



Nokia Container Services

Release 20 FP2

Operations and Administration Manual

DN1000040981

Issue 1-6

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

This document includes Nokia proprietary and confidential information, which may not be distributed or disclosed to any third parties without the prior written consent of Nokia.

This document is intended for use by Nokia's customers ("You"/"Your") in connection with a product purchased or licensed from any company within Nokia Group of Companies. Use this document as agreed. You agree to notify Nokia of any errors you may find in this document; however, should you elect to use this document for any purpose(s) for which it is not intended, You understand and warrant that any determinations You may make or actions You may take will be based upon Your independent judgment and analysis of the content of this document.

Nokia reserves the right to make changes to this document without notice. At all times, the controlling version is the one available on Nokia's site.

No part of this document may be modified.

NO WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF AVAILABILITY, ACCURACY, RELIABILITY, TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, IS MADE IN RELATION TO THE CONTENT OF THIS DOCUMENT. IN NO EVENT WILL NOKIA BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL OR ANY LOSSES, SUCH AS BUT NOT LIMITED TO LOSS OF PROFIT, REVENUE, BUSINESS INTERRUPTION, BUSINESS OPPORTUNITY OR DATA THAT MAY ARISE FROM THE USE OF THIS DOCUMENT OR THE INFORMATION IN IT, EVEN IN THE CASE OF ERRORS IN OR OMISSIONS FROM THIS DOCUMENT OR ITS CONTENT.

Copyright and trademark: Nokia is a registered trademark of Nokia Corporation. Other product names mentioned in this document may be trademarks of their respective owners.

©2022 Nokia.

Table of Contents

List of Figures.....	15
List of Tables.....	16
1 Guide to documentation.....	19
1.1 Documentation conventions.....	19
1.2 Document list for Nokia Container Services (NCS).....	19
1.3 Documentation feedback.....	21
2 Overview.....	22
3 Getting started.....	23
3.1 Cluster definition.....	23
3.1.1 Storage node.....	23
3.1.1.1 Limitations.....	23
3.1.1.2 Configuration.....	24
3.1.1.3 Networking.....	24
3.2 Cluster node requirements.....	24
3.3 Cluster configuration.....	27
3.4 Run-time resource usage.....	28
3.4.1 Single mode.....	28
3.4.2 Mixed Mode.....	29
4 NCS command line (CLI).....	31
5 NCS API.....	32
6 Dashboards.....	33
6.1 Accessing the NCS dashboard.....	33
6.2 Accessing the Kubernetes dashboard.....	33
6.3 Accessing the Keycloak dashboard.....	33
6.4 Integrating CSF dashboard components into NCS.....	33
7 User management.....	35
7.1 Initial user login.....	35
7.2 Managing users.....	38
7.3 Adding a user.....	39
7.3.1 Add extra user on cluster node.....	40

7.4 User roles.....	42
7.5 Using a Non-Root user.....	43
7.5.1 Management of Operating System users.....	45
8 Adding and Removing Physical Resources to a host group.....	51
8.1 Scale-Out for Worker, Storage and Edge Roles.....	51
8.2 Scale-In for Worker, Storage and Edge Roles.....	52
9 Cluster management.....	53
9.1 Cluster inventory.....	53
9.2 Cluster re-install.....	54
9.2.1 Re-try cluster install.....	55
9.2.2 Reset cluster.....	55
9.2.3 Re-create cluster nodes.....	56
9.2.4 Uninstall cluster.....	56
9.3 Cluster scale-in.....	57
9.4 Cluster scale-out.....	60
9.5 Cluster scale-out-new-group.....	64
9.6 Cluster auto-scaling.....	69
9.7 Cluster health check.....	70
9.8 Cluster kubernetes setting (KubeSetting).....	87
9.9 Cluster backup and restore.....	91
9.10 Cluster graceful shutdown and startup.....	91
9.10.1 Cluster graceful shutdown startup for virtual deployments.....	91
9.11 Refresh cluster root certificate.....	93
9.12 Recover a node.....	100
9.13 Restart a Node.....	102
9.14 Shutdown a node.....	103
9.15 Startup a node.....	105
9.16 Cluster node resize.....	106
9.16.1 BCMT Control node resize in OpenStack.....	109
9.17 Uninstall the cluster.....	121
10 Networking.....	123
10.1 Network topology.....	123
10.2 Network stack.....	124
10.2.1 Enable IPv4 only using a configuration file.....	124
10.2.2 Enable dualstack using a configuration file.....	125
10.2.3 Enable IPv6 only using a configuration file.....	125
10.2.4 Manage dualstack using CLI commands.....	126
10.3 Container network interface (CNI).....	126
10.3.1 Calico.....	128
10.3.1.1 Enable Calico using a configuration file.....	130
10.3.2 ipvlan.....	130
10.3.2.1 ipvlan usage example.....	130

10.3.2.1.1 Network Attachment Definition.....	131
10.3.2.1.2 Pod annotation.....	131
10.3.2.2 ipvlan network configuration reference.....	131
10.3.3 macvlan.....	132
10.3.3.1 macvlan usage example.....	132
10.3.3.1.1 Network Attachment Definition.....	132
10.3.3.1.2 Pod annotation.....	132
10.3.3.2 macvlan network configuration reference.....	132
10.3.4 host-device.....	133
10.3.5 SR-IOV.....	134
10.3.5.1 SR-IOV usage example.....	136
10.3.5.2 Network Attachment Definition.....	136
10.3.5.3 Pod annotation.....	136
10.3.5.4 SR-IOV network configuration reference.....	136
10.3.6 vhost-user.....	137
10.3.7 Multus.....	137
10.3.7.1 Enable Multus using a configuration file.....	138
10.3.7.2 Multus usage example.....	138
10.3.7.2.1 Network attachment definition.....	139
10.3.7.2.2 Pod annotation.....	139
10.3.7.2.3 Multus examples for NCS clusters running in VMs (no VLAN tagging is needed)....	139
10.3.7.2.4 Multus examples for NCS clusters running on baremetal without SR-IOV (VLAN tagging is required).....	147
10.3.7.2.5 Multus examples for NCS clusters running on baremetal with SR-IOV (VLAN tagging is required).....	154
10.3.7.3 Enable Multus by node group.....	160
10.3.8 Source based routing (SBR).....	161
10.3.9 DANM - Virtual Deployment.....	162
10.3.9.1 Enable DANM using a configuration file.....	163
10.3.9.2 DanmNet operations.....	164
10.3.9.3 DanmNet specifications in pod annotations.....	164
10.3.10 DANM - Baremetal Deployment.....	165
10.3.10.1 DANM Overview.....	165
10.3.10.2 DANM Usage.....	168
10.3.10.3 Enable DANM by node group.....	168
10.3.11 DANM - Renewing certificate.....	169
10.3.12 Enable Multus and DANM by node group.....	170
10.3.12.1 Configuring Multus and DANM together.....	171
10.3.13 IPv6 Support.....	172
10.3.13.1 Dual Stack.....	172
10.3.13.2 IPv6 only.....	173
10.3.14 Tuning.....	174
10.3.14.1 Tuning usage example.....	174
10.3.14.1.1 Network Attachment Definition.....	174
10.3.14.1.2 Pod annotation.....	175
10.3.14.2 Tuning network configuration reference.....	175
10.3.15 Whereabouts.....	175
10.3.15.1 Whereabouts usage example.....	175

10.3.15.1.1 Network Attachment Definition.....	175
10.3.15.1.2 Pod annotation.....	176
10.3.15.2 Whereabouts IPAM configuration reference.....	176
10.3.15.2.1 Required.....	176
10.3.15.2.2 Range end syntax.....	176
10.3.15.2.3 Optional.....	176
10.3.16 static.....	177
10.3.16.1 static IPAM usage example.....	177
10.3.16.1.1 Network Attachment Definition.....	177
10.3.16.1.2 Pod annotation.....	178
10.3.16.2 static IPAM configuration reference.....	178
10.3.17 host-local.....	178
10.3.17.1 host-local usage example.....	178
10.3.17.1.1 Network Attachment Definition.....	178
10.3.17.1.2 Pod annotation.....	179
10.3.17.2 host-local IPAM configuration reference.....	179
10.4 Ingress.....	179
10.5 Egress.....	179
10.5.1 Enable cluster egress using a configuration file.....	181
10.5.2 Enable cluster egress using a CRD.....	181
10.5.3 Manage cluster egress using CLI commands.....	182
10.5.4 Reset cluster egress configuration settings.....	183
10.6 Istio.....	184
10.6.1 Istio in Nokia Container Services.....	184
10.6.2 CITM usage with Istio.....	186
10.6.3 Components in NCS Istio.....	187
10.6.3.1 Kiali.....	189
10.6.4 Life Cycle Events.....	191
10.6.5 Operations.....	197
10.6.5.1 Egress Gateways with TLS Origination.....	211
10.6.6 Limitation.....	212
10.6.7 Plugins.....	214
10.6.8 Multicluster deployment.....	217
10.6.8.1 Prerequisites.....	218
10.6.8.2 Deploy the Istio control plane in each cluster.....	218
10.6.8.3 Configure application services.....	219
10.6.8.3.1 Deploy the httpbin service in <i>cluster2</i>	219
10.6.8.3.2 Get the information of <i>cluster2</i> gateway.....	221
10.6.8.3.3 Create a service entry for the httpbin service in <i>cluster1</i>	221
10.6.8.3.4 Verify that httpbin is accessible from the <i>cluster1</i>	223
10.6.8.3.5 Send remote traffic via an egress gateway.....	224
10.7 Jaeger.....	225
10.7.1 Jaeger in NCS.....	226
10.7.2 Life Cycle Events.....	227
10.7.3 Install/Uninstall.....	227
10.7.3.1 Install Jaeger.....	228
10.7.3.1.1 Install all-in-one.....	230

10.7.4 Upgrade/Rollback.....	230
10.7.5 Operations.....	230
10.7.5.1 With Istio.....	231
10.7.5.1.1 With Jaeger Client Libraries.....	232
10.8 Dynamic DNS.....	235
10.8.1 Usage.....	236
10.8.2 Provider Information in Secret.....	238
10.9 Border Gateway Protocol.....	241
10.10 Static Route.....	245
10.10.1 Dynamic static route management via ConfigMap.....	247
11 Services.....	250
11.1 NTP.....	250
11.1.1 Manage NTP using a configuration file.....	250
11.1.2 Manage NTP using CLI commands.....	250
11.1.3 Updating poll interval and open port 123 for localhost.....	251
11.2 DNS.....	253
11.2.1 Support different DNS server for different zones.....	254
11.2.2 Manage DNS using a configuration file.....	255
11.2.3 Manage DNS using CLI commands.....	255
11.2.4 Usage.....	257
11.3 High availability.....	257
11.3.1 Enable high availability using a configuration file.....	258
11.3.2 Enable high availability using a CRD.....	259
11.3.3 Manage high availability using CLI commands.....	261
11.3.4 Define a default Pod prio class for applications forbidding preemption.....	263
11.3.5 Static VIP address for edges on vCenter.....	263
11.3.6 Kubernetes cluster HA.....	264
11.3.7 ETCD Cluster HA.....	264
11.3.8 Edge Node VIP.....	265
11.3.9 API Service HA.....	265
11.3.10 DNS HA.....	265
11.3.11 Registry/chart repo HA.....	266
11.3.12 Living upgrade/rollback support.....	266
11.3.13 Disaster recovery support.....	266
11.3.14 NCS heartbeat.....	266
11.4 Docker registry and chart repository.....	267
11.4.1 NCS Docker Registry.....	267
11.4.2 Manage the Docker registry using CLI commands.....	268
11.4.3 NCS Chart Repository.....	269
11.5 Log collection.....	269
11.5.1 Manage fluentd using CLI commands.....	270
11.6 Certificate management.....	270
11.6.1 HA Configuration.....	276
11.7 ETCD.....	279
11.7.1 RAM ETCD.....	282
11.7.1.1 Limitation.....	283

11.7.2 ETCD backend storage size.....	285
11.7.3 ETCD on disk recovery procedure.....	285
11.8 Registry server.....	288
11.8.1 CLI commands.....	288
12 Storage.....	289
12.1 Overview.....	289
12.2 Software RAID.....	292
12.2.1 Software RAID on an operating system device.....	292
12.2.1.1 Enable Software RAID1 on a management node.....	293
12.2.1.2 Enable Software RAID1 for other nodes.....	293
12.2.1.2.1 SW RAID1 devices should not conflict with other settings.....	294
12.2.1.3 Post deployment of Software RAID.....	295
12.2.1.4 Replace OS drive when Software RAID1 is configured.....	295
12.2.1.4.1 Identify a failed drive.....	296
12.2.1.4.2 Replacing a failed drive.....	296
12.3 Cinder.....	298
12.3.1 Cinder usage example.....	298
12.3.2 PVC resize.....	300
12.3.3 Snapshot create and store.....	301
12.4 Manila.....	302
12.4.1 Manila usage example.....	303
12.5 vSphere.....	305
12.5.1 Permissions.....	305
12.5.2 vSphere usage example.....	305
12.5.3 Known issues and troubleshooting.....	307
12.6 EBS.....	307
12.6.1 EBS usage example.....	308
12.7 Azure Disk.....	309
12.7.1 Azure Disk usage example.....	310
12.8 Rook.....	311
12.8.1 Verification.....	312
12.8.2 Rook usage example.....	314
12.8.3 Rook troubleshooting.....	316
12.9 GlusterFS.....	316
12.9.1 GlusterFS usage example.....	317
12.9.1.1 PVC resize.....	319
12.10 Local storage.....	321
12.10.1 Local storage usage example.....	321
12.11 Ceph CSI.....	322
12.11.1 Precondition.....	322
12.11.2 Installation procedure.....	322
12.11.3 Verification.....	323
12.11.4 Upgrade of ceph rbd-csi.....	326
12.12 Dynamic NFS.....	331
12.13 Implementation of S3 on NCS.....	331
12.13.1 S3 Support.....	331

12.13.1.1 How to use S3 from pod.....	331
12.13.1.1.1 S3 towards the local radosgw s3.....	332
12.13.1.1.2 Accessing AWS S3.....	336
12.14 S3cmd to access s3 server.....	338
12.15 Docker.....	343
12.15.1 Project quota functionality.....	343
12.16 Storage Classes.....	344
12.17 Resize partition size.....	346
12.17.1 OpenStack Environment.....	346
12.18 Rack Enablement for CaaS.....	348
12.19 Storage Node Disk Failure and Replacement.....	348
12.19.1 Replacing the Storage Node Disk.....	348
12.19.1.1 Disabling and stopping the failed OSD service.....	352
12.19.2 Locating the disk that must be replaced (AirFrame OR/RM products).....	352
12.19.3 Locating the disk that must be replaced (HP Storage Nodes).....	354
12.19.4 Locating the disk that must be replaced (DELL PowerEdge R740xd with HP740P RAID Controller).....	355
12.19.5 Locating the Physical Storage Node and its JBOD AirFrame Storage Nodes.....	358
12.19.6 Replacing the Physical Disk.....	358
12.19.6.1 Nokia AirFrame Products (RM/OR) and Dell PowerEdge 740xd Support.....	358
12.19.6.2 HP Storage Nodes Support.....	361
12.19.7 Creating a new OSD on the New Disks.....	362
12.20 Creating and Recreating a Specific OSD.....	363
12.20.1 Symptoms of Unhealthy OSDs and when to Recreate them.....	363
12.20.2 Creating and Recreating an OSD- Overview.....	364
12.20.3 Example 1 - Recreating an OSD (block.db and journal partitions reside on the same disk).....	365
12.20.3.1 Preparation Phase for Example 1.....	365
12.20.3.1.1 Scenario 1 - Preparation Phase for Example 1.....	366
12.20.3.1.2 Scenario 2 - Preparation Phase for Example 1.....	368
12.20.4 Example 2 - Recreating an OSD that uses other disks for Journal and block.dbd.....	369
12.20.5 Example 3 - Creating an OSD on a Root Disk.....	370
13 Firewall management.....	372
13.1 Enable firewall.....	372
13.2 Add firewall rules.....	374
13.2.1 Add Firewall Rules for Multiports.....	374
13.3 Delete firewall rules.....	375
13.4 SELinux enforcement rules.....	376
14 Operators.....	380
14.1 Sysctl.....	380
14.2 Limit.....	381
14.3 Password.....	381
14.4 KernelModule.....	382
14.5 RegistryCertificate.....	383
14.6 Application POD TimeZone.....	384

15 Custom installation/NCS hooks.....	386
16 Application management.....	388
16.1 Harbor.....	388
16.1.1 What is Harbor.....	388
16.1.2 Harbor on NCS.....	388
16.1.3 What's New Harbor NCS FP2.....	388
16.1.4 Installation.....	389
16.1.5 Administration.....	410
16.1.6 Usage.....	413
16.1.6.1 Harbor access endpoint.....	413
16.1.6.2 Docker repository.....	414
16.1.6.3 Helm Chart repository.....	414
16.1.6.4 API.....	415
16.1.6.5 Harbor portal exposure.....	415
16.1.6.6 Access Harbor repo from outside NCS cluster.....	416
16.1.6.7 Add trusted CA to Harbor.....	417
16.1.7 Design Items.....	417
16.1.7.1 DB.....	417
16.1.7.2 HA.....	418
16.1.7.3 TLS.....	419
16.1.8 FAQ.....	419
16.1.8.1 Why can't add user to a harbor project.....	419
16.1.8.2 How Harbor uses Keycloak.....	420
16.1.8.3 Is Keycloak necessary.....	420
16.1.8.4 How to generate Clair offline database.....	421
16.1.8.5 How to update Trivy offline vulnerability database.....	421
16.1.8.6 How to increase harbor PVC size.....	421
16.1.9 Vulnerability Scanning.....	422
16.2 Chart application management.....	422
16.3 Integrated NCS application manager.....	423
16.3.1 Enable integrated NCS application.....	423
16.3.2 ncs_app_list.....	423
16.3.3 ncs_app_values.....	423
17 Tenant management.....	426
17.1 Tenant management overview.....	426
17.1.1 Multi-tenancy.....	426
17.1.1.1 NCS multi-tenancy limitations in NCS20FP2SU2.....	427
17.1.1.2 Workaround for multitenancy + cpu pooler case.....	427
17.1.1.3 Feature Flag.....	428
17.1.1.4 Dimensioning.....	429
17.1.1.4.1 Services List.....	429
17.1.1.4.2 Disk Requirements.....	430
17.1.1.4.3 Increasing registry size.....	430

17.1.1.5 User & Role.....	431
17.1.1.6 Kubernetes Multi-tenancy Profile.....	431
17.1.1.7 Tenant Level Isolations.....	432
17.1.1.7.1 UI Changes.....	432
17.1.1.7.2 REST API MT.....	432
17.1.1.7.3 CLI MT.....	432
17.1.1.7.4 Management Portal MT.....	432
17.1.1.8 Resource Quota.....	433
17.1.1.9 Limit Range.....	433
17.1.1.10 Network.....	433
17.1.1.11 Container Image Registry.....	433
17.1.1.12 Chart Repo.....	433
17.1.1.13 Harbor RBAC Roles.....	433
17.1.1.14 Multi-Project and Multi-Tenant Admins.....	439
17.1.1.15 Rest API for Multi tenancy.....	439
17.1.1.16 CLI with Multi tenancy.....	440
17.1.1.17 Tenant Creation Flow.....	440
17.2 Tenant Management Operation.....	441
18 Fault Management and Performance Management.....	445
18.1 Fault Management.....	445
18.2 Performance Management.....	445
18.3 Configuration.....	446
18.4 NCS Logging.....	446
18.5 Logs Location.....	457
18.6 Audit.....	458
18.7 Monitoring.....	459
18.7.1 Monitoring for both VM and Bare Metal.....	460
18.7.1.1 Node problem detector.....	460
18.7.1.1.1 Node problem detector architecture.....	460
18.7.1.1.2 User Facing API.....	461
18.7.1.1.3 Node problem detection rule.....	461
18.7.1.1.4 Starting and Stopping the Node problem detector.....	463
18.7.1.1.5 Node condition reported by Node problem detector.....	463
18.7.1.2 Escalator.....	464
18.7.1.2.1 Enable the Escalator using a configuration file.....	467
18.7.2 Monitoring for VM.....	467
18.7.2.1 Prometheus.....	467
18.7.2.1.1 Prometheus metrics.....	467
18.7.2.1.2 Prometheus alert rules.....	534
18.7.2.1.3 Prometheus adapter configuration.....	539
18.7.2.1.4 Dimensioning Prometheus.....	541
18.7.2.2 BTEL Telemetry.....	544
18.7.2.2.1 Connection to NetAct.....	544
18.7.2.3 Monitoring for Bare-metal.....	545
18.7.2.3.1 Zabbix.....	545
18.7.2.3.2 Alarms.....	584

18.7.2.3.3 ELK.....	589
18.7.2.4 Splunk.....	591
18.7.2.4.1 Using Splunk.....	592
18.7.2.5 NCS Virtualized Deployment Alarms.....	593
18.8 Alarm Manager on BareMetal.....	594
18.8.1 MTU Size.....	594
19 Cluster Graceful Shutdown and Startup for Bare-metal implementation.....	595
19.1 Prerequisites for executing Graceful Shutdown.....	595
19.2 Cluster Shutdown.....	596
19.2.1 NCS Cluster shutdown.....	596
19.2.2 Ceph and services shutdown.....	597
19.2.3 Redhat "NotReady" Status issue.....	597
19.3 Cluster Startup.....	598
20 Backup and Restore Operations for Bare-metal Implementation.....	600
20.1 Storage dimensioning for Backup and Restore.....	600
20.2 Virtualized Deployments.....	601
20.2.1 Configuring CBUR before running any backup or restore.....	602
20.2.2 Procedure to Back up the NCS Cluster.....	609
20.2.3 Restore Cluster.....	612
20.2.4 App backups.....	615
20.2.5 Additional procedures.....	615
20.2.6 Backup or Restore with Backend Mode SFTP.....	615
20.2.7 Copying the Public Key for the Backup and Restore Tool to the SFTP Server.....	617
20.2.8 Backup or Restore with Avamar Server.....	618
20.3 Containerized Application Backup and Recovery (the Containerized Backup and Restore tool)....	646
20.3.1 Introduction to Backup and Recovery.....	646
20.3.2 Prerequisites.....	647
20.3.3 NCS installation.....	649
20.3.4 Backup and Recovery Infrastructure in NCS.....	650
20.3.5 Back Up and Restore Process.....	651
20.3.5.1 BCMT Cluster Backup and Restore.....	651
20.3.5.1.1 Introduction.....	651
20.3.5.1.2 Back up BCMT cluster.....	652
20.3.5.1.3 Recover BCMT Cluster in the event of disaster.....	659
20.3.5.1.4 Recover APPs.....	660
20.3.5.1.5 Recover APP PV Data.....	664
20.3.5.2 Backup or Restore with Avamar Server.....	664
20.3.5.3 Backup or Restore with Backend Mode SFTP.....	692
20.3.5.4 Backup or Restore for cbur-master self volume.....	694
20.3.5.5 Namespace Backup and Restore.....	700
20.3.5.6 Restore between clusters.....	705
20.3.5.7 To configure Avamar if using AVAMAR as the backup or restore repository.....	706
20.3.5.8 To set ncm endpoint and login ncm.....	706
20.3.5.9 To begin the back up as Kubernetes cronjob.....	706

20.3.5.10 To disable a scheduled cluster backup.....	709
20.3.5.11 To begin the immediate cluster backup.....	709
20.3.6 Deployment.....	709
20.3.7 Back Up and Restore via NCM CLI.....	710
20.3.8 Back Up and Restore via REST API.....	710
20.3.9 Install or Upgrade Helm Plugins.....	711
20.3.10 Helm Version 2 Back Up and Restore Plugins Usage.....	711
20.3.11 Helm Version 3 Back Up and Restore Plugins Usage.....	715
20.3.12 Scheduled Backup.....	718
20.3.13 Retrieve RequestHistory.....	724
20.3.14 Retrieve backup_id.....	724
20.3.15 Backup and Recovery Scope.....	726
20.3.16 The Containerized Backup and Restore tool-a sidecar.....	728
20.3.16.1 Pre-defined the Containerized Backup and Restore tool-a sidecar.....	729
20.3.16.2 Attach volume for sidecar.....	736
20.3.16.3 Support of Special Characters for Login and Password.....	737
20.3.17 BrPolicy.....	738
20.3.18 Async.....	743
20.3.19 Containerized Backup and Restore tool Installation Configuration.....	745
20.3.20 Alarm Management.....	768
20.3.21 The Containerized Backup and Restore Tool API Reference Guide.....	769
20.4 Backup.....	775
20.4.1 Nokia Container Services Backup.....	775
20.4.2 CM backup.....	776
20.5 NCS Manager Restore Central Deployment Type.....	777
20.6 NCS Manager Reinstall/Restore Procedure Cluster Deployment Type.....	777
20.7 NCS on Virtualized Deployments: Backup the Deploy-server using Snapshots.....	778
20.7.1 Introduction.....	778
20.7.2 NCS on top of Openstack deployment.....	778
20.7.3 NCS on top of vCenter deployment.....	780
20.7.4 NCS on top of vCloud deployment.....	781
20.8 Troubleshooting.....	782
20.8.1 Administration or Operations Issues.....	782
20.8.1.1 glusterfs volume almost full, occupied nearly 85% disk, manual cleaning required.....	782
20.8.1.2 glusterfs self heal daemon occupied nearly 50% CPU, Glusterfs volume self heal daemon occupied too CPU, manual cleaning required.....	783
20.8.1.3 With restoration, errors are encountered due to previous request history.....	785
20.8.2 Upgrade/Rollback failures.....	785
21 Cluster Scale-in and Scale-out operations for Bare-metal implementation.....	787
21.1 Scale-out and scale-in failed for one worker node.....	787
21.2 Baremetal Cluster Scale-In.....	788
21.2.1 Removing a Node.....	788
21.3 Baremetal Cluster Scale-out.....	789
21.3.1 General.....	789
21.3.1.1 Hardware.....	789
21.3.2 Customize Host Groups.....	790

21.3.2.1 Monitor BM.....	790
21.3.3 IPMI.....	792
21.3.3.1 Add IPMIs.....	792
21.3.3.2 Multiple pools.....	793
21.3.3.3 Define Rack Names.....	793
21.3.3.4 Define Zone Labels.....	793
21.3.3.5 Assign Nodes.....	794
22 Kubernetes configuration.....	795
22.1 Configure ImagePullSecret.....	795
22.2 Topology Manager.....	795
22.2.1 How the Topology Manager works.....	796
22.2.1.1 Topology Manager Scopes.....	797
22.2.1.2 Topology Manager Policies.....	798
22.2.2 Configurations of Kubernetes Topology Manager in NCS.....	799
22.2.2.1 Provisioning the Topology Manager policy on the NCS manager.....	801
22.2.2.2 Examples for SRIOV and CPU usage with Topology Manager.....	801
22.2.2.3 Resource Pooling.....	807
22.2.3 Known Limitations.....	809
23 Appendix: Example for cpro-values.yaml files.....	810
24 Appendix: Example for grafana-values.yaml files.....	870
25 Appendix: Example of cpro-rest-api-srv-values.yaml.....	892
26 Appendix: Example of cpro-gen3gpp-values.yaml.....	900
27 Appendix: Example Telemetry profile yaml.....	918
28 Appendix: Baremetal Security - Communications matrix.....	989
29 Appendix: Changing the BCMT server timezone from UTC to US (PST, CST, EST).....	1002

List of Figures

Figure 1:	NCS Manager.....	72
Figure 2:	Network topology.....	124
Figure 3:	Calico routing.....	130
Figure 4:	External Ingress /Egress configuration.....	147
Figure 5:	Customize Host Groups.....	148
Figure 6:	Customize Host Groups.....	154
Figure 7:	Control/Worker node with direct acces to an external network.....	180
Figure 8:	Control/Worker node with indirect access to an external nework.....	181
Figure 9:	Istio Multicluster.....	218
Figure 10:	Jaeger direct-to-storage architecture.....	226
Figure 11:	Trace and spans on Jaeger portal.....	235
Figure 12:	Update NTP example.....	251
Figure 13:	DNS query architecture.....	254
Figure 14:	Update DNS example.....	256
Figure 15:	Keepalived use cases (2/1).....	258
Figure 16:	Keepalived use cases (2/2).....	258
Figure 17:	NCS support for GlusterFS.....	317
Figure 18:	Harbor architecture.....	388
Figure 19:	Load balancer.....	418
Figure 20:	Project/tenant level RBAC roles.....	434
Figure 21:	Users.....	435
Figure 22:	Zabbix Proxy.....	583
Figure 23:	Proxies.....	583
Figure 24:	List of the Containerized Backup and Restore tool APIs.....	770
Figure 25:	Create a snapshot.....	779
Figure 26:	Create Snapshot.....	779
Figure 27:	Name snapshot.....	780
Figure 28:	Snapshot creation.....	781
Figure 29:	vCenter additional resources.....	781
Figure 30:	Create a Snapshot.....	782

List of Tables

Table 1:	Product and Release documents.....	19
Table 2:	Installation and Upgrade documents.....	19
Table 3:	System Acceptance Tests (SATE) documents.....	20
Table 4:	Integrating documents.....	20
Table 5:	Operation and Administration documents.....	21
Table 6:	Legal, Safety, and Environment documents.....	21
Table 7:	Cluster node minimum requirements.....	25
Table 8:	Common cluster configurations.....	27
Table 9:	NCS default testings.....	28
Table 10:	Turn on Istio.....	29
Table 11:	Node role.....	29
Table 12:	NCS default settings.....	29
Table 13:	Turn on Istio.....	30
Table 14:	Node role.....	30
Table 15:	Node Role.....	30
Table 16:	Supported special characters.....	36
Table 17:	Default RBAC roles.....	43
Table 18:	Account parameters description.....	43
Table 19:	NCS Cluster Roles.....	51
Table 20:	Network plugin comparison.....	127
Table 21:	Meta plugins.....	127
Table 22:	IPAM plugins.....	128
Table 23:	SR-IOV Resource Naming for static resource pooling (default).....	135
Table 24:	SR-IOV Resource Naming for dynamic resource pooling.....	135
Table 25:	SR-IOV Resource Naming for FPGA FEC devices.....	135
Table 26:	Multus config file and result.....	140
Table 27:	Multus config file and result.....	149
Table 28:	Multus config file and result.....	155
Table 29:	NCS CLI Category.....	172
Table 30:	CRDs changes.....	185
Table 31:	NCS Istio Components.....	188
Table 32:	File and secret names.....	192
Table 33:	Components in NCS Jaeger.....	227
Table 34:	Default StorageClass.....	291
Table 35:	Steps.....	332
Table 36:	crd fields explanation.....	334
Table 37:	s3 target fields.....	336

Table 38:	Default storage class.....	344
Table 39:	Storage class name.....	345
Table 40:	Services list.....	429
Table 41:	NodeCondition.....	463
Table 42:	API-server metrics.....	467
Table 43:	ETCD metrics.....	474
Table 44:	Kubernetes metrics.....	480
Table 45:	cAdviser metrics.....	487
Table 46:	Node metrics.....	490
Table 47:	Local_volume metrics.....	501
Table 48:	Ceph metrics.....	502
Table 49:	CoreDNS metrics.....	514
Table 50:	Golang metrics.....	515
Table 51:	Calico metrics.....	518
Table 52:	Fluentd metrics.....	521
Table 53:	Prometheus metrics.....	522
Table 54:	CPRO Alertmanager metrics.....	530
Table 55:	General metrics.....	533
Table 56:	Kube-apiserver alert rules.....	534
Table 57:	ETCD alert rules.....	534
Table 58:	Kube-state-metrics alert rules.....	535
Table 59:	Kubelet alert rules.....	536
Table 60:	Node alert rules.....	536
Table 61:	Fluentd alert rules.....	537
Table 62:	Calico alert rules.....	537
Table 63:	Service alert rules.....	538
Table 64:	Prometheus alert rules.....	538
Table 65:	Grafana alert rules.....	539
Table 66:	CPRO Alertmanager alert rules.....	539
Table 67:	General alert rules.....	539
Table 68:	Port for communication with NetAct.....	545
Table 69:	XML Field Descriptions.....	552
Table 70:	Matrix.....	726
Table 71:	Baremetal Cluster Scale-In Parameters.....	788
Table 72:	Hardware Parameters.....	790
Table 73:	Monitor BM Parameters.....	790
Table 74:	Multiple pools.....	793
Table 75:	Rack names.....	793
Table 76:	Topology Manager configurations.....	799

Table 77: fsfsfsf.....	802
Table 78: Security Communications Matrix.....	989
Table 79: Ports that require external access.....	1000

1 Guide to documentation

1.1 Documentation conventions

 **Warning:** Warning provides information that can affect performance or service.

 **Note:** Note provides information that is important and may require action.

 **Tip:** Tip provides information that is of special interest.

1.2 Document list for Nokia Container Services (NCS)

This topic lists all the documents available in the Nokia Container Services (NCS) document set. You can find related information such as document issue number or identifiers.

