Optional: Specifies the name of a container. When a pod has more than one container, you must specify the container name.

For example:

```
$ oc logs ruby-58cd97df55-mww7r
```

```
$ oc logs -f ruby-57f7f4855b-znl92 -c ruby
```

The contents of log files are printed out.

- View the log for a specific resource:

```
$ oc logs <object_type>/<resource_name> 1
```

1 Specifies the resource type and name.

For example:

```
$ oc logs deployment/ruby
```

The contents of log files are printed out.

CHAPTER 11. VIEWING CLUSTER LOGS BY USING KIBANA

The logging subsystem includes a web console for visualizing collected log data. Currently, OpenShift Container Platform deploys the Kibana console for visualization.

Using the log visualizer, you can do the following with your data:

- search and browse the data using the **Discover** tab.
- chart and map the data using the **Visualize** tab.
- create and view custom dashboards using the **Dashboard** tab.

Use and configuration of the Kibana interface is beyond the scope of this documentation. For more information, on using the interface, see the [Kibana documentation](#).



NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the [Log Forwarding API](#) to configure a pipeline that uses the **default** output for audit logs.

11.1. DEFINING KIBANA INDEX PATTERNS

An index pattern defines the Elasticsearch indices that you want to visualize. To explore and visualize data in Kibana, you must create an index pattern.

Prerequisites

- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.

If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

```
$ oc auth can-i get pods --subresource log -n <project>
```

Example output

```
yes
```




NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

- Elasticsearch documents must be indexed before you can create index patterns. This is done automatically, but it might take a few minutes in a new or updated cluster.

Procedure

To define index patterns and create visualizations in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher  and select **Logging**.
2. Create your Kibana index patterns by clicking **Management** → **Index Patterns** → **Create index pattern**:
 - Each user must manually create index patterns when logging into Kibana the first time to see logs for their projects. Users must create an index pattern named **app** and use the **@timestamp** time field to view their container logs.
 - Each admin user must create index patterns when logged into Kibana the first time for the **app**, **infra**, and **audit** indices using the **@timestamp** time field.
3. Create Kibana Visualizations from the new index patterns.

11.2. VIEWING CLUSTER LOGS IN KIBANA

You view cluster logs in the Kibana web console. The methods for viewing and visualizing your data in Kibana that are beyond the scope of this documentation. For more information, refer to the [Kibana documentation](#).

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.
- Kibana index patterns must exist.
- A user must have the **cluster-admin** role, the **cluster-reader** role, or both roles to view the **infra** and **audit** indices in Kibana. The default **kubeadmin** user has proper permissions to view these indices.

If you can view the pods and logs in the **default**, **kube-** and **openshift-** projects, you should be able to access these indices. You can use the following command to check if the current user has appropriate permissions:

```
$ oc auth can-i get pods --subresource log -n <project>
```

Example output

```
yes
```




NOTE

The audit logs are not stored in the internal OpenShift Container Platform Elasticsearch instance by default. To view the audit logs in Kibana, you must use the Log Forwarding API to configure a pipeline that uses the **default** output for audit logs.

Procedure

To view logs in Kibana:

1. In the OpenShift Container Platform console, click the Application Launcher  and select **Logging**.
2. Log in using the same credentials you use to log in to the OpenShift Container Platform console.
The Kibana interface launches.
3. In Kibana, click **Discover**.
4. Select the index pattern you created from the drop-down menu in the top-left corner: **app**, **audit**, or **infra**.
The log data displays as time-stamped documents.
5. Expand one of the time-stamped documents.
6. Click the **JSON** tab to display the log entry for that document.

Example 11.1. Sample infrastructure log entry in Kibana

```
{
  "_index": "infra-000001",
  "_type": "_doc",
  "_id": "YmJmYTBINDkZTRmLTliMGQtMjE3NmFiOGUyOWM3",
  "_version": 1,
  "_score": null,
  "_source": {
    "docker": {
      "container_id": "f85fa55bbef7bb783f041066be1e7c267a6b88c4603dfce213e32c1"
    },
    "kubernetes": {
      "container_name": "registry-server",
      "namespace_name": "openshift-marketplace",
      "pod_name": "redhat-marketplace-n64gc",
      "container_image": "registry.redhat.io/redhat/redhat-marketplace-index:v4.7",
      "container_image_id": "registry.redhat.io/redhat/redhat-marketplace-index@sha256:65fc0c45aabb95809e376feb065771ecda9e5e59cc8b3024c4545c168f",
      "pod_id": "8f594ea2-c866-4b5c-a1c8-a50756704b2a",
      "host": "ip-10-0-182-28.us-east-2.compute.internal",
      "master_url": "https://kubernetes.default.svc",
      "namespace_id": "3abab127-7669-4eb3-b9ef-44c04ad68d38",
      "namespace_labels": {
        "openshift_io/cluster-monitoring": "true"
      },
      "flat_labels": [
        "catalogsource_operators_coreos_com/update=redhat-marketplace"
      ]
    },
    "message": "time=\\\"2020-09-23T20:47:03Z\\\" level=info msg=\\\"serving registry\\\" database=/database/index.db port=50051",
    "level": "unknown",
    "hostname": "ip-10-0-182-28.internal",
    "pipeline_metadata": {
      "collector": {
        "ipaddr4": "10.0.182.28",
        "inputname": "fluent-plugin-systemd",
        "name": "fluentd",
```

```
    "received_at": "2020-09-23T20:47:15.007583+00:00",
    "version": "1.7.4 1.6.0"
  }
},
"@timestamp": "2020-09-23T20:47:03.422465+00:00",
"viaq_msg_id": "YmJmYTBINDktMDMGQtMjE3NmFiOGUyOWM3",
"openshift": {
  "labels": {
    "logging": "infra"
  }
}
},
"fields": {
  "@timestamp": [
    "2020-09-23T20:47:03.422Z"
  ],
  "pipeline_metadata.collector.received_at": [
    "2020-09-23T20:47:15.007Z"
  ]
},
"sort": [
  1600894023422
]
}
```

CHAPTER 12. LOG COLLECTION AND FORWARDING

12.1. ABOUT LOG COLLECTION AND FORWARDING

Administrators can create **ClusterLogForwarder** resources that specify which logs are collected, how they are transformed, and where they are forwarded to.

ClusterLogForwarder resources can be used up to forward container, infrastructure, and audit logs to specific endpoints within or outside of a cluster. Transport Layer Security (TLS) is supported so that log forwarders can be configured to send logs securely.

Administrators can also authorize RBAC permissions that define which service accounts and users can access and forward which types of logs.

12.1.1. Sending audit logs to the internal log store

By default, the logging subsystem sends container and infrastructure logs to the default internal log store defined in the **ClusterLogging** custom resource. However, it does not send audit logs to the internal store because it does not provide secure storage. If this default configuration meets your needs, you do not need to configure the Cluster Log Forwarder.



NOTE

To send audit logs to the internal Elasticsearch log store, use the Cluster Log Forwarder as described in [Forward audit logs to the log store](#).

12.1.2. About forwarding logs to third-party systems

To send logs to specific endpoints inside and outside your OpenShift Container Platform cluster, you specify a combination of *outputs* and *pipelines* in a **ClusterLogForwarder** custom resource (CR). You can also use *inputs* to forward the application logs associated with a specific project to an endpoint. Authentication is provided by a Kubernetes *Secret* object.

output

The destination for log data that you define, or where you want the logs sent. An output can be one of the following types:

- **elasticsearch**. An external Elasticsearch instance. The **elasticsearch** output can use a TLS connection.
- **fluentdForward**. An external log aggregation solution that supports Fluentd. This option uses the Fluentd **forward** protocols. The **fluentForward** output can use a TCP or TLS connection and supports shared-key authentication by providing a **shared_key** field in a secret. Shared-key authentication can be used with or without TLS.
- **syslog**. An external log aggregation solution that supports the syslog [RFC3164](#) or [RFC5424](#) protocols. The **syslog** output can use a UDP, TCP, or TLS connection.
- **cloudwatch**. Amazon CloudWatch, a monitoring and log storage service hosted by Amazon Web Services (AWS).
- **loki**. Loki, a horizontally scalable, highly available, multi-tenant log aggregation system.
- **kafka**. A Kafka broker. The **kafka** output can use a TCP or TLS connection.

- **default.** The internal OpenShift Container Platform Elasticsearch instance. You are not required to configure the default output. If you do configure a **default** output, you receive an error message because the **default** output is reserved for the Red Hat OpenShift Logging Operator.

pipeline

Defines simple routing from one log type to one or more outputs, or which logs you want to send. The log types are one of the following:

- **application.** Container logs generated by user applications running in the cluster, except infrastructure container applications.
- **infrastructure.** Container logs from pods that run in the **openshift***, **kube***, or **default** projects and journal logs sourced from node file system.
- **audit.** Audit logs generated by the node audit system, **auditd**, Kubernetes API server, OpenShift API server, and OVN network.

You can add labels to outbound log messages by using **key:value** pairs in the pipeline. For example, you might add a label to messages that are forwarded to other data centers or label the logs by type. Labels that are added to objects are also forwarded with the log message.

input

Forwards the application logs associated with a specific project to a pipeline.

In the pipeline, you define which log types to forward using an **inputRef** parameter and where to forward the logs to using an **outputRef** parameter.

Secret

A **key:value map** that contains confidential data such as user credentials.

Note the following:

- If a **ClusterLogForwarder** CR object exists, logs are not forwarded to the default Elasticsearch instance, unless there is a pipeline with the **default** output.
- By default, the logging subsystem sends container and infrastructure logs to the default internal Elasticsearch log store defined in the **ClusterLogging** custom resource. However, it does not send audit logs to the internal store because it does not provide secure storage. If this default configuration meets your needs, do not configure the Log Forwarding API.
- If you do not define a pipeline for a log type, the logs of the undefined types are dropped. For example, if you specify a pipeline for the **application** and **audit** types, but do not specify a pipeline for the **infrastructure** type, **infrastructure** logs are dropped.
- You can use multiple types of outputs in the **ClusterLogForwarder** custom resource (CR) to send logs to servers that support different protocols.
- The internal OpenShift Container Platform Elasticsearch instance does not provide secure storage for audit logs. We recommend you ensure that the system to which you forward audit logs is compliant with your organizational and governmental regulations and is properly secured. The logging subsystem does not comply with those regulations.

The following example forwards the audit logs to a secure external Elasticsearch instance, the infrastructure logs to an insecure external Elasticsearch instance, the application logs to a Kafka broker, and the application logs from the **my-apps-logs** project to the internal Elasticsearch instance.

Sample log forwarding outputs and pipelines

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  outputs:
    - name: elasticsearch-secure ❸
      type: "elasticsearch"
      url: https://elasticsearch.secure.com:9200
      secret:
        name: elasticsearch
    - name: elasticsearch-insecure ❹
      type: "elasticsearch"
      url: http://elasticsearch.insecure.com:9200
    - name: kafka-app ❺
      type: "kafka"
      url: tls://kafka.secure.com:9093/app-topic
  inputs: ❻
    - name: my-app-logs
      application:
        namespaces:
          - my-project
  pipelines:
    - name: audit-logs ❼
      inputRefs:
        - audit
      outputRefs:
        - elasticsearch-secure
        - default
      parse: json ❽
      labels:
        secure: "true" ❾
        datacenter: "east"
    - name: infrastructure-logs ❿
      inputRefs:
        - infrastructure
      outputRefs:
        - elasticsearch-insecure
      labels:
        datacenter: "west"
    - name: my-app ⓫
      inputRefs:
        - my-app-logs
      outputRefs:
        - default
    - inputRefs: ⓬
      - application
      outputRefs:
        - kafka-app
      labels:
        datacenter: "south"

```

- 1 The name of the **ClusterLogForwarder** CR must be **instance**.
- 2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- 3 Configuration for an secure Elasticsearch output using a secret with a secure URL.
 - A name to describe the output.
 - The type of output: **elasticsearch**.
 - The secure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.
 - The secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project.
- 4 Configuration for an insecure Elasticsearch output:
 - A name to describe the output.
 - The type of output: **elasticsearch**.
 - The insecure URL and port of the Elasticsearch instance as a valid absolute URL, including the prefix.
- 5 Configuration for a Kafka output using a client-authenticated TLS communication over a secure URL
 - A name to describe the output.
 - The type of output: **kafka**.
 - Specify the URL and port of the Kafka broker as a valid absolute URL, including the prefix.
- 6 Configuration for an input to filter application logs from the **my-project** namespace.
- 7 Configuration for a pipeline to send audit logs to the secure external Elasticsearch instance:
 - A name to describe the pipeline.
 - The **inputRefs** is the log type, in this example **audit**.
 - The **outputRefs** is the name of the output to use, in this example **elasticsearch-secure** to forward to the secure Elasticsearch instance and **default** to forward to the internal Elasticsearch instance.
 - Optional: Labels to add to the logs.
- 8 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- 9 Optional: String. One or more labels to add to the logs. Quote values like "true" so they are recognized as string values, not as a boolean.
- 10 Configuration for a pipeline to send infrastructure logs to the insecure external Elasticsearch instance.

- 11 Configuration for a pipeline to send logs from the **my-project** project to the internal Elasticsearch instance.
 - A name to describe the pipeline.
 - The **inputRefs** is a specific input: **my-app-logs**.
 - The **outputRefs** is **default**.
 - Optional: String. One or more labels to add to the logs.
- 12 Configuration for a pipeline to send logs to the Kafka broker, with no pipeline name:
 - The **inputRefs** is the log type, in this example **application**.
 - The **outputRefs** is the name of the output to use.
 - Optional: String. One or more labels to add to the logs.

Fluentd log handling when the external log aggregator is unavailable

If your external logging aggregator becomes unavailable and cannot receive logs, Fluentd continues to collect logs and stores them in a buffer. When the log aggregator becomes available, log forwarding resumes, including the buffered logs. If the buffer fills completely, Fluentd stops collecting logs. OpenShift Container Platform rotates the logs and deletes them. You cannot adjust the buffer size or add a persistent volume claim (PVC) to the Fluentd daemon set or pods.

Supported Authorization Keys

Common key types are provided here. Some output types support additional specialized keys, documented with the output-specific configuration field. All secret keys are optional. Enable the security features you want by setting the relevant keys. You are responsible for creating and maintaining any additional configurations that external destinations might require, such as keys and secrets, service accounts, port openings, or global proxy configuration. Open Shift Logging will not attempt to verify a mismatch between authorization combinations.

Transport Layer Security (TLS)

Using a TLS URL ('http://...' or 'ssl://...') without a Secret enables basic TLS server-side authentication. Additional TLS features are enabled by including a Secret and setting the following optional fields:

- **tls.crt**: (string) File name containing a client certificate. Enables mutual authentication. Requires **tls.key**.
- **tls.key**: (string) File name containing the private key to unlock the client certificate. Requires **tls.crt**.
- **passphrase**: (string) Passphrase to decode an encoded TLS private key. Requires **tls.key**.
- **ca-bundle.crt**: (string) File name of a customer CA for server authentication.

Username and Password

- **username**: (string) Authentication user name. Requires **password**.
- **password**: (string) Authentication password. Requires **username**.

Simple Authentication Security Layer (SASL)

- **sasl.enable** (boolean) Explicitly enable or disable SASL. If missing, SASL is automatically enabled when any of the other **sasl.** keys are set.
- **sasl.mechanisms**: (array) List of allowed SASL mechanism names. If missing or empty, the system defaults are used.
- **sasl.allow-insecure**: (boolean) Allow mechanisms that send clear-text passwords. Defaults to false.

12.1.2.1. Creating a Secret

You can create a secret in the directory that contains your certificate and key files by using the following command:

```
$ oc create secret generic -n openshift-logging <my-secret> \
  --from-file=tls.key=<your_key_file>
  --from-file=tls.crt=<your_crt_file>
  --from-file=ca-bundle.crt=<your_bundle_file>
  --from-literal=username=<your_username>
  --from-literal=password=<your_password>
```



NOTE

Generic or opaque secrets are recommended for best results.

12.1.3. Forwarding JSON logs from containers in the same pod to separate indices

You can forward structured logs from different containers within the same pod to different indices. To use this feature, you must configure the pipeline with multi-container support and annotate the pods. Logs are written to indices with a prefix of **app-**. It is recommended that Elasticsearch be configured with aliases to accommodate this.



IMPORTANT

JSON formatting of logs varies by application. Because creating too many indices impacts performance, limit your use of this feature to creating indices for logs that have incompatible JSON formats. Use queries to separate logs from different namespaces, or applications with compatible JSON formats.

Prerequisites

- Logging subsystem for Red Hat OpenShift: 5.5

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
```

```

outputDefaults:
  elasticsearch:
    enableStructuredContainerLogs: true 1
pipelines:
- inputRefs:
  - application
  name: application-logs
  outputRefs:
  - default
  parse: json

```

- 1** Enables multi-container outputs.

2. Create or edit a YAML file that defines the **Pod** CR object:

```

apiVersion: v1
kind: Pod
metadata:
  annotations:
    containerType.logging.openshift.io/heavy: heavy 1
    containerType.logging.openshift.io/low: low
spec:
  containers:
  - name: heavy 2
    image: heavyimage
  - name: low
    image: lowimage

```

- 1** Format: **containerType.logging.openshift.io/<container-name>: <index>**
- 2** Annotation names must match container names



WARNING

This configuration might significantly increase the number of shards on the cluster.

Additional Resources

- [Kubernetes Annotations](#)

12.1.4. Forwarding logs to an external Elasticsearch instance

You can optionally forward logs to an external Elasticsearch instance in addition to, or instead of, the internal OpenShift Container Platform Elasticsearch instance. You are responsible for configuring the external log aggregator to receive log data from OpenShift Container Platform.

To configure log forwarding to an external Elasticsearch instance, you must create a **ClusterLogForwarder** custom resource (CR) with an output to that instance, and a pipeline that uses

the output. The external Elasticsearch output can use the HTTP (insecure) or HTTPS (secure HTTP) connection.

To forward logs to both an external and the internal Elasticsearch instance, create outputs and pipelines to the external instance and a pipeline that uses the **default** output to forward logs to the internal instance. You do not need to create a **default** output. If you do configure a **default** output, you receive an error message because the **default** output is reserved for the Red Hat OpenShift Logging Operator.



NOTE

If you want to forward logs to **only** the internal OpenShift Container Platform Elasticsearch instance, you do not need to create a **ClusterLogForwarder** CR.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

- Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: elasticsearch-insecure 3
      type: "elasticsearch" 4
      url: http://elasticsearch.insecure.com:9200 5
    - name: elasticsearch-secure
      type: "elasticsearch"
      url: https://elasticsearch.secure.com:9200 6
  secret:
    name: es-secret 7
  pipelines:
    - name: application-logs 8
      inputRefs: 9
      - application
      - audit
      outputRefs:
        - elasticsearch-secure 10
        - default 11
      parse: json 12
      labels:
        myLabel: "myValue" 13
    - name: infrastructure-audit-logs 14
      inputRefs:
        - infrastructure
      outputRefs:
```

```
- elasticsearch-insecure
labels:
  logs: "audit-infra"
```

- 1 The name of the **ClusterLogForwarder** CR must be **instance**.
- 2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- 3 Specify a name for the output.
- 4 Specify the **elasticsearch** type.
- 5 Specify the URL and port of the external Elasticsearch instance as a valid absolute URL. You can use the **http** (insecure) or **https** (secure HTTP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP Address.
- 6 For a secure connection, you can specify an **https** or **http** URL that you authenticate by specifying a **secret**.
- 7 For an **https** prefix, specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent. Otherwise, for **http** and **https** prefixes, you can specify a secret that contains a username and password. For more information, see the following "Example: Setting secret that contains a username and password."
- 8 Optional: Specify a name for the pipeline.
- 9 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- 10 Specify the name of the output to use when forwarding logs with this pipeline.
- 11 Optional: Specify the **default** output to send the logs to the internal Elasticsearch instance.
- 12 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- 13 Optional: String. One or more labels to add to the logs.
- 14 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:
 - A name to describe the pipeline.
 - The **inputRefs** is the log type to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
 - The **outputRefs** is the name of the output to use.
 - Optional: String. One or more labels to add to the logs.

2. Create the CR object:

—

```
$ oc create -f <file-name>.yaml
```

Example: Setting a secret that contains a username and password

You can use a secret that contains a username and password to authenticate a secure connection to an external Elasticsearch instance.

For example, if you cannot use mutual TLS (mTLS) keys because a third party operates the Elasticsearch instance, you can use HTTP or HTTPS and set a secret that contains the username and password.

1. Create a **Secret** YAML file similar to the following example. Use base64-encoded values for the **username** and **password** fields. The secret type is opaque by default.