Table 1: Product and Release documents

Document title	Filename	Issue	Document Identifier
Nokia Guide to Documentation	NCS_20FP2-Guide_to_Documentation.pdf	1-0	DN1000048602
Release Notes	NCS_20FP2_V5-Release_Notes.pdf	1-7	DN1000040970
Product Description	NCS_20FP2-Product_Description_ver_6.pdf	1-6	DN1000039717

Table 2: Installation and Upgrade documents

Document title	Filename	Issue	Document Identifier
OpenStack Installation	NCS_20FP2-OpenStack_Installation.pdf	1-2	DN1000041003

Table 2: Installation and Upgrade documents (continued)

Document title	Filename	Issue	Document Identifier
VMware Installation	NCS_20FP2_VMware_Installation.pdf	1-2	DN1000041014
Bare-metal Installation	NCS_20FP2_V6-Bare_metal_Installation.pdf	1-6	DN1000051561
Upgrade Guide	NCS_20FP2_V2-Upgrading.pdf	1-2	DN1000041036

Table 3: System Acceptance Tests (SATE) documents

Document title	Filename	Issue	Document Identifier
System Acceptance Tests - OpenStack	NCS_20FP2-SATE_OS.pdf	1-0	DN1000047602
System Acceptance Tests - Bare Metal	NCS_20FP2_V2-SATE_BM.pdf	1-1	DN1000047591

Table 4: Integrating documents

Document title	Filename	Issue	Document Identifier
Onboarding and Managing Applications	NCS_20FP2-Onboard_Manage_Apps.pdf	1-0	DN1000052737
Integration Guide	NCS_20FP2-Integrating.pdf	1-1	DN1000041047

Table 5: Operation and Administration documents

Document title	Filename	Issue	Document Identifier
Operations and Administration Manual	NCS_20FP2-Operations_and_Administration_Manual.pdf	1-7	DN1000040981
NCS Manager on Bare Metal	NCS_20FP2-NCS_Manager_Guide.pdf	1-2	DN1000040981
Security Reference Guide for NCS	NCS_20FP2-Security_Reference.pdf	1-0	DN1000055705

Table 6: Legal, Safety, and Environment documents

Document title	Filename	Issue	Document Identifier
NCS Privacy Considerations	NCS_20FP2-Privacy_Considerations.pdf	1-1	DN1000040947
Free and Open Source Software Information	NCS_20FP2-Licenses_of_FOSS.pdf	1-1	DN1000040969

1.3 Documentation feedback

If you have questions or comments about this documentation, contact:

documentation.feedback@nokia.com

2 Overview

The Nokia Container Services is a highly available platform providing Container-as-a-Service (CaaS) functionality for on premises deployment of containerized applications. NCS is an infrastructure independent solution that can operate in bare metal or cloud environments, such as AWS and VMware. NCS is security hardened and includes log and monitoring interfaces.

NCS uses Kubernetes as the container orchestration system for automating deployment, scaling and basic management functionality. The management layer includes additional functions above Kubernetes, including package management (Helm), service mesh (Istio), backup, and a REST API to expose the management functions. NCS leverages integrated Kubernetes pluggable interfaces for network and storage options.

3 Getting started

3.1 Cluster definition

NCS comprises of a Deployment Server and a cluster of nodes, which can be physical or virtual systems, depending on the production environment.

The Deployment Server is used to perform the initial deployment of NCS software components, and can be used to instantiate the cluster nodes in cloud environments. It is not considered as part of the cluster.

Cluster nodes are assigned one or more node roles, depending on the cluster configuration. There are 4 node roles: Control, Worker, Edge and Storage.

- Control nodes run the Kubernetes management processes and the Docker registry. These nodes also perform cluster, node and application management.
- Worker nodes run the application service loads.
- Edge nodes normally manage the communication between the cluster and external service providers. They provide an interface to the public network while managing access control and perform load balancing.
- Storage nodes provide access to local storage or GlusterFS volumes. Storage nodes can be dedicated, standalone nodes, or combined with Control, Worker or Edge roles.

3.1.1 Storage node

Storage nodes can be defined independently from Control nodes, or be combined with Worker and Edge nodes. Previously, the Storage role had to be combined with Control nodes.

The dedicated Storage node is used for the following:

- GlusterFS storage back end for applications (configure raw device).
- Rook storage back end for applications (configure rook device).
- Docker registry or yum repository, NCS shared storage, and glusterFS storage back end for NCS.
- Local storage, if configured.

The combined Storage with Worker or Edge nodes is used for rook storage back end only. Raw devices configured on a worker or edge combined storage node is not used for GlusterFS.

3.1.1.1 Limitations

- Configuring dedicated Storage nodes and combined Storage nodes (with Control, Worker or Edge nodes) is not supported.
- Dedicated Storage nodes do not support scale-in or scale-out events.

- Upgrading from a combined Control or Storage node to a dedicated Storage node configuration is not supported.

3.1.2 Configuration

Storage nodes can be configured as being either 1 or 3 nodes. For applications to use GlusterFS, 3 nodes are required.

If there is a single node, only one Docker registry and yum repo is deployed (this was previously deployed on the Control node.)

3.1.3 Networking

An internal storage network is used to connect storage resources to other nodes in the cluster, allowing for customized quality of service (QoS) and MTU configurations. This network can be combined with other internal networks. If a separate internal storage network is defined, it is used for all nodes to connect to the Docker registry, yum repo and GlusterFS. Otherwise, the internal service network is marked as the default internal storage network.

3.2 Cluster node requirements

The Deployment Server requires at least 2 CPUs, 8GB of memory, and 100GB of disk space.

Most of the NCS services are run on Control nodes. Additional resources may be needed for Worker nodes, depending on application requirements.

 **Note:** In case of an overlap in IP ranges between existing and new entities, do not create duplicate IPs in the cluster.

The server does not support removal or reconfiguration of existing subnets. Make sure the IP range of the new subnet matches IP ranges (explicit or as whole netmasks) with the new and existing subnets.

Overlapping IP ranges are not allowed. There is risk of having the same IP on different VLANs, and on the same server, which may cause routing issues for the router locally.

Validation should be able to mostly reuse:

```
_validate_all_cluster_addresses from/cmdutils/networking/config.py
```

 **Warning:** The deployment fails immediately when overlapping or out of range IP addresses are used.

Table 7: Cluster node minimum requirements

Node role	CPU	Memory (GB)	Storage (GB)
Control	4	8	300
Worker	4	4	200
Edge	8	4	200
Storage	4	8	300

ETCD Disk requirements



Warning:

- Make sure the disk I/O is good when installing an NCS cluster. ETCD is disk sensitive.
- If the ETCD issue is caused by a hardware mismatch (slow disk IO that does not meet criteria), the user can refer to the ETCD performance impact documentation, but these type of issues are outside of the scope of standard Nokia technical services for NCS operations.

In production environments hardware guidelines must be considered for proper operations.

These guidelines support the creation of a robust production deployment. All deployments must be tested with simulated workloads before running in production.

Fast disks are the most critical factor for etcd deployment performance and stability. A slow disk increases ETCD request latency and potentially causes cluster stability defects. Typically 50 sequential IOPS (for example a 7200 RPM disk) is required.

For clusters under heavy load, 500 sequential IOPS (local SSD or high performance virtualized block device) is recommended.



Tip: Backing storage of ETCD with an SSD resolves this particular issue. An SSD usually provides lower write latencies with less variance than a spinning hard drive. All these improve the stability and reliability of ETCD.

If using spinning hard drives, the fastest disks are recommended (high-end 15,000 RPM). Where available, a vendor published 6,000 IOPS parameter is the minimum requirement.

It is recommended to run the benchmark test when setting up an ETCD cluster for the first time in a new environment to ensure the cluster achieves adequate performance; cluster latency and throughput can be sensitive to minor environment differences.



Tip: Details of open source ETCD documentation can be retrieved from [Hardware recommendations](#).

In general, to test if a disk is fast enough for ETCD, a benchmarking tool such as *fio* can be used.

As a rule of thumb at 99th percentile of this metric it should be below 10ms and no r/w would take more than 2s for storage to consider a storage device fast enough.

Example command:

```
mkdir test-data;fio --rw=write --ioengine=sync --fdatasync=1 --directory=test-data --size=1024m --bs=2300 --name=mytest
```

In the output it becomes visible if the 99th percentile of *fdatasync* durations are less than 10ms.

Example output:

```
mytest: (g=0): rw=write, bs=(R) 2300B-2300B, (W) 2300B-2300B, (T) 2300B-2300B,
  ioengine=sync, iodepth=1
fio-3.7
Starting 1 process
Jobs: 1 (f=1): [W(1)][100.0%][r=0KiB/s,w=3196KiB/s][r=0,w=1423 IOPS][eta
  00m:00s]
mytest: (groupid=0, jobs=1): err= 0: pid=3305: Tue May 12 08:22:09 2020
  write: IOPS=1126, BW=2531KiB/s (2592kB/s)(22.0MiB/8901msec)
    clat (usec): min=3, max=23643, avg=243.00, stdev=683.77
    lat (usec): min=4, max=23644, avg=244.12, stdev=684.13
    clat percentiles (usec):
      | 1.00th=[     4], 5.00th=[     5], 10.00th=[     6], 20.00th=[     7],
      | 30.00th=[     7], 40.00th=[    13], 50.00th=[   221], 60.00th=[   269],
      | 70.00th=[  334], 80.00th=[  396], 90.00th=[  469], 95.00th=[  515],
      | 99.00th=[  709], 99.50th=[ 2900], 99.90th=[10421], 99.95th=[13960],
      | 99.99th=[23462]
    bw ( KiB/s): min= 1013, max= 3184, per=100.00%, avg=2648.63, stdev=592.06,
samples=16
  iops          : min=  451, max= 1418, avg=1179.44, stdev=263.64, samples=16
  lat (usec)    : 4=1.22%, 10=37.50%, 20=4.65%, 50=0.39%, 100=0.02%
  lat (usec)    : 250=11.53%, 500=38.21%, 750=5.54%, 1000=0.17%
  lat (msec)    : 2=0.22%, 4=0.12%, 10=0.33%, 20=0.10%, 50=0.02%
  fsync/fdatasync/sync_file_range:
    sync (usec): min=271, max=672245, avg=634.83, stdev=7981.19
    sync percentiles (usec):
      | 1.00th=[   302], 5.00th=[   338], 10.00th=[   367], 20.00th=[   408],
      | 30.00th=[   437], 40.00th=[   465], 50.00th=[   486], 60.00th=[   515],
      | 70.00th=[   553], 80.00th=[   586], 90.00th=[   635], 95.00th=[   693],
      | 99.00th=[   816], 99.50th=[  1156], 99.90th=[  7963], 99.95th=[  8979],
      | 99.99th=[429917]
  cpu           : usr=1.37%, sys=5.13%, ctx=41304, majf=0, minf=16
  IO depths     : 1=200.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
    submit       : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%,
  >=64=0.0%
```

```

complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%,
>=64=0.0%
issued rwts: total=0,10030,0,0 short=10029,0,0,0 dropped=0,0,0,0
latency    : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
WRITE: bw=2531KiB/s (2592kB/s), 2531KiB/s-2531KiB/s (2592kB/s-2592kB/s),
io=22.0MiB (23.1MB), run=8901-8901msec

Disk stats (read/write):
vdc: ios=5589/19908, merge=0/1, ticks=2252/6068, in_queue=1719, util=80.38%

```

3.3 Cluster configuration

To support high availability (HA), it is recommended to define at least 3 Control nodes, 2 Workers nodes, 2 Edge nodes and 3 Storage nodes.

Table 8: Common cluster configurations

Node	Single mode (HA)	Single mode (w/o HA)	Mixed mode (HA)	Mixed mode (w/o HA)
1	Control	Control	Control, Worker, Storage	Control, Worker, Edge, Storage
2	Control	Worker	Control, Worker, Storage	
3	Control	Edge	Control, Worker, Edge, Storage	
4	Worker	Storage	Worker, Edge	
5	Worker			
6	Edge			
7	Edge			
8	Storage			

Table 8: Common cluster configurations (continued)

Node	Single mode (HA)	Single mode (w/o HA)	Mixed mode (HA)	Mixed mode (w/o HA)
9	Storage			
10	Storage			

3.4 Run-time resource usage

The following tables show a summarized resource usage of different nodes in an NCS cluster. The CPU usage is based on the following model:

```
CPU: Topology: 4x Single Core model: Intel Xeon (Skylake IBRS) bits: 64 type:  
SMP L2 cache: 32.0 MiB  
Speed: 2095 MHz min/max: N/A Core speeds (MHz): 1: 2095 2: 2095
```

Special notes for nodes that are assigned storage role, CPU and memory resource usage will vary depending on the IOPS load of ROOKIO or GlusterFS. Please consider reservations to ensure there is no starve IO.

Take ROOKIO/Ceph cluster as an example, Red Hat recommends that each OSD container has at least 1 CPU core and 5GB RAM, each mon container has at least 1 CPU core and 3GB RAM. According to the test result from NSW CTO, 2 CPU cores and 6G RAM are enough to keep the bottleneck lies in disk (Intel SSD DC P3500, PCIe NVMe 3.0 x4).

3.4.1 Single mode

- NCS default settings

Table 9: NCS default testings

Node Role	CPU	Memory	Disk
Control	10%	2.2GB	11GB
Worker/Edge	2%	350MB	5GB

- Turn on Istio

Table 10: Turn on Istio

Node Role	CPU	Memory	Disk
Control	15%	3GB	12GB
Worker/Edge	2.5%	350MB	5GB

- Turn on ROOKIO, GlusterFS and DANM

Table 11: Node role

Node Role	CPU	Memory	Disk
Control	16%	2.3GB	12GB
Storage	6%	1.1G	10GB
Worker/Edge	5%	500MB	5GB

3.4.2 Mixed Mode

- NCS default settings

Table 12: NCS default settings

Node Role	CPU	Memory	Disk
Control/Worker/ Edge/Storage	14%	2.3GB	11GB

- Turn on Istio

Table 13: Turn on Istio

Node Role	CPU	Memory	Disk
Control/Worker/ Edge/Storage	14%	3GB	11GB

- Turn on ROOKIO, GlusterFS and DANM

Table 14: Node role

Node Role	CPU	Memory	Disk
Control/Worker/ Edge/Storage	16%	3.4GB	13GB

- Turn on ROOKIO, GlusterFS, DANM and Istio

Table 15: Node Role

Node Role	CPU	Memory	Disk
Control/Worker/ Edge/Storage	20%	4GB	13GB

In order to provide more reliable scheduling and minimize node resource over commitment, each node can reserve a portion of its resources for use by all these underlying services. The minimum resource usage listed above can be used as the tuning input of `--system-reserved` or `--kube-reserved` in kubelet.

4 NCS command line (CLI)

NCS provides a command line interface (CLI) for use in updating cluster configuration information. There are multiple ways to access the CLI:

- Via a Control node.
- Via the Deployment Server.

5 NCS API

NCS provides methods to access an NCS API and a Kubernetes API, and also support customization of NCS port.

6 Dashboards

6.1 Accessing the NCS dashboard

Log in to the NCS dashboard by using port 8082 and the public IP address of one of the Control nodes.

URL syntax:

```
https://<control node IP>:8082
```

Example:

```
https://10.0.2.1:8082
```



Note: For Baremetal deployment, please use port 8084 instead of 8082.



Note: If this is the user's first login, they may be required to reset the default password.

6.2 Accessing the Kubernetes dashboard

Log into the NCS dashboard and go to the **PORTAL** menu and click on **Kubernetes PORTAL** option.



Note: If you see the error *Your connection is not private*, click **Advanced** and choose **Proceed to <IP address>**.

6.3 Accessing the Keycloak dashboard

Log into the NCS dashboard and go to the **PORTAL** menu and click the **KEYCLOAK PORTAL** option.

Users must log in to Keycloak Dashboard using an ncs user account again due to Single-Sign-On through NCS Web Portal not implemented yet.



Note:

Do not change the default password of the Keycloak admin account.

6.4 Integrating CSF dashboard components into NCS

CSF components such as CPRO, Grafana and Kibana can register their dashboards with NCS Web Portal.

1. Get an access token through the login API:

Example commands:

```
accessToken=`curl -s https://<your_control_ip>:8082/ncm/api/v1/users/login -X POST -d '{"username": "ncs-admin", "password": "<your_password>"}' -H"Content-Type:application/json" | jq ".accessToken" | sed 's//g'`
```

2. Register Dashboard URL using accessToken:**Example commands:**

```
curl -X POST -H"Content-Type:application/json" -d'{"gui": "Kibana Dashboard", "endpoint": "https://<Kibana_dashboard_url>:<Kibana_dashboard_port>"}' https://<your_control_ip>:8082/ncm/api/v1/guis -H"Authorization: Bearer $accessToken"
```

After the above steps, users can see your dashboard in the **PORTAL** menu on NCS Web Portal.

NCS does not support CSF Components Dashboard SSO through NCS Web Portal yet. Since CSF components can install CKEY for their own purposes, users in NCS and CSF components may be different. It's impossible to implement SSO in different user systems.

NCS has embedded CKEY by default, so CSF components can use CKEY inside NCS to share the same user system. The following is the default CKEY information inside NCS:

```
server_url: https://bcmt-ckey-ckey.ncms.svc:8443/auth/  
  
ncm_realm_server_url: https://bcmt-ckey-ckey.ncms.svc:8443/auth/realms/ncs  
  
keycloak_admin_user_name: admin  
  
keycloak_admin_password: saved in etcd /BCMTClusterManager/bcmt_config/cluster_config/keycloak_admin_password  
  
keycloak_root_ca: /opt/bcmt/config/bcmt-ckey/certs/ckey.pem  
  
keycloak_root_ca_key: /opt/bcmt/config/bcmt-ckey/certs/ckey-key.pem  
  
keycloak.jks: /opt/bcmt/config/bcmt-ckey/certs/keycloak.jks
```

7 User management

The Nokia Container Services (NCS) uses Web Single-Sign-On (keycloak) to perform user management. This service is started as a Kubernetes pod on one of the Control nodes. The keycloak pod has a replica. If the Control node which has the primary keycloak pod is down, another Control node restarts the keycloak pod. The data for keycloak is in a shared storage area so all Control nodes have access to the same data.

 **Note:** If there is a schedule existing for NCS cluster backups, ncs-admin user password change will make them fail unless the 'ncs cluster backup' command/Cluster backup flow (from UI) is invoked again.

Steps that CSFP needs to be executed when the password for the username ncs-admin is changed are the following:

Update password in secrets when ncs-admin password has been updated:

- ncs-ansible-runner creates a Kubernetes secret object with the name ncs-secret. It stores the password as data.password and the encrypted config.json as data.config in this secret.
- Operator, might have update ncs-admin user credentials through NCS portal, but this ncs-secret is not synchronized. Follow the following procedure to synchronize:

```
$ # Login to one of the control nodes
$ read -sp "Enter password: " new_password
$ Enter password:
$ kubectl patch secret -n ncms ncs-secret --type='json' -p='[ { "op" :
"replace", "path" : "/data/password" , "value" : "'$(echo -n $new_password
| base64 )'" } ]'
$ kubectl patch secret -n ncms ncs-secret --type='json' -p='[ { "op" :
"remove", "path" : "/data/config" } ]'
```

If the CSFP is not running in the mentioned NCS cluster, these steps are not needed.

 **Note:** LDAP(S) can be integrated into Keycloak, which allows federated (LDAP/AD) users to login into NCS Portal. See *Server Administration Guide*, section **LDAP and Active Directory** at www.keycloak.org for more details.

7.1 Initial user login

Whenever a tenant is created by the cluster admin, a default tenant admin user is created: tenant-name - admin. For example for tenant team1, the admin user name is: team1-admin. The default password for tenant admin is: ncs@CSF_k8s, which is the same as the default password of ncs-admin user. For the first login, the password must be reset either via cli or the management portal NCS CLI or by logging in to the NCS Dashboard.

About this task

This task describes how to change the default password using NCS CLI.

1. Log in to one of the Control nodes.
2. Log in to the NCS CLI using the default username and password. You are prompted to reset the default password.

```
ncs user login --username=ncs-admin --password=ncs@CSF_k8s
```

Example output:

```
Please reset password first
Please use below command to reset password
ncs user password reset --user_id=8da89207-1d18-45a3-9351-570b3a74592c --code=LIOZVuC --password=
```

3. Use **ncs user password-policy get** to display the current password policy details.
4. Reset the password using the provided command string, including the **user_id**, and specify a valid value for the new password.



Note: The following ASCII characters are supported by the password field:

Table 16: Supported special characters

Name of the Character	Character
at sign	@
percent sign	%
plus sign	+
single quotation mark	'
exclamation point	!
number sign	#
dollar sign	\$
caret	^

Table 16: Supported special characters (continued)

Name of the Character	Character
question mark	?
colon	:
semicolon	;
comma	,
left parenthesis	(
right parenthesis)
left brace	{
right brace	}
left bracket	[
right bracket	
tilde	~
hyphen	-
underscore	_
period	.
star	*
ampersand	&
pipe	

Table 16: Supported special characters (continued)

Name of the Character	Character
equals	=

Example command:

```
ncs user password reset --user_id=8da89207-1d18-45a3-9351-570b3a74592c --code=LIOZVuC --
password>NewPassword@1234!
```

Example output:

```
Reset password successfully
```

5. Optional: To prevent the password from being displayed on the screen, the environment value NEW_NCSPASSWORD can be defined to store the new password string.

- a) Define the NEW_NCSPASSWORD environment variable.

```
export NEW_NCSPASSWORD=<password>
```

- b) Execute the reset command without specifying the password field.

```
ncs user password reset --user_id=8da89207-1d18-45a3-9351-570b3a74592c --code=LIOZVuC
```

Example output:

```
Reset password successfully
```

7.2 Managing users

1. Log in to one of the Control nodes.
2. Use the **ncs user** command to add, update, retrieve, or delete user information.

```
NAME: ncs user - user commands

USAGE:
  ncs user command [command options] [arguments...]

COMMANDS:
  add           add a new user
  update        update user info
  list          get user info
  show          show user info
  login         change user
  delete        delete user info
```

```
password      password operation
password-policy password-policy operations
```

Example

```
ncs user list
```

Example output:

```
{
  "userList": [
    {
      "username": "ncs-user",
      "firstName": "",
      "lastName": "",
      "enabled": true,
      "email": "",
      "id": "8540ec27-623a-406e-b483-5c4f2f116aa9"
    },
    {
      "username": "ncs-admin",
      "firstName": "",
      "lastName": "",
      "enabled": true,
      "email": "ncs-admin@nokia.com",
      "id": "8da89207-1d18-45a3-9351-570b3a74592c"
    }
  ],
  "total": 2,
  "pageSize": 2,
  "pageNo": 1
}
```

7.3 Adding a user

About this task

Use the **ncs user add** command to add a new user.

```
NAME:
  ncs user add - add a new user

USAGE:
  ncs user add [command options] [arguments...]

OPTIONS:
  --username <user name>      specifies the user name
  --email    <email>          specifies the user email of the user(optional)
  --firstname <user name>     specifies first name of the user(optional)
  --lastname  <user name>     specifies last name of the user(optional)
```

1. Log in to a Control node.

2. Use the **ncs user add** command:

```
ncs user add --username <user name> --email <email>
```

Example:

```
ncs user add --username=ncs-user
```

3. The default password is **ncs@CSF_k8s**. To reset this password, use the **ncs user login** command.

```
ncs user login --username=ncs-user --password=ncs@CSF_k8s
```

Example output:

```
Please reset password first  
Please use below command to reset password  
ncs user password reset --user_id=8540ec27-623a-406e-b483-5c4f2f116aa9 --code=idnufEEEd --password=
```

4. Reset the password using the provided command string and specify a new password.

Example command:

```
ncs user password reset --user_id=8540ec27-623a-406e-b483-5c4f2f116aa9 --code=idnufEEEd --  
password>NewPassword@1234!
```

Example output:

```
Reset password successfully
```

5. Optional: To prevent the password from being displayed on the screen, the environment value **NEW_NCPASSWORD** can be defined to store the new password string.

a) Define the **NEW_NCPASSWORD** environment variable

```
export NEW_NCPASSWORD=<password>
```

b) Execute the reset command without specifying the password field.

```
ncs user password reset --user_id=8540ec27-623a-406e-b483-5c4f2f116aa9 --code=idnufEEEd
```

Example output:

```
Reset password successfully
```

7.3.1 Add extra user on cluster node

The default user in Cluster nodes are root and cloud-user. Take following procedure to add more users.

Pay attention to the security when adding more users such as what usage purpose and permission of the new users are.

There are two parts, the user needs to decide what to do with the new user to select corresponding procedure.

1. Add user to Linux OS.

a. Option1: Add user without sudo.

```
#useradd user3
```

b. Option2: Add user with sudo, but cannot login directly.

```
Use root login:  
1.# useradd newuser --comment 'ssh_user' --groups wheel --shell /bin/  
bash -m  
// If want to add more permission of newuser, add it to group  
adm,systemd-journal,  
//like: useradd cloud-user --comment 'ssh_user' --groups adm,systemd-  
journal,wheel --shell /bin/bash -m  
  
2.# vi /etc/sudoers.d/90-cloud-init-users  
//add line like:  
    newuser ALL=(ALL) NOPASSWD:ALL  
3.# passwd -l newuser  
//switch to newuser  
  
4.# su - newuser  
//sudo to root  
5.# sudo su -
```

c. Option3: Add user with sudo, and can login directly.

The following is based on the second option

```
1. #sudo newuser  
2. # mkdir .ssh -m 700 && cd .ssh  
3. # touch authorized_keys && chmod 644 authorized_keys  
4. # vi authorized_keys  
//copy and add your public key from /root/.ssh/authorized_keys.
```

2. Add user permission to access Kubernetes cluster.

Add user with a kubectl permission. The user now has permission to access the kubernetes cluster.

```
1. $ sudo newuser  
2. $ mkdir .kube -m 700
```

```

3. $ sudo cp /root/.kube/config /etc/kubernetes/ssl/ca.pem /etc/
kubernetes/ssl/cluster-admin.pem /etc/kubernetes/ssl/cluster-admin-
key.pem ./kube/ && sudo chown -R newuser:newuser .kube
4. $ vi .kube/config
    certificate-authority: /etc/kubernetes/ssl/ca.pem
    client-certificate: /etc/kubernetes/ssl/kubectl.pem
    client-key: /etc/kubernetes/ssl/kubectl-key.pem
=>
    certificate-authority: ./ca.pem
    client-certificate: ./kubectl.pem
    client-key: ./kubectl-key.pem
Example for updated configuration file:

```

```

apiVersion: v1
kind: Config
clusters:
- cluster:
  certificate-authority: ./ca.pem
  server: https://k8s-apiserver:8443
  name: bcmt-kubernetes
contexts:
- context:
  cluster: bcmt-kubernetes
  namespace: default
  user: kubectl
  name: kubectl-context
  current-context: kubectl-context
  preferences: {}
users:
- name: kubectl
  user:
    client-certificate: ./cluster-admin.pem
    client-key: ./cluster-admin-key.pem

```

```

5. $ kubectl get pods --all-namespaces
// to verify kubectl command.

```

7.4 User roles

NCS supports Role Based Access Control (RBAC).



Note: Refer to *Onboarding and Managing Applications on Nokia Container Services* and *Dashboard User Guide for Nokia Container Services (NCS)* documents for more information

regarding when multi-tenancy (MT), Harbor image and chart registry is used instead of bcmt-registry.

The following default roles are provided:

Table 17: Default RBAC roles

Role Name	Description
administrator	Administrator role for NCS, has all permissions
viewer	Viewer role for NCS, has read permissions



Note: *ncs-admin* does not have any (system or project/tenant level) Harbor RBAC roles assigned. This also means that by default *ncs-admin* is not created in Harbor. Check with NOKIA support if it is recommended to create an *ncs-admin* user account for both NCS and Harbor users.

7.5 Using a Non-Root user

Non-root users can add/modify/delete OS accounts in a single node, several nodes, role group, or all nodes, and also handle its life cycle.

For each account: password, su permission, sudo privilege, and ssh authorized keys can be configured.

Table 18: Account parameters description

Account parameters	Is it mandatory	Description
user_name	Yes	Account name
password	No	Password for the account. If you want to add it during deployment, this field needs encryption. The RSA can be recognized by Linux. NCS CLI does not need encryption. If the password is forgotten, please reset the password.

Table 18: Account parameters description (continued)

Account parameters	Is it mandatory	Description
		<p>The following commands can generate a password. Use one of them:</p> <pre data-bbox="1033 557 1314 586">openssl passwd -1</pre> <p>or</p> <pre data-bbox="1033 714 1389 887">python -c 'import crypt; print(crypt.crypt("yourpassword", crypt.METHOD_SHA512))'</pre>
su_enabled	No	<p>When <code>su enable</code>, the user can <code>su</code> to other users with the target user's password. For example,</p> <pre data-bbox="1033 1140 1394 1365">[bcmt@lilian-dev-control-01 ~]\$ su root - Enter login password: [root@lilian-dev-control-01]#</pre>
sudo_privilege	NO	<p>Sudo privilege can configure sudo permission. The sudo privilege list should start with PASSWD or NOPASSWD. For example</p> <pre data-bbox="1033 1686 1362 1837">'NOPASWD:/usr/local/bin/kubectl,!/usr/local/bin/kubectl delete'</pre> <p>When current account can run <code>sudo kubectl xxx</code>, except <code>sudo kubectl delete xxx</code></p>

Table 18: Account parameters description (continued)

Account parameters	Is it mandatory	Description
		without current account's password.
ssh_authorized_key	NO	The ssh authorized key. It is a pub key.

7.5.1 Management of Operating System users

OS accounts can be added or modified through setting the new *os_users* field in the NCS node inventory during an NCS Installation and supported either by using the NCS Manager or an alternate CLI command, if it exists. Preference is always where possible to use the NCS Manager.

Installation

A `node_inventory.json` sample is shown here:

```
{
    "ssh_private_key": "",
    "control": {
        "bcmk-k8s-control-01": {
            "oam_ip": "10.10.10.10",
            "internal_service_ip": "172.16.1.128",
            "internal_storage_ip": "172.16.1.128",
            "internal_service_ipv6": "2001:db8:0:f101::1",
            "internal_service_network_name": "BCMT-INTERNAL-SVC",
            "ssh_user": "test",
            "internal_oam_ip": "172.16.1.128",
            "group_index": "group_01",
            "data0_device": "/dev/vdb",
            "etcd_device": "/dev/vdc",
            "local_storage_devices": [ { "/dev/vdd": "ext4" }, { "/dev/vde": "xfs" } ],
            "os_users": [
                {
                    "password": "$6$K7Xuvfb9OV4JNlMY$FkA14uF.JL5pzlf9fmTceFmGBuLLHeS0wYJgmWIom9MrHL3CjI0v/MrsOCWMyb4F4Zgy9d1Fjqc5JbZMuesk./",
                    "ssh_authorized_key": "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCl6Bmlk6kHBfmtbz0CfQknu8PvEAj/h1oh2ERN9yI+tKGQJC0tI/TEYwbL/"
                }
            ]
        }
    }
}
```

```
UBfaGWldhXK381fTIY6cxnFX0BAWIJQ5Pz5bjXV4JXWH46RuU9Yfwg9xaf6TB8AWLfX41uAftfkSVFSXj
+HrdPUtWZsy+Ii1Tj7EvF5m1TMrG5sedMkx+jLis7O2dqH1P+3SGDgpGqNSgtWhFLqtoIZQ0R3L/Yk
+1V5plWFjAu8Ae2KX2Qn8VZMhE0RvDv5cc0cGRmehSraIfuul74tBgeOmjzcN06Ovd0RsJSPH7QorR5Bj2XtHw
FiTMmTp8bxVAQKhRze8J69H1",
    "su_enabled": "yes",
    "sudo_privilege": "NOPASSWD:/usr/local/bin/kubectl,/usr/local/
bin/helm",
    "user_name": "ncs-user"
},
{
    "password": "$1$F6P1EWSm$/GyrIUSpUw4ED88oumWMJ/",
    "user_name": "root"
}
]
}
},
{
    "worker": {
        "bcmt-k8s-worker-01": {
            "internal_service_ip": "172.16.1.13",
            "internal_storage_ip": "172.16.1.13",
            "internal_service_ipv6": "2001:db8:0:f101::2",
            "internal_service_network_name": "BCMT-INTERNAL-SVC",
            "ssh_user": "test",
            "internal_oam_ip": "172.16.1.13",
            "group_index": "group_02",
            "data0_device": "/dev/vdb",
            "local_storage_devices": [ { "/dev/vdd": "ext4" }, { "/dev/
vde": "xfs" } ],
            "os_users": [
{
            "password": "$6$1dJWKN30YwGQ957S
$PD.d6czO.JUlSAENbIlzobrtkdFIUYvwKlfqFW16o9eWhAShrnhYdpseD73a39SExdI7/4nCpKZ6htWMrRXgf.",
                "sshAuthorizedKey": "ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCyd1U9yqm2LFgr/
dUF9G00vmuBf9+US+k6kNCjeiJiY4Hnux4Nj+v1w2jLiHm/
mNRmD003qzXVQvV6oSCBBwmYsgCVWcn23fTSxcExJPdbFB09XZVm8KKyHBfKraHSKenFFksxNg3m4nuf jueX9ZNTN
+kEYFDszw/nCmiywWhs5AIb3R0tq2347AF+ONnA96ApTnaCjRBCNZHVnKbhTm/
dPYVIMFdg4sNcDSK21mFfgQVd8doHqEViB0P",
                "su_enabled": "no",
                "sudo_privilege": "NOPASSWD:/usr/local/bin/kubectl,!/usr/
local/bin/kubectl delete",
                "user_name": "ncs-user"
}
,
{

```

```

        "sshAuthorizedKey": "ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQC2OJaDIUfWVDT1Ts3fLDDzebmJ8UPs1VKafN0h099Tm9yVYIw/
kMRWkb4bgLn5Qu0W8+vA3F0mWncjU6+LDkeC3bnsqgFU9252GVgkVVVNRSdlPbBZEW70Odh5CCNXMYemFMOtG
u4L4F4kas41yM1sYvPtMj03rqXf9qcGfnSilgoVapTP6VkgRitNAXPULCDahiJy4fk6J70Qp5+cgBG7fRdvkdg
+FTd0zkKiHsflcB47qFaQihicxWCyausvT",
        "suEnabled": "yes",
        "userName": "debug-user"
    },
    {
        "password": "$1$ViOb9iT$MVLFmlJQ/FcCqxelEpNyD1",
        "userName": "root"
    }
]
}
},
"edge": {
    "bcmt-k8s-edge-01": {
        "internalServiceIp": "172.16.1.129",
        "internalStorageIp": "172.16.1.129",
        "internalServiceIpv6": "2001:db8:0:f101::3",
        "internalServiceNetworkName": "BCMT-INTERNAL-SVC",
        "sshUser": "test",
        "groupIndex": "group_03",
        "internalOamIp": "172.16.1.129",
        "data0Device": "/dev/vdb",
        "localStorageDevices": [ { "/dev/vdd": "ext4" }, { "/dev/
vde": "xfs" } ]
    }
},
{
    "storage": {
        "bcmt-k8s-control-01": {
            "internalServiceIp": "172.16.1.128",
            "internalStorageIp": "172.16.1.128",
            "internalServiceIpv6": "2001:db8:0:f101::1",
            "internalServiceNetworkName": "BCMT-INTERNAL-SVC",
            "sshUser": "test",
            "groupIndex": "group_01",
            "internalOamIp": "172.16.1.128",
            "rawDevices": [ "/dev/vdc", "/dev/sdd1" ],
            "rookStorageDevices": [
{
            "/dev/vde": "rook-cluster1"
}
],
"osUsers": [

```

```
{
    "password": "$6$K7Xuvfb9OV4JNlMY
$FkA14uF.JL5pzlf9fmTceFmGBuLLHeS0wYJgmWIom9MrHL3CjI0v/
MrsOCWMyb4F4Zgy9d1Fjqc5JbZMuesk./",
        "ssh_authorized_key": "ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCl6Bmlk6kHBfmtbz0CfQknu8PvEAj/h1oh2ERN9yI
+tKGQJC0tI/TEYwbL/
UBfaGWldhXK381fTIY6cxnFX0BAWIJQ5Pz5bjXV4JXWH46RuU9Yfwg9xaf6TB8AWLfX41uAftfkSVFSXj
+HrdPUtWZsy+Iiltj7EvF5m1TMrG5sedMkx+jLis702dqH1P+3SGDgpGqNSgtWhFLqtOIZQ0R3L/Yk
+1V5plWFjAu8Ae2KX2Qn8VZMhE0RvDv5cc0cGRmehSraIfuul74tBgeOmjzcN060vd0RsJSPH7QorR5Bj2XtHv
FiTMmTp8bxVAQKhRze8J69H1",
        "su_enabled": "yes",
        "sudo_privilege": "NOPASSWD:/usr/local/bin/kubectl,/usr/local/
bin/helm",
        "user_name": "ncs-user"
    },
    {
        "password": "$1$F6P1EWSm$/GyrIUSpUw4ED88oumWMJ/",
        "user_name": "root"
    }
]
}
}
}
```



Note:

- Cloud-user cannot be changed using this feature.
- For the root user, only the password can be modified.
- Scale-out will not handle NCS newly added accounts.
- Role group only supports the following:

- role_control
- role_edge
- role_worker
- role_storage
- role_all

There is a default account *ncs-user* in the NCS OS image, which is locked by default.