```
apiVersion: v1
kind: Secret
metadata:
  name: openshift-test-secret
data:
  username: <username>
  password: <password>
```

2. Create the secret:

```
$ oc create secret -n openshift-logging openshift-test-secret.yaml
```

3. Specify the name of the secret in the **ClusterLogForwarder** CR:

```
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
    - name: elasticsearch
      type: "elasticsearch"
      url: https://elasticsearch.secure.com:9200
      secret:
        name: openshift-test-secret
```



NOTE

In the value of the **url** field, the prefix can be **http** or **https**.

4. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

12.1.5. Forwarding logs using the Fluentd forward protocol

You can use the Fluentd **forward** protocol to send a copy of your logs to an external log aggregator that is configured to accept the protocol instead of, or in addition to, the default Elasticsearch log store. You are responsible for configuring the external log aggregator to receive the logs from OpenShift

Container Platform.

To configure log forwarding using the **forward** protocol, you must create a **ClusterLogForwarder** custom resource (CR) with one or more outputs to the Fluentd servers, and pipelines that use those outputs. The Fluentd output can use a TCP (insecure) or TLS (secure TCP) connection.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: fluentd-server-secure 3
      type: fluentdForward 4
      url: 'tls://fluentdserver.security.example.com:24224' 5
      secret: 6
        name: fluentd-secret
    - name: fluentd-server-insecure
      type: fluentdForward
      url: 'tcp://fluentdserver.home.example.com:24224'
  pipelines:
    - name: forward-to-fluentd-secure 7
      inputRefs: 8
        - application
        - audit
      outputRefs:
        - fluentd-server-secure 9
        - default 10
      parse: json 11
      labels:
        clusterId: "C1234" 12
    - name: forward-to-fluentd-insecure 13
      inputRefs:
        - infrastructure
      outputRefs:
        - fluentd-server-insecure
      labels:
        clusterId: "C1234"
```

- 1 The name of the **ClusterLogForwarder** CR must be **instance**.
- 2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- 3 Specify a name for the output.

- 4 Specify the **fluentdForward** type.
- 5 Specify the URL and port of the external Fluentd instance as a valid absolute URL. You can use the **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.
- 6 If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent. Otherwise, for http and https prefixes, you can specify a secret that contains a username and password. For more information, see the following "Example: Setting secret that contains a username and password."
- 7 Optional: Specify a name for the pipeline.
- 8 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- 9 Specify the name of the output to use when forwarding logs with this pipeline.
- 10 Optional: Specify the **default** output to forward logs to the internal Elasticsearch instance.
- 11 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- 12 Optional: String. One or more labels to add to the logs.
- 13 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type:
 - A name to describe the pipeline.
 - The **inputRefs** is the log type to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
 - The **outputRefs** is the name of the output to use.
 - Optional: String. One or more labels to add to the logs.

2. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

12.1.5.1. Enabling nanosecond precision for Logstash to ingest data from fluentd

For Logstash to ingest log data from fluentd, you must enable nanosecond precision in the Logstash configuration file.

Procedure

- In the Logstash configuration file, set **nanosecond_precision** to **true**.

Example Logstash configuration file

```
input { tcp { codec => fluent { nanosecond_precision => true } port => 24114 } }
filter { }
output { stdout { codec => rubydebug } }
```

12.1.6. Forwarding logs using the syslog protocol

You can use the **syslog** [RFC3164](#) or [RFC5424](#) protocol to send a copy of your logs to an external log aggregator that is configured to accept the protocol instead of, or in addition to, the default Elasticsearch log store. You are responsible for configuring the external log aggregator, such as a syslog server, to receive the logs from OpenShift Container Platform.

To configure log forwarding using the **syslog** protocol, you must create a **ClusterLogForwarder** custom resource (CR) with one or more outputs to the syslog servers, and pipelines that use those outputs. The syslog output can use a UDP, TCP, or TLS connection.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  outputs:
    - name: rsyslog-east ❸
      type: syslog ❹
      syslog: ❺
        facility: local0
        rfc: RFC3164
        payloadKey: message
        severity: informational
      url: 'tls://rsyslogserver.east.example.com:514' ❻
      secret: ❼
        name: syslog-secret
    - name: rsyslog-west
      type: syslog
      syslog:
        appName: myapp
        facility: user
        msgID: mymsg
        procID: myproc
        rfc: RFC5424
        severity: debug
      url: 'udp://rsyslogserver.west.example.com:514'
  pipelines:
    - name: syslog-east ❽
```

```

inputRefs: 9
- audit
- application
outputRefs: 10
- rsyslog-east
- default 11
parse: json 12
labels:
  secure: "true" 13
  syslog: "east"
- name: syslog-west 14
  inputRefs:
  - infrastructure
  outputRefs:
  - rsyslog-west
  - default
  labels:
    syslog: "west"

```

- 1 The name of the **ClusterLogForwarder** CR must be **instance**.
- 2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- 3 Specify a name for the output.
- 4 Specify the **syslog** type.
- 5 Optional: Specify the syslog parameters, listed below.
- 6 Specify the URL and port of the external syslog instance. You can use the **udp** (insecure), **tcp** (insecure) or **tls** (secure TCP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP address.
- 7 If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have keys of: **tls.crt**, **tls.key**, and **ca-bundle.crt** that point to the respective certificates that they represent.
- 8 Optional: Specify a name for the pipeline.
- 9 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- 10 Specify the name of the output to use when forwarding logs with this pipeline.
- 11 Optional: Specify the **default** output to forward logs to the internal Elasticsearch instance.
- 12 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- 13 Optional: String. One or more labels to add to the logs. Quote values like "true" so they are recognized as string values, not as a boolean.
- 14 Optional: Configure multiple outputs to forward logs to other external log aggregators of any supported type.

any supported type.

- A name to describe the pipeline.
- The **inputRefs** is the log type to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- The **outputRefs** is the name of the output to use.
- Optional: String. One or more labels to add to the logs.

2. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

12.1.6.1. Adding log source information to message output

You can add **namespace_name**, **pod_name**, and **container_name** elements to the **message** field of the record by adding the **AddLogSource** field to your **ClusterLogForwarder** custom resource (CR).

```
spec:
  outputs:
  - name: syslogout
    syslog:
      addLogSource: true
      facility: user
      payloadKey: message
      rfc: RFC3164
      severity: debug
      tag: mytag
      type: syslog
      url: tls://syslog-receiver.openshift-logging.svc:24224
  pipelines:
  - inputRefs:
    - application
    name: test-app
    outputRefs:
    - syslogout
```



NOTE

This configuration is compatible with both RFC3164 and RFC5424.

Example syslog message output without **AddLogSource**

```
<15>1 2020-11-15T17:06:14+00:00 fluentd-9hkb4 mytag - - - {"msgcontent"=>"Message Contents",
"timestamp"=>"2020-11-15 17:06:09", "tag_key"=>"rec_tag", "index"=>56}
```

Example syslog message output with **AddLogSource**

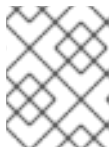
```
<15>1 2020-11-16T10:49:37+00:00 crc-j55b9-master-0 mytag - - - namespace_name=clo-test-
6327,pod_name=log-generator-ff9746c49-qxm7l,container_name=log-generator,message=
{"msgcontent":"My life is my message", "timestamp":"2020-11-16 10:49:36", "tag_key":"rec_tag",
```

```
"index":76}
```

12.1.6.2. Syslog parameters

You can configure the following for the **syslog** outputs. For more information, see the syslog [RFC3164](#) or [RFC5424](#) RFC.

- facility: The [syslog facility](#). The value can be a decimal integer or a case-insensitive keyword:
 - **0** or **kern** for kernel messages
 - **1** or **user** for user-level messages, the default.
 - **2** or **mail** for the mail system
 - **3** or **daemon** for system daemons
 - **4** or **auth** for security/authentication messages
 - **5** or **syslog** for messages generated internally by syslogd
 - **6** or **lpr** for the line printer subsystem
 - **7** or **news** for the network news subsystem
 - **8** or **uucp** for the UUCP subsystem
 - **9** or **cron** for the clock daemon
 - **10** or **authpriv** for security authentication messages
 - **11** or **ftp** for the FTP daemon
 - **12** or **ntp** for the NTP subsystem
 - **13** or **security** for the syslog audit log
 - **14** or **console** for the syslog alert log
 - **15** or **solaris-cron** for the scheduling daemon
 - **16–23** or **local0** – **local7** for locally used facilities
- Optional: **payloadKey**: The record field to use as payload for the syslog message.



NOTE

Configuring the **payloadKey** parameter prevents other parameters from being forwarded to the syslog.

- rfc: The RFC to be used for sending logs using syslog. The default is RFC5424.
- severity: The [syslog severity](#) to set on outgoing syslog records. The value can be a decimal integer or a case-insensitive keyword:
 - **0** or **Emergency** for messages indicating the system is unusable

- **1** or **Alert** for messages indicating action must be taken immediately
- **2** or **Critical** for messages indicating critical conditions
- **3** or **Error** for messages indicating error conditions
- **4** or **Warning** for messages indicating warning conditions
- **5** or **Notice** for messages indicating normal but significant conditions
- **6** or **Informational** for messages indicating informational messages
- **7** or **Debug** for messages indicating debug-level messages, the default
- tag: Tag specifies a record field to use as a tag on the syslog message.
- trimPrefix: Remove the specified prefix from the tag.

12.1.6.3. Additional RFC5424 syslog parameters

The following parameters apply to RFC5424:

- appName: The APP-NAME is a free-text string that identifies the application that sent the log. Must be specified for **RFC5424**.
- msgID: The MSGID is a free-text string that identifies the type of message. Must be specified for **RFC5424**.
- procID: The PROCID is a free-text string. A change in the value indicates a discontinuity in syslog reporting. Must be specified for **RFC5424**.

12.1.7. Forwarding logs to Amazon CloudWatch

You can forward logs to Amazon CloudWatch, a monitoring and log storage service hosted by Amazon Web Services (AWS). You can forward logs to CloudWatch in addition to, or instead of, the default log store.

To configure log forwarding to CloudWatch, you must create a **ClusterLogForwarder** custom resource (CR) with an output for CloudWatch, and a pipeline that uses the output.

Procedure

1. Create a **Secret** YAML file that uses the **aws_access_key_id** and **aws_secret_access_key** fields to specify your base64-encoded AWS credentials. For example:

```
apiVersion: v1
kind: Secret
metadata:
  name: cw-secret
  namespace: openshift-logging
data:
  aws_access_key_id: QUtJQUIPU0ZPRE5ON0VYQU1QTEUK
  aws_secret_access_key:
    d0phbHJYVXRuRkVNSS9LN01ERU5HL2JQeFJmaUNZRVhBTvBMRUtFWQo=
```

2. Create the secret. For example:

```
$ oc apply -f cw-secret.yaml
```

3. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object. In the file, specify the name of the secret. For example:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: cw 3
      type: cloudwatch 4
      cloudwatch:
        groupBy: logType 5
        groupPrefix: <group prefix> 6
        region: us-east-2 7
      secret:
        name: cw-secret 8
  pipelines:
    - name: infra-logs 9
      inputRefs: 10
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw 11
```

- 1 The name of the **ClusterLogForwarder** CR must be **instance**.
- 2 The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- 3 Specify a name for the output.
- 4 Specify the **cloudwatch** type.
- 5 Optional: Specify how to group the logs:
 - **logType** creates log groups for each log type
 - **namespaceName** creates a log group for each application name space. It also creates separate log groups for infrastructure and audit logs.
 - **namespaceUUID** creates a new log groups for each application namespace UUID. It also creates separate log groups for infrastructure and audit logs.
- 6 Optional: Specify a string to replace the default **infrastructureName** prefix in the names of the log groups.
- 7 Specify the AWS region.
- 8 Specify the name of the secret that contains your AWS credentials.

- 9 Optional: Specify a name for the pipeline.
- 10 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- 11 Specify the name of the output to use when forwarding logs with this pipeline.

4. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

Example: Using ClusterLogForwarder with Amazon CloudWatch

Here, you see an example **ClusterLogForwarder** custom resource (CR) and the log data that it outputs to Amazon CloudWatch.

Suppose that you are running an OpenShift Container Platform cluster named **mycluster**. The following command returns the cluster's **infrastructureName**, which you will use to compose **aws** commands later on:

```
$ oc get Infrastructure/cluster -ojson | jq .status.infrastructureName
"mycluster-7977k"
```

To generate log data for this example, you run a **busybox** pod in a namespace called **app**. The **busybox** pod writes a message to stdout every three seconds:

```
$ oc run busybox --image=busybox -- sh -c 'while true; do echo "My life is my message"; sleep 3; done'
$ oc logs -f busybox
My life is my message
My life is my message
My life is my message
...
```

You can look up the UUID of the **app** namespace where the **busybox** pod runs:

```
$ oc get ns/app -ojson | jq .metadata.uid
"794e1e1a-b9f5-4958-a190-e76a9b53d7bf"
```

In your **ClusterLogForwarder** custom resource (CR), you configure the **infrastructure**, **audit**, and **application** log types as inputs to the **all-logs** pipeline. You also connect this pipeline to **cw** output, which forwards the logs to a CloudWatch instance in the **us-east-2** region:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
  - name: cw
    type: cloudwatch
    cloudwatch:
      groupBy: logType
```

```

    region: us-east-2
    secret:
      name: cw-secret
  pipelines:
    - name: all-logs
      inputRefs:
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw

```

Each region in CloudWatch contains three levels of objects:

- log group
 - log stream
 - log event

With **groupBy: logType** in the **ClusterLogForwarding** CR, the three log types in the **inputRefs** produce three log groups in Amazon Cloudwatch:

```

$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.application"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"

```

Each of the log groups contains log streams:

```

$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.application | jq
.logStreams[].logStreamName
"kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log"

```

```

$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.audit | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.k8s-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.linux-audit.log"
"ip-10-0-131-228.us-east-2.compute.internal.openshift-audit.log"
...

```

```

$ aws --output json logs describe-log-streams --log-group-name mycluster-7977k.infrastructure | jq
.logStreams[].logStreamName
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-69f9fd9b58-
zqzw5_openshift-oauth-apiserver_oauth-apiserver-
453c5c4ee026fe20a6139ba6b1cdd1bed25989c905bf5ac5ca211b7cbb5c3d7b.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-
ce51532df7d4e4d5f21c4f4be05f6575b93196336be0027067fd7d93d70f66a4.log"
"ip-10-0-131-228.us-east-2.compute.internal.kubernetes.var.log.containers.apiserver-797774f7c5-
lftrx_openshift-apiserver_openshift-apiserver-check-endpoints-
82a9096b5931b5c3b1d6dc4b66113252da4a6472c9fff48623baee761911a9ef.log"
...

```

Each log stream contains log events. To see a log event from the **busybox** Pod, you specify its log stream from the **application** log group:

```
$ aws logs get-log-events --log-group-name mycluster-7977k.application --log-stream-name
kubernetes.var.log.containers.busybox_app_busybox-
da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76.log
{
  "events": [
    {
      "timestamp": 1629422704178,
      "message": "{\"docker\":
{\\\"container_id\\\":\\\"da085893053e20beddd6747acdbaf98e77c37718f85a7f6a4facf09ca195ad76\\\"},\\\"kub
ernetes\\\":
{\\\"container_name\\\":\\\"busybox\\\",\\\"namespace_name\\\":\\\"app\\\",\\\"pod_name\\\":\\\"busybox\\\",\\\"container_ima
ge\\\":\\\"docker.io/library/busybox:latest\\\",\\\"container_image_id\\\":\\\"docker.io/library/busybox@sha256:0f35
4ec1728d9ff32edcd7d1b8bbdfc798277ad36120dc3dc683be44524c8b60\\\",\\\"pod_id\\\":\\\"870be234-
90a3-4258-b73f-4f4d6e2777c7\\\",\\\"host\\\":\\\"ip-10-0-216-3.us-east-2.compute.internal\\\",\\\"labels\\\":
{\\\"run\\\":\\\"busybox\\\"},\\\"master_url\\\":\\\"https://kubernetes.default.svc\\\",\\\"namespace_id\\\":\\\"794e1e1a-
b9f5-4958-a190-e76a9b53d7bf\\\",\\\"namespace_labels\\\":
{\\\"kubernetes_io/metadata_name\\\":\\\"app\\\"}},\\\"message\\\":\\\"My life is my
message\\\",\\\"level\\\":\\\"unknown\\\",\\\"hostname\\\":\\\"ip-10-0-216-3.us-east-
2.compute.internal\\\",\\\"pipeline_metadata\\\":{\\\"collector\\\":
{\\\"ipaddr4\\\":\\\"10.0.216.3\\\",\\\"inputname\\\":\\\"fluent-plugin-
systemd\\\",\\\"name\\\":\\\"fluentd\\\",\\\"received_at\\\":\\\"2021-08-
20T01:25:08.085760+00:00\\\",\\\"version\\\":\\\"1.7.4 1.6.0\\\"}},\\\"@timestamp\\\":\\\"2021-08-
20T01:25:04.178986+00:00\\\",\\\"viaq_index_name\\\":\\\"app-
write\\\",\\\"viaq_msg_id\\\":\\\"NWRjZmUyMWQtdzJgZNC00Mjl4LTk3MjMtNTk3NmY3ZjU4NDk1\\\",\\\"log_type\\\":
\\\"application\\\",\\\"time\\\":\\\"2021-08-20T01:25:04+00:00\\\"}\",
      "ingestionTime": 1629422744016
    },
    ...
  ]
}
```

Example: Customizing the prefix in log group names

In the log group names, you can replace the default **infrastructureName** prefix, **mycluster-7977k**, with an arbitrary string like **demo-group-prefix**. To make this change, you update the **groupPrefix** field in the **ClusterLogForwarding** CR:

```
cloudwatch:
  groupBy: logType
  groupPrefix: demo-group-prefix
  region: us-east-2
```

The value of **groupPrefix** replaces the default **infrastructureName** prefix:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"demo-group-prefix.application"
"demo-group-prefix.audit"
"demo-group-prefix.infrastructure"
```

Example: Naming log groups after application namespace names

For each application namespace in your cluster, you can create a log group in CloudWatch whose name is based on the name of the application namespace.

If you delete an application namespace object and create a new one that has the same name, CloudWatch continues using the same log group as before.

If you consider successive application namespace objects that have the same name as equivalent to each other, use the approach described in this example. Otherwise, if you need to distinguish the resulting log groups from each other, see the following "Naming log groups for application namespace UUIDs" section instead.

To create application log groups whose names are based on the names of the application namespaces, you set the value of the **groupBy** field to **namespaceName** in the **ClusterLogForwarder** CR:

```
cloudwatch:
  groupBy: namespaceName
  region: us-east-2
```

Setting **groupBy** to **namespaceName** affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

In Amazon Cloudwatch, the namespace name appears at the end of each log group name. Because there is a single application namespace, "app", the following output shows a new **mycluster-7977k.app** log group instead of **mycluster-7977k.application**:

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.app"
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

If the cluster in this example had contained multiple application namespaces, the output would show multiple log groups, one for each namespace.

The **groupBy** field affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

Example: Naming log groups after application namespace UUIDs

For each application namespace in your cluster, you can create a log group in CloudWatch whose name is based on the UUID of the application namespace.

If you delete an application namespace object and create a new one, CloudWatch creates a new log group.

If you consider successive application namespace objects with the same name as different from each other, use the approach described in this example. Otherwise, see the preceding "Example: Naming log groups for application namespace names" section instead.

To name log groups after application namespace UUIDs, you set the value of the **groupBy** field to **namespaceUUID** in the **ClusterLogForwarder** CR:

```
cloudwatch:
  groupBy: namespaceUUID
  region: us-east-2
```

In Amazon Cloudwatch, the namespace UUID appears at the end of each log group name. Because there is a single application namespace, "app", the following output shows a new **mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf** log group instead of **mycluster-7977k.application**:

■

```
$ aws --output json logs describe-log-groups | jq .logGroups[].logGroupName
"mycluster-7977k.794e1e1a-b9f5-4958-a190-e76a9b53d7bf" // uid of the "app" namespace
"mycluster-7977k.audit"
"mycluster-7977k.infrastructure"
```

The **groupBy** field affects the application log group only. It does not affect the **audit** and **infrastructure** log groups.

12.1.7.1. Forwarding logs to Amazon CloudWatch from STS enabled clusters

For clusters with AWS Security Token Service (STS) enabled, you can create an AWS service account manually or create a credentials request by using the [Cloud Credential Operator\(CCO\)](#) utility **ccoctl**.

Prerequisites

- Logging subsystem for Red Hat OpenShift: 5.5 and later

Procedure

1. Create a **CredentialsRequest** custom resource YAML by using the template below:

CloudWatch credentials request template

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: <your_role_name>-credrequest
  namespace: openshift-cloud-credential-operator
spec:
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
      - action:
          - logs:PutLogEvents
          - logs:CreateLogGroup
          - logs:PutRetentionPolicy
          - logs:CreateLogStream
          - logs:DescribeLogGroups
          - logs:DescribeLogStreams
        effect: Allow
        resource: arn:aws:logs:*:*:*
  secretRef:
    name: <your_role_name>
    namespace: openshift-logging
  serviceAccountNames:
    - logcollector
```

2. Use the **ccoctl** command to create a role for AWS using your **CredentialsRequest** CR. With the **CredentialsRequest** object, this **ccoctl** command creates an IAM role with a trust policy that is tied to the specified OIDC identity provider, and a permissions policy that grants permissions to perform operations on CloudWatch resources. This command also creates a YAML configuration file in `/<path_to_ccoctl_output_dir>/manifests/openshift-logging-<your_role_name>-credentials.yaml`. This secret file contains the **role_arn** key/value used during authentication with the AWS IAM identity provider.

```
$ ccoctl aws create-iam-roles \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \
--identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com ❶
```

- ❶ <name> is the name used to tag your cloud resources and should match the name used during your STS cluster install

3. Apply the secret created:

```
$ oc apply -f output/manifests/openshift-logging-<your_role_name>-credentials.yaml
```

4. Create or edit a **ClusterLogForwarder** custom resource:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  outputs:
    - name: cw ❸
      type: cloudwatch ❹
      cloudwatch:
        groupBy: logType ❺
        groupPrefix: <group prefix> ❻
        region: us-east-2 ❼
      secret:
        name: <your_role_name> ❽
  pipelines:
    - name: to-cloudwatch ❾
      inputRefs: ❿
        - infrastructure
        - audit
        - application
      outputRefs:
        - cw ⓫
```

- ❶ The name of the **ClusterLogForwarder** CR must be **instance**.
- ❷ The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- ❸ Specify a name for the output.
- ❹ Specify the **cloudwatch** type.
- ❺ Optional: Specify how to group the logs:
 - **logType** creates log groups for each log type

- **namespaceName** creates a log group for each application name space. Infrastructure and audit logs are unaffected, remaining grouped by **logType**.
 - **namespaceUUID** creates a new log groups for each application namespace UUID. It also creates separate log groups for infrastructure and audit logs.
- 6 Optional: Specify a string to replace the default **infrastructureName** prefix in the names of the log groups.
 - 7 Specify the AWS region.
 - 8 Specify the name of the secret that contains your AWS credentials.
 - 9 Optional: Specify a name for the pipeline.
 - 10 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
 - 11 Specify the name of the output to use when forwarding logs with this pipeline.

Additional resources

- [AWS STS API Reference](#)

12.1.7.2. Creating a secret for AWS CloudWatch with an existing AWS role

If you have an existing role for AWS, you can create a secret for AWS with STS using the **oc create secret --from-literal** command.

Procedure

- In the CLI, enter the following to generate a secret for AWS:

```
$ oc create secret generic cw-sts-secret -n openshift-logging --from-literal=role_arn=arn:aws:iam::123456789012:role/my-role_with-permissions
```

Example Secret

```
apiVersion: v1
kind: Secret
metadata:
  namespace: openshift-logging
  name: my-secret-name
stringData:
  role_arn: arn:aws:iam::123456789012:role/my-role_with-permissions
```

12.1.8. Forwarding logs to Loki

You can forward logs to an external Loki logging system in addition to, or instead of, the internal default OpenShift Container Platform Elasticsearch instance.

To configure log forwarding to Loki, you must create a **ClusterLogForwarder** custom resource (CR) with an output to Loki, and a pipeline that uses the output. The output to Loki can use the HTTP (insecure) or HTTPS (secure HTTP) connection.

Prerequisites

- You must have a Loki logging system running at the URL you specify with the **url** field in the CR.

Procedure

- Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```

apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance 1
  namespace: openshift-logging 2
spec:
  outputs:
    - name: loki-insecure 3
      type: "loki" 4
      url: http://loki.insecure.com:3100 5
      loki:
        tenantKey: kubernetes.namespace_name
        labelKeys: kubernetes.labels.foo
    - name: loki-secure 6
      type: "loki"
      url: https://loki.secure.com:3100
      secret:
        name: loki-secret 7
      loki:
        tenantKey: kubernetes.namespace_name 8
        labelKeys: kubernetes.labels.foo 9
  pipelines:
    - name: application-logs 10
      inputRefs: 11
      - application
      - audit
      outputRefs: 12
      - loki-secure

```

- The name of the **ClusterLogForwarder** CR must be **instance**.
- The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- Specify a name for the output.
- Specify the type as **"loki"**.
- Specify the URL and port of the Loki system as a valid absolute URL. You can use the **http** (insecure) or **https** (secure HTTP) protocol. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name or FQDN, not an IP Address. Loki's default port for HTTP(S) communication is 3100.
- For a secure connection, you can specify an **https** or **http** URL that you authenticate by specifying a **secret**.
-

For an **https** prefix, specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project, and must have

- 8 Optional: Specify a meta-data key field to generate values for the **TenantID** field in Loki. For example, setting **tenantKey: kubernetes.namespace_name** uses the names of the Kubernetes namespaces as values for tenant IDs in Loki. To see which other log record fields you can specify, see the "Log Record Fields" link in the following "Additional resources" section.
- 9 Optional: Specify a list of meta-data field keys to replace the default Loki labels. Loki label names must match the regular expression **[a-zA-Z-:][a-zA-Z0-9-:]***. Illegal characters in meta-data keys are replaced with **_** to form the label name. For example, the **kubernetes.labels.foo** meta-data key becomes Loki label **kubernetes_labels_foo**. If you do not set **labelKeys**, the default value is: **[log_type, kubernetes.namespace_name, kubernetes.pod_name, kubernetes_host]**. Keep the set of labels small because Loki limits the size and number of labels allowed. See [Configuring Loki, limits_config](#). You can still query based on any log record field using query filters.
- 10 Optional: Specify a name for the pipeline.
- 11 Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- 12 Specify the name of the output to use when forwarding logs with this pipeline.



NOTE

Because Loki requires log streams to be correctly ordered by timestamp, **labelKeys** always includes the **kubernetes_host** label set, even if you do not specify it. This inclusion ensures that each stream originates from a single host, which prevents timestamps from becoming disordered due to clock differences on different hosts.

2. Create the CR object:

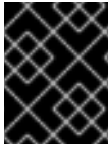
```
$ oc create -f <file-name>.yaml
```

12.1.8.1. Troubleshooting Loki rate limit errors

If the Log Forwarder API forwards a large block of messages that exceeds the rate limit to Loki, Loki generates rate limit (**429**) errors.

These errors can occur during normal operation. For example, when adding the logging subsystem to a cluster that already has some logs, rate limit errors might occur while the logging subsystem tries to ingest all of the existing log entries. In this case, if the rate of addition of new logs is less than the total rate limit, the historical data is eventually ingested, and the rate limit errors are resolved without requiring user intervention.

In cases where the rate limit errors continue to occur, you can fix the issue by modifying the **LokiStack** custom resource (CR).



IMPORTANT

The **LokiStack** CR is not available on Grafana-hosted Loki. This topic does not apply to Grafana-hosted Loki servers.

Conditions

- The Log Forwarder API is configured to forward logs to Loki.
- Your system sends a block of messages that is larger than 2 MB to Loki. For example:

```
"values":[[{"1630410392689800468","{\\"kind\\":\\"Event\\",\\"apiVersion\\":\\"
.....
.....
.....
.....
\\"received_at\\":\\"2021-08-31T11:46:32.800278+00:00\\",\\"version\\":\\"1.7.4
1.6.0\\"}},\\"@timestamp\\":\\"2021-08-
31T11:46:32.799692+00:00\\",\\"viaq_index_name\\":\\"audit-
write\\",\\"viaq_msg_id\\":\\"MzFjYjJkZjltNjY0MCM0YUWU4LWlWMTetNGNmM2E5ZmViMGU4\\",\\"lo
g_type\\":\\"audit\\"}]]]]}
```

- After you enter **oc logs -n openshift-logging -l component=collector**, the collector logs in your cluster show a line containing one of the following error messages:

```
429 Too Many Requests Ingestion rate limit exceeded
```

Example Vector error message

```
2023-08-25T16:08:49.301780Z WARN sink{component_kind="sink"
component_id=default_loki_infra component_type=loki component_name=default_loki_infra}:
vector::sinks::util::retries: Retrying after error. error=Server responded with an error: 429 Too
Many Requests internal_log_rate_limit=true
```

Example Fluentd error message

```
2023-08-30 14:52:15 +0000 [warn]: [default_loki_infra] failed to flush the buffer. retry_times=2
next_retry_time=2023-08-30 14:52:19 +0000
chunk="604251225bf5378ed1567231a1c03b8b"
error_class=Fluent::Plugin::LokiOutput::LogPostError error="429 Too Many Requests
Ingestion rate limit exceeded for user infrastructure (limit: 4194304 bytes/sec) while
attempting to ingest '4082' lines totaling '7820025' bytes, reduce log volume or contact your
Loki administrator to see if the limit can be increased\n"
```

The error is also visible on the receiving end. For example, in the LokiStack ingester pod:

Example Loki ingester error message

```
level=warn ts=2023-08-30T14:57:34.155592243Z caller=grpc_logging.go:43
duration=1.434942ms method=/logproto.Pusher/Push err="rpc error: code = Code(429) desc
= entry with timestamp 2023-08-30 14:57:32.012778399 +0000 UTC ignored, reason: 'Per
stream rate limit exceeded (limit: 3MB/sec) while attempting to ingest for stream
```

Procedure

- Update the **ingestionBurstSize** and **ingestionRate** fields in the **LokiStack** CR:

```
apiVersion: loki.grafana.com/v1
kind: LokiStack
metadata:
  name: logging-loki
  namespace: openshift-logging
spec:
  limits:
    global:
      ingestion:
        ingestionBurstSize: 16 1
        ingestionRate: 8 2
# ...
```

- ¹ The **ingestionBurstSize** field defines the maximum local rate-limited sample size per distributor replica in MB. This value is a hard limit. Set this value to at least the maximum logs size expected in a single push request. Single requests that are larger than the **ingestionBurstSize** value are not permitted.
- ² The **ingestionRate** field is a soft limit on the maximum amount of ingested samples per second in MB. Rate limit errors occur if the rate of logs exceeds the limit, but the collector retries sending the logs. As long as the total average is lower than the limit, the system recovers and errors are resolved without user intervention.

Additional resources

- [Log Record Fields](#)
- [Configuring Loki server](#)

12.1.9. Forwarding logs to Google Cloud Platform (GCP)

You can forward logs to [Google Cloud Logging](#) in addition to, or instead of, the internal default OpenShift Container Platform log store.



NOTE

Using this feature with Fluentd is not supported.

Prerequisites

- Logging subsystem for Red Hat OpenShift Operator 5.5.1 and later

Procedure

- Create a secret using your [Google service account key](#).

```
$ oc -n openshift-logging create secret generic gcp-secret --from-file google-application-credentials.json=<your_service_account_key_file.json>
```

2. Create a **ClusterLogForwarder** Custom Resource YAML using the template below:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  outputs:
    - name: gcp-1
      type: googleCloudLogging
      secret:
        name: gcp-secret
      googleCloudLogging:
        projectId : "openshift-gce-devel" 1
        logId : "app-gcp" 2
  pipelines:
    - name: test-app
      inputRefs: 3
      - application
      outputRefs:
        - gcp-1
```

- 1** Set either a **projectId**, **folderId**, **organizationId**, or **billingAccountId** field and its corresponding value, depending on where you want to store your logs in the [GCP resource hierarchy](#).
- 2** Set the value to add to the **logName** field of the [Log Entry](#).
- 3** Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.

Additional resources

- [Google Cloud Billing Documentation](#)
- [Google Cloud Logging Query Language Documentation](#)

12.1.10. Forwarding logs to Splunk

You can forward logs to the [Splunk HTTP Event Collector \(HEC\)](#) in addition to, or instead of, the internal default OpenShift Container Platform log store.



NOTE

Using this feature with Fluentd is not supported.

Prerequisites

- Red Hat OpenShift Logging Operator 5.6 and higher
- ClusterLogging instance with vector specified as collector
- Base64 encoded Splunk HEC token

Procedure

1. Create a secret using your Base64 encoded Splunk HEC token.

```
$ oc -n openshift-logging create secret generic vector-splunk-secret --from-literal hecToken=
<HEC_Token>
```

2. Create or edit the **ClusterLogForwarder** Custom Resource (CR) using the template below:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: "instance" ❶
  namespace: "openshift-logging" ❷
spec:
  outputs:
    - name: splunk-receiver ❸
      secret:
        name: vector-splunk-secret ❹
        type: splunk ❺
        url: <http://your.splunk.hec.url:8088> ❻
  pipelines: ❼
    - inputRefs:
        - application
        - infrastructure
      name: ❽
      outputRefs:
        - splunk-receiver ❾
```

- ❶ The name of the ClusterLogForwarder CR must be **instance**.
- ❷ The namespace for the ClusterLogForwarder CR must be **openshift-logging**.
- ❸ Specify a name for the output.
- ❹ Specify the name of the secret that contains your HEC token.
- ❺ Specify the output type as **splunk**.
- ❻ Specify the URL (including port) of your Splunk HEC.
- ❼ Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
- ❽ Optional: Specify a name for the pipeline.
- ❾ Specify the name of the output to use when forwarding logs with this pipeline.

12.1.11. Forwarding logs over HTTP

Forwarding logs over HTTP is supported for both fluentd and vector collectors. To enable, specify **http** as the output type in the **ClusterLogForwarder** custom resource (CR).

Procedure

- Create or edit the ClusterLogForwarder Custom Resource (CR) using the template below:

Example ClusterLogForwarder CR

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  outputs:
    - name: httpout-app
      type: http
      url: ❶
      http:
        headers: ❷
          h1: v1
          h2: v2
        method: POST
      secret:
        name: ❸
      tls:
        insecureSkipVerify: ❹
  pipelines:
    - name:
      inputRefs:
        - application
      outputRefs:
        - ❺
```

- ❶ Destination address for logs.
- ❷ Additional headers to send with the log record.
- ❸ Secret name for destination credentials.
- ❹ Values are either **true** or **false**.
- ❺ This value should be the same as the output name.

12.1.12. Forwarding application logs from specific projects

You can use the Cluster Log Forwarder to send a copy of the application logs from specific projects to an external log aggregator. You can do this in addition to, or instead of, using the default Elasticsearch log store. You must also configure the external log aggregator to receive log data from OpenShift Container Platform.

To configure forwarding application logs from a project, you must create a **ClusterLogForwarder** custom resource (CR) with at least one input from a project, optional outputs for other log aggregators, and pipelines that use those inputs and outputs.