Application cases

- Change the root password during installation and inject the ssh pub key for the ncs-user.
- Enable the su permission.

- Using the ssh private key to log in node, and su root to debug.
- After debugging, use ncs user osuser to disable the su permission.
- Change the root password during installation.
- After installation, use ncs user osuser modify to modify ncs-user and set the sudo privilege list, for example, 'NOPASWD:/usr/local/bin/kubectl,/usr/local/bin/helm,/usr/local/bin/ncs'.
- After debugging, use ncs user osuser to disable the sudo permission.
- Using the ncs command to modify the root default password and the ncs-user info as a debugging user.
- Create a new user ncs-admin, and enable su permission.

Operating System Accounts Management CLI

Use the ncs user osuser command to add, modify or delete OS user information.

```
ncs user osuser
NAME:
  ncs user osuser - add/modify/delete os user

USAGE:
  ncs user osuser command [command options] [arguments...]

COMMANDS:
  add      add os user
  modify   modify os user
  delete   delete os user
  help, h  Shows a list of commands or help for one command

OPTIONS:
  --help, -h    show help (default: false)
  --version, -v print the version (default: false)
```

ncs user osuser add

Example:

```
ncs user osuser add --nodes role_edge --user_name bcmt-
user --su_enabled yes --sshAuthorizedKey 'ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCC2OJaDIUFWVDT1Ts3fLDDzebmJ8UPs1VKafN0h099Tm9yVYIw/
kMRWkb4bgLn5Qu0W8+vA3FOmWncjU6+LDkeC3bnsqgFU9252GVgkVVVNRsd1PbBZEW700dh5CCNXMYemFMOCTgGuX
u4L4F4kas4lyM1sYvPtMj03rqXf9qcGfznSi1goVapTP6VkgRitNAXPU1CDahiJy4fk6J70Qp5+cgBG7fRdvkquFV
+FTd0zkKiHsflcB47qFaQihicxWCyausvT' --password
Password:
```

```
Verifying Password:  
{ "task-status": "running" }  
...  
{ "task-status": "done" }
```



Note: *--password* specifies setting the password, click **Enter** to input password, */* means lock password.

ncs user osuser modify

Example:

```
ncs user osuser modify --nodes bcmt-edge-01 --user_name bcmt-user --su_enabled  
no --password --sudo_privilege "NOPASSWD:/usr/local/bin/kubelet"  
Password:  
Verifying Password:  
{ "task-status": "running" }  
...  
{ "task-status": "done" }
```



Note: *--password* specifies setting the password, click **Enter** to input password, */* means delete password.

ncs user osuser delete

Example:

```
ncs user osuser delete --nodes lilian-dev-edge-02 --user_name bcmt-user  
{ "task-status": "running" }  
. .  
{ "task-status": "done" }
```

8 Adding and Removing Physical Resources to a host group

This section describes how a user can add or remove a server to a host group. This can be used to increase physical resources in the cluster, or remove a faulty server. The following table describes which operations are allowed in the NCS cluster per role.

Table 19: NCS Cluster Roles

Host Group Role	Scale in	Scale out	Replace	Notes
Manager	Not Supported	Not Supported	Supported	Manual procedure
Master	Not Supported	Not Supported	Supported	
Worker	Supported	Supported	With scale in and scale out	
Storage	Supported	Supported	With scale in and scale out	You must have at least as many storage nodes as the replication count, otherwise the Ceph cluster fails. Node count \geq replication count.
Edge	Supported	Supported	With scale in and scale out	

To change the configuration of a node, you need to scale it in from a host group and scale it out to a host group which has a preferred configuration. You cannot edit existing host group configurations after deployment, but you can create new hostgroups with custom configurations on scale-out.

8.1 Scale-Out for Worker, Storage and Edge Roles

This procedure adds the given physical servers to the cluster with the host group configurations.

Note:

- It is important to perform a backup before any critical action, so that you are able to roll back actions, in case of failure.
- Scale Out or In for Worker, Storage or Edge is available on successful NCS Installation only.
- **VERY IMPORTANT:** Ensure that new compute hardware has no bootable operating system or disk partitions. The disk can be wiped with any utility that you choose.
- When scaling out a monitoring controller, the ELK database will reset and the old ELK data will be lost.
- **Multiple Ceph Pools (for AirFrame RM and OR hardware).** Before performing the scale out operation, when the multiple pools feature is enabled, it is recommended to see *Considerations when selecting Multiple Ceph Pools in Appendix - Ceph Multiple Pools for AirFrame RM and OR.*
- If scale-out has failed before the NCS steps, you need to scale-in to remove the node, and fix the error, and then run scale-out again.

Related information

[Baremetal Cluster Scale-out](#) on page 789

8.2 Scale-In for Worker, Storage and Edge Roles

This procedure will remove the chosen servers from the cluster as described in the NCS Manager Guide.



- Note:** If scale-in has failed after NCS steps, you should perform a scale-out, and fix the error and then run scale-in again.

Related information

[Baremetal Cluster Scale-In](#) on page 788

9 Cluster management

Many of these operations in a Bare Metal environment can also be performed by the **NCS Manager**. See **Nokia Container Services Manager Reference Guide for Bare Metal implementation**.

9.1 Cluster inventory

There are three key configuration files to manage the cluster inventory, variables and ssh private key. Ansible playbooks can be executed using these files.

- To view the current configuration, log in to access the NCS CLI and use the **ncs cluster inventory** command:

Example command:

```
ncs cluster inventory
```

Example output:

```
*****
inventory tarball has been downloaded in current path.
Name is inventory.tgz
*****
```

- To execute Ansible playbooks perform the following steps:

1. Unzip the tarball.

Example command:

```
tar -zxvf inventory.tgz
```

Expected output:

```
bcmt_var-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_inventory-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

2. Change the permissions for the `bcmt_vm_key-<uuid>.json` file using the `chmod` command.

Example command:

```
chmod 600 bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

3. If variables are required, execute an Ansible playbook using the following command syntax.

```
ansible-playbook -i bcmt_inventory-<uuid>.json -e@./bcmt_var-<uuid>.json run.yml
```

4. If variables are not required, remove the "-e" argument.

```
ansible-playbook -i bcmt_inventory-<uuid>.json run.yml
```

Example playbook: os_password_update.yml

```
- hosts: "{{ run_hosts|default('role_all') }}"
become: yes
gather_facts: "{{ run_gather_facts|default(True) }}"
serial: "{{ run_serial|default('100%') }}"
any_errors_fatal: "{{ run_any_errors_fatal|default(True) }}"
tasks:
- name: change root password
  user:
    name: root
    password: "{{ 'NewPassword@1234!' | password_hash('sha512') }}"
```

9.2 Cluster re-install

- When NCS deployment fails, you can re-install a cluster with the following steps:

1. *Re-try cluster install* on page 55

This option applies to the following cases:

- You do not need to modify *user_input* or *bcmt_config* files to correct configuration.
- You want to run the installation from the failed step in the last deployment.

2. *Reset cluster* on page 55

Reset cleans up the cluster data generated by the last installation. After reset is done, you can deploy again.

This option applies to the following cases:

- you need to modify *user_input* or *bcmt_config* files to correct configuration.
- You want to keep your own data in cluster nodes.

In general, reset is not efficient (costs more time than re-creating cluster nodes and may have issues, like old ssl left).

3. *Re-create cluster nodes* on page 56

Recreating cluster nodes can avoid installation failure caused by legacy data, and it is faster than *cluster reset/uninstall* for cloud VMs.

This option applies to the following cases:

- You do not need to keep your own data in cluster nodes.
- Whether to modify the *user_input* or the *bcmt_config* file or not.

After recreating cluster nodes, you can deploy again.

- After NCS is installed successfully, you may need to run [Uninstall the cluster](#) on page 121:

– Cluster uninstall used to clean cluster nodes after successful installation.

In general, uninstall is not efficient (cost more time than re-create cluster nodes and may has issues, like old ssl left).

Recommend [Re-create cluster nodes](#) on page 56 if there is no need to retain data in cluster nodes.

9.2.1 Re-try cluster install

Follow the respective commands according to your installation type.

- Use NCS cli

Command:

```
ncs cluster install --config=xxx
```

Example command:

```
ncs cluster install --config=/opt/bcmt/bcmt_config_openstack.json
```

- Use OneKey

Command:

```
ncs onekey retry --deployip=<deployserver ip>
```

Example command:

```
ncs onekey retry --deployip=10.10.10.10
```

9.2.2 Reset cluster

- Use NCS cli

Command:

```
ncs cluster reset
```

- Use OneKey

Command:

```
ncs onekey reset --deployip <deployserver ip>;  
ncs onekey clear
```



Note:

Reset cleans data created during the NCS installation. *onekey clear* deletes old configuration files.

9.2.3 Re-create cluster nodes

- Use NCS cli

Steps:

1. Log in to the Cluster admin container.

```
docker exec -it clcm-admin bash
```

2. Recreate target nodes.

Commands:

```
clcm-<platform> cluster destroy <cluster name>; clcm-<platform>
cluster create <cluster name>
```

Available platforms: aws/azure/baremetal/openstack/vcenter/vcd

3. Log out from the Cluster admin container.

4. Clean old ETCD data on the deployment server and restart *bcmt-admin*.

Commands:

```
rm -rf /opt/bcmt/config/etcd/*;
docker restart bcmt-admin
```



Note: If cluster nodes were created by users, users must re-create the nodes manually.

- Use OneKey

- The *onekey reset* command deletes cluster nodes that were created by *onekey*.

Commands:

```
ncs onekey reset --deployip <deployserver ip>;
ncs onekey clear
```

onekey clear deletes old configuration files.

9.2.4 Uninstall cluster

- Use NCS cli

Command:

```
ncs cluster uninstall --control_api_server https://10.10.10.10:8082 --  
password ***
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

- Use Onekey

Command:

```
ncs onekey uninstall --deployip=<deployserver ip> --cluster-name=<BCMT  
cluster name> --password=<BCMT cluster admin user password>
```

Example command:

```
ncs onekey uninstall --deployip=10.10.10.10 --cluster-name=bcmt-cluster --  
password
```

9.3 Cluster scale-in

About this task

This task is best performed with NCS Manager. See the *Nokia Container Services Manager Reference Guide for Bare Metal implementation*.

1. Using account, *cloud-user*, type the following command to log in to the Deployment Server.



Note:

Scale-in control node is not supported.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

2. Change to the root user using the following command:

```
sudo su -
```

3. Log in to the NCS admin container using the following command:

```
docker exec -it bcmt-admin bash
```

4. Set the endpoint to point to one of the Control nodes using the **ncs config command:**

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

5. Optional: Log in using the **ncs user login command:**

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password>NewPassword@1234 !
```

6. Remove a node from the cluster using the **ncs cluster scale-in command.** Multiple nodes can be specified by separating each node name with a comma.

NAME:

```
ncs cluster scale-in - scale-in ncs VMs
```

USAGE:

```
ncs cluster scale-in [command options] [arguments...]
```

OPTIONS:

```
--node_names      <node names>      specifies the name of nodes need to be
scaled in(optional)
--continue        <continue>        specifies whether to continue last
scale-in or not(optional, value range(true, false))
```

Example command:

```
ncs cluster scale-in --node_names=bcmt-01-worker-edge-01
```

Example output:

```
{"task-status": "pending"}
{"task-status": "running"}
{"task-status": "running"}
```

```
{
  "task-status": "running"
  ...
{
  "task-status": "done"
}
```

**Note:**

Skip the following steps if *embedded_clcm* is *true*, *ncs cluster scale-in* command deletes and destroys the node automatically.

7. Delete the node's information from the cluster inventory using the **ncs node delete** command.

Command syntax:

```
NAME:
  ncs node delete - delete node

USAGE:
  ncs node delete [command options] [arguments...]

OPTIONS:
  --node_name      <node name>          specifies the node name to be deleted
  --force          <force delete>        force delete the node (optional)
```

Example command:

```
ncs node delete --node_name=bcmt-01-worker-edge-01
```

Example output:

```
Delete node successfully
```

8. Destroy the node. Skip this step if you do not want to destroy the node.

Use Cluster admin scale-in command to destroy the node if the cluster nodes were created by the cluster admin.

- a) Log out from the NCS admin container and log in to the cluster admin container.

```
exit
docker exec -it clcm-admin bash
```

- b) Scale-in the node, available platforms are:

- aws
- azure
- baremetal
- openstack

- vcenter

Command:

```
clcm-<platform\> cluster scale-in <CLUSTER_NAME\> <GROUP_NAME\>
<NODE_NAMES\>
```

Example command:

```
clcm-vcenter cluster scale-in bcmt-01 group_02 bcmt-01-worker-edge-01
```

9.4 Cluster scale-out

About this task

This task is best performed with NCS Manager. See the ***Nokia Container Services Manager Reference Guide for Bare Metal implementation.***

1. Using account, *cloud-user*, type the following command to log in to the Deployment Server.

**Note:**

Scale-out control node is not supported.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

2. Switch to the root user using the following command:

```
sudo su -
```

3. Use the cluster admin scale-out command to create a node if the cluster nodes are created by the cluster admin. Skip this step if `embedded_clcm` is true.

- a) Log in to Cluster admin container.

```
docker exec -it clcm-admin bash
```

- b) Scale-out the node, the available platforms are:

- aws

- azure
- baremetal
- openstack
- vcenter

Command, using --help to see usage for different platforms:

Example command:

```
clcm-openstack cluster scale-out bcmt-01 group_02 2
```

c) After the node scale-out successful, the node's information is added to <cluster name>_node_inventory.json file in the /root/CSF-CLCM/terraform/<platform> directory.

The available platforms are:

- aws
- azure
- baremetal
- openstack
- vcenter

d) Copy the node_inventory JSON file to /opt/clcm.

Command:

```
cp <cluster_name>_node_inventory.json /opt/clcm
```

Example command:

```
cp bcmt-01_node_inventory.json /opt/clcm
```

e) Log out from the Cluster admin container.

f) Move the node_inventory JSON file to /opt/bcmt.

Command:

```
mv /opt/clcm/<cluster_name>_node_inventory.json /opt/bcmt
```

Example command:

```
mv /opt/clcm/bcmt-01_node_inventory.json /opt/bcmt
```

4. Log in to the NCS admin container using the following command:

```
docker exec -it bcmt-admin bash
```

5. Set the endpoint to point to one of the Control nodes using the **ncs config command.:**

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

6. Log in using the **ncs user login command.**

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password>NewPassword@1234 !
```

7. Add the node's information to the cluster inventory using the **ncs node add command. (Skip this step if `embedded_clcm` is true.)**

Command syntax:

```
NAME:  
      ncs node add - add node to cluster  
  
USAGE:  
      ncs node add [command options] [arguments...]  
  
OPTIONS:  
      --inventory    <node inventory file>      specifies the node inventory  
      file to be loaded
```

Example command:

```
ncs node add --inventory=/opt/bcmt/bcmt-01_node_inventory.json
```

Example output:

```
Add node successfully
```

- 8. Add a node to the cluster using the **ncs cluster scale-out** command.** Multiple nodes can be specified by separating each node name with a comma.

```

NAME:
  ncs cluster scale-out - scale-out ncs VMs

USAGE:
  ncs cluster scale-out [command options] [arguments...]

OPTIONS:
  --group_index           <group index>           specifies the index
  of VM groups.If embedded_clcm is true, this is mandatory. For others,
  this is optional
  --count value            <count>                 specifies the
  number of the new nodes.If openstack and embedded_clcm is true, this is
  mandatory.For others, this is optional
  --zones value            <zone>                 specifies the zones
  of the new nodes on AWS and Azure. Only used when embedded_clcm is true
  (optional)
  --node_names             <node name>            specifies the name
  of nodes need to be scaled out, use this param when not using Cluster
  admin.If openstack and embedded_clcm is true, this is optional.For
  others, this is mandatory.
  --static_ip_addresses    <static_ip_addresses>
  specifies the static ip addresses if related
  group is created with static ip(optional)Exmaple:for
  openstack:'{"net-01":{"ipv4":"10.75.56.41","ipv6":""}, "net-02":
  {"ipv4":"192.168.4.24", "ipv6":""}}' for vcenter:'{"NIC_1":
  {"ipv4":"10.75.56.41", "ipv6":""}, "NIC_2":
  {"ipv4":"192.168.4.24", "ipv6":""}}'
  --continue                <continue>              specifies whether to
  continue last scale-out or not(optional, value range(true, false))

```

Example command, **embedded_clcm is true:**

```

ncs cluster scale-out --group_index=group_01 --count=1 --
node_names=bcmt-01-worker-edge-01

```

Example command, **embedded_clcm is false:**

```

ncs cluster scale-out --node_names=bcmt-01-worker-edge-01

```

Example output:

```

{
  "task-status": "pending"
}
{
  "task-status": "running"
}

```

```
{
  "task-status": "running"
  {"task-status": "running"}
  ...
  {"task-status": "done"}
```

9.5 Cluster scale-out-new-group

Note:

This function is only available when `embedded_clcm` is `true`. BareMetal platform does not support scale-out-new-group and uses the existing scale-out method.

1. Customize the `*-new-group.json` configuration file. Refer to the following json files to customize your configuration file of new group.

- `aws-new-group.json` !!! example `aws-new-group.json`

```
{
  "group_04": {
    "node_number": 2,
    "zone": [
      "a",
      "a"
    ],
    "hostname_prefix": "bcmt-w",
    "node_roles": [
      "worker"
    ],
    "node_labels": [],
    "node_taints": [],
    "network_index": [
      "net-01"
    ],
    "default_route": "net-01",
    "default_route_v6": null,
    "static_route": {
      "net-01": {
        "az0": null,
        "az1": null
      }
    },
    "root_device": {
      "instance_type": "t2.large",
      "volume_size": 60,
      "volume_type": "gp2"
    }
  }
}
```

```

        } ,
        "non_root_device": {
            "volume_sizes": [
                {
                    "swap_device": 0
                },
                {
                    "data0_device": 100
                },
                {
                    "local_storage_devices": "0|ext4 0|xfs"
                },
                0
            ],
            "volume_type": "gp2"
        },
        "assign_eip": null
    }
}

```

- azure-new-group.json !!! example azure-new-group.json

```

{
    "group_04": {
        "node_number": 2,
        "zone": [],
        "hostname_prefix": "w",
        "node_roles": [
            "worker"
        ],
        "node_labels": [],
        "node_taints": [],
        "image_index": "image-01",
        "vm_size": null,
        "os_disk": {
            "disk_type": "Standard_LRS",
            "disk_size": null
        },
        "data_disks": {
            "disk_type": "Standard_LRS",
            "volume_sizes": [
                {
                    "swap_device": 0
                },
                {

```

```

        "data0_device": 100
    },
    {
        "local_storage_devices": "0|ext4 0|xfs"
    },
    0
]
},
"network_index": [
    "net-01"
],
"assign_public_ip": false,
"network_security_group": {
    "net-01": null
}
}
}
}

```

- openstack-new-group.json !!! example openstack-new-group.json

```
{
"group_04": {
    "hostname_prefix": "worker",
    "node_number": 2,
    "node_roles": "worker",
    "flavor_name": "m1.large",
    "image_uuid_or_name": null,
    "boot_from_cinder_volume": false,
    "root_volume_size": null,
    "node_labels": null,
    "node_taints": null,
    "network_index": "net-01 net-02",
    "network_no_fixed_ip": null,
    "gateway": "net-01",
    "gateway_v6": null,
    "static_route": {
        "net-01": {
            "ipv4": null,
            "ipv6": null
        }
    },
    "static_ip_addresses": {
        "net-01": {
            "ipv4": null,
            "ipv6": null,
            "mac": null
        }
    }
}
}
```

```

        "mtu": null
    }
},
"allowed_address_pairs": {
    "net-01": {
        "ipv4": null,
        "ipv6": null
    }
},
"server_group_index": null,
"floating_network_name": null,
"security_group_options": null,
"volume_options": {
    "volume_size": [
        {
            "data0_device": 50
        },
        {
            "local_storage_devices": "0|ext4 0|xfs"
        },
        0
    ],
    "zone_of_volume": null,
    "volume_type": null
},
"extra_options": {
    "zone_of_instance": null
}
}
}
}

```

- vcenter-new-group.json !!! example vcenter-new-group.json ./bcmt_topics_admin/scale_out_group/vcenter_new_group.md

2. Using account **cloud-user**, type the following command to log in to the Deployment Server.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

3. Change to the root user using the following command:

```
sudo su -
```

4. Log in to the NCS admin container using the following command:

```
docker exec -it bcmt-admin bash
```

5. Set the endpoint to point to one of the Control nodes using the ncs config command:

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

6. Run ncs cluster scale-out-group command with customized *-new-group.json configuration file.

Command syntax:

```
NAME:  
  ncs cluster scale-out-group - scale out a new group  
  
USAGE:  
  ncs cluster scale-out-group [command options] [arguments...]  
  
OPTIONS:  
  --config      <configuration file>      specifies the path and filename  
  for the scale out configuration JSON file  
  --continue    <continue>                  specifies whether to continue  
  last scale-out-group or not(optional, value range(true, false))  
  --reset       <reset>                   reset last scale-out-group if  
  scale-out-group failed(optional, value range(true, false))
```

Example command:

```
ncs cluster scale-out-group --config ./openstack_new_group.json
```

Example output:

```
Open another terminal and use below command to check detail progress:  
tail -f /opt/bcmt/log/bcmt.log  
{ "task-status": "running" }
```

```
.....  
{"task-status": "done"}
```

9.6 Cluster auto-scaling

The cluster supports auto-scaling new nodes when resources are insufficient to respond to resource requests from Pods. When a Pod's status is pending, and it requests more resources, NCS scales out a Worker node, or a Worker & Edge combined node.



Note: Auto-scaling is only supported when `embedded_clcm` is set to `true`.

User can enable auto-scaling during NCS deployment and enable or disable auto-scaling after deployment.

1. To enable cluster auto-scaling prior to NCS deployment, set `autoscaling_enabled` to `true` in the `bcmt_config.json` configuration file.

```
"autoscaling_enabled": true
```

2. To enable or disable cluster auto-scaling using CLI commands after installation, use the `ncs cluster autoscaling` command.

```
ncs cluster autoscaling <enable|disable>
```

Example command:

```
ncs cluster autoscaling enable
```

Example output:

```
{"task-status": "pending"}  
  
{"task-status": "running"}  
. .  
{"task-status": "done"}
```

3. To check auto-scaling status in NCS cluster.

```
ncs cluster autoscaling status
```

Example output:

```
{  
    "status": "enabled"  
}
```

To enable cluster auto-scaling prior to NCS deployment, set **autoscaling_enabled** to *true* in the `bcmt_config.json` configuration file.

To enable cluster auto-scaling using CLI commands, use the **ncs cluster autoscaling** command.

```
ncs cluster autoscaling <enable|disable>
```

9.7 Cluster health check

Executing this command immediately checks the cluster status, as well as schedule a cron job to verify the cluster every 15 minutes, from a random Control node. The NCS cluster health check includes:

- Checks if all pods are in a running status
- Checks if all nodes are in a ready status
- Checks ETCD performance status
- Checks if the Kubernetes service is in a good status
- Checks if the shared storage is in a good status

The following enhancements are implemented to the health check tool to meet the requirements of the above mentioned functions:

- Resources [CPU, RAM, File System] utilization check across all NCS cluster nodes.
- NCS running on which platform and its Version.
- Resources [CPU, RAM, File System] utilization check across all NCS cluster nodes.
- List of nodes type running in NCS cluster [Control, Worker, Edge, Storage], Nodes Status and node count.
- Status of all services [systemd services, Pods and docker] across all NCS cluster nodes.
- Chronyd sync status across all nodes [NTP sync].
- DNS verification.

Procedure

1. Using account, *cloud-user*, type the following command to log in to the Deployment Server.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

2. Change to the root user using the following command.

```
sudo su -
```

3. Log in to the admin container using the following command.

```
docker exec -it bcmt-admin bash
```

4. Set the endpoint to point to one of the Control nodes using the **ncs config command.**

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

5. Log in using the **ncs user login command.**

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password=NewPassword@1234!
```

6. A healthcheck can be scheduled using the **ncs cluster health-check command that returns the latest available result of the cron job, which means it can show the result of the health-check conducted fifteen minutes ago.**

Command syntax:

```
ncs cluster health-check
```

For the real-time health check result, run the following command:

```
ncs cluster health-check --real_time=true
```

NCS 20FP2SU2 version and above have option to validate the cluster status as a post deployment step. This step is set by default to *enable* in Cluster deployment, and tests the following:

- **ncs components health check**

Tests the different Kubernetes infra components and verifies they are running and functioning, including the kube-dns component validating the capability of the dns resolution.

- **Verify the pods can be initiated**

Create simple Kubernetes pods with storage claims, and different ceph-rbd, ceph-fs and ceph-s3 volume claims can be successfully initiated.

You can use the following option to rerun this validation only, as a post deployment step, at any given time.



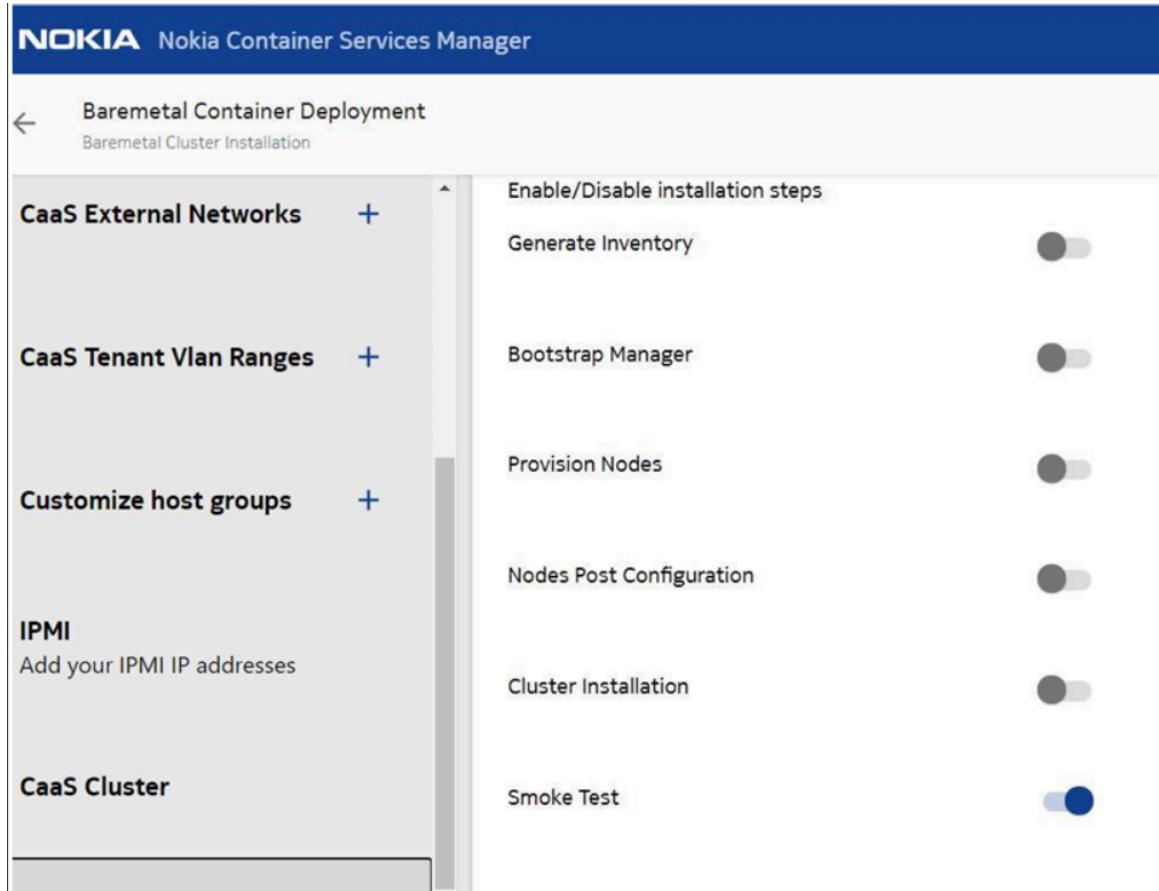
Note: This step is automatically being executed after Scale-out operation.

To initiate this step, use the following recommended option for ncs-manager User Interface:

1. From NCS-Manager UI log in to ncs-manager UI using the credentials.
2. Click **Deployment** and the relevant cluster.
3. Select only the *Smoke test* step.
4. Select **Deploy**.

The deployment displays a log window and reports success or failure with details.

Figure 1: NCS Manager



5. From ncs-manager api: smoke test command

The smoke test can be initiated from API sending a POST request to the ncs-manger site IP address.

```
Format: http POST request url: https://[ncs-manager-ip-address]]/api/
test/smoke
body: json
body content: cloud_name ,
password
```

Example with curl utility:

```
curl -u "ncs-admin:password" -H  
"Content-Type: application/json" -k -d '{"cloud_name": "bvt-  
baremetal",  
"password": "goNokia123$"}'  
https://10.11.232.67/api/tests/smoke
```

6. From ncs-manager api: health check command

The health check command can be initiated also from API sending a POST request to the ncs-manager site IP address.