Prerequisites

- You must have a logging server that is configured to receive the logging data using the specified protocol or format.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object:

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  outputs:
    - name: fluentd-server-secure ❸
      type: fluentdForward ❹
      url: 'tls://fluentdserver.security.example.com:24224' ❺
      secret: ❻
        name: fluentd-secret
    - name: fluentd-server-insecure
      type: fluentdForward
      url: 'tcp://fluentdserver.home.example.com:24224'
  inputs: ❼
    - name: my-app-logs
      application:
        namespaces:
          - my-project
  pipelines:
    - name: forward-to-fluentd-insecure ❸
      inputRefs: ❾
        - my-app-logs
      outputRefs: ❿
        - fluentd-server-insecure
      parse: json 11
      labels:
        project: "my-project" 12
    - name: forward-to-fluentd-secure 13
      inputRefs:
        - application
        - audit
        - infrastructure
      outputRefs:
        - fluentd-server-secure
        - default
      labels:
        clusterId: "C1234"

```

- ❶ The name of the **ClusterLogForwarder** CR must be **instance**.
- ❷ The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- ❸ Specify a name for the output.
- ❹ Specify the output type: **elasticsearch**, **fluentdForward**, **syslog**, or **kafka**.

- 5 Specify the URL and port of the external log aggregator as a valid absolute URL. If the cluster-wide proxy using the CIDR annotation is enabled, the output must be a server name
- 6 If using a **tls** prefix, you must specify the name of the secret required by the endpoint for TLS communication. The secret must exist in the **openshift-logging** project and have **tls.crt**, **tls.key**, and **ca-bundle.crt** keys that each point to the certificates they represent.
- 7 Configuration for an input to filter application logs from the specified projects.
- 8 Configuration for a pipeline to use the input to send project application logs to an external Fluentd instance.
- 9 The **my-app-logs** input.
- 10 The name of the output to use.
- 11 Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- 12 Optional: String. One or more labels to add to the logs.
- 13 Configuration for a pipeline to send logs to other log aggregators.
 - Optional: Specify a name for the pipeline.
 - Specify which log types to forward by using the pipeline: **application**, **infrastructure**, or **audit**.
 - Specify the name of the output to use when forwarding logs with this pipeline.
 - Optional: Specify the **default** output to forward logs to the internal Elasticsearch instance.
 - Optional: String. One or more labels to add to the logs.

2. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

12.1.13. Forwarding application logs from specific pods

As a cluster administrator, you can use Kubernetes pod labels to gather log data from specific pods and forward it to a log collector.

Suppose that you have an application composed of pods running alongside other pods in various namespaces. If those pods have labels that identify the application, you can gather and output their log data to a specific log collector.

To specify the pod labels, you use one or more **matchLabels** key-value pairs. If you specify multiple key-value pairs, the pods must match all of them to be selected.

Procedure

1. Create or edit a YAML file that defines the **ClusterLogForwarder** CR object. In the file, specify the pod labels using simple equality-based selectors under **inputs[].name.application.selector.matchLabels**, as shown in the following example.

Example ClusterLogForwarder CR YAML file

```

apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance ❶
  namespace: openshift-logging ❷
spec:
  pipelines:
    - inputRefs: [ myAppLogData ] ❸
      outputRefs: [ default ] ❹
      parse: json ❺
  inputs: ❻
    - name: myAppLogData
      application:
        selector:
          matchLabels: ❼
            environment: production
            app: nginx
          namespaces: ❽
            - app1
            - app2
  outputs: ❾
    - default
    ...

```

- ❶ The name of the **ClusterLogForwarder** CR must be **instance**.
- ❷ The namespace for the **ClusterLogForwarder** CR must be **openshift-logging**.
- ❸ Specify one or more comma-separated values from **inputs[].name**.
- ❹ Specify one or more comma-separated values from **outputs[]**.
- ❺ Optional: Specify whether to forward structured JSON log entries as JSON objects in the **structured** field. The log entry must contain valid structured JSON; otherwise, OpenShift Logging removes the **structured** field and instead sends the log entry to the default index, **app-00000x**.
- ❻ Define a unique **inputs[].name** for each application that has a unique set of pod labels.
- ❼ Specify the key-value pairs of pod labels whose log data you want to gather. You must specify both a key and value, not just a key. To be selected, the pods must match all the key-value pairs.
- ❽ Optional: Specify one or more namespaces.
- ❾ Specify one or more outputs to forward your log data to. The optional **default** output shown here sends log data to the internal Elasticsearch instance.

2. Optional: To restrict the gathering of log data to specific namespaces, use **inputs[].name.application.namespaces**, as shown in the preceding example.
3. Optional: You can send log data from additional applications that have different pod labels to the same pipeline.
 - a. For each unique combination of pod labels, create an additional **inputs[].name** section similar to the one shown.
 - b. Update the **selectors** to match the pod labels of this application.
 - c. Add the new **inputs[].name** value to **inputRefs**. For example:

```
- inputRefs: [ myAppLogData, myOtherAppLogData ]
```

4. Create the CR object:

```
$ oc create -f <file-name>.yaml
```

Additional resources

- For more information on **matchLabels** in Kubernetes, see [Resources that support set-based requirements](#).

Additional resources

- [Logging for egress firewall and network policy rules](#)

12.1.14. Troubleshooting log forwarding

When you create a **ClusterLogForwarder** custom resource (CR), if the Red Hat OpenShift Logging Operator does not redeploy the Fluentd pods automatically, you can delete the Fluentd pods to force them to redeploy.

Prerequisites

- You have created a **ClusterLogForwarder** custom resource (CR) object.

Procedure

- Delete the Fluentd pods to force them to redeploy.

```
$ oc delete pod --selector logging-infra=collector
```

12.2. LOG OUTPUT TYPES

Log outputs specified in the **ClusterLogForwarder** CR can be any of the following types:

default

The on-cluster, Red Hat managed log store. You are not required to configure the default output.

**NOTE**

If you configure a **default** output, you receive an error message, because the **default** output name is reserved for referencing the on-cluster, Red Hat managed log store.

loki

Loki, a horizontally scalable, highly available, multi-tenant log aggregation system.

kafka

A Kafka broker. The **kafka** output can use a TCP or TLS connection.

elasticsearch

An external Elasticsearch instance. The **elasticsearch** output can use a TLS connection.

fluentdForward

An external log aggregation solution that supports Fluentd. This option uses the Fluentd **forward** protocols. The **fluentForward** output can use a TCP or TLS connection and supports shared-key authentication by providing a **shared_key** field in a secret. Shared-key authentication can be used with or without TLS.

**IMPORTANT**

The **fluentdForward** output is only supported if you are using the Fluentd collector. It is not supported if you are using the Vector collector. If you are using the Vector collector, you can forward logs to Fluentd by using the **http** output.

syslog

An external log aggregation solution that supports the syslog [RFC3164](#) or [RFC5424](#) protocols. The **syslog** output can use a UDP, TCP, or TLS connection.

cloudwatch

Amazon CloudWatch, a monitoring and log storage service hosted by Amazon Web Services (AWS).

12.2.1. Supported log data output types in OpenShift Logging 5.7

Red Hat OpenShift Logging 5.7 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range of target log collectors that ingest these protocols.

Table 12.1. Logging 5.7 outputs

Output	Protocol	Tested with	Fluentd	Vector
Cloudwatch	REST over HTTP(S)		✓	✓
Elasticsearch v6		v6.8.1	✓	✓
Elasticsearch v7		v7.12.2, 7.17.7	✓	✓

Output	Protocol	Tested with	Fluentd	Vector
Elasticsearch v8		v8.4.3	✓	✓
Fluent Forward	Fluentd forward v1	Fluentd 1.14.6, Logstash 7.10.1	✓	
Google Cloud Logging				✓
HTTP	HTTP 1.1	Fluentd 1.14.6, Vector 0.21	✓	✓
Kafka	Kafka 0.11	Kafka 2.4.1, 2.7.0, 3.3.1	✓	✓
Loki	REST over HTTP(S)	Loki 2.3.0, 2.7	✓	✓
Splunk	HEC	v8.2.9, 9.0.0		✓
Syslog	RFC3164, RFC5424	Rsyslog 8.37.0- 9.e17	✓	✓

12.2.2. Supported log data output types in OpenShift Logging 5.6

Red Hat OpenShift Logging 5.6 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch
elasticsearch	elasticsearch	Elasticsearch 6.8.23 Elasticsearch 7.10.1 Elasticsearch 8.6.1
fluentdForward	fluentd forward v1	fluentd 1.14.6 logstash 7.10.1
Loki	REST over HTTP and HTTPS	Loki 2.5.0 deployed on OCP

Output types	Protocols	Tested with
kafka	kafka 0.11	kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0



IMPORTANT

Fluentd doesn't support Elasticsearch 8 as of 5.6.2. Vector doesn't support fluentd/logstash/rsyslog before 5.7.0.

12.2.3. Supported log data output types in OpenShift Logging 5.5

Red Hat OpenShift Logging 5.5 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch
elasticsearch	elasticsearch	Elasticsearch 7.10.1
fluentdForward	fluentd forward v1	fluentd 1.14.6 logstash 7.10.1
Loki	REST over HTTP and HTTPS	Loki 2.5.0 deployed on OCP
kafka	kafka 0.11	kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

12.2.4. Supported log data output types in OpenShift Logging 5.4

Red Hat OpenShift Logging 5.4 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch

Output types	Protocols	Tested with
elasticsearch	elasticsearch	Elasticsearch 7.10.1
fluentdForward	fluentd forward v1	fluentd 1.14.5 logstash 7.10.1
Loki	REST over HTTP and HTTPS	Loki 2.2.1 deployed on OCP
kafka	kafka 0.11	kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

12.2.5. Supported log data output types in OpenShift Logging 5.3

Red Hat OpenShift Logging 5.3 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch
elasticsearch	elasticsearch	Elasticsearch 7.10.1
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
Loki	REST over HTTP and HTTPS	Loki 2.2.1 deployed on OCP
kafka	kafka 0.11	kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

12.2.6. Supported log data output types in OpenShift Logging 5.2

Red Hat OpenShift Logging 5.2 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
Amazon CloudWatch	REST over HTTPS	The current version of Amazon CloudWatch
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 6.8.4 Elasticsearch 7.12.2
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
Loki	REST over HTTP and HTTPS	Loki 2.3.0 deployed on OCP and Grafana labs
kafka	kafka 0.11	kafka 2.4.1 kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

12.2.7. Supported log data output types in OpenShift Logging 5.1

Red Hat OpenShift Logging 5.1 provides the following output types and protocols for sending log data to target log collectors.

Red Hat tests each of the combinations shown in the following table. However, you should be able to send log data to a wider range target log collectors that ingest these protocols.

Output types	Protocols	Tested with
elasticsearch	elasticsearch	Elasticsearch 6.8.1 Elasticsearch 6.8.4 Elasticsearch 7.12.2
fluentdForward	fluentd forward v1	fluentd 1.7.4 logstash 7.10.1
kafka	kafka 0.11	kafka 2.4.1 kafka 2.7.0
syslog	RFC-3164, RFC-5424	rsyslog-8.39.0

**NOTE**

Previously, the syslog output supported only RFC-3164. The current syslog output adds support for RFC-5424.

12.3. ENABLING JSON LOG FORWARDING

You can configure the Log Forwarding API to parse JSON strings into a structured object.

12.3.1. Parsing JSON logs

Logs including JSON logs are usually represented as a string inside the **message** field. That makes it hard for users to query specific fields inside a JSON document. OpenShift Logging's Log Forwarding API enables you to parse JSON logs into a structured object and forward them to either OpenShift Logging-managed Elasticsearch or any other third-party system supported by the Log Forwarding API.

To illustrate how this works, suppose that you have the following structured JSON log entry.

Example structured JSON log entry

```
{"level":"info","name":"fred","home":"bedrock"}
```

Normally, the **ClusterLogForwarder** custom resource (CR) forwards that log entry in the **message** field. The **message** field contains the JSON-quoted string equivalent of the JSON log entry, as shown in the following example.

Example message field

```
{"message":"'{"level":"info","name":"fred","home":"bedrock"}',  
  "more fields..."}
```

To enable parsing JSON log, you add **parse: json** to a pipeline in the **ClusterLogForwarder** CR, as shown in the following example.

Example snippet showing parse: json

```
pipelines:  
- inputRefs: [ application ]  
  outputRefs: myFluentd  
  parse: json
```

When you enable parsing JSON logs by using **parse: json**, the CR copies the JSON-structured log entry in a **structured** field, as shown in the following example. This does not modify the original **message** field.

Example structured output containing the structured JSON log entry

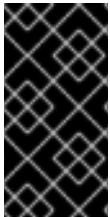
```
{"structured": { "level": "info", "name": "fred", "home": "bedrock" },  
  "more fields..."}
```


**IMPORTANT**

If the log entry does not contain valid structured JSON, the **structured** field will be absent.

12.3.2. Configuring JSON log data for Elasticsearch

If your JSON logs follow more than one schema, storing them in a single index might cause type conflicts and cardinality problems. To avoid that, you must configure the **ClusterLogForwarder** custom resource (CR) to group each schema into a single output definition. This way, each schema is forwarded to a separate index.

**IMPORTANT**

If you forward JSON logs to the default Elasticsearch instance managed by OpenShift Logging, it generates new indices based on your configuration. To avoid performance issues associated with having too many indices, consider keeping the number of possible schemas low by standardizing to common schemas.

Structure types

You can use the following structure types in the **ClusterLogForwarder** CR to construct index names for the Elasticsearch log store:

- **structuredTypeKey** (string, optional) is the name of a message field. The value of that field, if present, is used to construct the index name.
 - **kubernetes.labels.<key>** is the Kubernetes pod label whose value is used to construct the index name.
 - **openshift.labels.<key>** is the **pipeline.label.<key>** element in the **ClusterLogForwarder** CR whose value is used to construct the index name.
 - **kubernetes.container_name** uses the container name to construct the index name.
- **structuredTypeName**: (string, optional) If **structuredTypeKey** is not set or its key is not present, OpenShift Logging uses the value of **structuredTypeName** as the structured type. When you use both **structuredTypeKey** and **structuredTypeName** together, **structuredTypeName** provides a fallback index name if the key in **structuredTypeKey** is missing from the JSON log data.

**NOTE**

Although you can set the value of **structuredTypeKey** to any field shown in the "Log Record Fields" topic, the most useful fields are shown in the preceding list of structure types.

A structuredTypeKey: kubernetes.labels.<key> example

Suppose the following:

- Your cluster is running application pods that produce JSON logs in two different formats, "apache" and "google".
- The user labels these application pods with **logFormat=apache** and **logFormat=google**.

- You use the following snippet in your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
  elasticsearch:
    structuredTypeKey: kubernetes.labels.logFormat ❶
    structuredTypeName: nologformat
  pipelines:
    - inputRefs: <application>
      outputRefs: default
      parse: json ❷
```

❶ Uses the value of the key-value pair that is formed by the Kubernetes **logFormat** label.

❷ Enables parsing JSON logs.

In that case, the following structured log record goes to the **app-apache-write** index:

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "apache", ...}}
}
```

And the following structured log record goes to the **app-google-write** index:

```
{
  "structured":{"name":"wilma","home":"bedrock"},
  "kubernetes":{"labels":{"logFormat": "google", ...}}
}
```

A structuredTypeKey: openshift.labels.<key> example

Suppose that you use the following snippet in your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
  elasticsearch:
    structuredTypeKey: openshift.labels.myLabel ❶
    structuredTypeName: nologformat
  pipelines:
    - name: application-logs
      inputRefs:
        - application
        - audit
      outputRefs:
        - elasticsearch-secure
        - default
      parse: json
      labels:
        myLabel: myValue ❷
```

❶ Uses the value of the key-value pair that is formed by the OpenShift **myLabel** label.

❷ The **myLabel** element gives its string value, **myValue**, to the structured log record.

In that case, the following structured log record goes to the **app-myValue-write** index:

```
{
  "structured":{"name":"fred","home":"bedrock"},
  "openshift":{"labels":{"myLabel": "myValue", ...}}
}
```

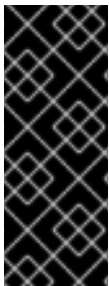
Additional considerations

- The Elasticsearch *index* for structured records is formed by prepending "app-" to the structured type and appending "-write".
- Unstructured records are not sent to the structured index. They are indexed as usual in the application, infrastructure, or audit indices.
- If there is no non-empty structured type, forward an *unstructured* record with no **structured** field.

It is important not to overload Elasticsearch with too many indices. Only use distinct structured types for distinct log *formats*, **not** for each application or namespace. For example, most Apache applications use the same JSON log format and structured type, such as **LogApache**.

12.3.3. Forwarding JSON logs to the Elasticsearch log store

For an Elasticsearch log store, if your JSON log entries *follow different schemas*, configure the **ClusterLogForwarder** custom resource (CR) to group each JSON schema into a single output definition. This way, Elasticsearch uses a separate index for each schema.



IMPORTANT

Because forwarding different schemas to the same index can cause type conflicts and cardinality problems, you must perform this configuration before you forward data to the Elasticsearch store.

To avoid performance issues associated with having too many indices, consider keeping the number of possible schemas low by standardizing to common schemas.

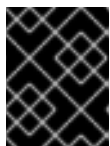
Procedure

1. Add the following snippet to your **ClusterLogForwarder** CR YAML file.

```
outputDefaults:
  elasticsearch:
    structuredTypeKey: <log record field>
    structuredTypeName: <name>
  pipelines:
  - inputRefs:
    - application
    outputRefs: default
    parse: json
```

2. Optional: Use **structuredTypeKey** to specify one of the log record fields, as described in the documentation about "Configuring JSON log data for Elasticsearch". Otherwise, remove this line.

- Optional: Use **structuredTypeName** to specify a **<name>**, as described in the documentation about "Configuring JSON log data for Elasticsearch". Otherwise, remove this line.



IMPORTANT

To parse JSON logs, you must set either **structuredTypeKey** or **structuredTypeName**, or both **structuredTypeKey** and **structuredTypeName**.

- For **inputRefs**, specify which log types to forward by using that pipeline, such as **application**, **infrastructure**, or **audit**.
- Add the **parse: json** element to pipelines.
- Create the CR object:

```
$ oc create -f <filename>.yaml
```

The Red Hat OpenShift Logging Operator redeploys the Fluentd pods. However, if they do not redeploy, delete the Fluentd pods to force them to redeploy.

```
$ oc delete pod --selector logging-infra=collector
```

Additional resources

- [About log forwarding](#)

12.4. CONFIGURING THE LOGGING COLLECTOR

Logging subsystem for Red Hat OpenShift collects operations and application logs from your cluster and enriches the data with Kubernetes pod and project metadata.

You can configure the CPU and memory limits for the log collector and [move the log collector pods to specific nodes](#). All supported modifications to the log collector can be performed through the **spec.collection.log.fluentd** stanza in the **ClusterLogging** custom resource (CR).

12.4.1. Viewing logging collector pods

You can view the Fluentd logging collector pods and the corresponding nodes that they are running on. The Fluentd logging collector pods run only in the **openshift-logging** project.

Procedure

- Run the following command in the **openshift-logging** project to view the Fluentd logging collector pods and their details:

```
$ oc get pods --selector component=collector -o wide -n openshift-logging
```

Example output

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED
fluentd-8d69v	1/1	Running	0	134m	10.130.2.30	master1.example.com	<none>

```

<none>
fluentd-bd225 1/1 Running 0      134m 10.131.1.11 master2.example.com <none>
<none>
fluentd-cvrzs 1/1 Running 0      134m 10.130.0.21 master3.example.com <none>
<none>
fluentd-gpqg2 1/1 Running 0      134m 10.128.2.27 worker1.example.com <none>
<none>
fluentd-l9j7j 1/1 Running 0      134m 10.129.2.31 worker2.example.com <none>
<none>

```

12.4.2. Configure log collector CPU and memory limits

The log collector allows for adjustments to both the CPU and memory limits.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc -n openshift-logging edit ClusterLogging instance
```

```

apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: openshift-logging
...
spec:
  collection:
    logs:
      fluentd:
        resources:
          limits: 1
          memory: 736Mi
        requests:
          cpu: 100m
          memory: 736Mi

```

- 1 Specify the CPU and memory limits and requests as needed. The values shown are the default values.

12.4.3. Advanced configuration for the Fluentd log forwarder

The logging subsystem for Red Hat OpenShift includes multiple Fluentd parameters that you can use for tuning the performance of the Fluentd log forwarder. With these parameters, you can change the following Fluentd behaviors:

- Chunk and chunk buffer sizes
- Chunk flushing behavior
- Chunk forwarding retry behavior

Fluentd collects log data in a single blob called a *chunk*. When Fluentd creates a chunk, the chunk is considered to be in the *stage*, where the chunk gets filled with data. When the chunk is full, Fluentd moves the chunk to the *queue*, where chunks are held before being flushed, or written out to their destination. Fluentd can fail to flush a chunk for a number of reasons, such as network issues or capacity issues at the destination. If a chunk cannot be flushed, Fluentd retries flushing as configured.

By default in OpenShift Container Platform, Fluentd uses the *exponential backoff* method to retry flushing, where Fluentd doubles the time it waits between attempts to retry flushing again, which helps reduce connection requests to the destination. You can disable exponential backoff and use the *periodic* retry method instead, which retries flushing the chunks at a specified interval.

These parameters can help you determine the trade-offs between latency and throughput.

- To optimize Fluentd for throughput, you could use these parameters to reduce network packet count by configuring larger buffers and queues, delaying flushes, and setting longer times between retries. Be aware that larger buffers require more space on the node file system.
- To optimize for low latency, you could use the parameters to send data as soon as possible, avoid the build-up of batches, have shorter queues and buffers, and use more frequent flush and retries.

You can configure the chunking and flushing behavior using the following parameters in the **ClusterLogging** custom resource (CR). The parameters are then automatically added to the Fluentd config map for use by Fluentd.



NOTE

These parameters are:

- Not relevant to most users. The default settings should give good general performance.
- Only for advanced users with detailed knowledge of Fluentd configuration and performance.
- Only for performance tuning. They have no effect on functional aspects of logging.

Table 12.2. Advanced Fluentd Configuration Parameters

Parameter	Description	Default
chunkLimitSize	The maximum size of each chunk. Fluentd stops writing data to a chunk when it reaches this size. Then, Fluentd sends the chunk to the queue and opens a new chunk.	8m

Parameter	Description	Default
totalLimitSize	The maximum size of the buffer, which is the total size of the stage and the queue. If the buffer size exceeds this value, Fluentd stops adding data to chunks and fails with an error. All data not in chunks is lost.	8G
flushInterval	The interval between chunk flushes. You can use s (seconds), m (minutes), h (hours), or d (days).	1s
flushMode	The method to perform flushes: <ul style="list-style-type: none"> ● lazy: Flush chunks based on the timekey parameter. You cannot modify the timekey parameter. ● interval: Flush chunks based on the flushInterval parameter. ● immediate: Flush chunks immediately after data is added to a chunk. 	interval
flushThreadCount	The number of threads that perform chunk flushing. Increasing the number of threads improves the flush throughput, which hides network latency.	2
overflowAction	The chunking behavior when the queue is full: <ul style="list-style-type: none"> ● throw_exception: Raise an exception to show in the log. ● block: Stop data chunking until the full buffer issue is resolved. ● drop_oldest_chunk: Drop the oldest chunk to accept new incoming chunks. Older chunks have less value than newer chunks. 	block

Parameter	Description	Default
retryMaxInterval	The maximum time in seconds for the exponential_backoff retry method.	300s
retryType	<p>The retry method when flushing fails:</p> <ul style="list-style-type: none"> • exponential_backoff: Increase the time between flush retries. Fluentd doubles the time it waits until the next retry until the retry_max_interval parameter is reached. • periodic: Retries flushes periodically, based on the retryWait parameter. 	exponential_backoff
retryTimeOut	The maximum time interval to attempt retries before the record is discarded.	60m
retryWait	The time in seconds before the next chunk flush.	1s

For more information on the Fluentd chunk lifecycle, see [Buffer Plugins](#) in the Fluentd documentation.

Procedure

1. Edit the **ClusterLogging** custom resource (CR) in the **openshift-logging** project:

```
$ oc edit ClusterLogging instance
```

2. Add or modify any of the following parameters:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  forwarder:
    fluentd:
      buffer:
        chunkLimitSize: 8m 1
        flushInterval: 5s 2
        flushMode: interval 3
        flushThreadCount: 3 4
```



```

overflowAction: throw_exception 5
retryMaxInterval: "300s" 6
retryType: periodic 7
retryWait: 1s 8
totalLimitSize: 32m 9

```

...

- 1 Specify the maximum size of each chunk before it is queued for flushing.
- 2 Specify the interval between chunk flushes.
- 3 Specify the method to perform chunk flushes: **lazy**, **interval**, or **immediate**.
- 4 Specify the number of threads to use for chunk flushes.
- 5 Specify the chunking behavior when the queue is full: **throw_exception**, **block**, or **drop_oldest_chunk**.
- 6 Specify the maximum interval in seconds for the **exponential_backoff** chunk flushing method.
- 7 Specify the retry type when chunk flushing fails: **exponential_backoff** or **periodic**.
- 8 Specify the time in seconds before the next chunk flush.
- 9 Specify the maximum size of the chunk buffer.

3. Verify that the Fluentd pods are redeployed:

```
$ oc get pods -l component=collector -n openshift-logging
```

4. Check that the new values are in the **fluentd** config map:

```
$ oc extract configmap/fluentd --confirm
```

Example fluentd.conf

```

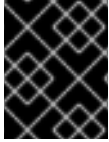
<buffer>
  @type file
  path '/var/lib/fluentd/default'
  flush_mode interval
  flush_interval 5s
  flush_thread_count 3
  retry_type periodic
  retry_wait 1s
  retry_max_interval 300s
  retry_timeout 60m
  queued_chunks_limit_size "#{ENV['BUFFER_QUEUE_LIMIT'] || '32'}"
  total_limit_size 32m
  chunk_limit_size 8m
  overflow_action throw_exception
</buffer>

```

12.5. COLLECTING AND STORING KUBERNETES EVENTS

The OpenShift Container Platform Event Router is a pod that watches Kubernetes events and logs them for collection by the logging subsystem. You must manually deploy the Event Router.

The Event Router collects events from all projects and writes them to **STDOUT**. The collector then forwards those events to the store defined in the **ClusterLogForwarder** custom resource (CR).



IMPORTANT

The Event Router adds additional load to Fluentd and can impact the number of other log messages that can be processed.

12.5.1. Deploying and configuring the Event Router

Use the following steps to deploy the Event Router into your cluster. You should always deploy the Event Router to the **openshift-logging** project to ensure it collects events from across the cluster.

The following Template object creates the service account, cluster role, and cluster role binding required for the Event Router. The template also configures and deploys the Event Router pod. You can use this template without making changes, or change the deployment object CPU and memory requests.

Prerequisites

- You need proper permissions to create service accounts and update cluster role bindings. For example, you can run the following template with a user that has the **cluster-admin** role.
- The logging subsystem for Red Hat OpenShift must be installed.

Procedure

1. Create a template for the Event Router:

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: eventrouter-template
  annotations:
    description: "A pod forwarding kubernetes events to OpenShift Logging stack."
    tags: "events,EFK,logging,cluster-logging"
objects:
  - kind: ServiceAccount 1
    apiVersion: v1
    metadata:
      name: eventrouter
      namespace: ${NAMESPACE}
  - kind: ClusterRole 2
    apiVersion: rbac.authorization.k8s.io/v1
    metadata:
      name: event-reader
    rules:
      - apiGroups: [""]
        resources: ["events"]
        verbs: ["get", "watch", "list"]
```

```

- kind: ClusterRoleBinding 3
  apiVersion: rbac.authorization.k8s.io/v1
  metadata:
    name: event-reader-binding
  subjects:
  - kind: ServiceAccount
    name: eventrouter
    namespace: ${NAMESPACE}
  roleRef:
    kind: ClusterRole
    name: event-reader
- kind: ConfigMap 4
  apiVersion: v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  data:
    config.json: |-
      {
        "sink": "stdout"
      }
- kind: Deployment 5
  apiVersion: apps/v1
  metadata:
    name: eventrouter
    namespace: ${NAMESPACE}
  labels:
    component: "eventrouter"
    logging-infra: "eventrouter"
    provider: "openshift"
  spec:
    selector:
      matchLabels:
        component: "eventrouter"
        logging-infra: "eventrouter"
        provider: "openshift"
    replicas: 1
    template:
      metadata:
        labels:
          component: "eventrouter"
          logging-infra: "eventrouter"
          provider: "openshift"
        name: eventrouter
      spec:
        serviceAccount: eventrouter
        containers:
        - name: kube-eventrouter
          image: ${IMAGE}
          imagePullPolicy: IfNotPresent
          resources:
            requests:
              cpu: ${CPU}
              memory: ${MEMORY}
          volumeMounts:
          - name: config-volume

```

```

        mountPath: /etc/eventrouter
      volumes:
      - name: config-volume
        configMap:
          name: eventrouter
    parameters:
    - name: IMAGE 6
      displayName: Image
      value: "registry.redhat.io/openshift-logging/eventrouter-rhel8:v0.4"
    - name: CPU 7
      displayName: CPU
      value: "100m"
    - name: MEMORY 8
      displayName: Memory
      value: "128Mi"
    - name: NAMESPACE
      displayName: Namespace
      value: "openshift-logging" 9

```

- 1** Creates a Service Account in the **openshift-logging** project for the Event Router.
- 2** Creates a ClusterRole to monitor for events in the cluster.
- 3** Creates a ClusterRoleBinding to bind the ClusterRole to the service account.
- 4** Creates a config map in the **openshift-logging** project to generate the required **config.json** file.
- 5** Creates a deployment in the **openshift-logging** project to generate and configure the Event Router pod.
- 6** Specifies the image, identified by a tag such as **v0.4**.
- 7** Specifies the minimum amount of CPU to allocate to the Event Router pod. Defaults to **100m**.
- 8** Specifies the minimum amount of memory to allocate to the Event Router pod. Defaults to **128Mi**.
- 9** Specifies the **openshift-logging** project to install objects in.

2. Use the following command to process and apply the template:

```
$ oc process -f <templatefile> | oc apply -n openshift-logging -f -
```

For example:

```
$ oc process -f eventrouter.yaml | oc apply -n openshift-logging -f -
```

Example output

```

serviceaccount/eventrouter created
clusterrole.authorization.openshift.io/event-reader created
clusterrolebinding.authorization.openshift.io/event-reader-binding created

```

```
configmap/eventrouter created
deployment.apps/eventrouter created
```

3. Validate that the Event Router installed in the **openshift-logging** project:

- a. View the new Event Router pod:

```
$ oc get pods --selector component=eventrouter -o name -n openshift-logging
```

Example output

```
pod/cluster-logging-eventrouter-d649f97c8-qvv8r
```

- b. View the events collected by the Event Router:

```
$ oc logs <cluster_logging_eventrouter_pod> -n openshift-logging
```

For example:

```
$ oc logs cluster-logging-eventrouter-d649f97c8-qvv8r -n openshift-logging
```

Example output

```
{"verb":"ADDED","event":{"metadata":{"name":"openshift-service-catalog-controller-
manager-remover.1632d931e88fcd8f","namespace":"openshift-service-catalog-
removed","selfLink":"/api/v1/namespaces/openshift-service-catalog-
removed/events/openshift-service-catalog-controller-manager-
remover.1632d931e88fcd8f","uid":"787d7b26-3d2f-4017-b0b0-
420db4ae62c0","resourceVersion":"21399","creationTimestamp":"2020-09-
08T15:40:26Z"},"involvedObject":{"kind":"Job","namespace":"openshift-service-catalog-
removed","name":"openshift-service-catalog-controller-manager-
remover","uid":"fac9f479-4ad5-4a57-8adc-
cb25d3d9cf8f","apiVersion":"batch/v1","resourceVersion":"21280"},"reason":"Completed","
message":"Job completed","source":{"component":"job-
controller"},"firstTimestamp":"2020-09-08T15:40:26Z","lastTimestamp":"2020-09-
08T15:40:26Z","count":1,"type":"Normal"}}
```

You can also use Kibana to view events by creating an index pattern using the Elasticsearch **infra** index.

CHAPTER 13. UPDATING OPENSIFT LOGGING

13.1. SUPPORTED VERSIONS

For version compatibility and support information, see [Red Hat OpenShift Container Platform Life Cycle Policy](#)

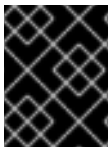
To upgrade from cluster logging in OpenShift Container Platform version 4.6 and earlier to OpenShift Logging 5.x, you update the OpenShift Container Platform cluster to version 4.7 or 4.8. Then, you update the following operators:

- From Elasticsearch Operator 4.x to OpenShift Elasticsearch Operator 5.x
- From Cluster Logging Operator 4.x to Red Hat OpenShift Logging Operator 5.x

To upgrade from a previous version of OpenShift Logging to the current version, you update OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator to their current versions.

13.2. UPDATING LOGGING TO THE CURRENT VERSION

To update Logging to the current version, you change the subscriptions for the OpenShift Elasticsearch Operator and Red Hat OpenShift Logging Operator.



IMPORTANT

You must update the OpenShift Elasticsearch Operator *before* you update the Red Hat OpenShift Logging Operator. You must also update *both* Operators to the same version.

If you update the Operators in the wrong order, Kibana does not update and the Kibana custom resource (CR) is not created. To work around this problem, you delete the Red Hat OpenShift Logging Operator pod. When the Red Hat OpenShift Logging Operator pod redeploys, it creates the Kibana CR and Kibana becomes available again.

Prerequisites

- The OpenShift Container Platform version is 4.7 or later.
- The Logging status is healthy:
 - All pods are **ready**.
 - The Elasticsearch cluster is healthy.
- Your [Elasticsearch and Kibana data is backed up](#).

Procedure

1. Update the OpenShift Elasticsearch Operator:
 - a. In the OpenShift Container Platform web console, click **Operators** → **Installed Operators**.
 - b. Select the **openshift-operators-redhat** project.
 - c. Click the **OpenShift Elasticsearch Operator**.

- d. Click **Subscription → Channel**.
 - e. In the **Change Subscription Update Channel** window, select **stable-5.x** and click **Save**.
 - f. Wait for a few seconds, then click **Operators → Installed Operators**.
 - g. Verify that the OpenShift Elasticsearch Operator version is 5.x.x.
 - h. Wait for the **Status** field to report **Succeeded**.
2. Update the Red Hat OpenShift Logging Operator:
 - a. In the OpenShift Container Platform web console, click **Operators → Installed Operators**.
 - b. Select the **openshift-logging** project.
 - c. Click the **Red Hat OpenShift Logging Operator**.
 - d. Click **Subscription → Channel**.
 - e. In the **Change Subscription Update Channel** window, select **stable-5.x** and click **Save**.
 - f. Wait for a few seconds, then click **Operators → Installed Operators**.
 - g. Verify that the Red Hat OpenShift Logging Operator version is 5.y.z
 - h. Wait for the **Status** field to report **Succeeded**.
 3. Check the logging components:
 - a. Ensure that all Elasticsearch pods are in the **Ready** status:

```
$ oc get pod -n openshift-logging --selector component=elasticsearch
```

Example output

NAME	READY	STATUS	RESTARTS	AGE
elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk	2/2	Running	0	31m
elasticsearch-cdm-1pbrl44l-2-5c6d87589f-gx5hk	2/2	Running	0	30m
elasticsearch-cdm-1pbrl44l-3-88df5d47-m45jc	2/2	Running	0	29m

- b. Ensure that the Elasticsearch cluster is healthy:

```
$ oc exec -n openshift-logging -c elasticsearch elasticsearch-cdm-1pbrl44l-1-55b7546f4c-mshhk -- health
```

```
{
  "cluster_name" : "elasticsearch",
  "status" : "green",
}
```

- c. Ensure that the Elasticsearch cron jobs are created:

```
$ oc project openshift-logging
```

```
$ oc get cronjob
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
elasticsearch-im-app	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-audit	*/15 * * * *	False	0	<none>	56s
elasticsearch-im-infra	*/15 * * * *	False	0	<none>	56s

- d. Verify that the log store is updated to 5.x and the indices are **green**:

```
$ oc exec -c elasticsearch <any_es_pod_in_the_cluster> -- indices
```

- e. Verify that the output includes the **app-00000x**, **infra-00000x**, **audit-00000x**, **.security** indices.

Example 13.1. Sample output with indices in a green status

```
Tue Jun 30 14:30:54 UTC 2020
health status index                                uuid                                pri rep
docs.count docs.deleted store.size pri.store.size
green open  infra-000008
bnBvUFEXTWi92z3zWAZieQ 3 1      222195      0      289      144
green open  infra-000004
rtDSzoqsSl6saisSK7Au1Q 3 1      226717      0      297      148
green open  infra-000012
RSf_kUwDSR2xEuKRZMPqZQ 3 1      227623      0      295      147
green open  .kibana_7
1SJdCqIZTPWIIAaOUd78yg 1 1        4        0        0        0
green open  infra-000010
iXwL3bnqTuGEABbUDa6OVw 3 1      248368      0      317      158
green open  infra-000009
YN9EsULWSNaxWeeNvOs0RA 3 1      258799      0      337      168
green open  infra-000014
YP0U6R7FQ_GVQVQZ6Yh9lg 3 1      223788      0      292      146
green open  infra-000015
JRBbAbEmSMqK5X40df9HbQ 3 1      224371      0      291      145
green open  .orphaned.2020.06.30
n_xQC2dWQzConkvQqei3YA 3 1        9        0        0        0
green open  infra-000007
llkAVSszSOMosWTSAJM_hg 3 1      228584      0      296      148
green open  infra-000005
d9BoGQdiQASsS3BBFm2iRA 3 1      227987      0      297      148
green open  infra-000003
goREK1QUKIQPAIVkWWaQ 3 1      226719      0      295      147
green open  .security
zeT65uOuRTKZMjg_bbUc1g 1 1        5        0        0        0
green open  .kibana-377444158_kubeadmin      wwMhDwJkR-
mRZQO84K0gUQ 3 1        1        0        0
green open  infra-000006
KBSXGQKiO7hdapDE23g 3 1      226676      0      295      147
green open  infra-000001
bSxSWR5xYZB6IVg 3 1      341800      0      443      220
green open  .kibana-6
RVp7TemSSemGJcsSUmf3A 1 1        4        0        0        0
green open  infra-000011
J7XWBauWSTe0jnzX02fU6A 3 1      226100      0      293      146
```



```

green open app-000001
axSAfONQDmKwatkjPXdtw 3 1 103186 0 126 57
green open infra-000016
m9c1iRLtStWSF1GopaRyCg 3 1 13685 0 19 9
green open infra-000002 Hz6WvINtTvKcQzw-
ewmbYg 3 1 228994 0 296 148
green open infra-000013 KR9mMFUpQI-
jraYtanyIGw 3 1 228166 0 298 148
green open audit-000001
eERqLdLmQOiQDFES1LBATQ 3 1 0 0 0 0

```

- f. Verify that the log collector is updated:

```
$ oc get ds collector -o json | grep collector
```

- g. Verify that the output includes a **collectort** container:

```
"containerName": "collector"
```

- h. Verify that the log visualizer is updated to 5.x using the Kibana CRD:

```
$ oc get kibana kibana -o json
```

- i. Verify that the output includes a Kibana pod with the **ready** status:

Example 13.2. Sample output with a ready Kibana pod

```

[
  {
    "clusterCondition": {
      "kibana-5fdd766ffd-nb2jj": [
        {
          "lastTransitionTime": "2020-06-30T14:11:07Z",
          "reason": "ContainerCreating",
          "status": "True",
          "type": ""
        },
        {
          "lastTransitionTime": "2020-06-30T14:11:07Z",
          "reason": "ContainerCreating",
          "status": "True",
          "type": ""
        }
      ]
    },
    "deployment": "kibana",
    "pods": {
      "failed": [],
      "notReady": [],
      "ready": []
    },
    "replicaSets": [
      "kibana-5fdd766ffd"
    ]
  }
]

```

```
    "replicas": 1
  }
]
```

CHAPTER 14. VIEWING CLUSTER DASHBOARDS

The **Logging/Elasticsearch Nodes** and **OpenShift Logging** dashboards in the OpenShift Container Platform web console contain in-depth details about your Elasticsearch instance and the individual Elasticsearch nodes that you can use to prevent and diagnose problems.

The **OpenShift Logging** dashboard contains charts that show details about your Elasticsearch instance at a cluster level, including cluster resources, garbage collection, shards in the cluster, and Fluentd statistics.

The **Logging/Elasticsearch Nodes** dashboard contains charts that show details about your Elasticsearch instance, many at node level, including details on indexing, shards, resources, and so forth.

14.1. ACCESSING THE ELASTICSEARCH AND OPENSIFT LOGGING DASHBOARDS

You can view the **Logging/Elasticsearch Nodes** and **OpenShift Logging** dashboards in the OpenShift Container Platform web console.

Procedure

To launch the dashboards:

1. In the OpenShift Container Platform web console, click **Observe → Dashboards**.
2. On the **Dashboards** page, select **Logging/Elasticsearch Nodes** or **OpenShift Logging** from the **Dashboard** menu.
For the **Logging/Elasticsearch Nodes** dashboard, you can select the Elasticsearch node you want to view and set the data resolution.