Note: This command only validates Kubernetes components states, and does not initiate pods creation to validate sanity of the components, it is recommended to run the smoke test for a more accurate result.

```
format:  
http POST request  
url:      https://[ncs-manager-ip-address]]/api/test/health  
body: json  
body content: cloud_name , password
```

Example with curl utility:

```
curl -u "ncs-admin:password" -H  
"Content-Type: application/json" -k -d '{"cloud_name": "bvt-  
baremetal",  
"password": "goNokia123$"}'  
https://10.11.232.67/api/tests/health
```

Example of Cluster health check output:

The readable output from a cluster health check should resemble the following example. This summarizes the health check output and can be downloadable via the NCS Manager. This can still be actioned by the Smoke test UI/API of the NCS Manager.

```
The health-check status is OK

root_ca_status is
+-----+-----+
|           | 0
+-----+-----+
| status    | OK
| expired_time | Valid before: 2023-04-05 20:55:11 |
| check_time  | 2021-01-03 22:59:28 |
```

```
+-----+-----+
dns_server_status is
+-----+-----+
| | 0 |
|-----+-----|
| status | OK |
| check_time | 2021-01-03 22:59:28 |
+-----+-----+

etcd_endpoints_status is
+-----+-----+
| | 0 |
|-----+-----|
| status | OK |
+-----+-----+

glusterFS_status is
+-----+-----+
| | 0 |
|-----+-----|
| status | OK |
| check_time | 2021-01-03 22:59:28 |
+-----+-----+

kube_svc_status is
+-----+-----+
| | 0 |
|-----+-----|
| status | OK |
| check_time | 2021-01-03 22:59:28 |
+-----+-----+

k8s_status
+-----+-----+
| | 0 |
|-----+-----|
| status | OK |
| pod_status | Running |
| nodes_status | Ready |
| check_time | 2021-01-03 22:59:28 |
+-----+-----+

pods list
```

namespace	name	restart_count	start_time	status
0 multustest-alexa		0 2021-01-02 22:08:43	Running	default
1 multustest-alexa-danm		0 2021-01-02 23:01:31	Running	default
2 multustest-alexa-danm-2		0 2021-01-03 13:53:15	Running	default
3 gatekeeper-audit-685c575b58-7xk87				
gatekeeper-system		0 2021-01-02 21:53:38	Running	
4 gatekeeper-controller-manager-5665796bc7-lwqls				
gatekeeper-system		0 2021-01-02 18:15:26	Running	
5 calico-kube-controllers-667c5977cb-fnhvq				kube-
system		0 2021-01-02 18:10:29	Running	system
6 coredns-7w8t2				kube-
system		0 2021-01-02 18:10:32	Running	system
7 coredns-fcgzc				kube-
system		0 2021-01-02 18:10:32	Running	system
8 coredns-h4m21				kube-
system		0 2021-01-02 18:10:32	Running	system
9 cpu-dev-pod-mutator-deployment-gmwxc				kube-
system		0 2021-01-02 18:25:37	Running	system
10 cpu-dev-pod-mutator-deployment-jxsjx				kube-
system		0 2021-01-02 18:25:37	Running	system
11 cpu-dev-pod-mutator-deployment-r78qb				kube-
system		0 2021-01-02 18:25:37	Running	system
12 cpu-device-plugin-fc5qh				kube-
system		0 2021-01-02 18:25:38	Running	system
13 cpu-device-plugin-ksk6v				kube-
system		0 2021-01-02 18:25:38	Running	system
14 cpu-setter-tcnn8				kube-
system		0 2021-01-02 18:25:37	Running	system
15 cpu-setter-vm5wx				kube-
system		0 2021-01-02 18:25:37	Running	system
16 csi-s3-controllerplugin-0				kube-
system		0 2021-01-02 18:14:38	Running	system

17 csi-s3-controllerplugin-1					kube-
system		0	2021-01-02 18:15:04	Running	
18 csi-s3-nodeplugin-2bjgv					kube-
system		0	2021-01-02 18:14:38	Running	
19 csi-s3-nodeplugin-bl7c9					kube-
system		0	2021-01-02 18:14:38	Running	
20 csi-s3-nodeplugin-gsj7m					kube-
system		0	2021-01-02 18:14:38	Running	
21 csi-s3-nodeplugin-mdkkz					kube-
system		0	2021-01-02 18:14:38	Running	
22 csi-s3-nodeplugin-tmb8j					kube-
system		0	2021-01-02 18:14:38	Running	
23 csi-s3-nodeplugin-znt84					kube-
system		0	2021-01-02 18:14:38	Running	
24 danm-cleaner-2l2pm					kube-
system		2	2021-01-02 18:24:33	Running	
25 danm-cleaner-6ffrs					kube-
system		1	2021-01-02 18:24:33	Running	
26 danm-cleaner-7qgtz					kube-
system		0	2021-01-02 18:24:33	Running	
27 danm-webhook-75c4d87cb8-qxfp8					kube-
system		0	2021-01-02 18:24:31	Running	
28 dashboard-metrics-scraper-6554464489-n66t4					kube-
system		0	2021-01-02 21:53:38	Running	
29 default-net-generator-6dd7f94498-1ldtx					kube-
system		0	2021-01-02 18:24:29	Running	
30 kube-apiserver-helsinki-ncs1251-masterbm-0					kube-
system		0	2021-01-02 18:06:48	Running	
31 kube-apiserver-helsinki-ncs1251-masterbm-1					kube-
system		0	2021-01-02 18:06:48	Running	
32 kube-apiserver-helsinki-ncs1251-masterbm-2					kube-
system		0	2021-01-02 18:06:48	Running	
33 kube-controller-manager-helsinki-ncs1251-masterbm-0					kube-
system		1	2021-01-02 18:06:48	Running	
34 kube-controller-manager-helsinki-ncs1251-masterbm-1					kube-
system		1	2021-01-02 18:06:48	Running	
35 kube-controller-manager-helsinki-ncs1251-masterbm-2					kube-
system		1	2021-01-02 18:06:48	Running	
36 kube-dashboard-6bfd8c97bd-h4951					kube-
system		0	2021-01-02 18:10:48	Running	
37 kube-proxy-4pcgb					kube-
system		0	2021-01-02 18:07:26	Running	
38 kube-proxy-6pc6p					kube-
system		0	2021-01-02 18:07:26	Running	

39	kube-proxy-6ww5q					kube-
system			0	2021-01-02 18:07:55	Running	
40	kube-proxy-hrvvq					kube-
system			0	2021-01-02 18:07:55	Running	
41	kube-proxy-qvmbr					kube-
system			0	2021-01-02 18:07:26	Running	
42	kube-proxy-s7srr					kube-
system			0	2021-01-02 18:07:55	Running	
43	kube-scheduler-helsinki-ncs1251-masterbm-0					kube-
system			0	2021-01-02 18:06:48	Running	
44	kube-scheduler-helsinki-ncs1251-masterbm-1					kube-
system			2	2021-01-02 18:06:48	Running	
45	kube-scheduler-helsinki-ncs1251-masterbm-2					kube-
system			1	2021-01-02 18:06:49	Running	
46	local-volume-provisioner-g8s8f					kube-
system			0	2021-01-02 18:11:10	Running	
47	local-volume-provisioner-jz6v5					kube-
system			0	2021-01-02 18:11:11	Running	
48	local-volume-provisioner-k26x6					kube-
system			0	2021-01-02 18:11:10	Running	
49	local-volume-provisioner-kt7wx					kube-
system			0	2021-01-02 18:11:10	Running	
50	local-volume-provisioner-mh9jj					kube-
system			0	2021-01-02 18:11:11	Running	
51	local-volume-provisioner-t982s					kube-
system			0	2021-01-02 18:11:10	Running	
52	metrics-server-6887b6855c-79ksg					kube-
system			0	2021-01-02 18:11:11	Running	
53	netwatcher-g9swl					kube-
system			0	2021-01-02 18:24:22	Running	
54	snapshot-controller-0					kube-
system			0	2021-01-02 18:11:38	Running	
55	snapshot-validation-cbc96b8f9-2w28p					kube-
system			0	2021-01-02 18:11:41	Running	
56	sriov-device-plugin-9dc5					kube-
system			0	2021-01-02 18:08:36	Running	
57	sriov-device-plugin-htrq8					kube-
system			0	2021-01-02 18:08:36	Running	
58	sriov-device-plugin-p4kxf					kube-
system			0	2021-01-02 18:08:36	Running	
59	storage-controller-57f6ffd4bc-1d9p8					kube-
system			0	2021-01-02 18:14:29	Running	
60	svcwatcher-4rhb4					kube-
system			2	2021-01-02 18:24:23	Running	

61 svcwatcher-9hrgz						kube-
system			0 2021-01-02 18:24:23 Running			
62 svcwatcher-c7dhd						kube-
system			1 2021-01-02 18:24:23 Running			
63 tiller-deploy-55d6d67597-wtvlt						kube-
system			0 2021-01-02 18:11:50 Running			
64 app-api-ncms-app-67b7bc6c5c-s9hwq						ncms
		0 2021-01-02 18:14:11 Running				
65 bcmt-api-6klgt						ncms
		1 2021-01-02 18:12:20 Running				
66 bcmt-api-tvcqr						ncms
		1 2021-01-02 18:12:20 Running				
67 bcmt-api-x6jhr						ncms
		1 2021-01-02 18:12:20 Running				
68 bcmt-citm-ingress-dcp19						ncms
		0 2021-01-02 18:30:00 Running				
69 bcmt-citm-ingress-default404-5cf89b78bc-vvv56						ncms
		0 2021-01-02 21:53:38 Running				
70 bcmt-citm-ingress-sdvp7						ncms
		0 2021-01-02 18:30:00 Running				
71 bcmt-citm-ingress-ttvpg						ncms
		0 2021-01-02 18:30:00 Running				
72 bcmt-ckey-ckey-0						ncms
		0 2021-01-02 18:22:34 Running				
73 bcmt-ckey-ckey-1						ncms
		0 2021-01-02 18:35:30 Running				
74 bcmt-cmdb-admin-0						ncms
		0 2021-01-02 18:17:17 Running				
75 bcmt-cmdb-mariadb-0						ncms
		0 2021-01-02 18:17:19 Running				
76 bcmt-cmdb-mariadb-1						ncms
		0 2021-01-02 18:17:19 Running				
77 bcmt-cmdb-mariadb-2						ncms
		0 2021-01-02 18:17:20 Running				
78 bcmt-fluentd-belk-fluentd-daemonset-472mw						ncms
		0 2021-01-02 18:14:15 Running				
79 bcmt-fluentd-belk-fluentd-daemonset-167tt						ncms
		0 2021-01-02 18:14:14 Running				
80 bcmt-fluentd-belk-fluentd-daemonset-r7pwj						ncms
		0 2021-01-02 18:14:15 Running				
81 bcmt-fluentd-belk-fluentd-daemonset-w4dmc						ncms
		0 2021-01-02 18:14:15 Running				
82 bcmt-fluentd-belk-fluentd-daemonset-wvl5c						ncms
		0 2021-01-02 18:14:15 Running				

83	bcmt-fluentd-belk-fluentd-daemonset-xbvlk				ncms
		0	2021-01-02 18:14:15	Running	
84	bcmt-ip-man-agent-677v7				ncms
		0	2021-01-02 18:19:27	Running	
85	bcmt-ip-man-agent-b4r8r				ncms
		0	2021-01-02 18:19:27	Running	
86	bcmt-ip-man-agent-cbd48				ncms
		0	2021-01-02 18:19:27	Running	
87	bcmt-ip-man-agent-h2q49				ncms
		0	2021-01-02 18:19:27	Running	
88	bcmt-ip-man-agent-nhx6s				ncms
		0	2021-01-02 18:19:27	Running	
89	bcmt-ip-man-agent-zx958				ncms
		0	2021-01-02 18:19:27	Running	
90	bcmt-ip-man-master-0				ncms
		0	2021-01-02 18:19:27	Running	
91	bcmt-kubesetting-2bxxl				ncms
		0	2021-01-02 18:26:38	Running	
92	bcmt-kubesetting-786jn				ncms
		0	2021-01-02 18:26:38	Running	
93	bcmt-kubesetting-hldgx				ncms
		0	2021-01-02 18:26:38	Running	
94	bcmt-kubesetting-k2m28				ncms
		0	2021-01-02 18:26:38	Running	
95	bcmt-kubesetting-l8zrl				ncms
		0	2021-01-02 18:26:38	Running	
96	bcmt-kubesetting-pxwlb				ncms
		0	2021-01-02 18:26:38	Running	
97	bcmt-mt-tenant-5f68f777b5-zqgn9				ncms
		0	2021-01-02 18:22:51	Running	
98	bcmt-operator-78r7h				ncms
		0	2021-01-02 18:19:06	Running	
99	bcmt-operator-8tkfx				ncms
		0	2021-01-02 18:19:06	Running	
100	bcmt-operator-96c92				ncms
		0	2021-01-02 18:19:05	Running	
101	bcmt-operator-97pmc				ncms
		0	2021-01-02 18:19:06	Running	
102	bcmt-operator-mbbfg				ncms
		0	2021-01-02 18:19:06	Running	
103	bcmt-operator-pfrtb				ncms
		0	2021-01-02 18:19:05	Running	
104	bcmt-redis-64cb479994-9sg6n				ncms
		0	2021-01-02 18:22:48	Running	

105 bcmt-yum-repo-55c6754c75-kbd91				ncms
0 2021-01-02 18:09:12 Running				
106 bcmt-yum-repo-55c6754c75-mhjxf				ncms
0 2021-01-02 18:09:12 Running				
107 bcmt-yum-repo-55c6754c75-r6574				ncms
0 2021-01-02 18:09:12 Running				
108 cbur-master-cbur-5f6544fd47-hx4gj				ncms
0 2021-01-02 18:16:40 Running				
109 cbur-master-cbur-k8swatcher-69d6d49f85-rrcq6				ncms
0 2021-01-02 18:16:20 Running				
110 cert-manager-7c87db4577-vkbx9				ncms
0 2021-01-02 21:53:30 Running				
111 cert-manager-cainjector-6f4995c84d-gnfg5				ncms
0 2021-01-02 21:53:38 Running				
112 cert-manager-webhook-5bb898b6f7-7zkvv				ncms
0 2021-01-02 21:53:30 Running				
113 csi-cephfs-nodeplugin-8hqvx				ncms
0 2021-01-02 18:15:59 Running				
114 csi-cephfs-nodeplugin-k779k				ncms
0 2021-01-02 18:15:59 Running				
115 csi-cephfs-nodeplugin-kxbm8				ncms
0 2021-01-02 18:15:59 Running				
116 csi-cephfs-nodeplugin-lfzjp				ncms
0 2021-01-02 18:15:59 Running				
117 csi-cephfs-nodeplugin-qctxs				ncms
0 2021-01-02 18:15:59 Running				
118 csi-cephfs-nodeplugin-rf82k				ncms
0 2021-01-02 18:15:59 Running				
119 csi-cephfs-provisioner-8d6bf8f99-9k7w9				ncms
0 2021-01-02 18:15:59 Running				
120 csi-cephfs-provisioner-8d6bf8f99-1hvmr				ncms
2 2021-01-02 18:15:59 Running				
121 csi-cephrbd-nodeplugin-8smf8				ncms
0 2021-01-02 18:15:46 Running				
122 csi-cephrbd-nodeplugin-j72zh				ncms
0 2021-01-02 18:15:46 Running				
123 csi-cephrbd-nodeplugin-pp7wl				ncms
0 2021-01-02 18:15:46 Running				
124 csi-cephrbd-nodeplugin-qqsrdf				ncms
0 2021-01-02 18:15:46 Running				
125 csi-cephrbd-nodeplugin-x8tg2				ncms
0 2021-01-02 18:15:46 Running				
126 csi-cephrbd-nodeplugin-zd9mw				ncms
0 2021-01-02 18:15:46 Running				

nodes list:

```
node helsinki-ncs1251-edgegbm-0
+-----+-----+
+-----+-----+
+-----+
|                               | allocatable   | capacity     |
| creation_time      | name           | status       | version
|                   |
|-----+-----+-----+
+-----+-----+
+-----+
| cpu                  | 16          | 48          |
| 2021-01-02 18:07:47 | helsinki-ncs1251-edgegbm-0 | Ready        | v1.19.5
|                   |
| ephemeral-storage    | 793871668753 | 861405892Ki |
| 2021-01-02 18:07:47 | helsinki-ncs1251-edgegbm-0 | Ready        | v1.19.5
|
```

hugepages-1Gi	50Gi	50Gi	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
memory	195575612Ki	263833404Ki	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/exclusive_numa_0_pool 4	4	4	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/exclusive_numa_1_pool 4	4	4	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/shared_pool	12k	12k	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_ens12f0	8	8	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_ens12f1	8	8	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_ens1f0	8	8	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_ens1f1	8	8	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_vfio_ens12f0	2	2	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_vfio_ens12f1	2	2	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_vfio_ens1f0	2	2	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
nokia.k8s.io/sriov_vfio_ens1f1	2	2	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
pods	110	110	
2021-01-02 18:07:47 helsinki-ncs1251-edgebm-0 Ready v1.19.5			
-----+-----+-----+			
-----+-----+-----+			
-----+-----+-----+			

node helsinki-ncs1251-edgebm-1			
		allocatable	capacity
creation_time	name	status	version
2021-01-02 18:07:47	cpu	16	48
2021-01-02 18:07:47	ephemeral-storage	793871668753	861405892Ki
2021-01-02 18:07:47	hugepages-1Gi	50Gi	50Gi
2021-01-02 18:07:47	memory	195575612Ki	263833404Ki
2021-01-02 18:07:47	nokia.k8s.io/exclusive numa_0_pool	4	4
2021-01-02 18:07:47	nokia.k8s.io/exclusive numa_1_pool	4	4
2021-01-02 18:07:47	nokia.k8s.io/shared_pool	12k	12k
2021-01-02 18:07:47	nokia.k8s.io/sriov_ens12f0	8	8
2021-01-02 18:07:47	nokia.k8s.io/sriov_ens12f1	8	8
2021-01-02 18:07:47	nokia.k8s.io/sriov_ens1f0	8	8
2021-01-02 18:07:47	nokia.k8s.io/sriov_ens1f1	8	8

```

| nokia.k8s.io/sriov_vfio_ens12f0 | 2 | 2 | 2 |
2021-01-02 18:07:47 | helsinki-ncs1251-edgebm-1 | Ready | v1.19.5
|
| nokia.k8s.io/sriov_vfio_ens12f1 | 2 | 2 | 2 |
2021-01-02 18:07:47 | helsinki-ncs1251-edgebm-1 | Ready | v1.19.5
|
| nokia.k8s.io/sriov_vfio_ens1f0 | 2 | 2 | 2 |
2021-01-02 18:07:47 | helsinki-ncs1251-edgebm-1 | Ready | v1.19.5
|
| nokia.k8s.io/sriov_vfio_ens1f1 | 2 | 2 | 2 |
2021-01-02 18:07:47 | helsinki-ncs1251-edgebm-1 | Ready | v1.19.5
|
| pods | 110 | 110 | 110 |
2021-01-02 18:07:47 | helsinki-ncs1251-edgebm-1 | Ready | v1.19.5
|
+-----+
+-----+
+-----+
node helsinki-ncs1251-masterbm-0
+-----+
+-----+
+-----+
| | allocatable | capacity | creation_time |
| name | status | version | 2021-01-02 |
+-----+-----+-----+
+-----+
+-----+
| cpu | 48 | 48 | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
| ephemeral-storage | 793871668753 | 861405892Ki | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
| hugepages-1Gi | 0 | 0 | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
| hugepages-2Mi | 0 | 0 | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
| memory | 237391292Ki | 263767484Ki | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
| pods | 110 | 110 | 2021-01-02 |
18:07:08 | helsinki-ncs1251-masterbm-0 | Ready | v1.19.5 |
+-----+
+-----+
+-----+
node helsinki-ncs1251-masterbm-1

```

		allocatable	capacity	creation_time
	name		status	version
18:07:11	cpu	48	48	2021-01-02
18:07:11	ephemeral-storage	793871668753	861405892Ki	2021-01-02
18:07:11	hugepages-1Gi	0	0	2021-01-02
18:07:11	hugepages-2Mi	0	0	2021-01-02
18:07:11	memory	237391292Ki	263767484Ki	2021-01-02
18:07:11	pods	110	110	2021-01-02
18:07:11	node helsinki-ncs1251-masterbm-2	Ready	v1.19.5	
18:07:09	cpu	48	48	2021-01-02
18:07:09	ephemeral-storage	793871668753	861405892Ki	2021-01-02
18:07:09	hugepages-1Gi	0	0	2021-01-02
18:07:09	hugepages-2Mi	0	0	2021-01-02
18:07:09	memory	237391292Ki	263767484Ki	2021-01-02
18:07:09	pods	110	110	2021-01-02
18:07:09	node helsinki-ncs1251-masterbm-2	Ready	v1.19.5	

node helsinki-ncs1251-workerdanm-0			
creation_time	name	allocatable	capacity
version			status
2021-01-02 18:07:47	v1.19.5	44	48
2021-01-02 18:07:47	v1.19.5	793871668753	861405892Ki
2021-01-02 18:07:47	v1.19.5	0	0
2021-01-02 18:07:47	v1.19.5	0	0
2021-01-02 18:07:47	v1.19.5	248004412Ki	263833404Ki
2021-01-02 18:07:47	v1.19.5	30	30
2021-01-02 18:07:47	v1.19.5	30	30
2021-01-02 18:07:47	v1.19.5	8	8
2021-01-02 18:07:47	v1.19.5	8	8
2021-01-02 18:07:47	v1.19.5	2	2
2021-01-02 18:07:47	v1.19.5	2	2

```

| nokia.k8s.io/sriov_vfio_ens1f1 | 2 | 2 |
2021-01-02 18:07:47 | helsinki-ncs1251-workerdanm-0 | Ready |
v1.19.5 |
| pods | 110 | 110 |
2021-01-02 18:07:47 | helsinki-ncs1251-workerdanm-0 | Ready |
v1.19.5 |
+-----+
+-----+
+-----+

```

Summary

The summary of the health test is shown in the Log of the Smoke test. If users see an error, they are able to use an API or command line instruction to fetch the full report to drill into the details of the issue.

9.8 Cluster kubernetes setting (KubeSetting)

The KubeSetting operator (kind: KubeSetting) is used to modify the /etc/kubernetes/kubelet-config.yaml or /etc/kubernetes/manifests/* for the selected nodes in a cluster. You can create a KubeSetting.yaml file to affect changes. you can customize kubernetes settings before or after NCS deployment.

- To customize kubernetes settings prior to NCS deployment, set values in the k8s_settings section in the bcmt_config.json configuration file.

```

"k8s_settings": {
    "kube_api_config": { "service-node-port-range": "30000-32767" },
    "kube_controller_manager_config": {},
    "kube_scheduler_config": {},
    "kubelet_config": {}
}

```

- To customize kubernetes settings using CLI commands after installation, use the ncs cluster kubesetting command.

Command syntax:

```

NAME:
ncs cluster kubesetting - modify the kube setting after deployment

USAGE:
ncs cluster kubesetting [command options] [arguments...]

OPTIONS:

```

```
--config    <configuration file>      specifies the path and filename for
the kube setting JSON file

Template:
kube-setting.json: set "allowUnsafeSysctls" for target nodes/groups

{
  "kubelet_config": {
    "allowedUnsafeSysctls": "net.ipv4.conf.*", "net.ipv6.conf.*"
  }
}

Set the value to "" to disable it.
```

Example Command:

```
ncs cluster kubesetting --config kube-setting.json
```

Example Output

```
{"task-status": "pending"}
{"task-status": "running"}
{"task-status": "done"}
```

Example: kube-setting.json

```
{
  "k8s_settings": {
    "kube_api_config": { "service-node-port-range": "30000-32767" },
    "kube_controller_manager_config": { "leader-elect-lease-
duration": "30s", "leader-elect-renew-deadline": "20s", "leader-elect-retry-
period": "5s" },
    "kube_scheduler_config": { "leader-elect-lease-duration": "30s",
"leader-elect-renew-deadline": "20s", "leader-elect-retry-period": "5s" },
    "kubelet_config": { "maxPods": 130, "allowedUnsafeSysctls": [
"net.ipv4.*", "net.ipv6.*" ] }
  }
}
```

References:

- *[kube-apiserver - kubernetes](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/)*
- *[kube-controller-manager - kubernetes](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/)*

- *[kube-scheduler - kubernetes](https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/)*
- kubelet_config keys are hump-named, please refer to *[kubelet-config - Github](https://github.com/kubernetes/kubernetes/blob/master/staging/src/k8s.io/kubelet/config/v1beta1/types.go)*

The nodeSelector parameter is optional. When specified, all nodes that match the label of the nodeSelector value applies KubeSettings. If nodeSelector is not specified, all cluster nodes apply KubeSettings.

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f KubeSetting.yaml
```

- To revert changes, use the **kubectl delete** command:

```
kubectl delete -f KubeSetting.yaml
```



Note: If you change the kubelet_config, you need to restart the kubelet service on the particular nodes.

```
ssh[node]
kubectl drain[node]
systemctl restart kubelet
kubectl uncordon[node]
```

Portal GUI:

Deploy:

- Add a new file under *targer VMs* (Create VMs via CLCM):
 - <check box> enable unsafe sysctls
 - <input> allowed unsafe sysctls
- Upload node inventory contains sysctl settings (use existing VMs).

Runtime: modify kube setting to allow unsafe sysctls.

- modify kubesettings:

- <input>kubesettings
- selector:
 - <comboBox>
 - node names

- labels

Configure this parameter during deployment:

- If create VMs via CLCM, set sysctl info in user_input.yml:

```
group_01:
  sysctl_settings:
    unsafe_sysctl_enabled: true
    allowed_unsafe_sysctls: net.ipv6.*, net.ipv4.*
```

- If do not create VMs via CLCM, specifies sysctl settings for each node in node_inventory.json

```
"control-01": {
  "data0_device": "/dev/vdb",
  "etcd_device": "/dev/vdc",
  ...
  "sysctl_settings": {
    "unsafe_sysctl_enabled": true,
    "allowed_unsafe_sysctls": "net.ipv6.*, net.ipv4.*"
  }
  ...
},
}
```

Example command:

```
ncs cluster kubesetting --config kube-setting.json
```

Example output:

```
{"task-status": "pending"}
{"task-status": "running"}.
 {"task-status": "done"}
```

Example: kube-setting.json

```
{
  "k8s_settings": {
    "kube_api_config": {"service-node-port-range": "30000-32767"},
    "kube_controller_manager_config": {"leader-elect-lease-duration": "30s", "leader-elect-renew-deadline": "20s", "leader-elect-retry-period": "5s"},
    "kube_scheduler_config": {"leader-elect-lease-duration": "30s", "leader-elect-renew-deadline": "20s", "leader-elect-retry-period": "5s"},
    "kubelet_config": {"maxPods": 130}
  }
}
```

```
}
```

References:

- [kube-apiserver - kubernetes](#)
- [kube-controller-manager - kubernetes](#)
- [kube-scheduler - kubernetes](#)
- kubelet_config keys are hump-named, refer to [kubelet-config - Github](#)

9.9 Cluster backup and restore

NCS supports cluster level data backup and restoration for applications:

9.10 Cluster graceful shutdown and startup

Refer to section *Baremetal Graceful shutdown and startup*

9.10.1 Cluster graceful shutdown startup for virtual deployments

This command executes the following before the NCS cluster shutdown:

- Disable RAM ETCD if it is enabled
- Cordon all cluster nodes
- Delete pods using rook or glusterfs
- Drain all cluster nodes
- Shut down NCS cluster in 2 minutes

Prerequisite:

- The NCS CLI should work normally
- User should log in CLI with NCS command

Command syntax:

```
ncs cluster shutdown
```

Sample output:

```
{"task-status": "running"}  
.....  
Broadcast message from root@bcmtn-06-control-01 (Wed 2020-07-15 02:49:01 UTC):
```

```
The system is going down for power-off at Wed 2020-07-15 02:51:01 UTC!
```

```
{"task-status": "done"}
```

How to startup the cluster after graceful shutdown:



Note: The following procedures require the user to operate manually.

1. Uncordon nodes using the `kubectl uncordon` command after starting the node.

```
kubectl uncordon <node names>
```

Example:

```
kubectl uncordon clcmapp-host01 clcmapp-host02 clcmapp-host03 clcmapp-host04
node/clcmapp-host01 uncordoned
node/clcmapp-host02 uncordoned
node/clcmapp-host03 uncordoned
node/clcmapp-host04 uncordoned
```

2. Run `kubectl get node` to make sure every node is in the Ready state.

```
kubectl get node
NAME        STATUS   ROLES      AGE        VERSION
clcmapp-host01  Ready    <none>    3d        v1.18.6
clcmapp-host02  Ready    <none>    3d        v1.18.6
clcmapp-host03  Ready    <none>    3d        v1.18.6
clcmapp-host04  Ready    <none>    3d        v1.18.6
```

3. If you enabled ram ETCD before shutdown, run the following command to re-enable it.

```
ncs service rametcd enable
```



Note:

- If rook is enabled, after cluster start up, some rook pods may run into error state. Restart the pod with prefix `rook-ceph-operator` will solve the issue.

Example:

```
# kubectl delete pod rook-ceph-operator-6dcbb78496c-hnn22 -nrook-ceph
```

- If using Red Hat, there may be nodes in `NotReady` status after cluster startup. The following log can be seen in the journal.

```
systemd-logind[1025]: Failed to enable subscription: Failed to
activate service 'org.freedesktop.systemd1': timed out
systemd-logind[1025]: Failed to fully start up daemon: Connection
timed out
```

```

systemd[1]: systemd-logind.service: main process exited,
            code=exited, status=1/FAILURE
systemd[1]: Failed to start Login Service.
systemd[1]: Unit systemd-logind.service entered failed state.
systemd-logind.service failed.
systemd[1]: systemd-logind.service has no holdoff time, scheduling
            restart.
systemd[1]: Stopped Login Service.

```

This is a known issue of Red Hat <https://access.redhat.com/solutions/3900301>, the workaround is to reboot the *NotReady* nodes.

```
# ncm node restart --node_names <notready node_names>
```

9.11 Refresh cluster root certificate

About this task

One option to install a new root certificate is to uninstall and then re-install the NCS cluster with the new root CA. This is recommended when the cluster can be easily uninstalled and re-installed.

The final renewCA process is not part NCS product, it is released as MOP, including patch file c4cbde3.tar.gz

 **Note:** The following procedure requires service and node to restart and must be performed when there is little to no traffic.

1. Save control-0 node labels into etcd with the following commands:

- Login to control-0 node:

```
ssh -i <private key file> cloud-user@< control-0 node oam IP \>
```

- Get the node labels, and save them into etcd:

```
etcdctl --endpoints=$(cat /etc/etcd/etcd_endpoints.yml|cut -d' '
-f2|tr -d '') --cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/
ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem put  /
BCMTClusterManager/renewcaLabels/<control-0 node hostname>/ $(kubectl
get node <control-0 node hostname> --show-labels | grep -v NAME | awk
'{print $6}')
```

Example command:

```
etcdctl --endpoints=$(cat /etc/etcd/etcd_endpoints.yml|cut -d' '
-f2|tr -d '') --cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/
ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem put  /
```

```
BCMTClusterManager/renewcaLabels/cbis-sut4-control-01/ $(kubectl get node cbis-sut4-control-01 --show-labels | grep -v NAME | awk '{print $6}')
```

Example output:

OK

2. Using account *cloud-user*, copy the provided patch file to the Deployment Server home folder.

3. Using account *cloud-user*, type the following command to log in to the Deployment Server.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP\>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

4. Change to the root user using the following command.

```
sudo su -
```

5. Extract the patch using the following commands:

```
cd ~

mkdir patch

tar -xzvf c4cbde3.tar.gz -C patch/

mkdir /opt/bcmt/patch

cp -r patch/src/ansible/commands/cluster/extendca/ /opt/bcmt/patch/

cp patch/bin/renewca.sh /opt/bcmt/patch/
```

6. Log in to the NCS admin container using the following command.

```
docker exec -it bcmt-admin bash
```

7. Apply the necessary scripts to bcmt-admin container using the following commands inside the container

```
cp -r /opt/bcmt/patch/extendca/ ansible/commands/cluster/
```

```
cp /opt/bcmt/patch/renewca.sh .
```

8. Set the endpoint to point to one of the Control nodes using the **ncs config command.**

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

9. Log in using the **ncs user login command.**

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password=NewPassword@1234!
```

10. Use the **ncs cluster inventory command to retrieve the cluster token.**

Example command:

```
ncs cluster inventory
```

Example output:

```
*****
inventory tarball has been downloaded in current path.
Name is inventory.tgz
*****
```

11. Unzip the tarball to the `/root/CSF-BCMT/ansible` directory.

Example command:

```
tar -zxvf inventory.tgz -C /root/CSF-BCMT/ansible/
```

Expected output:

```
bcmt_var-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_inventory-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

12. Change the permissions for the `bcmt_vm_key-<uuid>.json` file using the `chmod` command.

Example command:

```
chmod 600 /root/CSF-BCMT/ansible/bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

13. Prepare the root certificate files. Both the new CA and its associated key file are required. For example: ca.pem and ca-key.pem.



Note: Your new root certificate should be the same content except the expiry date. e.g. the same issuer, the same subject, the same DNS.

14. Use the `renewca.sh` with the `backupca` action to backup the current root CA to cluster nodes into the `/opt/bcmt/rootca/backup` folder.

Command syntax:

```
renewca.sh -i <inventory_file> -v <variable_file> -k <vm_key_file> -a  
<action\>
```

Example command:

```
./renewca.sh -i /root/CSF-BCMT/ansible/  
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \  
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json \  
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json -a backupca
```

15. Use the `renewca.sh` script to install the new certificate files .

Command syntax:

```
renewca.sh -i <inventory_file> -v <variable_file> -k <vm_key_file> -a  
<action\>
```

Example command:

```
./renewca.sh -i /root/CSF-BCMT/ansible/  
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \  
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json \  
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json \  
-a renewca -r ./ca.pem -p ./ca-key.pem
```

16. To renew a self-generated, self-signed CA, use the `-s` flag.

Example command:

```
./renewca.sh -i /root/CSF-BCMT/ansible/  
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \  
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json \  
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json -a renewca -s
```

17. Reboot control-0 node in the following way:

- Login to control-1 node:

```
ssh -i <private key file> cloud-user@<control-1 node oam IP>
```

- Set ncs endpoint config if necessary:

```
ncs config set --endpoint=https://<control-1 node oam IP>:<port>/ncm/  
api/v1
```



Note: The endpoint port is defined by the bcmt_portal_port parameter in the bcmt_config.json file (default value: 8084).

- Log in using ncs user login command:

```
ncs user login --username=<username> --password=<password>
```

- Restart control-0 node with ncs cli:

```
ncs node restart --node_names=<control-0 node name>
```

Example command:

```
ncs node restart --node_names=cbis-sut4-control-01
```

Example output:

```
[cloud-user@cbis-sut4-control-02 ~]$ ncs node restart --  
node_names=cbis-sut4-control-01  
  
This command will reboot the node, some service may get impact, are  
you sure to continue? [y/n]  
  
Y  
  
{ "task-status": "running" }
```

```
.....
{
  "task-status": "done"
}

{
  "cbis-sut4-control-01": "restart success"
}
```

- Get the node labels from etcd:

```
labels=$(ETCDCTL_API=3 etcdctl --endpoints=$(cat /etc/etcd/
etcd_endpoints.yml|cut -d' ' -f2|tr -d ''') --cacert=/etc/etcd/ssl/
ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-
client-key.pem get --print-value-only --prefix /BCMTClusterManager/
renewcaLabels/<control-0 node hostname>)
```

Example command:

```
labels=$(ETCDCTL_API=3 etcdctl --endpoints=$(cat /etc/etcd/
etcd_endpoints.yml|cut -d' ' -f2|tr -d ''') --cacert=/etc/etcd/ssl/
ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-
client-key.pem get --print-value-only --prefix /BCMTClusterManager/
renewcaLabels/cbis-sut4-control-01)
```

- Apply the saved labels on control-0 node:

```
for label in $(echo $labels | sed -n 1'p' | tr ',' '\n'); do kubectl
label nodes --overwrite <control-0 node hostname> $label; done
```

Example command:

```
for label in $(echo $labels | sed -n 1'p' | tr ',' '\n'); do kubectl
label nodes --overwrite cbis-sut4-control-01 $label; done
```

Example output:

```
node/cbis-sut4-control-01 not labeled
```

```

node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 not labeled
node/cbis-sut4-control-01 labeled

```

- 18.** If Escalator component is enabled on the NCS Cluster, issue the following commands to restart the bcmt-escalator deployment

```

kubectl rollout restart deployment -n ncms bcmt-escalator
kubectl rollout status deployment -n ncms bcmt-escalator

```

- 19.** If errors occur while replacing the CA in ETCD after a node reboot, use the replace_ca_in_etcd action to retry this step.

Example command:

```

./renewca.sh -i /root/CSF-BCMT/ansible/
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-
f1d54ee3b958.json \
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-
f1d54ee3b958.json -a replace_ca_in_etcd

```

- 20.** To restore a backup of the root CA, use the copybackca action. All services must be restarted after performing this action.

Example command:

```

./renewca.sh -i /root/CSF-BCMT/ansible/
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-
f1d54ee3b958.json \
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-
f1d54ee3b958.json -a copybackca

```

- 21.** After having restored a backup of the root CA, restart all services using the restartservices action.

Example command:

```
./renewca.sh -i /root/CSF-BCMT/ansible/  
bcmt_inventory-92bc876b-63fd-4637-9810-f1d54ee3b958.json \  
-v /root/CSF-BCMT/ansible/bcmt_var-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json \  
-k /root/CSF-BCMT/ansible/bcmt_vm_key-92bc876b-63fd-4637-9810-  
f1d54ee3b958.json -a restartservices
```

9.12 Recover a node

About this task

 **Note:** Recovery precondition is that Node OS is up as originally. All IP address, hostname should be the same as the corrupted node.

To bring up the corrupted node:

- If the corrupted node was created manually, the Node OS must be manually recovered or recreated.
- If the corrupted node was created via Cluster admin, use Cluster admin heal command to bring up the node.
- Skip this step if embedded_clcm is true, *ncs node recover* command will heal the node automatically.

1. Log in to the Deployment Server and log in to the Cluster admin container.

```
docker exec -it clcm-admin bash
```

2. Heal the node using the following command, available platforms are:

- aws
- azure
- baremetal
- openstack
- vcenter
- vcd

```
clcm-<platform\> cluster heal <CLUSTER_NAME> <NODE_NAMES>
```

To recover a node:

1. Using account, *cloud-user*, type the following command to log in to the Deployment Server.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

2. Change to the root user using the following command:

```
sudo su -
```

3. Log in to the NCS admin container using the following command:

```
docker exec -it bcmt-admin bash
```

4. Set the endpoint to point to one of the Control nodes using the **ncs config** command.

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

5. Log in using the **ncs user login** command,

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password>NewPassword@1234 !
```

6. Recover a node in the cluster using the **ncs node recover** command.

Command syntax:

```
ncs node recover --node_name=<node name>
```

Example command:

```
ncs node recover --node_name=bcmt-01-control-worker-edge-03
```

Example output:

```
Open another terminal and use below command to check detail progress:  
tail -f /opt/bcmt/log/bcmt.log  
{"task-status": "pending"}  
{"task-status": "running"}  
.....  
{"task-status": "done"}
```

9.13 Restart a Node



Note: This command is not applicable for DELL Hardware setups. By issuing this command (as well as *sudo reboot now*) causes the master node to become stuck until the watchdog timer restarts the node.

1. You must run the following command for each node that you are going to restart before running the node restart command:

```
sudo docker exec redis redis-cli info | grep role
```

For each node that you get *master*, run the following:

```
sudo docker exec -it redis redis-cli -p 26379 SENTINEL failover  
cbis_cluster
```

2. Restart cluster nodes using the *ncs node restart* command.

Command syntax:

```
NAME:  
  ncs node restart - restart nodes  
  
USAGE:  
  ncs node restart [command options] [arguments...]  
  
OPTIONS:  
  --node_names    <node name>      specifies the name of nodes need to be  
  restarted, if there are multiple nodes, split each with comma  
  --parallel       <boolean>        specifies whether restart in parallel,  
  yes or no. (Optional);Default is no
```

Example command:

```
ncs node restart --node_names bcmt-cwes-01,bcmt-cws-01,bcmt-cws-02,bcmt-  
we-01
```

Example printout:

```
This command will reboot the node, some service may get impact, are you
sure to continue? [y/n]

Y
{"task-status": "pending"}

{"task-status": "pending"}

{"task-status": "running"}


.....
{"task-status": "done"}
{
    "bcmt-cws-01": "restart success",
    "bcmt-cws-02": "restart success",
    "bcmt-we-01": "restart success",
    "bcmt-cwes-01": "current node, will restart in 1 minute."
}
```

3. After restart, from the Kubernetes CLI prompt, check the node names and status using the *kubectl get nodes* command.

```
kubectl get nodes
```

Example printout:

NAME	STATUS	ROLES	AGE	VERSION
bcmt-cwes-01	Ready	<none>	3d	v1.16.4
bcmt-cws-01	Ready	<none>	3d	v1.16.4
bcmt-cws-02	Ready	<none>	3d	v1.16.4
bcmt-we-01	Ready	<none>	3d	v1.16.4

4. Ensure all application Pods are in the *Running* state using the *kubectl get pod* command:

```
kubectl get pod --all-namespaces
```

5. Run any application specific verification script.

9.14 Shutdown a node

- From the Kubernetes CLI prompt, check the node names and status using the **kubectl get nodes** command.

```
kubectl get nodes
```

Sample output:

NAME	STATUS	ROLES	AGE	VERSION
clcmapp-host01	Ready	<none>	3d	v1.16.3
clcmapp-host02	Ready	<none>	3d	v1.16.3
clcmapp-host03	Ready	<none>	3d	v1.16.3
clcmapp-host04	Ready	<none>	3d	v1.16.3

- Cordon nodes using the **ncs node cordon** command.

Command syntax:

```
ncs node cordon --node_names=<node names>
```

Example command:

```
ncs node cordon --node_names=clcmapp-host01,clcmapp-host02,clcmapp-host03,clcmapp-host04
```

Sample output:

```
node/clcmapp-host01 cordoned  
node/clcmapp-host02 cordoned  
node/clcmapp-host03 cordoned  
node/clcmapp-host04 cordoned
```

- Drain traffic for all non-Control cluster nodes.

Command syntax:

```
kubectl drain -force --ignore-daemonsets --delete-local-data <node name>
```

Example command:

```
kubectl drain --force --ignore-daemonsets --delete-local-data clcmapp-host04
```

Sample output:

```
node "clcmapp-host04" cordoned
```

```
WARNING: Ignoring DaemonSet-managed pods: kube-proxy-k8ccv
node "clcmapp-host04" drained
```

- Check the new status of the node using the **kubectl get nodes** command. The node status should be *Ready,SchedulingDisabled*.

```
kubectl get nodes
```

Sample output:

NAME	STATUS	ROLES	AGE	VERSION
clcmapp-host01	Ready, SchedulingDisabled	<none>	3d	v1.16.3
clcmapp-host02	Ready, SchedulingDisabled	<none>	3d	v1.16.3
clcmapp-host03	Ready, SchedulingDisabled	<none>	3d	v1.16.3
clcmapp-host04	Ready, SchedulingDisabled	<none>	3d	v1.16.3

- Shutdown cluster nodes using the **ncs node shutdown** command.

Command syntax:

```
ncs node shutdown --node_name=<node names>
```

Example command:

```
ncs node shutdown --node_names=clcmapp-host01,clcmapp-host02,clcmapp-
host03,clcmapp-host04
```

Sample output:

```
This command will shutdown the node and need user to manually power on the
node, some service may get impact, are you sure to continue? [y/n]
y
"nodes clcmapp-host01,clcmapp-host02,clcmapp-host03,clcmapp-host04 will be
shutdown later."
```

9.15 Startup a node

- Uncordon nodes using the **ncs node uncordon** command.

Command syntax:

```
ncs node uncordon --node_names=<node names>
```

Example command:

```
ncs node uncordon --node_names=clcmapp-host01,clcmapp-host02,clcmapp-
host03,clcmapp-host04
```

Sample output:

```
node/clcmapp-host01 uncordoned  
node/clcmapp-host02 uncordoned  
node/clcmapp-host03 uncordoned  
node/clcmapp-host04 uncordoned
```

- From the Kubernetes CLI prompt, check the node names and status using the **kubectl get nodes** command. The node status should be *Ready*.

```
kubectl get nodes
```

Sample output:

NAME	STATUS	ROLES	AGE	VERSION
clcmapp-host01	Ready	<none>	3d	v1.16.3
clcmapp-host02	Ready	<none>	3d	v1.16.3
clcmapp-host03	Ready	<none>	3d	v1.16.3
clcmapp-host04	Ready	<none>	3d	v1.16.3

- Ensure all application Pods are in the *Running* state using the **kubectl get pod** command:

```
kubectl get pod --all-namespaces
```

- Run any application-specific verification script.

9.16 Cluster node resize

OpenStack platform

The user can change the size of cluster node on the OpenStack platform after NCS has deployed.

Example: change all edge nodes' flavor from m1.large to m1.xlarge

Procedure:

- Change flavor manually for some cluster nodes.
- Once the nodes are available remove the kubernetes `cpu_manager_state` file if it exist, skip this step if not exist:

```
rm "/var/lib/kubelet/cpu_manager_state"
```

- Update data in Cluster admin container, be clear how the BCMT cluster was installed, with `embedded_clcm` is true or false.

- a. Log in to the Cluster admin container.

- Case1: `embedded_clcm` is **false**, log in to the Cluster admin container from **Deployment Server** by using the following command:

```
docker exec -it clcm-admin bash
```

- Case2: `embedded_clcm` is **true**(deployed NCS via web portal or onekey), log in to the Cluster admin container from **a control node** by using the following command:

```
kubectl exec -it `kubectl get po -nncms | awk '{print $1}' | grep clcm` bash -nncms
```

b. Backup current data.

- This step is needed only for `embedded_clcm` is **true**, should get data from ETCD by using the following command:

```
curl http://0.0.0.0:8083/ncms/api/v1/clcm/cluster/resource/restore  
-X post
```

- Backup current data to `/opt/clcm/data_bk/`:

```
mkdir -p /opt/clcm/data_bk  
cp /root/CSF-CLCM/terraform/openstack/user_input.yml /opt/clcm/  
data_bk/  
cp -r /root/CSF-CLCM/terraform/openstack/resources /opt/clcm/  
data_bk/
```

c. Update the `flavor_name` under related node group in file `/root/CSF-CLCM/terraform/openstack/user_input.yml`.

Example: change `flavor_name` from change `m1.large` to `m1.xlarge` under `group_03` for edge nodes.

Before:

```
group_03:  
  node_roles    : edge  
  flavor_name   : m1.large
```

After:

```
group_03:  
  node_roles    : edge  
  flavor_name   : m1.xlarge
```

d. Get new `flavor_id`(`flavor_uuid`) by using the following command:

```
cd /root/CSF-CLCM/terraform/openstack/resources/
```

```
. keystone.rc  
openstack flavor list |grep <new-flavor-name>
```

Example command:

```
cd /root/CSF-CLCM/terraform/openstack/resources/  
. keystone.rc  
openstack flavor list |grep m1.xlarge
```

Sample output: flavor_id(flavor_uuid) is d8ac4c3d-0b37-4b9a-b624-4377bc88f70c

	d8ac4c3d-0b37-4b9a-b624-4377bc88f70c		m1.xlarge						
16384		160		0		8		True	

e. Update flavor_name and flavor_id under related nodes in file /root/CSF-CLCM/terraform/openstack/resources/terraform.tfstate.<cluster_name>.

Example: update flavor_name and flavor_id for edge nodes

Before:

```
"flavor_name": "m1.large",  
"flavor_id": "494ef0cf-4663-4912-afa9-856decc5770b",
```

After:

```
"flavor_name": "m1.xlarge",  
"flavor_id": "d8ac4c3d-0b37-4b9a-b624-4377bc88f70c",
```

f. Update flavor_uuid in file /root/CSF-CLCM/terraform/openstack/resources/openstack.tf.<cluster_name>.

Example: find the module for edge nodes, search flavor_uuid, then change it.

Before:

```
flavor_uuid = "494ef0cf-4663-4912-afa9-856decc5770b"
```

After:

```
flavor_uuid = "d8ac4c3d-0b37-4b9a-b624-4377bc88f70c"
```

g. Run terraform cli to verify by using the following commands:

```
cd /root/CSF-CLCM/terraform/openstack/resources/  
. keystone.rc  
cp openstack.tf.<cluster_name> openstack.tf  
terraform init
```

```
terraform plan -state=terraform.tfstate.<cluster_name>
```

- Expected results:

- No changes. Infrastructure is up-to-date. Example:

```
-----
No changes. Infrastructure is up-to-date.
This means that Terraform did not detect any differences
between your
configuration and real physical resources that exist. As a
result, no
actions need to be performed.
-----
```

- If there is change and change only for *False -> false*. Example:

```
~ metadata ={ "attached_mode" = "rw" ~ "readonly" = "False" ->
  "false" "usage" = "bcmt-cs-02-blockstorage" }
Plan: 0 to add, 9 to change, 0 to destroy.
```

- Unexpected result, Example

```
-----
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  ~ update in-place
Terraform will perform the following actions:
  ~ module. bcmt-edge-02.openstack_compute_instance_v2.instance
    flavor_id: " d8ac4c3d-0b37-4b9a-b624-4377bc88f70c " => "
      494ef0cf-4663-4912-afa9-856decc5770b "
Plan: 0 to add, 1 to change, 0 to destroy.
-----
```



Note: If you encounter an unexpected result, please check pre steps and correct it.

4. This step is needed only for `embedded_clcm` is **true**, update the ETCD by using following command:

```
curl http://0.0.0.0:8083/ncms/api/v1/clcm/etcd/update -X POST -H
"Content-Type:application/json" --data '{"Platform":"openstack"}'
```

9.16.1 BCMT Control node resize in OpenStack

1. Create a new flavor in CBIS cloud

```
[stack@undercloud (overcloudrc) ~]$ openstack flavor create --ram 16384 --disk 75 --vcpus 6 --public test-flv
+-----+-----+
| Field | Value |
+-----+-----+
| OS-FLV-DISABLED:disabled | False |
| OS-FLV-EXT-DATA:ephemeral | 0 |
| disk | 75 |
| id | 315abf9e-d0b8-458c-990c-2204a4fb9394 |
| name | test-flv |
| os-flavor-access:is_public | True |
| properties |
| ram | 16384 |
| rxtx_factor | 1.0 |
| swap |
| vcpus | 6 |
+-----+
```

2. Check Gluster health and Memory limit on Control node

```
[root@cbis-sut3-control-01 ~]# systemctl status glusterd
● glusterd.service - GlusterFS, a clustered file-system server
  Loaded: loaded (/etc/systemd/system/glusterd.service; enabled; vendor
  preset: disabled)
  Active: active (running) since Fri 2021-01-22 14:06:30 UTC; 5 days ago
    Main PID: 13091 (glusterd)
      Memory: 212.0M (limit: 5.7G)
        CGroup: /podruntime.slice/glusterd.service
                ├─13091 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level
          ERROR
                ├─13426 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id
          bcmt-glusterfs.172.16.130.25.data0-glusterfs -p /var/run/gluster/vols/
          bcmt-glusterfs/172.16.130.25-data0-glusterfs.pid -S /var/run/glust...
                ├─13447 /usr/sbin/glusterfs -s 172.16.130.25 --volfile-id shd/
          bcmt-glusterfs -p /var/run/gluster/shd/bcmt-glusterfs/bcmt-glusterfs-
          shd.pid -l /var/log/glusterfs/glustershd.log -S /var/run/gluster...
                ├─13720 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id
          cbur-glusterfs-repo.172.16.130.25.data0-glusterfs-cbur-repo -p /var/run/
          gluster/vols/cbur-glusterfs-repo/172.16.130.25-data0-glusterfs-c...
                ├─13937 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id
          cbur-glusterfs-backup.172.16.130.25.data0-glusterfs-cbur-backup -p /var/
          run/gluster/vols/cbur-glusterfs-backup/172.16.130.25-data0-glust...
```

```
└─14126 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id  
cbur-glusterfs-backup-cluster.172.16.130.25.data0-glusterfs-backup-  
cluster -p /var/run/gluster/vols/cbur-glusterfs-backup-cluster/172.16...  
  
Jan 22 14:06:30 cbis-sut3-control-01 systemd[1]: Starting GlusterFS, a  
clustered file-system server...  
Jan 22 14:06:30 cbis-sut3-control-01 systemd[1]: Started GlusterFS, a  
clustered file-system server.
```

3. Change MemoryLimit in Gluster service file

```
[root@cbis-sut3-control-01 ~]# vi /etc/systemd/system/glusterd.service  
[root@cbis-sut3-control-01 ~]# cat /etc/systemd/system/glusterd.service |  
grep Memory  
MemoryLimit=7000M
```

4. Check Gluster health - MemoryLimit should be unchanged

```
[root@cbis-sut3-control-01 ~]# systemctl status glusterd  
● glusterd.service - GlusterFS, a clustered file-system server  
  Loaded: loaded (/etc/systemd/system/glusterd.service; enabled; vendor  
  preset: disabled)  
  Active: active (running) since Fri 2021-01-22 14:06:30 UTC; 5 days ago  
    Main PID: 13091 (glusterd)  
      Memory: 212.0M (limit: 5.7G)  
        CGroup: /podruntime.slice/glusterd.service  
                ├─13091 /usr/sbin/glusterd -p /var/run/glusterd.pid --log-level  
          ERROR  
                ├─13426 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id  
          bcmt-glusterfs.172.16.130.25.data0-glusterfs -p /var/run/gluster/vols/  
          bcmt-glusterfs/172.16.130.25-data0-glusterfs.pid -S /var/run/glust...  
                ├─13447 /usr/sbin/glusterfs -s 172.16.130.25 --volfile-id shd/  
          bcmt-glusterfs -p /var/run/gluster/shd/bcmt-glusterfs/bcmt-glusterfs-  
          shd.pid -l /var/log/glusterfs/glustershd.log -S /var/run/gluster...  
                ├─13720 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id  
          cbur-glusterfs-repo.172.16.130.25.data0-glusterfs-cbur-repo -p /var/run/  
          gluster/vols/cbur-glusterfs-repo/172.16.130.25-data0-glusterfs-c...  
                ├─13937 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id  
          cbur-glusterfs-backup.172.16.130.25.data0-glusterfs-cbur-backup -p /var/  
          run/gluster/vols/cbur-glusterfs-backup/172.16.130.25-data0-glust...  
                └─14126 /usr/sbin/glusterfsd -s 172.16.130.25 --volfile-id  
          cbur-glusterfs-backup-cluster.172.16.130.25.data0-glusterfs-backup-  
          cluster -p /var/run/gluster/vols/cbur-glusterfs-backup-cluster/172.16...  
  
Jan 22 14:06:30 cbis-sut3-control-01 systemd[1]: Starting GlusterFS, a  
clustered file-system server...
```

```
Jan 22 14:06:30 cbis-sut3-control-01 systemd[1]: Started GlusterFS, a
clustered file-system server.
Warning: glusterd.service changed on disk. Run 'systemctl daemon-reload'
to reload units.
```

5. Setup ncm CLI environment on another Control node or deploy server bcmt-admin container

```
[root@cbis-sut3-control-02 ~]# ncm user login --username <> --password <>
[root@cbis-sut3-control-02 ~]# ncm config set --endpoint
'https://10.88.154.47:8082/ncm/api/v1;https://10.88.152.26:8082/ncm/api/
v1;https://10.88.152.23:8082/ncm/api/v1'
Set config successfully
```

6. Start node shutdown procedure

```
Shutdown a node - Nokia Container Services

[BCMT-20.12.0 root@cbis-sut3-deploy:~/CSF-BCMT]$ ncm node cordon --
node_names=cbis-sut3-control-01
node/cbis-sut3-control-01 cordoned

[BCMT-20.12.0 root@cbis-sut3-deploy:~/CSF-BCMT]$ ncm node shutdown --
node_names=cbis-sut3-control-02
This command will reboot the node, some service may get impact, are you
sure to continue? [y/n]
Y
"nodes cbis-sut3-control-02 will be shutdown later."
```

7. Check if node is in SHUTOFF state in CBIS cloud

```
[stack@undercloud (overcloudrc) ~]$ openstack server show
d105e360-6ee7-4a3f-8c11-2d1d044e0d76
+-----+
| Field | Value |
+-----+
| OS-DCF:diskConfig | MANUAL |
| OS-EXT-AZ:availability_zone | zone1 |
| OS-EXT-SRV-ATTR:host | overcloud-ovscompute-
gating-0.localdomain |
| OS-EXT-SRV-ATTR:hypervisor_hostname | overcloud-ovscompute-
gating-0.localdomain |
```

OS-EXT-SRV-ATTR:instance_name	instance-000014ea
OS-EXT-STS:power_state	Shutdown
OS-EXT-STS:task_state	None
OS-EXT-STS:vm_state	stopped
OS-SRV-USG:launched_at	2021-01-22T13:56:15.000000
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	cbis-sut3_int=172.16.130.25;
floating=10.88.154.47	
config_drive	True
created	2021-01-22T13:55:48Z
flavor	ncs_def_c1 (76628d1e-
a562-4d4f-9985-4a3cea5367ed)	
hostId	
60d7b75c70327f4777e5329bda9da8a84488feae6ac1e3dc1ee3c25	
id	
d105e360-6ee7-4a3f-8c11-2d1d044e0d76	
image	bcmt-baseos-2006
(96e602d5-6681-4bad-b857-b58c60a183b8)	
key_name	cbis-sut3-cluster
name	cbis-sut3-control-01
project_id	8dd4f966db5c4503b460c59698916e15
properties	dc='openstack', role='control
storage edge', ssh_user='cloud-user'	
security_groups	name='cbis-sut3-int-oam-default'
default'	name='cbis-sut3-int-storage-
default'	name='cbis-sut3-int-service-

```

|                               | name='cbis-sut3-ext-storage-
| default'                   |
|                               | name='cbis-sut3-ext-control-
| default'                   |
|                               | name='cbis-sut3-ext-edge-default'
|
| status                      | SHUTOFF
|
| updated                     | 2021-01-28T00:26:50Z
|
| user_id                     | adb1967302504221a81cb17116d06d64
|
| volumes_attached           |
| b93f-78d0fd7b4516'          | id='76481b63-3904-4dc1-
|                                |
| baf3fcb7e890'               | id='a6df01ac-35bf-4d4c-a786-
|                                |
| fa43e1a935b1'               | id='399cb162-b57d-4870-b971-
|                                |
| eee0bce971ff'               | id='c7e1b1e5-fa22-435d-80c2-
|                                |
+-----+
+-----+

```

8. Change VM flavor in CBIS cloud

```
[stack@undercloud (overcloudrc) ~]$ openstack server resize --flavor
315abf9e-d0b8-458c-990c-2204a4fb9394 d105e360-6ee7-4a3f-8c11-2d1d044e0d76
(It takes about 5 minutes)
```

9. Verify node status is back to SHUTOFF

The resize process may take 5-10 minutes.

```
[stack@undercloud (overcloudrc) ~]$ openstack server show
d105e360-6ee7-4a3f-8c11-2d1d044e0d76
+-----+
+-----+
| Field                  | Value
|                         |
+-----+
+-----+
| OS-DCF:diskConfig      | MANUAL
|                         |
| OS-EXT-AZ:availability_zone | zone1
|                         |
+-----+
```

OS-EXT-SRV-ATTR:host	overcloud-ovscompute-
gating-1.localdomain	
OS-EXT-SRV-ATTR:hypervisor_hostname	overcloud-ovscompute-
gating-1.localdomain	
OS-EXT-SRV-ATTR:instance_name	instance-000014ea
OS-EXT-STS:power_state	Shutdown
OS-EXT-STS:task_state	None
OS-EXT-STS:vm_state	stopped
OS-SRV-USG:launched_at	2021-01-28T00:33:25.000000
OS-SRV-USG:terminated_at	None
accessIPv4	
accessIPv6	
addresses	cbis-sut3_int=172.16.130.25;
floating=10.88.154.47	
config_drive	True
created	2021-01-22T13:55:48Z
flavor	test-flv (315abf9e-
d0b8-458c-990c-2204a4fb9394)	
hostId	
9b76e0e9fef4cd4fc52f6e24fb392e4e91447d8580dfe4570b9f97d7	
id	
d105e360-6ee7-4a3f-8c11-2d1d044e0d76	
image	bcmt-baseos-2006
(96e602d5-6681-4bad-b857-b58c60a183b8)	
key_name	cbis-sut3-cluster
name	cbis-sut3-control-01
project_id	8dd4f966db5c4503b460c59698916e15
properties	dc='openstack', role='control
storage edge', ssh_user='cloud-user'	
security_groups	name='cbis-sut3-int-oam-default'

```
|           | name='cbis-sut3-int-storage-  
| default' |           |  
|           | name='cbis-sut3-int-service-  
| default' |           |  
|           | name='cbis-sut3-ext-storage-  
| default' |           |  
|           | name='cbis-sut3-ext-control-  
| default' |           |  
|           | name='cbis-sut3-ext-edge-default'  
|           |  
| status    | SHUTOFF  
|           |  
| updated   | 2021-01-28T00:34:22Z  
|           |  
| user_id   | adb1967302504221a81cb17116d06d64  
|           |  
| volumes_attached | id='76481b63-3904-4dc1-  
b93f-78d0fd7b4516' |           |  
|           | id='a6df01ac-35bf-4d4c-a786-  
baf3fc7e890' |           |  
|           | id='399cb162-b57d-4870-b971-  
fa43e1a935b1' |           |  
|           | id='c7e1b1e5-fa22-435d-80c2-  
eee0bce971ff' |           |  
+-----+  
+-----+
```

10. Start VM

```
[stack@undercloud (overcloudrc) ~]$ openstack server start  
d105e360-6ee7-4a3f-8c11-2d1d044e0d76
```

11. Login and delete cpu pooler file if present

```
[root@cbis-sut3-control-01 ~]# rm -f /var/lib/kubelet/cpu_manager_state
```

12. Check Gluster status

Verify that all peers are available and volumes are accessible on all three nodes.

```
[root@cbis-sut3-control-02 ~]# gluster peer status  
Number of Peers: 2
```

```
Hostname: cbis-sut3-control-01.storage.bcmt  
Uuid: dbalbf08-b05a-46f8-95cb-1eaa4fefafa9f9
```

```
State: Peer in Cluster (Connected)

Hostname: 172.16.130.11
Uuid: 8bc23d35-1905-4e82-9e73-75b63978f9ac
State: Peer in Cluster (Connected)

[root@cbis-sut3-control-02 ~]# gluster volume status
Gluster process                                TCP Port  RDMA Port  Online
  Pid
-----
Brick 172.16.130.25:/data0/glusterfs          49152     0          Y
    3147
Brick 172.16.130.17:/data0/glusterfs          49153     0          Y
    12338
Brick 172.16.130.11:/data0/glusterfs          49152     0          Y
    12268
Self-heal Daemon on localhost                  N/A       N/A          Y
    12359
Self-heal Daemon on cbis-sut3-control-01.st
orange.bcmt                                    N/A       N/A          Y
    3206
Self-heal Daemon on 172.16.130.11             N/A       N/A          Y
    12289
```

13. Follow the node resize procedures accoring to your deployment type - the following is an example of embedded_clcm true, OneKey Deployment

```
Cluster node resize - Nokia Container Services

[root@cbis-sut3-control-01 ~]# kubectl exec -it `kubectl get po -nncms |
  awk '{print $1}' | grep clcm` bash -nncms
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ curl
http://0.0.0.0:8083/ncms/api/v1/clcm/cluster/resource/restore -X post
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ mkdir
-p /opt/clcm/data_bk
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ cp /
root/CSF-CLCM/terraform/openstack/user_input.yml /opt/clcm/data_bk/
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ cp -
r /root/CSF-CLCM/terraform/openstack/resources /opt/clcm/data_bk/
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ vi /
root/CSF-CLCM/terraform/openstack/user_input.yml
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ cat /
root/CSF-CLCM/terraform/openstack/user_input.yml | grep flavor
      flavor_name: test-flv
      flavor_name: ncs_min_wk
```

```
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM]$ cd /root/CSF-CLCM/terraform/openstack/resources/
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ . keystone.rc
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ openstack flavor list |grep test-flv
| 315abf9e-d0b8-458c-990c-2204a4fb9394 | test-flv | 16384 | 75 |
| 0 | 6 | True |
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ vi /root/CSF-CLCM/terraform/openstack/resources/terraform.tfstate.cbis-sut
    "flavor_id": "315abf9e-d0b8-458c-990c-2204a4fb9394",
    "flavor_name": "test-flv",
    "name": "cbis-sut3-control-01",
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ vi /root/CSF-CLCM/terraform/openstack/resources/openstack.tf.cbis-sut3
module "cbis-sut3-control-01" {
  flavor_uuid = "315abf9e-d0b8-458c-990c-2204a4fb9394"

[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ cp openstack.tf.cbis-sut3 openstack.tf
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/terraform/openstack/resources]$ terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.openstack: version = "~> 1.33"

Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to
see
any changes that are required for your infrastructure. All Terraform
commands
should now work.
```

```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget,
other
commands will detect it and remind you to do so if necessary.
```

```
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-bmbrt:~/CSF-CLCM/terraform/openstack/resources]$ terraform plan -state=terraform.tfstate.cbis-sut3
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

module.cbis-sut3-
control-01.openstack_blockstorage_volume_v3.blockstorage_volume-01:
  Refreshing state... [id=76481b63-3904-4dc1-b93f-78d0fd7b4516]
module.cbis-sut3-
worker-02.openstack_blockstorage_volume_v3.blockstorage_volume-01:
  Refreshing state... [id=8eb55739-c4d7-409a-91b4-df68ecc48fc2]
module.cbis-sut3-
control-03.openstack_blockstorage_volume_v3.blockstorage_volume-01:
  Refreshing state... [id=509febbe-9e65-4424-bf8e-112a8014fbab]
module.cbis-sut3-control-03.openstack_networking_port_v2.port_net-02_v4:
  Refreshing state... [id=491c5c5f-5c1d-4014-845e-960135e32b1b]
module.cbis-sut3-
control-02.openstack_blockstorage_volume_v3.blockstorage_volume-02:
  Refreshing state... [id=05a36df1-bd31-41db-ac5c-d5d0eab642d0]
module.cbis-sut3-worker-01.openstack_networking_port_v2.port_net-01_v4:
  Refreshing state... [id=b75d6e87-e943-48f7-9f73-bda09e6c16fb]
module.cbis-sut3-control-01.openstack_networking_port_v2.port_net-01_v4:
  Refreshing state... [id=2e7cf0a4-1afc-46c5-b914-faf4b6a82f34]
module.cbis-sut3-control-01.openstack_networking_port_v2.port_net-02_v4:
  Refreshing state... [id=1338b81a-edc2-4332-81cd-34aa1caf3e24]
module.cbis-sut3-
worker-01.openstack_blockstorage_volume_v3.blockstorage_volume-01:
  Refreshing state... [id=879af492-bd4e-49e2-9e9f-83c8dac6c8ff]
module.cbis-sut3-
control-03.openstack_blockstorage_volume_v3.blockstorage_volume-02:
  Refreshing state... [id=b074c0f2-01f1-4bac-8517-c82400d475df]
```

```

module.cbis-sut3-
control-01.openstack_blockstorage_volume_v3.blockstorage_volume-02:
  Refreshing state... [id=a6df01ac-35bf-4d4c-a786-baf3fc7e890]
module.cbis-sut3-worker-02.openstack_networking_port_v2.port_net-01_v4:
  Refreshing state... [id=e8228807-064a-4fb4-b181-d1c1cdc79f84]
module.cbis-sut3-control-03.openstack_networking_port_v2.port_net-01_v4:
  Refreshing state... [id=73fd9ad5-0b31-4589-bd35-cbd1741ccfba]
module.cbis-sut3-
control-02.openstack_blockstorage_volume_v3.blockstorage_volume-01:
  Refreshing state... [id=5225f723-64ad-45d7-8fba-a6a5e7311c7c]
module.cbis-sut3-control-02.openstack_networking_port_v2.port_net-01_v4:
  Refreshing state... [id=eb92687b-6636-449e-a868-a1b736cf8271]
module.cbis-sut3-worker-02.openstack_networking_port_v2.port_net-02_v4:
  Refreshing state... [id=f7dbb264-b357-42e2-a9ce-4a18dc49d4fd]
module.cbis-sut3-control-02.openstack_networking_port_v2.port_net-02_v4:
  Refreshing state... [id=96675878-10ea-40ef-a6ff-d0838b595433]
module.cbis-sut3-worker-01.openstack_networking_port_v2.port_net-02_v4:
  Refreshing state... [id=79530d5f-9fc1-4843-9c07-19fa849c560d]
module.cbis-sut3-control-03.openstack_compute_instance_v2.instance:
  Refreshing state... [id=5b12c6ab-8d70-4e5e-a16d-b4b258ee3b28]
module.cbis-sut3-worker-02.openstack_compute_instance_v2.instance:
  Refreshing state... [id=a0677dc6-3971-4234-8d9a-a18948ea6b44]
module.cbis-sut3-control-02.openstack_compute_instance_v2.instance:
  Refreshing state... [id=89a9050d-4f2d-401d-a562-b150f0f96918]
module.cbis-sut3-worker-01.openstack_compute_instance_v2.instance:
  Refreshing state... [id=00b829ef-bc27-4d90-a68a-e597266b2add]
module.cbis-sut3-control-01.openstack_compute_instance_v2.instance:
  Refreshing state... [id=d105e360-6ee7-4a3f-8c11-2d1d044e0d76]
-----
```

No changes. Infrastructure is up-to-date.

This means that Terraform did not detect any differences between your configuration and real physical resources that exist. As a result, no actions need to be performed.