The appropriate dashboard is displayed, showing multiple charts of data.

3. Optional: Select a different time range to display or refresh rate for the data from the **Time Range** and **Refresh Interval** menus.

For information on the dashboard charts, see [About the OpenShift Logging dashboard](#) and [About the Logging/Elasticsearch Nodes dashboard](#).

14.2. ABOUT THE OPENSIFT LOGGING DASHBOARD

The **OpenShift Logging** dashboard contains charts that show details about your Elasticsearch instance at a cluster-level that you can use to diagnose and anticipate problems.

Table 14.1. OpenShift Logging charts

Metric	Description
Elastic Cluster Status	<p>The current Elasticsearch status:</p> <ul style="list-style-type: none"> ● ONLINE - Indicates that the Elasticsearch instance is online. ● OFFLINE - Indicates that the Elasticsearch instance is offline.

Metric	Description
Elastic Nodes	The total number of Elasticsearch nodes in the Elasticsearch instance.
Elastic Shards	The total number of Elasticsearch shards in the Elasticsearch instance.
Elastic Documents	The total number of Elasticsearch documents in the Elasticsearch instance.
Total Index Size on Disk	The total disk space that is being used for the Elasticsearch indices.
Elastic Pending Tasks	The total number of Elasticsearch changes that have not been completed, such as index creation, index mapping, shard allocation, or shard failure.
Elastic JVM GC time	The amount of time that the JVM spent executing Elasticsearch garbage collection operations in the cluster.
Elastic JVM GC Rate	The total number of times that JVM executed garbage activities per second.
Elastic Query/Fetch Latency Sum	<ul style="list-style-type: none"> ● Query latency: The average time each Elasticsearch search query takes to execute. ● Fetch latency: The average time each Elasticsearch search query spends fetching data. <p>Fetch latency typically takes less time than query latency. If fetch latency is consistently increasing, it might indicate slow disks, data enrichment, or large requests with too many results.</p>
Elastic Query Rate	The total queries executed against the Elasticsearch instance per second for each Elasticsearch node.
CPU	The amount of CPU used by Elasticsearch, Fluentd, and Kibana, shown for each component.
Elastic JVM Heap Used	The amount of JVM memory used. In a healthy cluster, the graph shows regular drops as memory is freed by JVM garbage collection.
Elasticsearch Disk Usage	The total disk space used by the Elasticsearch instance for each Elasticsearch node.

Metric	Description
File Descriptors In Use	The total number of file descriptors used by Elasticsearch, Fluentd, and Kibana.
FluentD emit count	The total number of Fluentd messages per second for the Fluentd default output, and the retry count for the default output.
FluentD Buffer Availability	The percent of the Fluentd buffer that is available for chunks. A full buffer might indicate that Fluentd is not able to process the number of logs received.
Elastic rx bytes	The total number of bytes that Elasticsearch has received from FluentD, the Elasticsearch nodes, and other sources.
Elastic Index Failure Rate	The total number of times per second that an Elasticsearch index fails. A high rate might indicate an issue with indexing.
FluentD Output Error Rate	The total number of times per second that FluentD is not able to output logs.

14.3. CHARTS ON THE LOGGING/ELASTICSEARCH NODES DASHBOARD

The **Logging/Elasticsearch Nodes** dashboard contains charts that show details about your Elasticsearch instance, many at node-level, for further diagnostics.

Elasticsearch status

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the status of your Elasticsearch instance.

Table 14.2. Elasticsearch status fields

Metric	Description
--------	-------------

Metric	Description
Cluster status	<p>The cluster health status during the selected time period, using the Elasticsearch green, yellow, and red statuses:</p> <ul style="list-style-type: none"> ● 0 - Indicates that the Elasticsearch instance is in green status, which means that all shards are allocated. ● 1 - Indicates that the Elasticsearch instance is in yellow status, which means that replica shards for at least one shard are not allocated. ● 2 - Indicates that the Elasticsearch instance is in red status, which means that at least one primary shard and its replicas are not allocated.
Cluster nodes	The total number of Elasticsearch nodes in the cluster.
Cluster data nodes	The number of Elasticsearch data nodes in the cluster.
Cluster pending tasks	The number of cluster state changes that are not finished and are waiting in a cluster queue, for example, index creation, index deletion, or shard allocation. A growing trend indicates that the cluster is not able to keep up with changes.

Elasticsearch cluster index shard status

Each Elasticsearch index is a logical group of one or more shards, which are basic units of persisted data. There are two types of index shards: primary shards, and replica shards. When a document is indexed into an index, it is stored in one of its primary shards and copied into every replica of that shard. The number of primary shards is specified when the index is created, and the number cannot change during index lifetime. You can change the number of replica shards at any time.

The index shard can be in several states depending on its lifecycle phase or events occurring in the cluster. When the shard is able to perform search and indexing requests, the shard is active. If the shard cannot perform these requests, the shard is non-active. A shard might be non-active if the shard is initializing, reallocating, unassigned, and so forth.

Index shards consist of a number of smaller internal blocks, called index segments, which are physical representations of the data. An index segment is a relatively small, immutable Lucene index that is created when Lucene commits newly-indexed data. Lucene, a search library used by Elasticsearch, merges index segments into larger segments in the background to keep the total number of segments low. If the process of merging segments is slower than the speed at which new segments are created, it could indicate a problem.

When Lucene performs data operations, such as a search operation, Lucene performs the operation against the index segments in the relevant index. For that purpose, each segment contains specific data structures that are loaded in the memory and mapped. Index mapping can have a significant impact on

the memory used by segment data structures.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the Elasticsearch index shards.

Table 14.3. Elasticsearch cluster shard status charts

Metric	Description
Cluster active shards	The number of active primary shards and the total number of shards, including replicas, in the cluster. If the number of shards grows higher, the cluster performance can start degrading.
Cluster initializing shards	The number of non-active shards in the cluster. A non-active shard is one that is initializing, being reallocated to a different node, or is unassigned. A cluster typically has non-active shards for short periods. A growing number of non-active shards over longer periods could indicate a problem.
Cluster relocating shards	The number of shards that Elasticsearch is relocating to a new node. Elasticsearch relocates nodes for multiple reasons, such as high memory use on a node or after a new node is added to the cluster.
Cluster unassigned shards	The number of unassigned shards. Elasticsearch shards might be unassigned for reasons such as a new index being added or the failure of a node.

Elasticsearch node metrics

Each Elasticsearch node has a finite amount of resources that can be used to process tasks. When all the resources are being used and Elasticsearch attempts to perform a new task, Elasticsearch put the tasks into a queue until some resources become available.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about resource usage for a selected node and the number of tasks waiting in the Elasticsearch queue.

Table 14.4. Elasticsearch node metric charts

Metric	Description
ThreadPool tasks	The number of waiting tasks in individual queues, shown by task type. A long-term accumulation of tasks in any queue could indicate node resource shortages or some other problem.
CPU usage	The amount of CPU being used by the selected Elasticsearch node as a percentage of the total CPU allocated to the host container.

Metric	Description
Memory usage	The amount of memory being used by the selected Elasticsearch node.
Disk usage	The total disk space being used for index data and metadata on the selected Elasticsearch node.
Documents indexing rate	The rate that documents are indexed on the selected Elasticsearch node.
Indexing latency	The time taken to index the documents on the selected Elasticsearch node. Indexing latency can be affected by many factors, such as JVM Heap memory and overall load. A growing latency indicates a resource capacity shortage in the instance.
Search rate	The number of search requests run on the selected Elasticsearch node.
Search latency	The time taken to complete search requests on the selected Elasticsearch node. Search latency can be affected by many factors. A growing latency indicates a resource capacity shortage in the instance.
Documents count (with replicas)	The number of Elasticsearch documents stored on the selected Elasticsearch node, including documents stored in both the primary shards and replica shards that are allocated on the node.
Documents deleting rate	The number of Elasticsearch documents being deleted from any of the index shards that are allocated to the selected Elasticsearch node.
Documents merging rate	The number of Elasticsearch documents being merged in any of index shards that are allocated to the selected Elasticsearch node.

Elasticsearch node fielddata

Fielddata is an Elasticsearch data structure that holds lists of terms in an index and is kept in the JVM Heap. Because fielddata building is an expensive operation, Elasticsearch caches the fielddata structures. Elasticsearch can evict a fielddata cache when the underlying index segment is deleted or merged, or if there is not enough JVM HEAP memory for all the fielddata caches.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about Elasticsearch fielddata.

Table 14.5. Elasticsearch node fielddata charts

Metric	Description
Fielddata memory size	The amount of JVM Heap used for the fielddata cache on the selected Elasticsearch node.
Fielddata evictions	The number of fielddata structures that were deleted from the selected Elasticsearch node.

Elasticsearch node query cache

If the data stored in the index does not change, search query results are cached in a node-level query cache for reuse by Elasticsearch.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about the Elasticsearch node query cache.

Table 14.6. Elasticsearch node query charts

Metric	Description
Query cache size	The total amount of memory used for the query cache for all the shards allocated to the selected Elasticsearch node.
Query cache evictions	The number of query cache evictions on the selected Elasticsearch node.
Query cache hits	The number of query cache hits on the selected Elasticsearch node.
Query cache misses	The number of query cache misses on the selected Elasticsearch node.

Elasticsearch index throttling

When indexing documents, Elasticsearch stores the documents in index segments, which are physical representations of the data. At the same time, Elasticsearch periodically merges smaller segments into a larger segment as a way to optimize resource use. If the indexing is faster than the ability to merge segments, the merge process does not complete quickly enough, which can lead to issues with searches and performance. To prevent this situation, Elasticsearch throttles indexing, typically by reducing the number of threads allocated to indexing down to a single thread.

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about Elasticsearch index throttling.

Table 14.7. Index throttling charts

Metric	Description
Indexing throttling	The amount of time that Elasticsearch has been throttling the indexing operations on the selected Elasticsearch node.

Metric	Description
Merging throttling	The amount of time that Elasticsearch has been throttling the segment merge operations on the selected Elasticsearch node.

Node JVM Heap statistics

The **Logging/Elasticsearch Nodes** dashboard contains the following charts about JVM Heap operations.

Table 14.8. JVM Heap statistic charts

Metric	Description
Heap used	The amount of the total allocated JVM Heap space that is used on the selected Elasticsearch node.
GC count	The number of garbage collection operations that have been run on the selected Elasticsearch node, by old and young garbage collection.
GC time	The amount of time that the JVM spent running garbage collection operations on the selected Elasticsearch node, by old and young garbage collection.

CHAPTER 15. TROUBLESHOOTING LOGGING

15.1. VIEWING OPENSIFT LOGGING STATUS

You can view the status of the Red Hat OpenShift Logging Operator and for a number of logging subsystem components.

15.1.1. Viewing the status of the Red Hat OpenShift Logging Operator

You can view the status of your Red Hat OpenShift Logging Operator.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Change to the **openshift-logging** project.

```
$ oc project openshift-logging
```

2. To view the OpenShift Logging status:

- a. Get the OpenShift Logging status:

```
$ oc get clusterlogging instance -o yaml
```

Example output

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging

....

status: ❶
collection:
  logs:
    fluentdStatus:
      daemonSet: fluentd ❷
      nodes:
        fluentd-2rhqp: ip-10-0-169-13.ec2.internal
        fluentd-6fgjh: ip-10-0-165-244.ec2.internal
        fluentd-6l2ff: ip-10-0-128-218.ec2.internal
        fluentd-54nx5: ip-10-0-139-30.ec2.internal
        fluentd-flpnn: ip-10-0-147-228.ec2.internal
        fluentd-n2frh: ip-10-0-157-45.ec2.internal
      pods:
        failed: []
        notReady: []
        ready:
          - fluentd-2rhqp
          - fluentd-54nx5
          - fluentd-6fgjh
```

```

- fluentd-6l2ff
- fluentd-flpnn
- fluentd-n2frh
logstore: 3
elasticsearchStatus:
- ShardAllocationEnabled: all
cluster:
  activePrimaryShards: 5
  activeShards: 5
  initializingShards: 0
  numDataNodes: 1
  numNodes: 1
  pendingTasks: 0
  relocatingShards: 0
  status: green
  unassignedShards: 0
clusterName: elasticsearch
nodeConditions:
  elasticsearch-cdm-mkkdys93-1:
nodeCount: 1
pods:
  client:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  data:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
  master:
    failed:
    notReady:
    ready:
    - elasticsearch-cdm-mkkdys93-1-7f7c6-mjm7c
visualization: 4
kibanaStatus:
- deployment: kibana
pods:
  failed: []
  notReady: []
  ready:
  - kibana-7fb4fd4cc9-f2nls
replicaSets:
- kibana-7fb4fd4cc9
replicas: 1

```

- 1 In the output, the cluster status fields appear in the **status** stanza.
- 2 Information on the Fluentd pods.
- 3 Information on the Elasticsearch pods, including Elasticsearch cluster health, **green**, **yellow**, or **red**.
- 4 Information on the Kibana pods.

15.1.1.1. Example condition messages

The following are examples of some condition messages from the **Status.Nodes** section of the OpenShift Logging instance.

A status message similar to the following indicates a node has exceeded the configured low watermark and no shard will be allocated to this node:

Example output

```
nodes:
- conditions:
  - lastTransitionTime: 2019-03-15T15:57:22Z
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
      be allocated on this node.
    reason: Disk Watermark Low
    status: "True"
    type: NodeStorage
  deploymentName: example-elasticsearch-clientdatamaster-0-1
  upgradeStatus: {}
```

A status message similar to the following indicates a node has exceeded the configured high watermark and shards will be relocated to other nodes:

Example output

```
nodes:
- conditions:
  - lastTransitionTime: 2019-03-15T16:04:45Z
    message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated
      from this node.
    reason: Disk Watermark High
    status: "True"
    type: NodeStorage
  deploymentName: cluster-logging-operator
  upgradeStatus: {}
```

A status message similar to the following indicates the Elasticsearch node selector in the CR does not match any nodes in the cluster:

Example output

```
Elasticsearch Status:
Shard Allocation Enabled: shard allocation unknown
Cluster:
  Active Primary Shards: 0
  Active Shards:        0
  Initializing Shards:   0
  Num Data Nodes:       0
  Num Nodes:            0
  Pending Tasks:        0
  Relocating Shards:    0
  Status:                cluster health unknown
  Unassigned Shards:    0
  Cluster Name:          elasticsearch
```

```

Node Conditions:
  elasticsearch-cdm-mkkdys93-1:
    Last Transition Time: 2019-06-26T03:37:32Z
    Message:             0/5 nodes are available: 5 node(s) didn't match node selector.
    Reason:              Unschedulable
    Status:              True
    Type:               Unschedulable
  elasticsearch-cdm-mkkdys93-2:
Node Count: 2
Pods:
  Client:
    Failed:
    Not Ready:
      elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
      elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
    Ready:
  Data:
    Failed:
    Not Ready:
      elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
      elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
    Ready:
  Master:
    Failed:
    Not Ready:
      elasticsearch-cdm-mkkdys93-1-75dd69dccd-f7f49
      elasticsearch-cdm-mkkdys93-2-67c64f5f4c-n58vl
    Ready:

```

A status message similar to the following indicates that the requested PVC could not bind to PV:

Example output

```

Node Conditions:
  elasticsearch-cdm-mkkdys93-1:
    Last Transition Time: 2019-06-26T03:37:32Z
    Message:             pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
    Reason:              Unschedulable
    Status:              True
    Type:               Unschedulable

```

A status message similar to the following indicates that the Fluentd pods cannot be scheduled because the node selector did not match any nodes:

Example output

```

Status:
Collection:
Logs:
  Fluentd Status:
    Daemon Set: fluentd
  Nodes:
  Pods:

```

Failed:
Not Ready:
Ready:

15.1.2. Viewing the status of logging subsystem components

You can view the status for a number of logging subsystem components.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Change to the **openshift-logging** project.

```
$ oc project openshift-logging
```

2. View the status of the logging subsystem for Red Hat OpenShift environment:

```
$ oc describe deployment cluster-logging-operator
```

Example output

```
Name:          cluster-logging-operator
....

Conditions:
  Type           Status Reason
  ----           -
  Available      True  MinimumReplicasAvailable
  Progressing    True  NewReplicaSetAvailable
....

Events:
  Type    Reason          Age    From          Message
  ----    -
  Normal  ScalingReplicaSet 62m    deployment-controller Scaled up replica set cluster-logging-operator-574b8987df to 1----
```

3. View the status of the logging subsystem replica set:

- a. Get the name of a replica set:

Example output

```
$ oc get replicaset
```

Example output

```
NAME                                DESIRED  CURRENT  READY  AGE
```

```
cluster-logging-operator-574b8987df    1    1    1    159m
elasticsearch-cdm-uhr537yu-1-6869694fb  1    1    1    157m
elasticsearch-cdm-uhr537yu-2-857b6d676f 1    1    1    156m
elasticsearch-cdm-uhr537yu-3-5b6fdd8cfd 1    1    1    155m
kibana-5bd5544f87                      1    1    1    157m
```

- b. Get the status of the replica set:

```
$ oc describe replicaset cluster-logging-operator-574b8987df
```

Example output

```
Name:      cluster-logging-operator-574b8987df

....

Replicas:   1 current / 1 desired
Pods Status: 1 Running / 0 Waiting / 0 Succeeded / 0 Failed

....

Events:
Type Reason      Age From          Message
----
Normal SuccessfulCreate 66m replicaset-controller Created pod: cluster-logging-operator-574b8987df-qjhqv----
```

15.2. VIEWING THE STATUS OF THE ELASTICSEARCH LOG STORE

You can view the status of the OpenShift Elasticsearch Operator and for a number of Elasticsearch components.

15.2.1. Viewing the status of the log store

You can view the status of your log store.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

1. Change to the **openshift-logging** project.

```
$ oc project openshift-logging
```

2. To view the status:

- a. Get the name of the log store instance:

```
$ oc get Elasticsearch
```

Example output


```
NAME      AGE
elasticsearch 5h9m
```

- b. Get the log store status:

```
$ oc get Elasticsearch <Elasticsearch-instance> -o yaml
```

For example:

```
$ oc get Elasticsearch elasticsearch -n openshift-logging -o yaml
```

The output includes information similar to the following:

Example output

```
status: 1
cluster: 2
  activePrimaryShards: 30
  activeShards: 60
  initializingShards: 0
  numDataNodes: 3
  numNodes: 3
  pendingTasks: 0
  relocatingShards: 0
  status: green
  unassignedShards: 0
clusterHealth: ""
conditions: [] 3
nodes: 4
- deploymentName: elasticsearch-cdm-zjf34ved-1
  upgradeStatus: {}
- deploymentName: elasticsearch-cdm-zjf34ved-2
  upgradeStatus: {}
- deploymentName: elasticsearch-cdm-zjf34ved-3
  upgradeStatus: {}
pods: 5
  client:
    failed: []
    notReady: []
    ready:
      - elasticsearch-cdm-zjf34ved-1-6d7bf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
  data:
    failed: []
    notReady: []
    ready:
      - elasticsearch-cdm-zjf34ved-1-6d7bf844f-sn422
      - elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
      - elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
  master:
    failed: []
    notReady: []
    ready:
```

```
- elasticsearch-cdm-zjf34ved-1-6d7bf844f-sn422
- elasticsearch-cdm-zjf34ved-2-dfbd988bc-qkzjz
- elasticsearch-cdm-zjf34ved-3-c8f566f7c-t7zkt
shardAllocationEnabled: all
```

- 1 In the output, the cluster status fields appear in the **status** stanza.
- 2 The status of the log store:
 - The number of active primary shards.
 - The number of active shards.
 - The number of shards that are initializing.
 - The number of log store data nodes.
 - The total number of log store nodes.
 - The number of pending tasks.
 - The log store status: **green, red, yellow**.
 - The number of unassigned shards.
- 3 Any status conditions, if present. The log store status indicates the reasons from the scheduler if a pod could not be placed. Any events related to the following conditions are shown:
 - Container Waiting for both the log store and proxy containers.
 - Container Terminated for both the log store and proxy containers.
 - Pod unschedulable. Also, a condition is shown for a number of issues; see **Example condition messages**.
- 4 The log store nodes in the cluster, with **upgradeStatus**.
- 5 The log store client, data, and master pods in the cluster, listed under 'failed', **notReady**, or **ready** state.

15.2.1.1. Example condition messages

The following are examples of some condition messages from the **Status** section of the Elasticsearch instance.

The following status message indicates that a node has exceeded the configured low watermark, and no shard will be allocated to this node.

```
status:
  nodes:
    - conditions:
      - lastTransitionTime: 2019-03-15T15:57:22Z
        message: Disk storage usage for node is 27.5gb (36.74%). Shards will be not
          be allocated on this node.
        reason: Disk Watermark Low
```

```

status: "True"
type: NodeStorage
deploymentName: example-elasticsearch-cdm-0-1
upgradeStatus: {}

```

The following status message indicates that a node has exceeded the configured high watermark, and shards will be relocated to other nodes.

```

status:
  nodes:
  - conditions:
    - lastTransitionTime: 2019-03-15T16:04:45Z
      message: Disk storage usage for node is 27.5gb (36.74%). Shards will be relocated
        from this node.
      reason: Disk Watermark High
      status: "True"
      type: NodeStorage
    deploymentName: example-elasticsearch-cdm-0-1
    upgradeStatus: {}

```

The following status message indicates that the log store node selector in the CR does not match any nodes in the cluster:

```

status:
  nodes:
  - conditions:
    - lastTransitionTime: 2019-04-10T02:26:24Z
      message: '0/8 nodes are available: 8 node(s) didn't match node selector.'
      reason: Unschedulable
      status: "True"
      type: Unschedulable

```

The following status message indicates that the log store CR uses a non-existent persistent volume claim (PVC).

```

status:
  nodes:
  - conditions:
    - last Transition Time: 2019-04-10T05:55:51Z
      message: pod has unbound immediate PersistentVolumeClaims (repeated 5 times)
      reason: Unschedulable
      status: True
      type: Unschedulable

```

The following status message indicates that your log store cluster does not have enough nodes to support the redundancy policy.

```

status:
  clusterHealth: ""
  conditions:
  - lastTransitionTime: 2019-04-17T20:01:31Z
    message: Wrong RedundancyPolicy selected. Choose different RedundancyPolicy or
      add more nodes with data roles

```

```
reason: Invalid Settings
status: "True"
type: InvalidRedundancy
```

This status message indicates your cluster has too many control plane nodes:

```
status:
clusterHealth: green
conditions:
- lastTransitionTime: '2019-04-17T20:12:34Z'
  message: >-
    Invalid master nodes count. Please ensure there are no more than 3 total
    nodes with master roles
  reason: Invalid Settings
  status: 'True'
  type: InvalidMasters
```

The following status message indicates that Elasticsearch storage does not support the change you tried to make.

For example:

```
status:
clusterHealth: green
conditions:
- lastTransitionTime: "2021-05-07T01:05:13Z"
  message: Changing the storage structure for a custom resource is not supported
  reason: StorageStructureChangeIgnored
  status: 'True'
  type: StorageStructureChangeIgnored
```

The **reason** and **type** fields specify the type of unsupported change:

StorageClassNameChangeIgnored

Unsupported change to the storage class name.

StorageSizeChangeIgnored

Unsupported change the storage size.

StorageStructureChangeIgnored

Unsupported change between ephemeral and persistent storage structures.



IMPORTANT

If you try to configure the **ClusterLogging** custom resource (CR) to switch from ephemeral to persistent storage, the OpenShift Elasticsearch Operator creates a persistent volume claim (PVC) but does not create a persistent volume (PV). To clear the **StorageStructureChangeIgnored** status, you must revert the change to the **ClusterLogging** CR and delete the PVC.

15.2.2. Viewing the status of the log store components

You can view the status for a number of the log store components.

Elasticsearch indices

You can view the status of the Elasticsearch indices.

1. Get the name of an Elasticsearch pod:

```
$ oc get pods --selector component=elasticsearch -o name
```

Example output

```
pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Get the status of the indices:

```
$ oc exec elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -- indices
```

Example output

```
Defaulting container name to elasticsearch.
Use 'oc describe pod/elasticsearch-cdm-4vjor49p-2-6d4d7db474-q2w7z -n openshift-logging' to see all of the containers in this pod.

green open  infra-000002                                S4QANnf1QP6NgCegfnrnbQ
3 1 119926      0      157      78
green open  audit-000001                                8_EQx77iQCSTzFOXtxRqFw
3 1 0          0      0          0
green open  .security                                iDjscH7aSUGhldq0LheLBQ 1
1 5 0          0      0
green open  .kibana_-377444158_kubeadmin
yBywZ9GfSrKebz5gWBZbjw 3 1 1 0 0 0
green open  infra-000001                                z6Dpe__ORgiopEpW6YI44A
3 1 871000      0      874      436
green open  app-000001                                hlrazQCeSlSewG3c2VlvsQ
3 1 2453        0      3          1
green open  .kibana_1                                JCitcBMSQxKOvlq6iQW6wg
1 1 0          0      0          0
green open  .kibana_-1595131456_user1
ka0W3okS-mQ 3 1 1 0 0 0

```

Log store pods

You can view the status of the pods that host the log store.

1. Get the name of a pod:

```
$ oc get pods --selector component=elasticsearch -o name
```

Example output

```
pod/elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
pod/elasticsearch-cdm-1godmszn-2-5769cf-9ms2n
pod/elasticsearch-cdm-1godmszn-3-f66f7d-zqkz7
```

2. Get the status of a pod:

```
$ oc describe pod elasticsearch-cdm-1godmszn-1-6f8495-vp4lw
```

The output includes the following status information:

Example output

```
....
Status:          Running

....

Containers:
  elasticsearch:
    Container ID:  cri-o://b7d44e0a9ea486e27f47763f5bb4c39dfd2
    State:        Running
      Started:    Mon, 08 Jun 2020 10:17:56 -0400
    Ready:        True
    Restart Count: 0
    Readiness:    exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
                  period=5s #success=1 #failure=3
  ....

  proxy:
    Container ID:  cri-
o://3f77032abaddbb1652c116278652908dc01860320b8a4e741d06894b2f8f9aa1
    State:        Running
      Started:    Mon, 08 Jun 2020 10:18:38 -0400
    Ready:        True
    Restart Count: 0
  ....

Conditions:
  Type            Status
  Initialized     True
  Ready           True
  ContainersReady True
  PodScheduled    True
  ....

Events:          <none>
```

Log storage pod deployment configuration

You can view the status of the log store deployment configuration.

1. Get the name of a deployment configuration:

```
$ oc get deployment --selector component=elasticsearch -o name
```

Example output

```
deployment.extensions/elasticsearch-cdm-1gon-1
deployment.extensions/elasticsearch-cdm-1gon-2
deployment.extensions/elasticsearch-cdm-1gon-3
```

2. Get the deployment configuration status:

```
$ oc describe deployment elasticsearch-cdm-1gon-1
```

The output includes the following status information:

Example output

```
....
Containers:
  elasticsearch:
    Image: registry.redhat.io/openshift-logging/elasticsearch6-rhel8
    Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
               period=5s #success=1 #failure=3
....

Conditions:
  Type            Status  Reason
  ----            -
  Progressing     Unknown DeploymentPaused
  Available       True    MinimumReplicasAvailable
....

Events:          <none>
```

Log store replica set

You can view the status of the log store replica set.

1. Get the name of a replica set:

```
$ oc get replicaSet --selector component=elasticsearch -o name

replicaset.extensions/elasticsearch-cdm-1gon-1-6f8495
replicaset.extensions/elasticsearch-cdm-1gon-2-5769cf
replicaset.extensions/elasticsearch-cdm-1gon-3-f66f7d
```

2. Get the status of the replica set:

```
$ oc describe replicaSet elasticsearch-cdm-1gon-1-6f8495
```

The output includes the following status information:

Example output

```
....
Containers:
  elasticsearch:
```

```

Image: registry.redhat.io/openshift-logging/elasticsearch6-
rhel8@sha256:4265742c7cdd85359140e2d7d703e4311b6497eec7676957f455d6908e7b1
c25
Readiness: exec [/usr/share/elasticsearch/probe/readiness.sh] delay=10s timeout=30s
period=5s #success=1 #failure=3
....
Events:      <none>

```

15.2.3. Elasticsearch cluster status

A dashboard in the **Observe** section of the OpenShift Container Platform web console displays the status of the Elasticsearch cluster.

To get the status of the OpenShift Elasticsearch cluster, visit the dashboard in the **Observe** section of the OpenShift Container Platform web console at **<cluster_url>/monitoring/dashboards/grafana-dashboard-cluster-logging**.

Elasticsearch status fields

eo_elasticsearch_cr_cluster_management_state

Shows whether the Elasticsearch cluster is in a managed or unmanaged state. For example:

```

eo_elasticsearch_cr_cluster_management_state{state="managed"} 1
eo_elasticsearch_cr_cluster_management_state{state="unmanaged"} 0

```

eo_elasticsearch_cr_restart_total

Shows the number of times the Elasticsearch nodes have restarted for certificate restarts, rolling restarts, or scheduled restarts. For example:

```

eo_elasticsearch_cr_restart_total{reason="cert_restart"} 1
eo_elasticsearch_cr_restart_total{reason="rolling_restart"} 1
eo_elasticsearch_cr_restart_total{reason="scheduled_restart"} 3

```

es_index_namespaces_total

Shows the total number of Elasticsearch index namespaces. For example:

```

Total number of Namespaces.
es_index_namespaces_total 5

```

es_index_document_count

Shows the number of records for each namespace. For example:

```

es_index_document_count{namespace="namespace_1"} 25
es_index_document_count{namespace="namespace_2"} 10
es_index_document_count{namespace="namespace_3"} 5

```

The "Secret Elasticsearch fields are either missing or empty" message

If Elasticsearch is missing the **admin-cert**, **admin-key**, **logging-es.crt**, or **logging-es.key** files, the dashboard shows a status message similar to the following example:

```
message": "Secret \"elasticsearch\" fields are either missing or empty: [admin-cert, admin-key,
logging-es.crt, logging-es.key]",
"reason": "Missing Required Secrets",
```

15.3. UNDERSTANDING LOGGING SUBSYSTEM ALERTS

All of the logging collector alerts are listed on the Alerting UI of the OpenShift Container Platform web console.

15.3.1. Viewing logging collector alerts

Alerts are shown in the OpenShift Container Platform web console, on the **Alerts** tab of the Alerting UI. Alerts are in one of the following states:

- **Firing.** The alert condition is true for the duration of the timeout. Click the **Options** menu at the end of the firing alert to view more information or silence the alert.
- **Pending** The alert condition is currently true, but the timeout has not been reached.
- **Not Firing.** The alert is not currently triggered.

Procedure

To view the logging subsystem and other OpenShift Container Platform alerts:

1. In the OpenShift Container Platform console, click **Observe → Alerting**.
2. Click the **Alerts** tab. The alerts are listed, based on the filters selected.

Additional resources

- For more information on the Alerting UI, see [Managing alerts](#).

15.3.2. About logging collector alerts

The following alerts are generated by the logging collector. You can view these alerts in the OpenShift Container Platform web console on the **Alerts** page of the Alerting UI.

Table 15.1. Fluentd Prometheus alerts

Alert	Message	Description	Severity
FluentDHighErrorRate	<value> of records have resulted in an error by fluentd <instance>.	The number of FluentD output errors is high, by default more than 10 in the previous 15 minutes.	Warning

Alert	Message	Description	Severity
FluentdNodeDown	Prometheus could not scrape fluentd <instance> for more than 10m.	Fluentd is reporting that Prometheus could not scrape a specific Fluentd instance.	Critical
FluentdQueueLengthIncreasing	In the last 12h, fluentd <instance> buffer queue length constantly increased more than 1. Current value is <value>.	Fluentd is reporting that the queue size is increasing.	Critical
FluentDVeryHighErrorRate	<value> of records have resulted in an error by fluentd <instance>.	The number of FluentD output errors is very high, by default more than 25 in the previous 15 minutes.	Critical

15.3.3. About Elasticsearch alerting rules

You can view these alerting rules in Prometheus.

Table 15.2. Alerting rules

Alert	Description	Severity
ElasticsearchClusterNotHealthy	The cluster health status has been RED for at least 2 minutes. The cluster does not accept writes, shards may be missing, or the master node hasn't been elected yet.	Critical
ElasticsearchClusterNotHealthy	The cluster health status has been YELLOW for at least 20 minutes. Some shard replicas are not allocated.	Warning
ElasticsearchDiskSpaceRunningLow	The cluster is expected to be out of disk space within the next 6 hours.	Critical
ElasticsearchHighFileDescriptorUsage	The cluster is predicted to be out of file descriptors within the next hour.	Warning
ElasticsearchJVMHeapUsageHigh	The JVM Heap usage on the specified node is high.	Alert
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the low watermark due to low free disk space. Shards can not be allocated to this node anymore. You should consider adding more disk space to the node.	Info

Alert	Description	Severity
ElasticsearchNodeDiskWatermarkReached	The specified node has hit the high watermark due to low free disk space. Some shards will be re-allocated to different nodes if possible. Make sure more disk space is added to the node or drop old indices allocated to this node.	Warning
ElasticsearchNodeDiskFloodWatermarkReached	The specified node has hit the flood watermark due to low free disk space. Every index that has a shard allocated on this node is enforced a read-only block. The index block must be manually released when the disk use falls below the high watermark.	Critical
ElasticsearchJVMHeapUsageHigh	The JVM Heap usage on the specified node is too high.	Alert
ElasticsearchWriteRequestsRejectionJumps	Elasticsearch is experiencing an increase in write rejections on the specified node. This node might not be keeping up with the indexing speed.	Warning
AggregatedLoggingSystemCPUHigh	The CPU used by the system on the specified node is too high.	Alert
ElasticsearchProcessCPUHigh	The CPU used by Elasticsearch on the specified node is too high.	Alert

15.4. TROUBLESHOOTING FOR CRITICAL ALERTS

15.4.1. Elasticsearch Cluster Health is Red

At least one primary shard and its replicas are not allocated to a node.

Troubleshooting

1. Check the Elasticsearch cluster health and verify that the cluster **status** is red.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health
```

2. List the nodes that have joined the cluster.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/nodes?v
```

3. List the Elasticsearch pods and compare them with the nodes in the command output from the previous step.

```
oc -n openshift-logging get pods -l component=elasticsearch
```

4. If some of the Elasticsearch nodes have not joined the cluster, perform the following steps.

- a. Confirm that Elasticsearch has an elected control plane node.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/master?v
```

- b. Review the pod logs of the elected control plane node for issues.

```
oc logs <elasticsearch_master_pod_name> -c elasticsearch -n openshift-logging
```

- c. Review the logs of nodes that have not joined the cluster for issues.

```
oc logs <elasticsearch_node_name> -c elasticsearch -n openshift-logging
```

5. If all the nodes have joined the cluster, perform the following steps, check if the cluster is in the process of recovering.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/recovery?active_only=true
```

If there is no command output, the recovery process might be delayed or stalled by pending tasks.

6. Check if there are pending tasks.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- health |grep
number_of_pending_tasks
```

7. If there are pending tasks, monitor their status.

If their status changes and indicates that the cluster is recovering, continue waiting. The recovery time varies according to the size of the cluster and other factors.

Otherwise, if the status of the pending tasks does not change, this indicates that the recovery has stalled.

8. If it seems like the recovery has stalled, check if **cluster.routing.allocation.enable** is set to **none**.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty
```

9. If **cluster.routing.allocation.enable** is set to **none**, set it to **all**.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/settings?pretty -X PUT -d '{"persistent":
{"cluster.routing.allocation.enable":"all"}}'
```

10. Check which indices are still red.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

11. If any indices are still red, try to clear them by performing the following steps.

- a. Clear the cache.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_cache/clear?pretty
```

- b. Increase the max allocation retries.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.allocation.max_retries":10}'
```

- c. Delete all the scroll items.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_search/scroll/_all -X DELETE
```

- d. Increase the timeout.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name>/_settings?pretty -X PUT -d
'{"index.unassigned.node_left.delayed_timeout":"10m"}'
```

12. If the preceding steps do not clear the red indices, delete the indices individually.

- a. Identify the red index name.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cat/indices?v
```

- b. Delete the red index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_red_index_name> -X DELETE
```

13. If there are no red indices and the cluster status is red, check for a continuous heavy processing load on a data node.

- a. Check if the Elasticsearch JVM Heap usage is high.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_nodes/stats?pretty
```

In the command output, review the **node_name.jvm.mem.heap_used_percent** field to determine the JVM Heap usage.

- b. Check for high CPU utilization.

Additional resources

- Search for "Free up or increase disk space" in the Elasticsearch topic, [Fix a red or yellow cluster status](#).

15.4.2. Elasticsearch Cluster Health is Yellow

Replica shards for at least one primary shard are not allocated to nodes.

Troubleshooting

1. Increase the node count by adjusting **nodeCount** in the **ClusterLogging** CR.

Additional resources

- [About the Cluster Logging custom resource](#)
- [Configuring persistent storage for the log store](#)
- Search for "Free up or increase disk space" in the Elasticsearch topic, [Fix a red or yellow cluster status](#).

15.4.3. Elasticsearch Node Disk Low Watermark Reached

Elasticsearch does not allocate shards to nodes that [reach the low watermark](#).

Troubleshooting

1. Identify the node on which Elasticsearch is deployed.

```
oc -n openshift-logging get po -o wide
```

2. Check if there are **unassigned shards**.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --  
query=_cluster/health?pretty | grep unassigned_shards
```

3. If there are unassigned shards, check the disk space on each node.

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o  
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c  
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

4. Check the **nodes.node_name.fs** field to determine the free disk space on that node.
If the used disk percentage is above 85%, the node has exceeded the low watermark, and shards can no longer be allocated to this node.
5. Try to increase the disk space on all nodes.
6. If increasing the disk space is not possible, try adding a new data node to the cluster.
7. If adding a new data node is problematic, decrease the total cluster redundancy policy.
 - a. Check the current **redundancyPolicy**.

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```

**NOTE**

If you are using a **ClusterLogging** CR, enter:

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

8. If the preceding steps do not fix the issue, delete the old indices.

- a. Check the status of all indices on Elasticsearch.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. Identify an old index that can be deleted.

- c. Delete the index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in [About the Cluster Logging custom resource](#)

15.4.4. Elasticsearch Node Disk High Watermark Reached

Elasticsearch attempts to relocate shards away from a node [that has reached the high watermark](#).

Troubleshooting

1. Identify the node on which Elasticsearch is deployed.

```
oc -n openshift-logging get po -o wide
```

2. Check the disk space on each node.

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

3. Check if the cluster is rebalancing.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_cluster/health?pretty | grep relocating_shards
```

If the command output shows relocating shards, the High Watermark has been exceeded. The default value of the High Watermark is 90%.

The shards relocate to a node with low disk usage that has not crossed any watermark threshold limits.