```
[CLCM-20.12.0 root@clcm-api-deployment-7bcf558859-lhrc6:~/CSF-CLCM/
terraform/openstack/resources]$ curl http://0.0.0.0:8083/ncms/api/v1/
clcm/etcd/update -X POST -H "Content-Type:application/json" --data
'{"Platform":"openstack"}'
200
```

14. Uncordon node

```
[ BCMT-20.12.0 root@cbis-sut3-deploy:~/CSF-BCMT ]$ ncm node uncordon --  
node_names=cbis-sut3-control-01  
node/cbis-sut3-control-01 uncordoned
```

15. Verify node availability

```
[root@cbis-sut3-control-01 ~]# kubectl get nodes  
NAME           STATUS    ROLES   AGE     VERSION  
cbis-sut3-control-01  Ready    <none>  5d10h  v1.19.5  
cbis-sut3-control-02  Ready    <none>  5d10h  v1.19.5  
cbis-sut3-control-03  Ready    <none>  5d10h  v1.19.5  
cbis-sut3-worker-01   Ready    <none>  5d10h  v1.19.5  
cbis-sut3-worker-02   Ready    <none>  5d10h  v1.19.5
```

16. Repeat steps for the next Control node

9.17 Uninstall the cluster

1. Using account, *cloud-user*, type the following command to log in to the Deployment Server.

Command syntax:

```
ssh -i <private key file> cloud-user@<deployment server IP>
```

Example command:

```
ssh -i bcmt-deployserver.pem cloud-user@10.0.0.1
```

2. Change to the root user using the following command:

```
sudo su -
```

3. Log in to the NCS admin container using the following command:

```
docker exec -it bcmt-admin bash
```

4. Uninstall all software from the cluster using the **ncs cluster uninstall** command. For security reasons, you must specify your password when executing uninstall commands.

Command syntax:

```
ncs cluster uninstall --control_api_server https://<control node IP>:8082 --password  
<password>
```

Example command:

```
ncs cluster uninstall --control_api_server https://10.53.183.158:8082 --password  
NewPassword@1234!
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

After the uninstallation is finished, if `embedded_clcm=false` in the last installation, check if the cluster nodes are rebooted. Manually reboot all the cluster nodes if they are not rebooted successfully during uninstallation.

10 Networking

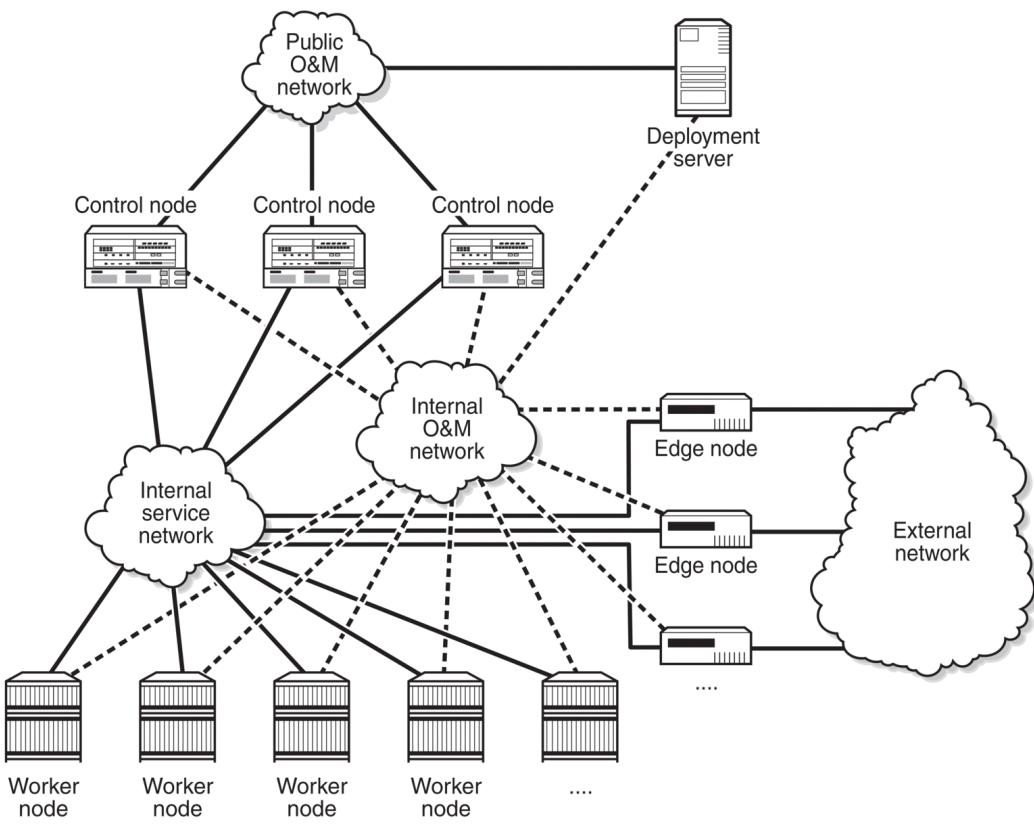
10.1 Network topology

NCS supports flexible network functions to accommodate product requirements. Networks are logically divided, and users can implement multiple functions within the same network. It is acceptable to use the same network for all traffic.

Networks typically consist of:

- an internal O&M network used for cluster management from the Deployment Server. Management tasks include cluster deploy, scale-in, and scale-out.
- an internal service network used for communication between cluster nodes. It provides the down-layer for container networking.
- an internal storage network used to access data on the storage nodes. The QoS and MTU of this network can be uniquely customized, independent of the other networks.
- the end users' external network. Nokia products provide service to the users in this network. Typically, Edge nodes act as gateways, providing an interface to the external network, as well as, access control and load balancing functionality.

Figure 2: Network topology



28120

10.2 Network stack

NCS supports three network stack options: IPv4 only, IPv4 dualstack and IPv6 only.

- IPv4 only: The cluster operates with IPv4 addresses only. All infrastructure components and applications use IPv4 addresses.
- IPv4/IPv6 dualstack: The cluster operates with both IPv4 and IPv6 addresses. Infrastructure components operate with IPv4 addresses, but application Pods are allocated IPv4/IPv6 dualstack addresses. Kubernetes is only aware of the IPv4 address for Pods and cluster nodes. The OAM network, internal OAM network and internal storage networks operate using IPv4. This option requires the Calico plugin, and an internal service network and an external network that operates with an IPv4/IPv6 dualstacks.
- IPv6 only: The cluster operates with IPv6 addresses only. All infrastructure components and applications use IPv6 addresses. This option requires the Calico plugin.
- See also [IPv6 Support](#) on page 172.

Calico is the only Cluster Network Interface (CNI) that supports IPv6.

10.2.1 Enable IPv4 only using a configuration file

Use this procedure to enable IPv4 support during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set `network_stack` to `ipv4_only`.
3. This feature is ready for deployment.

End of steps: this procedure is complete

10.2.2 Enable dualstack using a configuration file

Network elements require varying IP address support, as outlined below:

- Deployment Server does not require an IPv6 address.
- Control, Worker, and Edge nodes require IPv6 and IPv4 dual stack configurations.
- Kubernetes components require IPv4 addresses.
- Pods require IPv6 and IPv4 addresses.

Use this procedure to enable IPv4/IPv6 dualstack support during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set `network_stack` to `ipv4_dualstack`.
3. Set `network_mode` to `calico`.
4. Set `overlay_network_ipv6`, providing the IPv6 address and network mask value in CIDR notation.
5. Access the `node_inventory.json` file.
6. The following networks must be defined:
 - `oam_ip`
 - `internal_service_ip`
 - `internal_storage_ip`
 - `internal_oam_ip`
 - `internal_service_ipv6`
 - `internal_storage_ipv6`
7. This feature is ready for deployment.

10.2.3 Enable IPv6 only using a configuration file

In IPv6 only mode, all kubernetes components and nodes run with IPv6 addresses only.

Use this procedure to enable IPv6 only support during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set `network_stack` to `ipv6_only`.
3. Set `network_mode` to `calico`.
4. Set `overlay_network_ipv6`, providing the IPv6 address and network mask value in CIDR notation.
5. Access the `node_inventory.json` file.
6. Ensure the following IP addresses are set for each node:
 - `oam_ipv6`
 - `internal_service_ipv6`
 - `internal_storage_ipv6`
 - `internal_oam_ipv6`
7. This feature is ready for deployment.

End of steps: this procedure is complete

10.2.4 Manage dualstack using CLI commands

Use the `ncs network network-stack` command to manage dualstack functionality.

1. Get the current status of network using the command:

```
ncs network network-stack status
```

2. Define the current network settings using the command::

```
ncs network network-stack set
```

3. Get and define the IPv6 overlay-network using the command:

```
ncs network network-stack overlay-network list/set
```

10.3 Container network interface (CNI)

NCS follows the Kubernetes model for container networking. Kubernetes applies IP addresses at the Pod level. Containers within a Pod share their network namespaces, including their IP address. Containers within a Pod can all reach each other's ports on localhost. This is called the *IP-per-pod* model.

Container networks can be provided by using a series of network plugins.

Table 20: Network plugin comparison

Plugin	Performance	IPv6	Encryption	Built-in IPAM	Working independently	Special notes
Calico	Medium	Y	-	Y	Y	-
ipvlan	Medium	Y	-	-	-	Allowed address pairs needed in cloud
macvlan	Medium	Y	-	-	-	Disabled port security needed in cloud
host-device	High	Y	-	-	-	-
SR-IOV	High	Y	-	-	-	Bare metal only
vhost-user	High	Y	-	-	-	OVS/VPP needed

Table 21: Meta plugins

Plugin	Function
Multus	Create multiple interfaces for pod
DANM	Create multiple interfaces for pod
portmap	Forward traffic from one port on the host to the container

Table 21: Meta plugins (continued)

Plugin	Function
tuning	Change sysctls and interface attributes in the network namespace

Table 22: IPAM plugins

Plugin	Working Scope	Dynamic IP assignment	Static IP assignment
Whereabouts	Cluster	Y	N
static	Cluster	N	Y
host-local	Node	Y	N

Typical use cases

Common cloud native application:

- Calico is simple to configure and offers acceptable performance or external IPv6.
- Weave Net should be used if encryption is required.

Container application which require multiple data planes:

- Multus + Calico + ipvlan
- DANM + Calico + ipvlan

Container application which requires a high-performance network:

- Multus + Calico + host-device/SR-IOV
- DANM + Calico + SR-IOV

10.3.1 Calico

Calico provides a highly scalable networking and network policy solution for connecting Kubernetes pods based on the same IP networking principles as the internet. Calico can be deployed without encapsulation or overlays to provide high-performance, high-scale data center networking. Calico also provides fine-grained, intent based network security policy for Kubernetes pods via its distributed firewall.

In the Calico approach, IP packets to or from a workload are routed and firewalled by the Linux routing table and iptables infrastructure on the workload's host. For a workload that is sending packets, Calico ensures

that the host is always returned as the next hop MAC address regardless of whatever routing the workload itself might configure. For packets addressed to a workload, the last IP hop is that from the destination workload's host to the workload itself.

- Prevalent choice among cloud native platforms built for microservices.
- Leverages existing and internet-proven technologies – built using Linux Kernel capabilities and the Border Gateway Protocol (BGP) as its core
- Scalable, Secure, Open, and Easy to operate and troubleshoot
- No additional overlay (packet encapsulation) when running on a L3 or L2 network.
- Already supported within the Mantl composer (Kubernetes and Docker) – a good place to start and can be swapped out if needed.

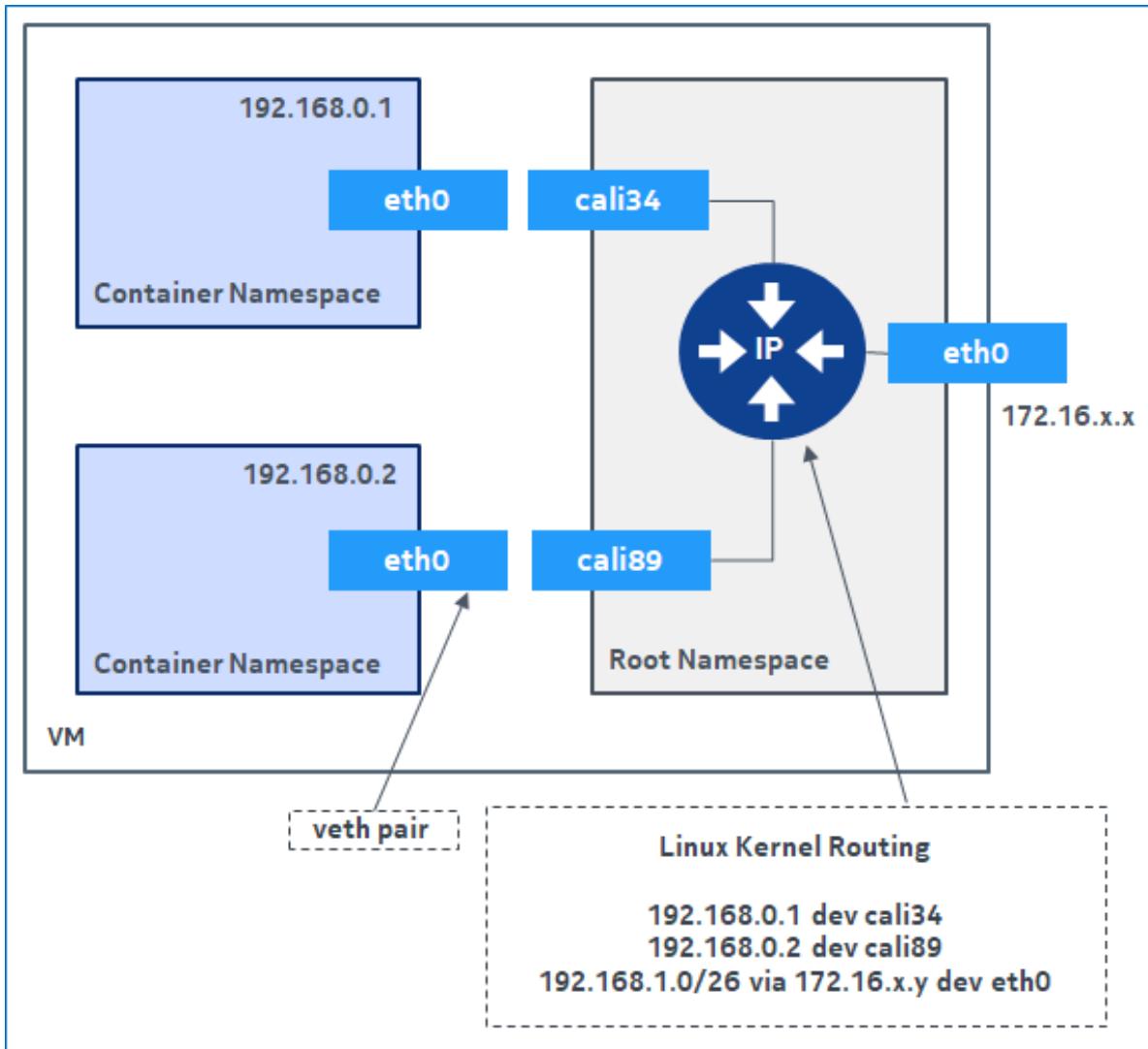
Functionality

When creating a pod, the Calico daemon creates a virtual Ethernet device (veth) pair between the pod and the node's Linux kernel.

Calico adds an explicit route in the kernel routing table on the node where the pod exists. Calico exchanges the routes with other nodes, so the correct routes will be added to every node in the cluster.

When the pod sends packets, the node's Linux kernel will forward the packets according to the routing table. If the destination is in the same node, the packets will be forwarded to it directly. If the destination is in another node, then packets will be transmitted to the node via the internal service network, and forwarded according to the routing table there, then passed to the destination pod.

Figure 3: Calico routing



10.3.1.1 Enable Calico using a configuration file

Use this procedure to enable calico during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set the `bcmt_config.json` file parameter **network_mode** to **calico**.

10.3.2 ipvlan

ipvlan is a new addition to the Linux kernel. Like its cousin macvlan, it virtualizes the host interface. However unlike macvlan which generates a new MAC address for each interface, ipvlan devices all share the same MAC. The kernel driver inspects the IP address of each packet when making a decision about which virtual interface should process the packet.

10.3.2.1 ipvlan usage example

The ipvlan plugin does not work independently in NCS, it works as a delegated plugin of Multus or DANM. Take Multus as example, a **NetworkAttachmentDefinition** should be created firstly, and then we can create pods with text **annotation** for NetworkAttachmentDefinition.

10.3.2.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "ipvlan-net1",
    "type": "ipvlan",
    "master": "eth2",
    "ipam": {
      "type": "whereabouts",
      "range": "172.18.100.10-172.18.100.50/24"
    }
  }'
```

10.3.2.1.2 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: ipvlan-net1
```

10.3.2.2 ipvlan network configuration reference

- **name** (string, required): the name of the network.
- **type** (string, required): *ipvlan*.
- **master** (string, required unless chained): name of the host interface to enslave.
- **mode** (string, optional): one of *I2*, *I3*, *I3s*. Defaults to *I2*.
- **mtu** (integer, optional): explicitly set MTU to the specified value. Defaults to the value chosen by the kernel.
- **ipam** (dictionary, required unless chained): IPAM configuration to be used for this network.

Note:

A single master interface can not be enslaved by both **macvlan** and **ipvlan**.

**Note:**

ipvlan does not allow virtual interfaces to communicate with the master interface. Therefore the container cannot reach the host via the ipvlan interface.

10.3.3 macvlan

macvlan functions like a switch that is already connected to the host interface. A host interface gets *enslaved* with the virtual interfaces sharing the physical device but having distinct MAC addresses.

10.3.3.1 macvlan usage example

The macvlan plugin does not work independently in NCS, it works as a delegated plugin of Multus or DANM. Take Multus as example, a **NetworkAttachmentDefinition** should be created first, and then create pods with text **annotation** for NetworkAttachmentDefinition.

10.3.3.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "macvlan-net1",
    "type": "macvlan",
    "master": "eth2",
    "ipam": {
      "type": "whereabouts",
      "range": "172.18.100.10-172.18.100.50/24"
    }
  }'
```

10.3.3.1.2 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: macvlan-net1
```

10.3.3.2 macvlan network configuration reference

- **name** (string, required): the name of the network
- **type** (string, required): *macvlan*

- `master` (string, optional): name of the host interface to enslave. Defaults to default route interface.
- `mode` (string, optional): one of `bridge`, `private`, `vepa`, `passthru`. Defaults to `bridge`.
- `mtu` (integer, optional): explicitly set MTU to the specified value. Defaults to the value chosen by the kernel. The value must be [0, master's MTU].
- `ipam` (dictionary, required): IPAM configuration to be used for this network. For interface only without an IP address, create empty dictionary.

**Note:**

A single master interface can not be enslaved by both `macvlan` and `ipvlan`.

**Note:**

Since each virtual interface generated by macvlan has unique MAC, traffic from the pod may get blocked by the cloud infrastructure because of the anti-mac-spoofing feature. So user may need to disable port security in cloud environment if macvlan is used.

10.3.4 host-device

This simple plugin will move the requested device from the host's network namespace to the container's.

host-device usage example

The host-device plugin does not work independently in NCS, it works as a delegated plugin of Multus or DANM. Take Multus as example, a **NetworkAttachmentDefinition** should be created firstly, and then we can create pods with text **annotation** for NetworkAttachmentDefinition

Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: device-eth2
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "host-device",
    "device": "eth2",
    "capabilities": { "ips": true },
    "ipam": {
      "type": "static",
      "routes": [
        { "dst": "172.18.0.0/24" }
      ]
    }
  }
```

```
}
```

Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: '[{"name": "device-eth2", "ips": ["172.18.200.104/24"], "interface": "external"}]'
```

host-device network configuration reference

The device can be specified with any one of four properties:

- **device**: The device name, for example *eth0*, *can0*
- **hwaddr**: A MAC address
- **kernelpath**: The kernel device kobj, for example */sys/devices/pci0000:00/0000:00:1f.6*
- **pciBusID**: A PCI address of network device, for example *0000:00:1f.6*

For this plugin, `CNI_IFNAME` is ignored. Upon DEL, the device is moved back.

The plugin also supports the following *capability argument*:

- **deviceID**: A PCI address of the network device, for example *0000:00:1f.6*

10.3.5 SR-IOV

SR-IOV VFs are managed by the SR-IOV Device Plugin component, which discovers the available VFs on each node and reports them to Kubelet as allocatable resources. For each VF the Device Plugin populates the `TopologyInfo` structure to store its NUMA location, which can be used by Topology Manager for scheduling decisions. The VFs are grouped into resource pools.

If the Kubernetes Topology Manager Policy is *disabled*, the resource pooling option is *static*, which means PF specific resource pools are created. Once the Kubernetes Topology Manager is enabled by selecting either *best-effort* or *strict* policy options, the resource pooling can be set to *dynamic* to merge the same type of VFs into a common resource pool. In this case the Kubernetes Topology Manager takes care of NUMA alignment across VFs, CPU cores, and Hugepages. The following naming structure is used:

Table 23: SR-IOV Resource Naming for static resource pooling (default)

resource pool name	drivers	purpose
nokia.k8s.io/sriov_<PF_interface_name>	mlx5_core, iavf, i40evf, ixgbevf, igbvf	pool for Mellanox & Intel non-DPDK VFs grouped by PF interface name
nokia.k8s.io/sriov_vfio_<PF_interface_name>	vfio-pci	pool for vfio-pci bound VFs for Intel DPDK use-case grouped by PF interface name

Table 24: SR-IOV Resource Naming for dynamic resource pooling

resource pool name	drivers	purpose
nokia.k8s.io/sriov	mlx5_core, iavf, i40evf, ixgbevf, igbvf	pool for Mellanox & Intel non-DPDK VFs
nokia.k8s.io/sriov_vfio	vfio-pci	pool for vfio-pci bound VFs for Intel DPDK use-case

Table 25: SR-IOV Resource Naming for FPGA FEC devices

resource pool name	drivers	purpose
nokia.k8s.io/sriov_fec	any (vfio-pci, igb_uio)	pool for FPGA (Intel Vista Creek & Mount Bryce) provided FEC VFs

Once the SR-IOV resource pools are created, containerized applications can request resources (for example, one or more VFs) from the SR-IOV resource pools.

NCS has automation to create VFs by specifying the `num_of_vfs` attribute of a port (PF). In case Intel Fortville NICs are present, the `num_of_dpdk_vfs` parameter is used to specify the number of VFs prepared for DPDK applications (for example, the `vfio-pci` driver is bound to those VFs during the host boot sequence). FPGA FEC VFs are also initialized during the host boot sequence.

 **Note:** For DPDK privileged mode should be used.

10.3.5.1 SR-IOV usage example

sriov CNI plugin is invoked by Multus or DANM, take Multus as example, before creating the pods that need VFs attached, a **network attachment definition** resource should be created, and then create pods with **text annotation** for NetworkAttachmentDefinition.

10.3.5.2 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: sriov-net1
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/resourceName: nokia.k8s.io/sriov_enslfo
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "sriov",
    "vlan": 1000,
    "spoofchk": "off",
    "trust": "off",
    "ipam": {
      "type": "whereabouts",
      "range": "10.200.10.0/24"
    }
  }'
```

10.3.5.3 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: sriov-net1
```

The pod will have one interface connected to the default network (Calico by default), and another attached with a VF.

10.3.5.4 SR-IOV network configuration reference

- **name** (string, required): the name of the network
- **type** (string, required): *sriov*

- `deviceID` (string, required): A valid pci address of an SRIOV NIC's VF. for example `0000:03:02.3`
- `vlan` (int, optional): VLAN ID to assign for the VF. Value must be in the range 0-4094 (0 for disabled, 1-4094 for valid VLAN IDs).
- `vlanQoS` (int, optional): VLAN QoS to assign for the VF. Value must be in the range 0-7. This option requires `vlan` field to be set to a non-zero value. Otherwise, the error will be returned.
- `mac` (string, optional): MAC address to assign for the VF.
- `ipam` (dictionary, optional): IPAM configuration to be used for this network.
- `spoofchk` (string, optional): turn packet spoof checking on or off for the VF
- `trust` (string, optional): turn trust setting on or off for the VF.
- `link_state` (string, optional): enforce link state for the VF. Allowed values: auto, enable, disable. Note that driver support may differ for this feature. For example, `i40e` is known to work but `igb` doesn't.
- `min_tx_rate` (int, optional): change the allowed minimum transmit bandwidth, in Mbps, for the VF. Setting this to 0 disables rate limiting. The `min_tx_rate` value should be <= `max_tx_rate`. Support of this feature depends on NICs and drivers.
- `max_tx_rate` (int, optional): change the allowed maximum transmit bandwidth, in Mbps, for the VF. Setting this to 0 disables rate limiting.

10.3.6 vhost-user

The vhost-user is a CNI plugin designed to implement userspace networking (as opposed to kernel space networking), like Data Plane Development Kit (DPDK) based applications. It is designed to run with either Open vSwitch with DPDK (OVS-DPDK) or Vector Packet Processor (VPP) along with the Multus CNI plugin in Kubernetes for bare metal container deployment model. It enhances high performance container Networking solution and Dataplane Acceleration for NFV Environment.

Userspace networking requires additional considerations. For one, the interface needs to be created/configured on a local vswitch (running on the host). There may also be a desire to add the interface to a specific network on the host through the local vswitch. Second, when the interface is inserted into the container, it is not owned by the kernel, so additional work needs to be done in the container to consume the interface and add to a network within the container. The Userspace CNI is designed to work with these additional considerations by provisioning the local vswitch (OVS-DPDK/VPP), and by pushing data into the container so the interface can be consumed.

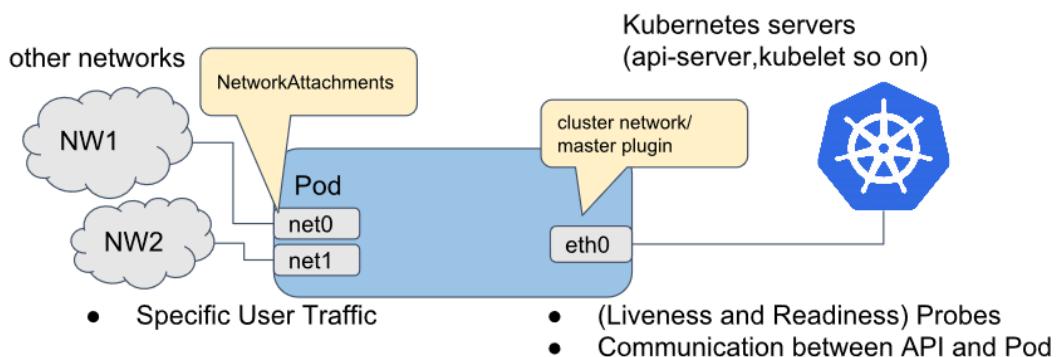
 **Note:** NCS does not install OVS or VPP on the nodes, so if a user wants to use this plugin to provide vhost-user type interfaces to pods, he should install OVS or VPP on the nodes firstly.

10.3.7 Multus

Multus CNI enables attaching multiple network interfaces to pods in Kubernetes.

Multus CNI is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. Typically, in Kubernetes each pod only has one network interface (apart from a loopback) -- with Multus you can create a multi-homed pod that has multiple interfaces. This is accomplished by Multus acting as a "meta-plugin", a CNI plugin that can call multiple other CNI plugins.

Multus CNI follows the Kubernetes Network Custom Resource Definition De-facto Standard to provide a standardized method by which to specify the configurations for additional network interfaces. This standard is put forward by the Kubernetes Network Plumbing Working Group. For SRIOV networks there is no need to create any Ingress/Egress networks with NCS manager GUI. With IPVLAN networks you need to create the Ingress/Egress networks with NCS manager GUI and the subnet can be some dummy subnet, because only the VLAN ID is important. The real IP subnet is given in the *NetworkAttachmentDefinition* entry that you need to create for the Multus network after the NCS deployment. The mapping of Ingress/Egress networks to NIC ports is done in table "Ingress / Egress per port configuration" in hostgroup configuration. The table is visible in NCS Manager only if the hostgroup has the edge role configured. IPVLAN networks can be used only in nodes which have edge role. SRIOV networks can be used in any worker or edge nodes.



10.3.7.1 Enable Multus using a configuration file

Use this procedure to enable Multus plugin support during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set the `bcmt_config.json` file parameter `k8s_use_multus` to `true`.
3. Set the `bcmt_config.json` file parameter `multus_node_group` to specify in which node groups Multus should be enabled, leave it "" if you want to enable Multus on all nodes in the cluster..

End of steps: this procedure is complete.

10.3.7.2 Multus usage example

For pod that does not need multiple interfaces, no additional operation is needed, Multus will invoke the default network plugin to provide interface for the pod. For pod that need additional interface, a

NetworkAttachmentDefinition should be created firstly, and then create pod with text **annotation** for NetworkAttachmentDefinition. For example: ipvlan.

10.3.7.2.1 Network attachment definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "ipvlan-net1",
    "type": "ipvlan",
    "master": "eth2",
    "ipam": {
      "type": "whereabouts",
      "range": "172.18.100.10-172.18.100.50/24"
    }
  }'
```

10.3.7.2.2 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: ipvlan-net1
```

 **Note:** NCS adds a label **cbcs.nokia.com/multus_node=true** on each Multus enabled node, APP pods can use it to choose the corresponding nodes.

10.3.7.2.3 Multus examples for NCS clusters running in VMs (no VLAN tagging is needed)

These example scenarios are tested on NCS 20 & NCS20FP2 releases.

eth0: default Calico interface

net1: Multus generated interface (based on the left column)

`kube-system` namespace + `tiller` service account is used to be able to ping as root inside busybox container. Only for debugging purposes.

 **Warning:** Do not set IP address on the IPVLAN host/master device, which is inside the CIDR being used in Pods' namespaces!

```
6: pf1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc mq state UP
  group default qlen 1000
  link/ether 00:50:56:b0:55:b9 brd ff:ff:ff:ff:ff:ff
```

```

inet6 fe80::250:56ff:feb0:55b9/64 scope link
  valid_lft forever preferred_lft forever
2: pf2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc mq state UP
  group default qlen 1000
  link/ether 00:50:56:b0:64:61 brd ff:ff:ff:ff:ff:ff
  inet6 fe80::250:56ff:feb0:6461/64 scope link
    valid_lft forever preferred_lft forever
4: pf3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc mq state UP
  group default qlen 1000
  link/ether 00:50:56:b0:a5:8d brd ff:ff:ff:ff:ff:ff
  inet6 fe80::250:56ff:feb0:a58d/64 scope link
    valid_lft forever preferred_lft forever

```

Table 26: Multus config file and result

Multus config file	Pod manifest file	Result (kubectl exec ... ip a; echo; ip r)
multustest1		
<pre> apiVersion: "k8s.cni.cncf.io/v1" kind: NetworkAttachmentDefinition metadata: name: device-pf1 namespace: kube-system spec: config: '{ "cniVersion": "0.3.1", "type": "host-device", "device": "pf1", "ipam": { "type": "static", "addresses": [{ "address": "10.10.10.10/24" }, { "address": "fe80:caa5::10/112" }] } }' </pre>	<pre> apiVersion: v1 kind: Pod metadata: name: multustest1 namespace: kube-system annotations: k8s.v1.cni.cncf.io/networks: device-pf1 spec: containers: - image: bcm-registry:5000/busybox:latest name: multustest1 command: ["/bin/sh", "-c", "while true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller </pre>	<pre> 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever inet6 ::1/128 scope host valid_lft forever preferred_lft forever 3: eth0@if2527: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 8900 qdisc noqueue link/ether f6:5a:ae:b8:c5:8e brd ff:ff:ff:ff:ff:ff </pre>

Table 26: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
<pre>}</pre>	<pre>terminationGracePeriodSeconds: 1</pre>	<pre>inet 192.168.107.132/32 scope global eth0 valid_lft forever preferred_lft forever 6: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc mq qlen 1000 link/ether 00:50:56:b0:c2:09 brd ff:ff:ff:ff:ff:ff inet 10.10.10.10/24 brd 10.10.10.255 scope global net1 valid_lft forever preferred_lft forever inet6 fe80::caa5::10/112 scope link valid_lft forever preferred_lft forever inet6 fe80::209:64 scope link valid_lft forever preferred_lft forever</pre>
multustest2		<pre>default via 169.254.1.1 dev eth0 10.10.10.0/24 dev net1 scope link src 10.10.10.10 169.254.1.1 dev eth0 scope link</pre>
<pre>apiVersion: "k8s.cni.cncf.io/v1"</pre>	<pre>apiVersion: v1 kind: Pod metadata:</pre>	<pre>1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536</pre>

Table 26: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (kubectl exec ... ip a; echo; ip r)
<pre>kind: Network AttachmentDefinition metadata: name: ipvlan-pf2 namespace: kube- system spec: config: '{ "cniVersion": "0.3. 1", "type": "ipvlan", "master": "pf2", "mode": "l2", "ipam": { "type": "static", "addresses": [{ "address": "10. 10.10.20/24" }, { "address": "fe80:caa5::20/112" }] } }'</pre>	<pre>name: multustest2 namespace: kube- system annotations: k8s.v1.cni.cncf.io/networks: ipvlan-pf2 spec: containers: - image: bcmt- registry:5000/busybox: latest name: multustest2 command: ["/bin/ sh", "-c", "while true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller terminationGrace PeriodSeconds: 1</pre>	<pre>qdisc noqueue qdisc 1000 link/loopback 00:00: 00:00:00:00 brd 00:00: 00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever inet6 ::1/128 scope host valid_lft forever preferred_lft forever 3: eth0@if2524: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 8900 qdisc noqueue link/ether 42:a7:8a: 8e:a1:18 brd ff:ff:ff: ff:ff:ff inet 192.168.107.188/ 32 scope global eth0 valid_lft forever preferred_lft forever 4: net1@if2: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 8900 qdisc noqueue link/ether 00:50:56: b0:33:bd brd ff:ff:ff: ff:ff:ff inet 10.10.10.20/ 24 brd 10.10.10.255 scope global net1 valid_lft forever preferred_lft forever inet6 fe80:caa5::20/ 112 scope link</pre>

Table 26: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)	
		<pre>valid_lft forever preferred_lft forever inet6 fe80::50:5600:: 1b0:33bd/64 scope link valid_lft forever preferred_lft forever</pre> <pre>default via 169.254.1. 1 dev eth0 10.10.10.0/24 dev net1 scope link src 10.10. 10.20 169.254.1.1 dev eth0 scope link</pre>	
multustest3		<pre>apiVersion: "k8s.cni.cncf.io/v1" kind: Network AttachmentDefinition metadata: name: ipvlan-pf3 namespace: kube- system spec: config: '{ "cniVersion": "0.3. 1", "type": "ipvlan", "master": "pf3", "mode": "l2", "ipam": { "type": "host- local", "subnet": "10.10. 10.0/24", "rangeStart": "10. 10.10.30", } }'</pre> <pre>apiVersion: v1 kind: Pod metadata: name: multustest3 namespace: kube- system annotations: k8s.v1.cni.cncf.io/networks: '[{ "name": "ipvlan- pf3", "ips": ["10.10. 10.30"] }]' spec: containers: - image: bcmt- registry:5000/busybox: latest name: multustest3 command: ["/bin/ sh", "-c", "while</pre>	<pre>1: lo: <LOOPBACK,UP, LOWER_UP> mtu 65536 qdisc noqueue qdisc 1000 link/loopback 00:00: 00:00:00:00 brd 00:00: 00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever 3: eth0@if2525: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 8900 qdisc noqueue link/ether 26:7d:4a: 35:c4:6a brd ff:ff:ff: ff:ff:ff inet 192.168.107.190/ 32 scope global eth0</pre>

Table 26: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (kubectl exec ... ip a; echo; ip r)
<pre>"rangeEnd": "10.10.10.39", "routes": [{ "dst": "10.30.0.0/16" }], "gateway": "10.10.10.1" } }'</pre>	<pre>true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller terminationGrace PeriodSeconds: 1</pre>	<pre>valid_lft forever preferred_lft forever 4: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc noqueue link/ether 00:50:56:b0:a0:f1 brd ff:ff:ff:ff:ff:ff inet 10.10.10.30/24 brd 10.10.10.255 scope global net1 valid_lft forever preferred_lft forever</pre>
multustest4		
<pre>apiVersion: "k8s.cni.cncf.io/v1" kind: NetworkAttachmentDefinition metadata: name: ipvlan-pf3-whereabouts namespace: kube-system spec: config: '{'</pre>	<pre>apiVersion: v1 kind: Pod metadata: name: multustest4 namespace: kube-system annotations: k8s.v1.cni.cncf.io/networks: ipvlan-pf3-whereabouts spec: containers:</pre>	<pre>1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qdisc 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever</pre>

Table 26: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (kubectl exec ... ip a; echo; ip r)
<pre>"cniVersion": "0.3.1", "type": "ipvlan", "master": "pf3", "mode": "l2", "ipam": { "type": "whereabouts", "range": "10.10.10.0/24", "range_start": "10.10.40", "range_end": "10.10.49", "routes": [{ "dst": "10.40.0.0/16" }], "gateway": "10.10.1" } }'</pre>	<pre>- image: bcmt- registry:5000/busybox:latest name: multustest3 command: ["/bin/sh", "-c", "while true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller terminationGracePeriodSeconds: 1</pre>	<pre>3: eth0@if2526: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 8900 qdisc noqueue link/ether a6:d9:6b:b3:ca:7f brd ff:ff:ff:ff:ff:ff inet 192.168.107.140/32 scope global eth0 valid_lft forever preferred_lft forever 4: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8900 qdisc noqueue link/ether 00:50:56:b0:a0:f1 brd ff:ff:ff:ff:ff:ff inet 10.10.10.40/24 brd 10.10.10.255 scope global net1 valid_lft forever preferred_lft forever default via 169.254.1.1 dev eth0 10.10.10.0/24 dev net1 scope link src 10.10.10.40 10.40.0.0/16 via 10.10.1 dev net1 169.254.1.1 dev eth0 scope link</pre>

Comments for each scenario:

1. `pf1` host device (e.g. vnic in the VM) is put to the Pod's namespace. From now on `pf1` is not seen in host namespace.

- no network sharing/virtualization is in place
- only one Pod can own this `pf1` interface, no more Pods can use `pf1` on the same node
- static IPv4 & IPv6 addresses are set

2. IPVLAN network virtualization is used on `pf2` host device

- many Pods can connect to the same host interface
- static IPv4 & IPv6 addresses are set

3. IPVLAN network virtualization is used on `pf3` host device

- many Pods can connect to the same host interface
- do not use `host-local` IPAM cluster-wide (it serves unique IP per node, not per cluster)
- static IP is selected in Pod annotation
- sample routing definition is set

4. IPVLAN network virtualization is used on `pf3` host device

- many Pods can connect to the same host interface
- *whereabouts* IPAM is used to serve cluster-wide unique IPs
- in NCS *whereabouts* is configured to store IP allocations in Kubernetes CRD (*ippools* resource)
- sample routing definition is set

Ping test:

```
[root@bcm-t-vc-sut-sandbox1-1 multus]# kubectl exec -ti multustest1 -n kube-system -- ping -c1 10.10.10.20
PING 10.10.10.20 (10.10.10.20): 56 data bytes
64 bytes from 10.10.10.20: seq=0 ttl=64 time=0.145 ms
--- 10.10.10.20 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.145/0.145/0.145 ms
[root@bcm-t-vc-sut-sandbox1-1 multus]# kubectl exec -ti multustest1 -n kube-system -- ping -c1 10.10.10.30
PING 10.10.10.30 (10.10.10.30): 56 data bytes
64 bytes from 10.10.10.30: seq=0 ttl=64 time=1.393 ms
--- 10.10.10.30 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.393/1.393/1.393 ms
[root@bcm-t-vc-sut-sandbox1-1 multus]# kubectl exec -ti multustest1 -n kube-system -- ping -c1 10.10.10.40
PING 10.10.10.40 (10.10.10.40): 56 data bytes
64 bytes from 10.10.10.40: seq=0 ttl=64 time=1.685 ms
--- 10.10.10.40 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
```

```
round-trip min/avg/max = 1.685/1.685/1.685 ms
[root@bcmt-vc-sut-sandbox1-1 multus]# kubectl exec -ti multustest1 -n kube-
system -- ping -c1 fe80:caa5::20
PING fe80:caa5::20 (fe80:caa5::20): 56 data bytes
64 bytes from fe80:caa5::20: seq=0 ttl=64 time=1.102 ms
--- fe80:caa5::20 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 1.102/1.102/1.102 ms
```

10.3.7.2.4 Multus examples for NCS clusters running on baremetal without SR-IOV (VLAN tagging is required)

These example scenarios are tested on NCS20FP2 release.

eth0: default Calico interface

net1: Multus generated interface (based on the left column)

`kube-system` namespace + `tiller` service account is used to be able to ping as root inside busybox container. Only for debugging purposes.

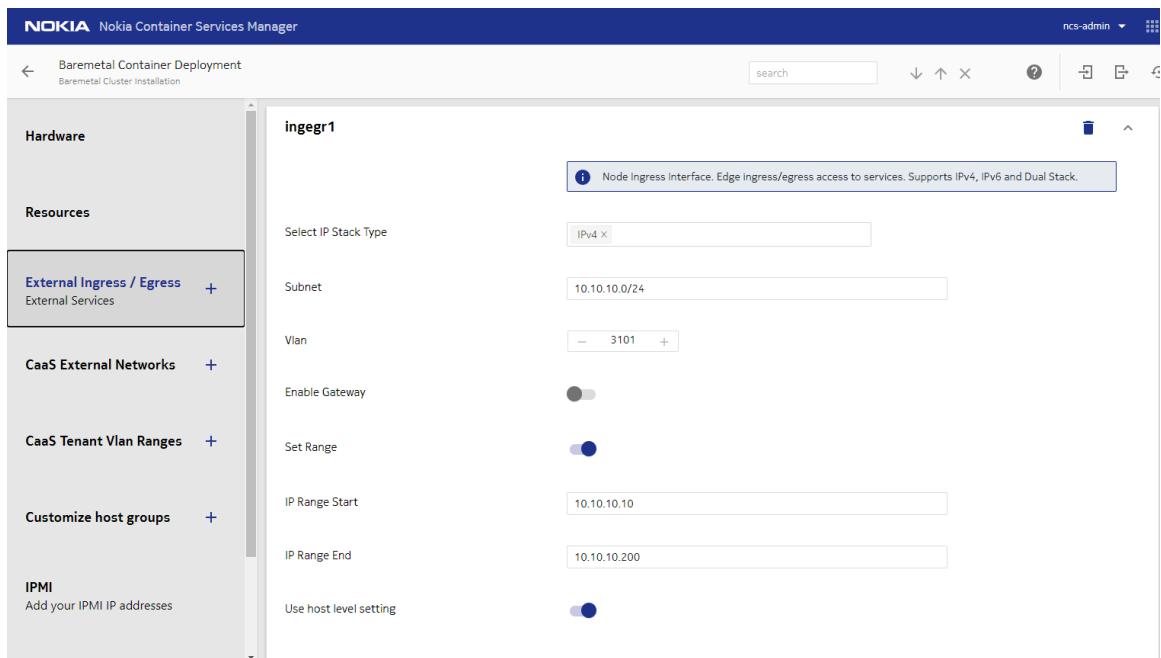
 **Warning:** Do not set IP address on the IPvLAN host/master device, which is inside the CIDR being used in Pods' namespaces!

Create VLAN interfaces via NCS Manager

Custom VLAN interfaces can be created at NCS deployment time. Use the NCS Manager GUI as follows:

step 1.: Define "External Ingress / Egress" configuration:

Figure 4: External Ingress /Egress configuration



If you will use IPVLAN or MACVLAN network type for Pod interface, specify a dummy CIDR which is outside of the CIDR what you would like to use for Pod networking. Multiple "External Ingress / Egress" definitions can be created. VLAN IDs must be unique.

step 2.: Assign the "External Ingress / Egress" definitions to bond interfaces on the host group panel:

Figure 5: Customize Host Groups

Bond Name	Internal Networks Mapping
nic_2_bond	
nic_3_bond	

Bond Name	Ingress/Egress Nets
nic_2_bond	ingegr1
nic_3_bond	ingegr2

Two predefined host groups have "Ingress / Egress per port configuration" table: "AllinOne" & "EdgeBM". In case user creates custom host groups, select edge role to reveal the "Ingress / Egress per port configuration" table.

In case of three NIC setup, the `nic_2_bond` become `tenant-bond`; `nic_3_bond` become `provider-bond` on host level.

The VLAN interfaces are created on those nodes which are assigned to these host groups:

```

38: wlan3101@tenant-bond: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc
noqueue state UP group default qlen 1000
                                link/ether 90:e2:ba:ae:29:08
brd ff:ff:ff:ff:ff:ff
                                inet 10.10.10.10/24 brd
10.10.10.255 scope global wlan3101
                                valid_lft forever
preferred_lft forever
                                inet6
fe80::92e2:baff:feae:2908/64 scope link
                                valid_lft forever
preferred_lft forever

```

step 3.: Multus config:

Table 27: Multus config file and result

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
multustest2		

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-vlan3101-1
  namespace: kube-system
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "ipvlan",
    "master": "vlan3101"
  ,
    "mode": "12",
    "ipam": {
      "type": "static",
      "addresses": [
        { "address": "10.100.10.20/24" },
        { "address": "fe80:caa5::20/112" }
      ]
    }
  }'
```

```
apiVersion: v1
kind: Pod
metadata:
  name: multustest2
  namespace: kube-system
  annotations:
    k8s.v1.cni.cncf.io/networks: ipvlan-vlan3101-1
spec:
  containers:
    - image: bcmt-registry:5000/busybox:latest
      name: multustest2
      command: [ "/bin/sh", "-c", "while true; do date; sleep 10; done" ]
      serviceAccount: tiller
      serviceAccountName: tiller
      terminationGracePeriodSeconds: 1
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever
3: eth0@if230: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether b2:90:3d:f3:0d:b9 brd ff:ff:ff:ff:ff:ff inet 192.168.75.155/32 scope global eth0 valid_lft forever preferred_lft forever
4: net1@if38: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether ec:0d:9a:6b:a2:dc brd ff:ff:ff:ff:ff:ff
```

Table 27: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)	
		<pre>inet 10.100.10.20/ 24 brd 10.100.10.255 scope global net1 valid_lft forever preferred_lft forever inet6 fe80::caa5::20/ 112 scope link valid_lft forever preferred_lft forever inet6 fe80::ec0d: 9a00:16b:a2dc/64 scope link valid_lft forever preferred_lft forever</pre> <pre>default via 169.254.1. 1 dev eth0 10.100.10.0/24 dev net1 scope link src 10.100.10.20 169.254.1.1 dev eth0 scope link</pre>	
multustest3		<pre>apiVersion: "k8s.cni.cncf. io/v1" kind: Network AttachmentDefinition metadata: name: ipvlan- vlan3101-2 namespace: kube- system spec: config: '{ "cniVersion": "0.3. 1",</pre> <pre>apiVersion: v1 kind: Pod metadata: name: multustest3 namespace: kube- system annotations: k8s.v1.cni.cncf.io/networks: '[{ "name": "ipvlan- vlan3101-2", "ips": ["10.100.10. 30"]</pre>	<pre>1: lo: <LOOPBACK,UP, LOWER_UP> mtu 65536 qdisc noqueue qdisc 1000 link/loopback 00:00: 00:00:00:00 brd 00:00: 00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever 3: eth0@if263: < BROADCAST,MULTICAST,</pre>

Table 27: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
<pre> "type": "ipvlan", "master": "vlan3101" , "mode": "l2", "ipam": { "type": "host- local", "subnet": "10.100. 10.0/24", "rangeStart": "10. 100.10.30", "rangeEnd": "10. 100.10.39", "routes": [{ "dst" : "10.30.0.0/16" }], "gateway": "10.100. 10.1" } } </pre>	<pre> }]' spec: containers: - image: bcmt- registry:5000/busybox: latest name: multustest3 command: ["/bin/ sh", "-c", "while true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller terminationGrace PeriodSeconds: 1 </pre>	<pre> UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether 9e:e6:3f: 89:b9:b4 brd ff:ff:ff: ff:ff:ff inet 192.168.197.116/ 32 scope global eth0 valid_lft forever preferred_lft forever 4: net1@if38: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether 90:e2:ba: ae:29:08 brd ff:ff:ff: ff:ff:ff inet 10.100.10.30/ 24 brd 10.100.10.255 scope global net1 valid_lft forever preferred_lft forever </pre>
multustest4		<pre> default via 169.254.1. 1 dev eth0 10.30.0.0/16 via 10. 100.10.1 dev net1 10.100.10.0/24 dev net1 scope link src 10.100.10.30 169.254.1.1 dev eth0 scope link </pre>
<pre> apiVersion: "k8s.cni.cncf. io/v1" </pre>	<pre> apiVersion: v1 kind: Pod </pre>	<pre> 1: lo: <LOOPBACK,UP, LOWER_UP> mtu 65536 </pre>

Table 27: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
<pre> kind: Network AttachmentDefinition metadata: name: ipvlan- vlan3101-whereabouts namespace: kube- system spec: config: '{ "cniVersion": "0.3. 1", "type": "ipvlan", "master": "vlan3101" , "mode": "l2", "ipam": { "type": "whereabouts", "range": "10.100. 10.0/24", "range_start": "10. 100.10.40", "range_end": "10. 100.10.49", "routes": [{ "dst" : "10.40.0.0/16" }], "gateway": "10.100. 10.1" } }' </pre>	<pre> metadata: name: multustest4 namespace: kube- system annotations: k8s.v1.cni.cncf.io/networks: ipvlan-vlan3101- whereabouts spec: containers: - image: bcmt- registry:5000/busybox: latest name: multustest3 command: ["/bin/ sh", "-c", "while true; do date; sleep 10; done"] serviceAccount: tiller serviceAccountName: tiller terminationGrace PeriodSeconds: 1 </pre>	<pre> qdisc noqueue qdisc 1000 link/loopback 00:00: 00:00:00:00 brd 00:00: 00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever 3: eth0@if231: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether 66:46:ba: 81:6c:8c brd ff:ff:ff: ff:ff:ff inet 192.168.75.156/ 32 scope global eth0 valid_lft forever preferred_lft forever 4: net1@if38: < BROADCAST,MULTICAST, UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether ec:0d:9a: 6b:a2:dc brd ff:ff:ff: ff:ff:ff inet 10.100.10.40/ 24 brd 10.100.10.255 scope global net1 valid_lft forever preferred_lft forever default via 169.254.1. 1 dev eth0 10.40.0.0/16 via 10. 100.10.1 dev net1 </pre>

Table 27: Multus config file and result (continued)

Multus config file	Pod manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
		<pre>10.100.10.0/24 dev net1 scope link src 10.100.10.40 169.254.1.1 dev eth0 scope link</pre>

Comments for each scenario:

1. IPVLAN network virtualization is used on `vlan3101` VLAN interface

- many Pods can connect to the same VLAN interface
- static IPv4 & IPv6 addresses are set

2. IPVLAN network virtualization is used on `vlan3101` VLAN interface

- many Pods can connect to the same VLAN interface
- do not use `host-local` IPAM cluster-wide (it serves unique IP per node, not per cluster)
- static IP is selected in Pod annotation
- sample routing definition is set

3. IPVLAN network virtualization is used on `vlan3101` VLAN interface

- many Pods can connect to the same VLAN interface
- `whereabouts` IPAM is used to serve cluster-wide unique IPs
- in NCS `whereabouts` is configured to store IP allocations in Kubernetes CRD (`ippools` resource)
- sample routing definition is set

Ping test:

```
[root@baremetal-airframe-sut2-allinone-0 multus-bm]# kubectl exec multustest2
-n kube-system -- ping -c1 10.100.10.30
PING 10.100.10.30 (10.100.10.30): 56 data bytes
64 bytes from 10.100.10.30: seq=0 ttl=64 time=0.102 ms
--- 10.100.10.30 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet
loss
round-trip min/avg/max = 0.102/0.102/0.102 ms
[root@baremetal-airframe-sut2-allinone-0 multus-bm]#
kubectl exec multustest2 -n kube-system -- ping -c1 10.100.10.40
```

```
PING 10.100.10.40 (10.100.10.40): 56 data bytes
64 bytes from 10.100.10.40: seq=0 ttl=64 time=0.053 ms
--- 10.100.10.40 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet
loss
round-trip min/avg/max = 0.053/0.053/0.053 ms
```

10.3.7.2.5 Multus examples for NCS clusters running on baremetal with SR-IOV (VLAN tagging is required)

These example scenarios are tested on NCS20FP2 release.

eth0: default Calico interface

net1: Multus generated interface (based on the left column)

`kube-system` namespace + `tiller` service account is used to be able to ping as root inside busybox container. Only for debugging purposes.

 **Warning:** Do not set IP address on the IPVLAN host/master device, which is inside the CIDR being used in Pods' namespaces!

Create SR-IOV VF devices via NCS Manager

step 1.: Specify the number of required non-DPDK and DPDK VF devices per physical interfaces:

Figure 6: Customize Host Groups



Port Name	Number of Non DPDK VFs on port	Number of DPDK VFs on port	Enable Trust on Port
nic_2_port_1	8	2	false
nic_2_port_2	8	2	false
nic_3_port_1	8	2	false
nic_3_port_2	8	2	false

step 2.: Once the deployment is ready, verify that SR-IOV Device Plugin (SRIOVDP) pools are populated properly:

```
# kubectl describe nodes | egrep "Name:|sriov_"
Name: baremetal-airframe-sut2-allinone-0
nokia.k8s.io/sriov_ens1f0: 8
nokia.k8s.io/sriov_ens1f1: 8
nokia.k8s.io/sriov_vfio_ens1f0: 2
nokia.k8s.io/sriov_vfio_ens1f1: 2
[...]
```

step 3.: Multus config:

Table 28: Multus config file and result

Multus config file	Deployment manifest file	Result (kubectl exec ... ip a; echo; ip r)
<pre> --- apiVersion: "k8s.cni.cncf.io/v1" kind: NetworkAttachmentDefinition metadata: name: sriov-a namespace: kube-system annotations: k8s.v1.cni.cncf.io/resourceName: nokia.k8s.io/sriov_ens1f0 spec: config: '{ "cniVersion": "0.3.1", "type": "sriov", "vlan": 1000, "spoofchk": "off", "trust": "off", "ipam": { "type": "whereabouts", "range": "10.200.10.0/24" } }' --- apiVersion: "k8s.cni.cncf.io/v1" kind: NetworkAttachmentDefinition metadata: name: sriov-b namespace: kube-system annotations: </pre>	<pre> apiVersion: apps/v1 kind: Deployment metadata: labels: run: multustest name: multustest namespace: kube-system spec: replicas: 3 selector: matchLabels: run: multustest template: metadata: namespace: kube-system labels: run: multustest annotations: k8s.v1.cni.cncf.io/networks: sriov-a@sriov-a, sriov-b@sriov-b spec: containers: - image: bcmt-registry:5000/busybox:latest name: multustest command: ["/bin/sh", "-c", "while true; do date; sleep 10; done"] resources: requests: nokia.k8s.io/sriov_ens1f0: '1' nokia.k8s.io/sriov_ens1f1: '1' </pre>	<p><u>sample Pod:</u></p> <pre> 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo valid_lft forever preferred_lft forever 3: eth0@if272: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 9000 qdisc noqueue link/ether 7a:62:96:35:75:bd brd ff:ff:ff:ff:ff:ff inet 192.168.197.125/32 scope global eth0 valid_lft forever preferred_lft forever 19: sriov-a: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq qlen 1000 link/ether 4e:ab:85:23:b8:4b brd ff:ff:ff:ff:ff:ff inet 10.200.10.1/24 brd 10.200.10.255 scope global sriov-a valid_lft forever preferred_lft forever 25: sriov-b: <BROADCAST,MULTICAST> </pre>

Table 28: Multus config file and result (continued)

Multus config file	Deployment manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
<pre> k8s.v1.cni.cncf.io/resource Name: nokia.k8s.io/sriov_ ens1f1 spec: config: '{ "cniVersion": "0.3. 1", "type": "sriov", "vlan": 1000, "spoofchk": "off", "trust": "off", "ipam": { "type": " whereabouts", "range": "10.200. 20.0/24" } }' </pre>	<pre> limits: nokia.k8s.io/sriov_ens1f0: '1' nokia.k8s.io/sriov_ens1f1: '1' serviceAccount: tiller serviceAccountName: tiller terminationGracePeriodSeconds: 1 </pre>	<pre> UP,LOWER_UP> mtu 1500 qdisc mq qlen 1000 link/ether 3a:a4:fb:ef:0f:ac brd ff:ff:ff:ff:ff:ff inet 10.200.20.1/24 brd 10.200.20.255 scope global sriov-b valid_lft forever preferred_lft forever </pre> <pre> default via 169.254.1.1 dev eth0 10.200.10.0/24 dev sriov-a scope link src 10.200.10.1 10.200.20.0/24 dev sriov-b scope link src 10.200.20.1 169.254.1.1 dev eth0 scope link </pre> <u>sample Node:</u> <pre> 4: ens1f0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 9000 qdisc mq master tenant-bond state UP mode DEFAULT group default qlen 1000 link/ether 90:e2:ba:ae:29:08 brd ff:ff:ff:ff:ff:ff vf 0 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off </pre>

Table 28: Multus config file and result (continued)

Multus config file	Deployment manifest file	Result (kubectl exec ... ip a; echo; ip r)
		<pre> vf 1 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off vf 2 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off vf 3 MAC 76:9c:b5:4b:bd:24, spoof checking on, link-state auto, trust off, query_rss off vf 4 MAC da:cb:d2:d9:c7:04, spoof checking on, link-state auto, trust off, query_rss off vf 5 MAC 00:00:00:00:00:00, spoof checking on, link-state auto, trust off, query_rss off vf 6 MAC 4e:ab:85:23:b8:4b, vlan 1000, spoof checking off, link-state auto, trust off, query_rss off vf 7 MAC 86:9f:82:98:15:89, spoof checking on, link-state auto, trust off, query_rss off vf 8 MAC ca:b6:c9:5d:4b:14, spoof checking on, link-state auto, </pre>

Table 28: Multus config file and result (continued)

Multus config file	Deployment manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
		<pre>trust off, query_rss off vf 9 MAC 1e:27:f6:ec: 58:f3, spoof checking on, link-state auto, trust off, query_rss off 5: ens1f1: <BROADCAST, MULTICAST,SLAVE,UP, LOWER_UP> mtu 9000 qdisc mq master tenant-bond state UP mode DEFAULT group default qlen 1000 link/ether 90:e2:ba: ae:29:08 brd ff:ff:ff: ff:ff:ff vf 0 MAC 00:00:00:00: 00:00, spoof checking on, link-state auto, trust off, query_rss off vf 1 MAC 00:00:00:00: 00:00, spoof checking on, link-state auto, trust off, query_rss off vf 2 MAC 3a:a4:fb: ef:0f:ac, vlan 1000, spoof checking off, link-state auto, trust off, query_rss off vf 3 MAC 72:85:8d:44: 1e:fd, spoof checking on, link-state auto, trust off, query_rss off vf 4 MAC 00:00:00:00: 00:00, spoof checking</pre>

Table 28: Multus config file and result (continued)

Multus config file	Deployment manifest file	Result (<code>kubectl exec ... ip a; echo; ip r</code>)
		<pre>on, link-state auto, trust off, query_rss off vf 5 MAC a2:3b:33:3d: b3:a2, spoof checking on, link-state auto, trust off, query_rss off vf 6 MAC 00:00:00:00: 00:00, spoof checking on, link-state auto, trust off, query_rss off vf 7 MAC 62:b8:8c:94: 54:db, spoof checking on, link-state auto, trust off, query_rss off vf 8 MAC 00:00:00:00: 00:00, spoof checking on, link-state auto, trust off, query_rss off vf 9 MAC 22:7f:41:b9: 94:1b, spoof checking on, link-state auto, trust off, query_rss off</pre>

Comments for the scenario:

- SRIOV CNI (<https://github.com/k8snetworkplumbingwg/sriov-cni>) is used to configure selected VF(s), like VLAN tagging
- SRIOVDP (<https://github.com/k8snetworkplumbingwg/sriov-network-device-plugin>) performs the node and VF selections according to the resource requests/limits
- `whereabouts` (<https://github.com/openshift/whereabouts-cni>) IPAM is used to serve cluster-wide unique IPs
- in NCS `whereabouts` is configured to store IP allocations in Kubernetes CRD (`ippools` resource)

- 1-1 VFs are requested from pools. No bonding is possible across VFs
- Multus generated interfaces are renamed to `sriov-a` and `sriov-b` respectively

ARP table on Leafs:

```
Codes: *N - VLT Peer Synced MAC
        *I - Internal MAC Address used for Inter Process
              Communication
      VlanId Mac Address Type Interface State
      1000 36:9c:ef:c9:ae:9a Dynamic Po 1781 Active
      1000 3a:a4:fb:ef:0f:ac Dynamic Po 1781 Active
      1000 42:26:db:0b:80:57 Dynamic Tf 1/11/3 Active
      1000 4e:ab:85:23:b8:4b Dynamic Te 1/18/3 Active
      1000 9a:b1:eb:62:de:1e Dynamic Te 1/18/4 Active
      1000 b6:f5:6f:75:70:10 Dynamic Po 1781 Active
```

10.3.7.3 Enable Multus by node group

NCS supports installing both Multus and DANM in the same cluster, the steps are below:

Create different groups of nodes

Due to the limitation that one node can have only one of this kind of meta plugin, so if we want to have both Multus and DANM in the same cluster, we should create different node groups for Multus and DANM nodes.

When create the VM resources, add the label **ncs.nokia.com/group=group_xx** about the node group for each node, which is configures with the Multus cni metaplugin, where **group_xx** is the group_name, for example: **ncs.nokia.com/group=WorkerBM**. With the **group_xx**, the **multus_node_group** will choose the right **group_xx**.

Configure Multus parameters in bcmt_config_<platform>.json file

- **k8s_use_multus** should be set as true if we need to enable Multus in the cluster
- **multus_node_group** specifies in which node group(s) Multus will be enabled, for example "group_01,group_04", if we want to enable Multus on all nodes, leave it ""



Note: "" means all groups, 'multus_node_group' and 'danm_node_group' should not overlap with each other.

Install the cluster

NCS will add corresponding label for the nodes during installation.

- **ncs.nokia.com/multus_node=true** for Multus enabled nodes

Pod settings

In previous versions, for APP pods that need to run on Multus enabled nodes, it was possible to use the nodeSelector:

```
nodeSelector:
  cbcs.nokia.com/multus_node: 'true'
```

From NCS20FP2, When APP pods use the Multus network, APP pods don't need to add nodeSelector, the scheduler will choose the Multus enabled nodes dynamically for APP pods.

If user wants to add the nodeSelector, use the following format:

```
nodeSelector:
  ncs.nokia.com/multus_node: 'true'
```

10.3.8 Source based routing (SBR)

This plugin performs Source Based Routing (SBR). In NCS it is intended to enhance *Multus* on page 137 with this functionality.

The most common and standard way to perform routing is to base it purely on the destination.

However, in some applications which use network separation for traffic management and security, there is no way to tell beforehand which interface should be used. However, the application can decide.

As an example, a Telco application might have two networks, a management network and a SIP (telephony) network for traffic, with rules which state that:

- SIP traffic (only) must be routed over the SIP network.
- All other traffic (but no SIP traffic) must be routed over the management network.

There is no way of configuring this based on a destination IP as there is no way of determining whether a destination IP on the internet is an address used for downloading updated software packages or a remote SIP endpoint.

Example of a typical NCS configuration

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-pf3-sbr
  namespace: kube-system
spec:
  config: '{
    "plugins": [
      {
        "cniVersion": "0.3.1",
        "type": "ipvlan",
```

```

"master": "pf3",
"mode": "l2",
"ipam":
{
    "type": "whereabouts",      "range": "10.10.10.0/24",      "range_start": "10.10.10.50",      "range_end": "10.10.10.59",      "routes": [
        { "dst": "10.50.0.0/16" }
    ],
    "gateway": "10.10.10.1"
},
{
    "type": "sbr"
}
}
apiVersion: v1
kind: Pod
metadata:
    name: multustest5
    namespace: kube-system
    annotations:
        k8s.v1.cni.cncf.io/networks: ipvlan-pf3-sbr
spec:
    containers:
        - image: bcmt-registry:5000/busybox:latest
          name: multustest5
          command: [ "/bin/sh", "-c", "while true; do date; sleep 10; done" ]
    serviceAccount: tiller
    serviceAccountName: tiller
    terminationGracePeriodSeconds: 1

```

10.3.9 DANM - Virtual Deployment

DANM is another solution from Nokia for attaching multiple network interfaces to pods in Kubernetes.