4. To allocate shards to a particular node, free up some space.
5. Try to increase the disk space on all nodes.
6. If increasing the disk space is not possible, try adding a new data node to the cluster.
7. If adding a new data node is problematic, decrease the total cluster redundancy policy.
 - a. Check the current **redundancyPolicy**.

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



NOTE

If you are using a **ClusterLogging** CR, enter:

```
oc -n openshift-logging get cl -o  
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.
8. If the preceding steps do not fix the issue, delete the old indices.
 - a. Check the status of all indices on Elasticsearch.
 - b. Identify an old index that can be deleted.
 - c. Delete the index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices  
query=<elasticsearch_index_name> -X DELETE
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in [About the Cluster Logging custom resource](#)

15.4.5. Elasticsearch Node Disk Flood Watermark Reached

Elasticsearch enforces a read-only index block on every index that has both of these conditions:

- One or more shards are allocated to the node.
- One or more disks exceed the [flood stage](#).

Troubleshooting

1. Check the disk space of the Elasticsearch node.


```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

Check the **nodes.node_name.fs** field to determine the free disk space on that node.

2. If the used disk percentage is above 95%, it signifies that the node has crossed the flood watermark. Writing is blocked for shards allocated on this particular node.
3. Try to increase the disk space on all nodes.
4. If increasing the disk space is not possible, try adding a new data node to the cluster.
5. If adding a new data node is problematic, decrease the total cluster redundancy policy.
 - a. Check the current **redundancyPolicy**.

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```



NOTE

If you are using a **ClusterLogging** CR, enter:

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.
6. If the preceding steps do not fix the issue, delete the old indices.
 - a. Check the status of all indices on Elasticsearch.


```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```
 - b. Identify an old index that can be deleted.
 - c. Delete the index.


```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```
7. Continue freeing up and monitoring the disk space until the used disk space drops below 90%. Then, unblock write to this particular node.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=_all/_settings?pretty -X PUT -d '{"index.blocks.read_only_allow_delete": null}'
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in [About the Cluster Logging custom resource](#)

15.4.6. Elasticsearch JVM Heap Use is High

The Elasticsearch node JVM Heap memory used is above 75%.

Troubleshooting

Consider [increasing the heap size](#).

15.4.7. Aggregated Logging System CPU is High

System CPU usage on the node is high.

Troubleshooting

Check the CPU of the cluster node. Consider allocating more CPU resources to the node.

15.4.8. Elasticsearch Process CPU is High

Elasticsearch process CPU usage on the node is high.

Troubleshooting

Check the CPU of the cluster node. Consider allocating more CPU resources to the node.

15.4.9. Elasticsearch Disk Space is Running Low

The Elasticsearch Cluster is predicted to be out of disk space within the next 6 hours based on current disk usage.

Troubleshooting

1. Get the disk space of the Elasticsearch node.

```
for pod in `oc -n openshift-logging get po -l component=elasticsearch -o
jsonpath='{.items[*].metadata.name}'`; do echo $pod; oc -n openshift-logging exec -c
elasticsearch $pod -- df -h /elasticsearch/persistent; done
```

2. In the command output, check the **nodes.node_name.fs** field to determine the free disk space on that node.
3. Try to increase the disk space on all nodes.
4. If increasing the disk space is not possible, try adding a new data node to the cluster.
5. If adding a new data node is problematic, decrease the total cluster redundancy policy.
 - a. Check the current **redundancyPolicy**.

```
oc -n openshift-logging get es elasticsearch -o jsonpath='{.spec.redundancyPolicy}'
```

**NOTE**

If you are using a **ClusterLogging** CR, enter:

```
oc -n openshift-logging get cl -o
jsonpath='{.items[*].spec.logStore.elasticsearch.redundancyPolicy}'
```

- b. If the cluster **redundancyPolicy** is higher than **SingleRedundancy**, set it to **SingleRedundancy** and save this change.

6. If the preceding steps do not fix the issue, delete the old indices.

- a. Check the status of all indices on Elasticsearch.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- indices
```

- b. Identify an old index that can be deleted.

- c. Delete the index.

```
oc exec -n openshift-logging -c elasticsearch <elasticsearch_pod_name> -- es_util --
query=<elasticsearch_index_name> -X DELETE
```

Additional resources

- Search for "redundancyPolicy" in the "Sample **ClusterLogging** custom resource (CR)" in [About the Cluster Logging custom resource](#)
- Search for "ElasticsearchDiskSpaceRunningLow" in [About Elasticsearch alerting rules](#).
- Search for "Free up or increase disk space" in the Elasticsearch topic, [Fix a red or yellow cluster status](#).

15.4.10. Elasticsearch FileDescriptor Usage is high

Based on current usage trends, the predicted number of file descriptors on the node is insufficient.

Troubleshooting

Check and, if needed, configure the value of **max_file_descriptors** for each node, as described in the Elasticsearch [File descriptors](#) topic.

Additional resources

- Search for "ElasticsearchHighFileDescriptorUsage" in [About Elasticsearch alerting rules](#).
- Search for "File Descriptors In Use" in [OpenShift Logging dashboards](#).

CHAPTER 16. UNINSTALLING OPENSIFT LOGGING

You can remove the logging subsystem from your OpenShift Container Platform cluster.

16.1. UNINSTALLING THE LOGGING SUBSYSTEM FOR RED HAT OPENSIFT

You can stop log aggregation by deleting the **ClusterLogging** custom resource (CR). After deleting the CR, there are other logging subsystem components that remain, which you can optionally remove.

Deleting the **ClusterLogging** CR does not remove the persistent volume claims (PVCs). To preserve or delete the remaining PVCs, persistent volumes (PVs), and associated data, you must take further action.

Prerequisites


- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

To remove OpenShift Logging:


1. Use the OpenShift Container Platform web console to remove the **ClusterLogging** CR:

- a. Switch to the **Administration** → **Custom Resource Definitions** page.
- b. On the **Custom Resource Definitions** page, click **ClusterLogging**.
- c. On the **Custom Resource Definition Details** page, click **Instances**.


- d. Click the Options menu  next to the instance and select **Delete ClusterLogging**.

2. Optional: Delete the custom resource definitions (CRD):

- a. Switch to the **Administration** → **Custom Resource Definitions** page.

- b. Click the Options menu  next to **ClusterLogForwarder** and select **Delete Custom Resource Definition**.

- c. Click the Options menu  next to **ClusterLogging** and select **Delete Custom Resource Definition**.

- d. Click the Options menu  next to **Elasticsearch** and select **Delete Custom Resource Definition**.

3. Optional: Remove the Red Hat OpenShift Logging Operator and OpenShift Elasticsearch Operator:

- a. Switch to the **Operators** → **Installed Operators** page.

- b. Click the Options menu



next to the Red Hat OpenShift Logging Operator and select

Uninstall Operator.

- c. Click the Options menu



next to the OpenShift Elasticsearch Operator and select

Uninstall Operator.

4. Optional: Remove the OpenShift Logging and Elasticsearch projects.

- a. Switch to the **Home → Projects** page.

- b. Click the Options menu



next to the **openshift-logging** project and select **Delete**

Project.

- c. Confirm the deletion by typing **openshift-logging** in the dialog box and click **Delete.**

- d. Click the Options menu



next to the **openshift-operators-redhat** project and select

Delete Project.



IMPORTANT

Do not delete the **openshift-operators-redhat** project if other global operators are installed in this namespace.

- e. Confirm the deletion by typing **openshift-operators-redhat** in the dialog box and click **Delete.**

5. To keep the PVCs for reuse with other pods, keep the labels or PVC names that you need to reclaim the PVCs.

6. Optional: If you do not want to keep the PVCs, you can delete them.



WARNING

Releasing or deleting PVCs can delete PVs and cause data loss.

- a. Switch to the **Storage → Persistent Volume Claims** page.

- b. Click the Options menu



next to each PVC and select **Delete Persistent Volume**

Claim.

- c. If you want to recover storage space, you can delete the PVs.

Additional resources

- [Reclaiming a persistent volume manually](#)

CHAPTER 17. LOG RECORD FIELDS

The following fields can be present in log records exported by the logging subsystem. Although log records are typically formatted as JSON objects, the same data model can be applied to other encodings.

To search these fields from Elasticsearch and Kibana, use the full dotted field name when searching. For example, with an Elasticsearch **/_search URL**, to look for a Kubernetes pod name, use **/_search/q=kubernetes.pod_name:name-of-my-pod**.

The top level fields may be present in every record.

CHAPTER 18. MESSAGE

The original log entry text, UTF-8 encoded. This field may be absent or empty if a non-empty **structured** field is present. See the description of **structured** for more.

Data type	text
Example value	HAPPY

CHAPTER 19. STRUCTURED

Original log entry as a structured object. This field may be present if the forwarder was configured to parse structured JSON logs. If the original log entry was a valid structured log, this field will contain an equivalent JSON structure. Otherwise this field will be empty or absent, and the **message** field will contain the original log message. The **structured** field can have any subfields that are included in the log message, there are no restrictions defined here.

Data type	group
Example value	map[message:starting fluentd worker pid=21631 ppid=21618 worker=0 pid:21631 ppid:21618 worker:0]

CHAPTER 20. @TIMESTAMP

A UTC value that marks when the log payload was created or, if the creation time is not known, when the log payload was first collected. The "@" prefix denotes a field that is reserved for a particular use. By default, most tools look for "@timestamp" with Elasticsearch.

Data type	date
Example value	2015-01-24 14:06:05.071000000 Z

CHAPTER 21. HOSTNAME

The name of the host where this log message originated. In a Kubernetes cluster, this is the same as **kubernetes.host**.

Data type	keyword
-----------	---------

CHAPTER 22. IPADDR4

The IPv4 address of the source server. Can be an array.

Data type	ip
-----------	----

CHAPTER 23. IPADDR6

The IPv6 address of the source server, if available. Can be an array.

Data type	ip
-----------	----

CHAPTER 24. LEVEL

The logging level from various sources, including **rsyslog(severitytext property)**, a Python logging module, and others.

The following values come from [syslog.h](#), and are preceded by their [numeric equivalents](#):

- **0 = emerg**, system is unusable.
- **1 = alert**, action must be taken immediately.
- **2 = crit**, critical conditions.
- **3 = err**, error conditions.
- **4 = warn**, warning conditions.
- **5 = notice**, normal but significant condition.
- **6 = info**, informational.
- **7 = debug**, debug-level messages.

The two following values are not part of **syslog.h** but are widely used:

- **8 = trace**, trace-level messages, which are more verbose than **debug** messages.
- **9 = unknown**, when the logging system gets a value it doesn't recognize.

Map the log levels or priorities of other logging systems to their nearest match in the preceding list. For example, from [python logging](#), you can match **CRITICAL** with **crit**, **ERROR** with **err**, and so on.

Data type	keyword
Example value	info

CHAPTER 25. PID

The process ID of the logging entity, if available.

Data type	keyword
-----------	---------

CHAPTER 26. SERVICE

The name of the service associated with the logging entity, if available. For example, syslog's **APP-NAME** and rsyslog's **programname** properties are mapped to the service field.

Data type	keyword
-----------	---------

CHAPTER 27. TAGS

Optional. An operator-defined list of tags placed on each log by the collector or normalizer. The payload can be a string with whitespace-delimited string tokens or a JSON list of string tokens.

Data type	text
-----------	------

CHAPTER 28. FILE

The path to the log file from which the collector reads this log entry. Normally, this is a path in the **/var/log** file system of a cluster node.

Data type	text
-----------	------

CHAPTER 29. OFFSET

The offset value. Can represent bytes to the start of the log line in the file (zero- or one-based), or log line numbers (zero- or one-based), so long as the values are strictly monotonically increasing in the context of a single log file. The values are allowed to wrap, representing a new version of the log file (rotation).

Data type	long
-----------	------

CHAPTER 30. KUBERNETES

The namespace for Kubernetes-specific metadata

Data type	group
-----------	-------

30.1. KUBERNETES.POD_NAME

The name of the pod

Data type	keyword
-----------	---------

30.2. KUBERNETES.POD_ID

The Kubernetes ID of the pod

Data type	keyword
-----------	---------

30.3. KUBERNETES.NAMESPACE_NAME

The name of the namespace in Kubernetes

Data type	keyword
-----------	---------

30.4. KUBERNETES.NAMESPACE_ID

The ID of the namespace in Kubernetes

Data type	keyword
-----------	---------

30.5. KUBERNETES.HOST

The Kubernetes node name

Data type	keyword
-----------	---------

30.6. KUBERNETES.CONTAINER_NAME

The name of the container in Kubernetes

Data type	keyword
-----------	---------

30.7. KUBERNETES.ANNOTATIONS

Annotations associated with the Kubernetes object

Data type	group
-----------	-------

30.8. KUBERNETES.LABELS

Labels present on the original Kubernetes Pod

Data type	group
-----------	-------

30.9. KUBERNETES.EVENT

The Kubernetes event obtained from the Kubernetes master API. This event description loosely follows **type Event** in [Event v1 core](#).

Data type	group
-----------	-------

30.9.1. kubernetes.event.verb

The type of event, **ADDED**, **MODIFIED**, or **DELETED**

Data type	keyword
Example value	ADDED

30.9.2. kubernetes.event.metadata

Information related to the location and time of the event creation

Data type	group
-----------	-------

30.9.2.1. kubernetes.event.metadata.name

The name of the object that triggered the event creation

Data type	keyword
Example value	java-mainclass-1.14d888a4cfc24890

30.9.2.2. kubernetes.event.metadata.namespace

The name of the namespace where the event originally occurred. Note that it differs from **kubernetes.namespace_name**, which is the namespace where the **eventrouter** application is deployed.

Data type	keyword
Example value	default

30.9.2.3. kubernetes.event.metadata.selfLink

A link to the event

Data type	keyword
Example value	/api/v1/namespaces/javaj/events/java-mainclass-1.14d888a4cfc24890

30.9.2.4. kubernetes.event.metadata.uid

The unique ID of the event

Data type	keyword
Example value	d828ac69-7b58-11e7-9cf5-5254002f560c

30.9.2.5. kubernetes.event.metadata.resourceVersion

A string that identifies the server's internal version of the event. Clients can use this string to determine when objects have changed.

Data type	integer
Example value	311987

30.9.3. kubernetes.event.involvedObject

The object that the event is about.

Data type	group
-----------	-------

30.9.3.1. kubernetes.event.involvedObject.kind

The type of object

Data type	keyword
Example value	ReplicationController

30.9.3.2. `kubernetes.event.involvedObject.namespace`

The namespace name of the involved object. Note that it may differ from **`kubernetes.namespace_name`**, which is the namespace where the **`eventrouter`** application is deployed.

Data type	keyword
Example value	default

30.9.3.3. `kubernetes.event.involvedObject.name`

The name of the object that triggered the event

Data type	keyword
Example value	java-mainclass-1

30.9.3.4. `kubernetes.event.involvedObject.uid`

The unique ID of the object

Data type	keyword
Example value	e6bff941-76a8-11e7-8193-5254002f560c

30.9.3.5. `kubernetes.event.involvedObject.apiVersion`

The version of kubernetes master API

Data type	keyword
Example value	v1

30.9.3.6. `kubernetes.event.involvedObject.resourceVersion`

A string that identifies the server's internal version of the pod that triggered the event. Clients can use this string to determine when objects have changed.

Data type	keyword
Example value	308882

30.9.4. kubernetes.event.reason

A short machine-understandable string that gives the reason for generating this event

Data type	keyword
Example value	SuccessfulCreate

30.9.5. kubernetes.event.source_component

The component that reported this event

Data type	keyword
Example value	replication-controller

30.9.6. kubernetes.event.firstTimestamp

The time at which the event was first recorded

Data type	date
Example value	2017-08-07 10:11:57.000000000 Z

30.9.7. kubernetes.event.count

The number of times this event has occurred

Data type	integer
Example value	1

30.9.8. kubernetes.event.type

The type of event, **Normal** or **Warning**. New types could be added in the future.

Data type	keyword
Example value	Normal

CHAPTER 31. OPENSIFT

The namespace for openshift-logging specific metadata

Data type	group
-----------	-------

31.1. OPENSIFT.LABELS

Labels added by the Cluster Log Forwarder configuration

Data type	group
-----------	-------

CHAPTER 32. GLOSSARY

This glossary defines common terms that are used in the OpenShift Container Platform Logging content.

annotation

You can use annotations to attach metadata to objects.

Cluster Logging Operator (CLO)

The Cluster Logging Operator provides a set of APIs to control the collection and forwarding of application, infrastructure, and audit logs.

Custom Resource (CR)

A CR is an extension of the Kubernetes API. To configure OpenShift Container Platform Logging and log forwarding, you can customize the **ClusterLogging** and the **ClusterLogForwarder** custom resources.

event router

The event router is a pod that watches OpenShift Container Platform events. It collects logs by using OpenShift Container Platform Logging.

Fluentd

Fluentd is a log collector that resides on each OpenShift Container Platform node. It gathers application, infrastructure, and audit logs and forwards them to different outputs.

garbage collection

Garbage collection is the process of cleaning up cluster resources, such as terminated containers and images that are not referenced by any running pods.

Elasticsearch

Elasticsearch is a distributed search and analytics engine. OpenShift Container Platform uses Elasticsearch as a default log store for OpenShift Container Platform Logging.

Elasticsearch Operator

Elasticsearch operator is used to run Elasticsearch cluster on top of OpenShift Container Platform. The Elasticsearch Operator provides self-service for the Elasticsearch cluster operations and is used by OpenShift Container Platform Logging.

indexing

Indexing is a data structure technique that is used to quickly locate and access data. Indexing optimizes the performance by minimizing the amount of disk access required when a query is processed.

JSON logging

OpenShift Container Platform Logging Log Forwarding API enables you to parse JSON logs into a structured object and forward them to either OpenShift Container Platform Logging-managed Elasticsearch or any other third-party system supported by the Log Forwarding API.

Kibana

Kibana is a browser-based console interface to query, discover, and visualize your Elasticsearch data through histograms, line graphs, and pie charts.

Kubernetes API server

Kubernetes API server validates and configures data for the API objects.

Labels

Labels are key-value pairs that you can use to organize and select subsets of objects, such as a pod.

Logging

With OpenShift Container Platform Logging you can aggregate application, infrastructure, and audit logs throughout your cluster. You can also store them to a default log store, forward them to third party systems, and query and visualize the stored logs in the default log store.

logging collector

A logging collector collects logs from the cluster, formats them, and forwards them to the log store or third party systems.

log store

A log store is used to store aggregated logs. You can use the default Elasticsearch log store or forward logs to external log stores. The default log store is optimized and tested for short-term storage.

log visualizer

Log visualizer is the user interface (UI) component you can use to view information such as logs, graphs, charts, and other metrics. The current implementation is Kibana.

node

A node is a worker machine in the OpenShift Container Platform cluster. A node is either a virtual machine (VM) or a physical machine.

Operators

Operators are the preferred method of packaging, deploying, and managing a Kubernetes application in an OpenShift Container Platform cluster. An Operator takes human operational knowledge and encodes it into software that is packaged and shared with customers.

pod

A pod is the smallest logical unit in Kubernetes. A pod consists of one or more containers and runs on a worker node..

Role-based access control (RBAC)

RBAC is a key security control to ensure that cluster users and workloads have access only to resources required to execute their roles.

shards

Elasticsearch organizes the log data from Fluentd into datastores, or indices, then subdivides each index into multiple pieces called shards.

taint

Taints ensure that pods are scheduled onto appropriate nodes. You can apply one or more taints on a node.

toleration

You can apply tolerations to pods. Tolerations allow the scheduler to schedule pods with matching taints.

web console

A user interface (UI) to manage OpenShift Container Platform.