DANM is a networking solution for telco workloads running in a Kubernetes cluster. It's built up from the following components:

- A CNI plugin capable of provisioning IPVLAN interfaces with advanced features
- An in-built IPAM module with the capability of managing multiple, cluster-wide, discontinuous L3 networks and provide a dynamic, static, or no IP allocation scheme on-demand
- A CNI metaplugin capable of attaching multiple network interfaces to a container, either through its own CNI, or through delegating the job to any of the popular CNI solution like SRI-OV, or Flannel in parallel

- A Kubernetes controller capable of centrally managing both VxLAN and VLAN interfaces of all Kubernetes hosts
- Another Kubernetes controller extending Kubernetes' Service-based service discovery concept to work over all network interfaces of a Pod

With this toolset DANM is able to provide multiple separated network interfaces, the possibility to use different networking back ends and advanced IPAM features for the pods

The DANM is composed of the following components:

- **danm** is the CNI plugin which can be directly integrated with kubelet. Internally it consists of the CNI metaplugin, the CNI plugin responsible for managing IPVLAN interfaces, and the in-built IPAM plugin. Danm binary is integrated to kubelet as any other CNI plugin.
- **fakeipam** is a little program used in natively integrating third party CNI plugins into the DANM ecosystem. It is basically used to echo the result of DANM's in-built IPAM to CNIs DANM delegates operations to. Fakeipam binary should be placed into kubelet's configured CNI plugin directory, next to danm. Fakeipam is a temporary solution, the long-term aim is to separate DANM's IPAM component into a full-fledged, standalone IPAM solution.
- **netwatcher** is a Kubernetes Controller watching the Kubernetes API for changes in the DANM related CRD network management APIs. This component is responsible for validating the semantics of network objects, and also for maintaining VxLAN and VLAN host interfaces of all Kubernetes nodes. Netwatcher binary is deployed in Kubernetes as a DaemonSet, running on all nodes.
- **svctracker** is another Kubernetes Controller monitoring Pod, Service, Endpoint, and DanmEp API paths. This Controller is responsible for extending Kubernetes native Service Discovery to work even for the non-primary networks of the Pod. Svctracker binary is deployed in Kubernetes as a DaemonSet, running only on the Kubernetes master nodes in a clustered setup.
- **webhook** is a standard Kubernetes Validating and Mutating Webhook. It has multiple, crucial responsibilities:
 - it validates all DANM introduced CRD APIs both syntactically, and semantically both during creation, and modification
 - it automatically mutates parameters only relevant to the internal implementation of DANM into the API objects
 - it automatically assigns physical network resources to the logical networks of tenant users in a production-grade infrastructure

10.3.9.1 Enable DANM using a configuration file

Use this procedure to enable DANM during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set the `bcmt_config.json` file parameter `k8s_use_danm` to "true".
3. Set the `bcmt_config.json` file parameter `danm_node_group` to specify in which node groups DANM should be enabled, leave it "" if you want to enable DANM on all nodes in the cluster.

End of steps: this procedure is complete

10.3.9.2 DANM usage example

The network that provided by DANM is called **danmnet**, a danmnet should be defined in advance, before any pod trying to use it.

A DANM network can be defined by using the following CRD schema: <https://github.com/nokia/danm/blob/master/schema/DanmNet.yaml>

- Create a danmnet

Example: net-oam.yaml

```
apiVersion: danm.k8s.io/v1
kind: DanmNet
metadata:
  name: oam
  namespace: default
spec:
  NetworkID: oam
  NetworkType: ipvlan
  Options:
    host_device: eth2
    cidr: 192.168.10.0/24
    allocation_pool:
      start: 192.168.10.10
      end: 192.168.10.100
    vxlan: 101
```



Note: The DanmNet **spec.NetworkID** value must exactly match the **metadata.name** field.

- Apply the danmnet using the following command:

```
kubectl apply -f net-oam.yaml
```

- Show the defined danmnet and detailed information

```
kubectl get danmnet
kubectl describe danmnet oam
```

10.3.9.3 DanmNet specifications in pod annotations

NCS creates a "default" danmnet in every namespace by default, it stands for the network plugin (Calico) that the user chose when installing NCS.

- If users want two interfaces in the pod, one from Calico and the other gets dynamic IP from the DANM network “ctrl”, the annotation file should be like this:

```
annotations:
danm.k8s.io/interfaces: |
[
{
  "network": "default",
  "ip": "dynamic"
},
{
  "network": "ctrl",
  "ip": "dynamic"
}
]
```

- If user want both the two interfaces in the pod from DANM networks, one from the network “oam” and has a static IP, the other from “ctrl”, and gets IP dynamically, then it should be like this:

```
annotations:
danm.k8s.io/interfaces: |
[
{
  "network": "oam",
  "ip": "192.168.10.10/24"
},
{
  "network": "ctrl",
  "ip": "dynamic"
}
]
```

 **Note:**

- The static ip must be in the range of the CIDR of the danmnet, and the first interface of a pod must be named as "eth0".
- The proute gateway must be reachable if proute is needed for the pod.

 **Note:** NCS adds a label **ncs.nokia.com/danm_node=true** on each DANM enabled node, APP pods can use it to choose the corresponding nodes.

10.3.10 DANM - Baremetal Deployment

10.3.10.1 DANM Overview

DANM is a networking solution for telco workloads running in a Kubernetes cluster. NCS20FP2SU2 integrates DANM 4.2 release as an optional “CNI metaplugin”. It can be enabled on the “CaaS Cluster” panel of the NCS Manager.

Main features:

- A CNI plugin capable of provisioning IPVLAN interfaces with advanced features
- An in-built IPAM module with the capability of managing multiple, cluster-wide, discontinuous L3 networks with managing up to 8M allocations per network; plus providing dynamic, static, or no IP allocation scheme on-demand for both IPv4, and IPv6
- A CNI metaplugin capable of attaching multiple network interfaces to a container, either through its own CNI, or through delegating the job to any of the popular CNI solution for example SR-IOV, Calico, Flannel and so on in parallel.
- A Kubernetes controller capable of centrally managing both VxLAN and VLAN interfaces of all Kubernetes hosts.
- Another Kubernetes controller extending Kubernetes' Service-based service discovery concept to work over all network interfaces of a Pod.
- A standard Kubernetes Validating and Mutating Webhook responsible for making you adhere to the schemas, furthermore automating network resource management for tenant users in a production-grade environment.

Supported network types:

- *Dynamic integration level*: parameters are passed over CNI API
- – **ipvlan** (built-in)
- **macvlan** (CNI)
- **sriov** (CNI + utilizes VF resource management by SR-IOV Device Plugin)
- *Static integration level*: parameters/config options are stored in static file on nodes
 - **calico**, **flannel** (typically the default CNI in the Kubernetes cluster)

The above functionalities are implemented by the following components:

- danm is the CNI plugin which can be directly integrated with kubelet. Internally it consists of the CNI metaplugin, the CNI plugin responsible for managing IPVLAN interfaces, and the in-built IPAM plugin. Danm binary is integrated to kubelet as any other CNI plugin.
- fakeipam is a little program used in natively integrating third party CNI plugins into the DANM ecosystem. It is basically used to echo the result of DANM's in-built IPAM to CNIs DANM delegates operations to. Fakeipam binary should be placed into kubelet's configured CNI plugin directory, next to danm. Fakeipam is a temporary solution, the long-term aim is to separate DANM's IPAM component into a full-fledged, standalone IPAM solution.
- netwatcher is a Kubernetes Controller watching the Kubernetes API for changes in the DANM related CRD network management APIs. This component is responsible for validating the semantics of

network objects, and also for maintaining VxLAN and VLAN host interfaces of all Kubernetes nodes. Netwatcher binary is deployed in Kubernetes as a DaemonSet, running on all nodes.

- svcwatcher is another Kubernetes Controller monitoring Pod, Service, Endpoint, and DanmEp API paths. This Controller is responsible for extending Kubernetes native Service Discovery to work even for the non-primary networks of the Pod. Svcwatcher binary is deployed in Kubernetes as a DaemonSet, running only on the Kubernetes master nodes in a clustered setup.
- webhook is a standard Kubernetes Validating and Mutating Webhook. It has multiple crucial responsibilities:
 - It validates all DANM introduced CRD APIs both syntactically, and semantically both during creation, and modification
 - It automatically mutates parameters only relevant to the internal implementation of DANM into the API objects
 - It automatically assigns physical network resources to the logical networks of tenant users in a production-grade infrastructure

DANM supports two kind of network management experiences: *lightweight* (the only supported mode before v4.0, that is DanmNet), and *production-grade* (which introduces ClusterNetworks and TenantNetworks). NCS integrates all of these CRD-based management APIs:

- **DanmNets:** Both administrators, and tenant users manage their networks in their own namespaces. Everyone has the same level of access, and can configure all the parameters supported by DANM. It is not possible to create networks across tenants/namespaces.
- **TenantNetworks** is a namespaced API, and can be freely created by tenant users. It basically is the same API as DanmNet, with one big difference: parameters any way related to host settings cannot be freely configured through this API. These parameters are automatically filled by DANM based on **TenantConfig** settings.
- **ClusterNetworks** on the other hand is a cluster-wide API, and as such, can be -or should be- only provisioned by administrator level users. Administrators can freely set all available configuration options, even the physical parameters. The other nice thing in ClusterNetworks is that all Pods, in any namespace can connect to them - unless the network administrator forbade it via the newly introduced AllowedTenants configuration list.

NCS automatically populates ClusterNetworks according to “CaaS External Networks” in NCS Manager. TenantConfig is set according to “CaaS Tenant VLAN Ranges”.

DANM's SR-IOV integration supports both Intel and Mellanox manufactured physical functions. Pods can use the allocated Virtual Functions for DPDK purposes. Intel VFs must be bound to the “vfio-pci” kernel driver before the Pod is instantiated. These special VFs are separated into dedicated device pool named “nokia.k8s.io/sriov_vfio_<interface>”.

DANM's IPAM module supports both pure IPv6, and dual-stack (one IPv4, and one IPv6 address provisioned to the same interface) addresses. To configure an IPv6 CIDR for a network, network administrators shall configure the "net6" attribute. Similarly to IPv4 address management operators can define a desired allocation pool for their V6 subnet via the "allocation_pool_v6" structure. Additionally, IP routes for IPv6

subnets can be configured via "routes6". If both "cidr", and "net6" are configured for the same network. Pods connecting to that network can ask either one IPv4 or IPv6 address - or even both at the same time.

10.3.10.2 DANM Usage

When you define CNF networks in “CaaS external networks” of Nokia Container Services (NCS) Manager GUI and map them to NIC ports in table “Interface per port configuration” table under hostgroup configurations, NCS cluster installation will create those networks as ClusterNetworks and create the VLAN interface for each network to all Edge and Worker nodes (no matter to which hostgroup they belong). For CNFs it is not necessary to create any DanmNets, CNF pods from any namespace can directly use the existing ClusterNetworks.

If the CNF creates its own DanmNet (which is namespace specific network) or if user wants to create namespace-specific DanmNet for the CNF, then verify that user does not have the ClusterNetwork with same name and VLAN ID already created. If it is present, delete that ClusterNetwork before deploying the CNF/creating a DanmNet entry for the CNF.

When creating the DanmNet, there is a netwatcher mechanism in NCS cluster which recognizes that new DanmNet and dynamically configures the corresponding VLAN interface to Edge and Worker nodes.

Because of this mechanism of automatic creation of VLAN interfaces into worker and edge nodes it is not necessary to define any CaaS External Networks in the NCS installation. So user does not need to know beforehand the kind of external networks that different CNFs will need. They can be added afterwards. In NCS Manager GUI during NCS installation it is enough to define in the “Caas Tenant Vlan Ranges” the range of DANM VLAN IDs that the CNFs will use. When deploying CNFs which uses DanmNet, VLAN interfaces will be automatically created to edge and worker nodes.

If DANM network is SRIOV network and user has defined the network in CaaS external networks and mapped it to NIC port in NCS Manager GUI table “SR-IOV to port configuration” of hostgroup, a ClusterNetwork will be created for that to all edge and worker nodes. One important thing to note is with “ip link show” the user does not see that VLAN in the VF. It only appears there when the user deploys some CNF that requests for VF for this VLAN.

10.3.10.3 Enable DANM by node group

NCS supports installing both Multus and DANM in the same cluster, the steps are below:

Create different groups of nodes¶

Due to the limitation that one node can have only one of this kind of meta plugin, so if we want to have both Multus and DANM in the same cluster, we should create different node groups for Multus and DANM nodes.

When create the VM resources, add the label **ncs.nokia.com/group=group_xx** about the node group for each node, which is configures with the DANM cnf metaplugin, where **group_xx** is the group_name, for example: **ncs.nokia.com/group=WorkerBM**. With the **group_xx**, the **danm_node_group** will choose the right **group_xx**.

Configure DANM parameters in bcmt_config_<platform>.json file¶

- **k8s_use_danm** should be set as true if we need to enable DANM in the cluster
- **danm_node_group** specifies in which node group(s) DANM will be enabled, for example "group_02,group_03", if we want to enable DANM on all nodes, leave it ""



Note: "" means all groups, 'multus_node_group' and 'danm_node_group' should not overlap with each other.

Install the cluster¶

NCS will add a corresponding label for the nodes during installation.

- **danm_net=true** for DANM enabled nodes

Pod settings¶

In previous versions, for APP pods that need to run on DANM enabled nodes, it was possible to use the nodeSelector:

```
nodeSelector:  
  danm_net: 'true'
```

From NCS20FP2, When APP pods use the DANM network, APP pods don't need to add nodeSelector, the scheduler will choose the DANM enabled nodes dynamically for APP pods.

10.3.11 DANM - Renewing certificate

In NCS 20FP2, the DANM certificate is required to be renewed annually to avoid DANM not allocating IP addresses to the pods.

To renew the certificate, execute the following:

1. On one of the controllers run:

```
find /data0 -name webhook-create-signed-cert.sh
```

2. Take the first result and copy it to the tmp directory:

```
cp -p /data0/...../webhook-create-signed-cert.sh /tmp/webhook-  
create-signed-cert.sh  
chmod +x /tmp/webhook-create-signed-cert.sh  
cd /tmp  
. webhook-create-signed-cert.sh
```

3. Restart the deployment:

```
kubectl rollout restart deployment danm-webhook -n kube-system
```

4. Checking if it is being renewed:

```
openssl s_client -connect danm-webhook-svc.kube-system.svc:443 | openssl
x509 -noout -dates
```

10.3.12 Enable Multus and DANM by node group

NCS supports installing both Multus and DANM in the same cluster, the steps are described below:

Create different groups of nodes

Due to the limitation that one node can have only one of this kind of meta plugin, to use both Multus and DANM in the same cluster, you need to create different node groups for Multus and DANM nodes.

When creatinge the VM resources, add the labels **ncs.nokia.com/group=group_01** about the node group for each node.

Configure Multus and DANM parameters in bcmt_config_<platform>.json file¶

- **k8s_use_multus** should be set as true if we need to enable Multus in the cluster
- **multus_node_group** specifies in which node group(s) Multus will be enabled, for example "group_01,group_04", if we want to enable Multus on all nodes, leave it ""
- **k8s_use_danm** should be set as true if we need to enable DANM in the cluster
- **danm_node_group** specifies in which node group(s) DANM will be enabled, for example "group_02,group_03", if we want to enable DANM on all nodes, leave it.*



Note: * means all groups, 'multus_node_group' and 'danm_node_group' should not overlap with each other.

Install the cluster

NCS will add a corresponding label for the nodes during installation.

- **ncs.nokia.com/multus_node=true** for Multus enabled nodes
- **danm_net=true** for DANM enabled nodes

Pod settings

Before NCS20FP2, APP pods that need to run on Multus enabled nodes, can utilize the `nodeSelector`:

```
nodeSelector:
  cbcs.nokia.com/multus_node: 'true'
```

For APP pods that need to run on DANM enabled nodes:

```
nodeSelector:
```

```
dannm_net: 'true'
```

From NCS20FP2, APP pods do not need to add the `nodeSelector`, as the scheduler will choose the Multus enabled nodes or DANM enabled nodes dynamically for the APP pods.

If the user wants to add the `nodeSelector`, use the following format:

```
nodeSelector:  
  ncs.nokia.com/multus_node: 'true'
```

10.3.12.1 Configuring Multus and DANM together

Requirements

For the sake of backward compatibility the cluster level config shall remain in the NCS API, and the HG level to be added in extra.

Nodes should be automatically labelled based on this config param. Labels already used in CN-A must be re-purposed

Pods submitted into the cluster must be mutated to ensure they are automatically scheduled to the appropriate host group.

Configure DANM+Multus in a cluster

1. Deploy the cluster with DANM and multus network.
2. After the deployment is done, check that a interface is created in ifconfig.

Configure DANM + Multus networks in a cluster and then reboot the nodes.

1. Configure multus+DANM networks from NCS manager.
2. After the deployment is done, do the ping test.
3. Now reboot the controller node.

Configure DANM+Multus network in a cluster and then perform backup/restore operations on the nodes.

1. Configure DANM+multus network.
2. After the deployment is done, do ping test.
3. Now backup of the node.
4. Remove the multus configuration from the node and check the IP interface is gone.
5. Restore the node and verify the DANM and multus networks are up.

Configure DANM+Multus network and then perform cluster heal on the node.

1. Configure DANM and multus network.
2. After the deployment is done, check IP interfaces are added on the node.
3. Now put the nodes in NotReady state (so we can perform heal on these nodes).

4. Then perform cluster heal operation on this node and verify the cluster is healed and the DANM and Multus configuration is intact.

Configure DANM +Multus and then perform scale out test the node.

1. Configure DANM and multus network.
2. After the deployment is done, check the IP interfaces are added on the node.
3. Now scale in the node(so we can perform scale out on this node).
4. Then perform scale out operation on this node and verify that the DANM +Multus configuration is present.

10.3.13 IPv6 Support

NCS supports IPv6 addresses for pods and external networks. (Note: There is a limitation with CBIS BM deployment where IPv6 cannot be support on the internal service network.)

Feature flag: `network_stack`.

The options for `network_stack`: `ipv4_only`, `ipv4_dualstack` and `ipv6_only`.

Description of each option:

Table 29: NCS CLI Category

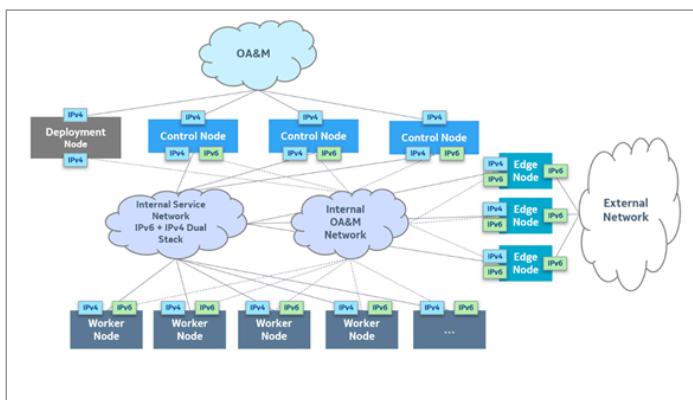
network_stack	description
<code>ipv4_only</code> (default)	The cluster operates with IPv4 address only. All infrastructure components (Kubernetes and other services) and applications run with IPv4 addresses.
<code>ipv4_dualstack</code>	The cluster operates with IPv4/IPv6 dualstack. Infrastructure components (Kubernetes and other services) still run with IPv4 addresses, but application Pods are allocated with IPv4/IPv6 dualstack addresses. Kubernetes is only aware of IPv4 address of Pods and cluster IP is of IPv4. This requires calico as the CNI plugin and internal service network and external network with dual stacks. NCS OAM networks still requires IPv4 network.
<code>ipv6_only</code>	The cluster operates with IPv6 address only. All infrastructure components (Kubernetes and other services) and applications run with IPv6 addresses.

10.3.13.1 Dual Stack

In dual stack mode, NCS supports IPv4/IPv6 addresses for pods and external networks. O&M, internal O&M and internal storage networks are restricted to using IPv4 addresses. If it is enabled in previous version, the feature flag will be changed to **network_stack: ipv4_dualstack** during NCS upgrade.

Network elements require varying IP address support, as outlined below:

- Deployment Server does not require an IPv6 address.
- Control, Worker and Edge nodes require IPv4 and IPv6 dual stack configurations.
- Kubernetes components require IPv4 addresses.
- Pods require IPv4 and IPv6 addresses.



10.3.13.2 IPv6 only

In IPv6 Only mode, all kubernetes components and nodes run with IPv6 addresses only. All nodes must have IPv6 addresses.

Note: As full IPv6 Only support in NCS is still in progress, the scope of IPv6 support is limited and due to some unresolved issues of k8s and other components, there are limitation when enabling IPv6 Only mode.

Scope:

- NCS life cycle events. Deployment, upgrade, recovery, scale and uninstall.
- Network. Calico, DANM, Multus.
- Storage. OpenStack Cinder (OpenStack API shall provide IPv6 access), Rook/Ceph, vSphere, Shared Storage (GlusterFS), Local storage.
- NCS Portal
- Onekey. OpenStack, vcenter, baremetal
- KVNF
- Cluster admin
- NCS Firewall.

- High availability.
- NTP (chrony)
- DNS
- NCS Egress.
- SELinux
- ETCD Ramdisk
- Container Backup and Restore (CBUR)

Out of scope:

- Storage: AWS.
- Network: SRIOV, DPDK.
- Onekey: AWS
- CITM
- CLOG
- Elk Stack (BELK)

10.3.14 Tuning

The tuning plugin can change some system controls (sysctls) and several interface attributes (promiscuous mode, MTU and MAC address) in the network namespace.

10.3.14.1 Tuning usage example

Tuning does not create any network interfaces and therefore does not bring connectivity by itself. It is only useful when used in addition to other plugins. Take macvlan as example, a multus **NetworkAttachmentDefinition** should be created firstly, and then we can create pods with text **annotation** for NetworkAttachmentDefinition.

10.3.14.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-tuning-net1
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-tuning-net1
```

```

spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "macvlan-tuning-net1",
    "plugins": [ {
      "type": "macvlan",
      "master": "eth2",
      "ipam": {
        "type": "whereabouts",
        "range": "172.18.100.10-172.18.100.11/24"
      }
    },
    {
      "type": "tuning",
      "mac": "c2:11:22:33:44:55",
      "mtu": 1454
    }
  ]
}'

```

10.3.14.1.2 Pod annotation

```

annotations:
  k8s.v1.cni.cncf.io/networks: macvlan-tuning-net1

```

10.3.14.2 Tuning network configuration reference

- `mac` (string, optional): MAC address (i.e. hardware address) of interface
- `mtu` (integer, optional): MTU of interface
- `promisc` (bool, optional): Change the promiscuous mode of interface
- `sysctl` (object, optional): Change system controls

10.3.15 Whereabouts

Whereabouts is an IP Address Management (IPAM) CNI plugin that assigns IP addresses cluster-wide. In NCS, `Whereabouts` is taken as an enhancement for Multus, so when Multus is enabled, `Whereabouts` is also installed.

10.3.15.1 Whereabouts usage example

Whereabouts can be used to allocate IP addresses for the network plugins that do not have built-in IPAM. Take `ipvlan` as example.

10.3.15.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "ipvlan-net1",
    "type": "ipvlan",
    "master": "eth2",
    "ipam": {
      "type": "whereabouts",
      "range": "192.168.2.225/28"
    }
  }'
```

10.3.15.1.2 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: ipvlan-net1
```

10.3.15.2 Whereabouts IPAM configuration reference

10.3.15.2.1 Required

- **type:** This should be set to "whereabouts".
- **range:** This specifies the range in which IP addresses will be allocated.

In this case the range is set to **192.168.2.225/28**, this will allocate IP addresses in the range.

10.3.15.2.2 Range end syntax

Additionally, the range parameter can support a CIDR notation that includes the last IP to use.

Example:

```
range: "192.168.2.225-192.168.2.230/28"
```

10.3.15.2.3 Optional

The following parameters are optional:

- `range_start`: First IP to use when allocating from the range. Optional, if unset is inferred from the range.
- `range_end`: Last IP to use when allocating from the range. Optional, if unset the last ip within the range is determined.
- `exclude`: This is a list of CIDRs to be excluded from being allocated.

In the example, we exclude IP addresses in the range **192.168.2.229/30** from being allocated (in this case it is 3 addresses, .229, .230, .231), as well as **192.168.2.236/32** (just a single address).



Note: It depends on you to properly set exclusion ranges that are within your subnet, there is no double checking for you (other than that the CIDR notation parses).

Additionally -- you can set the route, gateway and DNS using anything from the configurations for the static IPAM plugin, [static](#) on page 177, (as well as additional static IP addresses).

10.3.16 static

static IPAM is very simple IPAM plugin that assigns IPv4 and IPv6 addresses statically to container. This will be useful in debugging purpose and in case of assign same IP address in different vlan/vxlan to containers.

10.3.16.1 static IPAM usage example

static can be used to allocate IP addresses for the network plugins that do not have built-in IPAM. Take `ipvlan` as example.

10.3.16.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: net-static
spec:
  config: '{
    "cniVersion": "0.3.1",
    "type": "ipvlan",
    "capabilities": { "ips": true },
    "master": "eth2",
    "ipam": {
      "type": "static",
      "routes": [
        { "dst": "172.18.0.0/24" }
      ]
    }
  }'
```

10.3.16.1.2 Pod annotation

```
annotations:
  k8s.v1.cni.cncf.io/networks: '[{"name": "net-static", "ips": ["172.18.1.100/24"], "interface": "eth1"}]'
```

10.3.16.2 static IPAM configuration reference

- **type** (string, required): "static"
- **addresses** (array, optional): an array of ip address objects:
 - **address** (string, required): CIDR notation IP address.
 - **gateway** (string, optional): IP inside of "subnet" to designate as the gateway.
- **routes** (string, optional): list of routes add to the container namespace. Each route is a dictionary with "dst" and optional "gw" fields. If "gw" is omitted, value of "gateway" will be used.
- **dns** (string, optional): the dictionary with "nameservers", "domain" and "search"

10.3.17 host-local

host-local IPAM allocates IPv4 and IPv6 addresses out of a specified address range. Optionally, it can include a DNS configuration from a resolv.conf file on the host.

10.3.17.1 host-local usage example

host-local can be used to allocate IP addresses for the network plugins that do not have built-in IPAM. Take ipvlan as example.

10.3.17.1.1 Network Attachment Definition

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: local-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "local-net1",
    "type": "ipvlan",
    "master": "eth2",
    "ipam": {
```

```

    "type": "host-local",
    "subnet": "172.18.1.0/24",
    "rangeStart": "172.18.1.10",
    "rangeEnd": "172.18.1.100"
}
}'

```

10.3.17.1.2 Pod annotation

```

annotations:
  k8s.v1.cni.cncf.io/networks: local-net1

```

10.3.17.2 host-local IPAM configuration reference

- **type** (string, required): "host-local".
- **routes** (string, optional): list of routes to add to the container namespace. Each route is a dictionary with "dst" and optional "gw" fields. If "gw" is omitted, value of "gateway" will be used.
- **resolvConf** (string, optional): Path to a `resolv.conf` on the host to parse and return as the DNS configuration
- **dataDir** (string, optional): Path to a directory to use for maintaining state, e.g. which IPs have been allocated to which containers
- **ranges**, (array, required, nonempty) an array of arrays of range objects:
 - **subnet** (string, required): CIDR block to allocate out of.
 - **rangeStart** (string, optional): IP inside of "subnet" from which to start allocating addresses. Defaults to ".2" IP inside of the "subnet" block.
 - **rangeEnd** (string, optional): IP inside of "subnet" with which to end allocating addresses. Defaults to ".254" IP inside of the "subnet" block for ipv4, ".255" for IPv6
 - **gateway** (string, optional): IP inside of "subnet" to designate as the gateway. Defaults to ".1" IP inside of the "subnet" block.

10.4 Ingress

Ingress control on Edge nodes provides external network to internal container access. An nginx load balancer sits between the external network and the cluster accepting incoming network and application traffic, and distributes the traffic across multiple backend nodes.

For more information about Ingress requests, please consult the Istio Traffic Management document <https://istio.io/v1.7/docs/tasks/traffic-management>

10.5 Egress

While Worker nodes may be configured to have access to external networks, more typically, Edge nodes are configured to act as gateways for bi-directional traffic between internal containers and external networks. This limits the ability of Worker nodes to directly access external networks, thereby increasing system security.

Control of Edge node egress, from internal containers to external networks, is provided by Source Network Address Translation (SNAT).

Two egress options exist:

1. Control and Worker nodes can directly access the external network. In such a case, Control and Worker nodes must have an external network IP address assigned.
2. Control and Worker nodes cannot directly access the external network, and must use an Edge node, provided an external network IP address has been assigned to the Edge nodes.

Figure 7: Control/Worker node with direct access to an external network

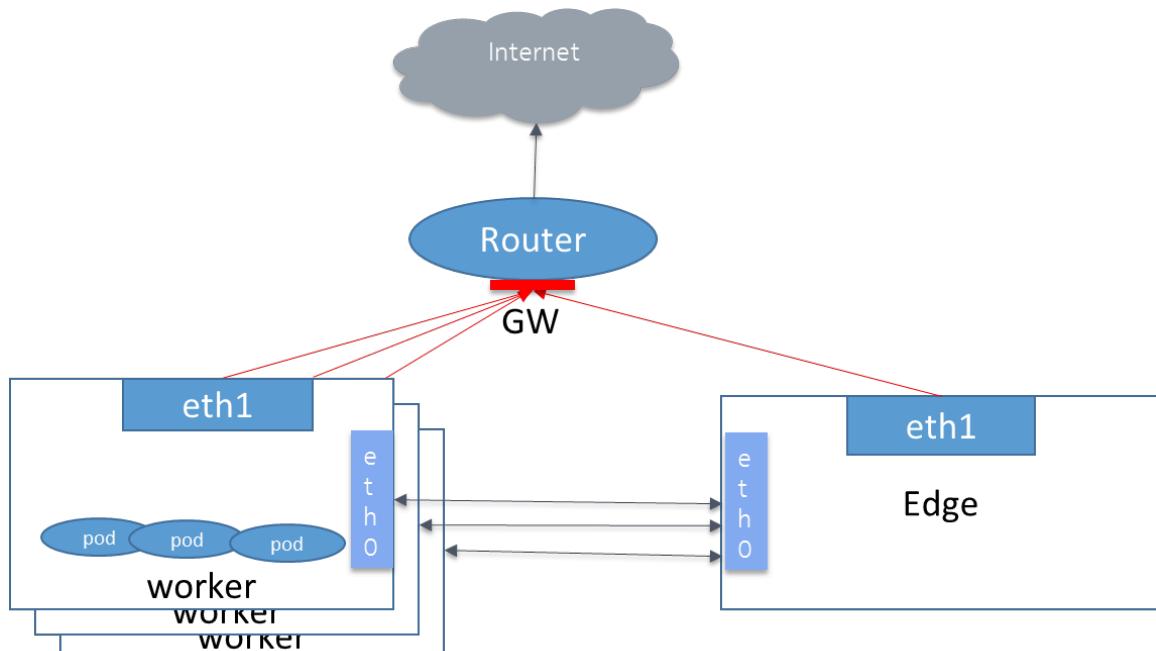
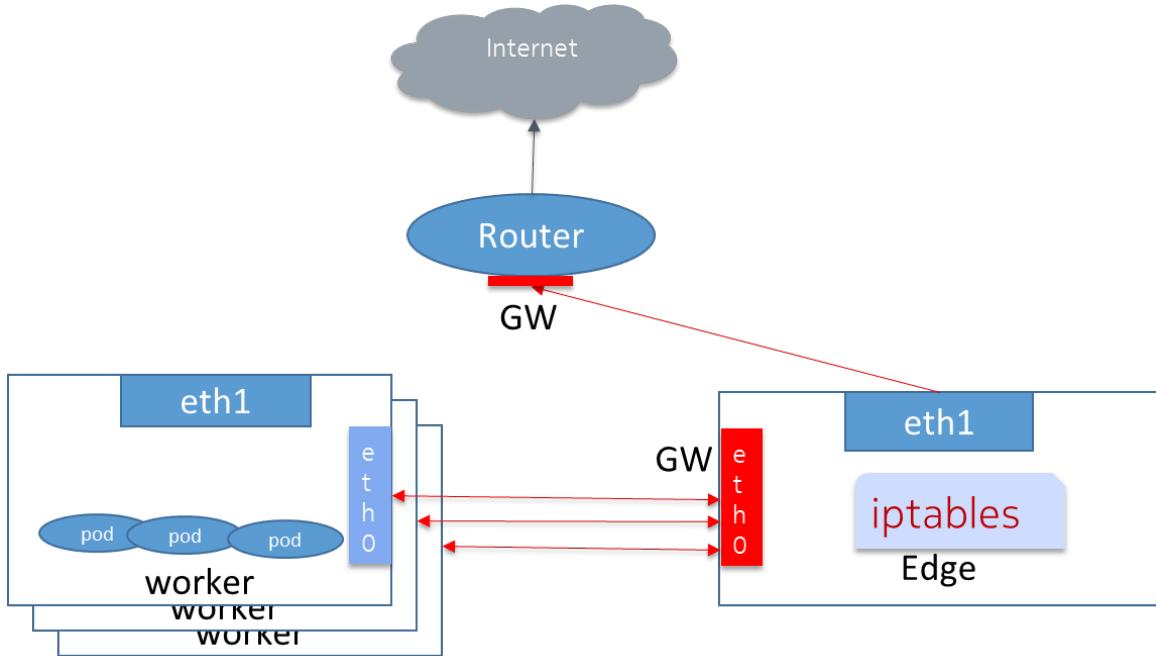


Figure 8: Control/Worker node with indirect access to an external network



10.5.1 Enable cluster egress using a configuration file

Use this procedure to enable cluster egress using Ansible tasks during NCS deployment.

1. Access the `bcmt_config.json` file.

2. Set the follow parameters:

- **egress_mode**
- **worker_gw**
- **control_gw**
- **edge_if**
- **egress_mode_v6**
- **worker_gw_v6**
- **control_gw_v6**
- **edge_if_v6**

3. This feature is ready for deployment.

10.5.2 Enable cluster egress using a CRD

Use this procedure to enable cluster egress using a CustomResourceDefinition (CRD) during NCS deployment.

1. (Optional) Create a NextHopGroup configuration file. For example:

```
ncm_v1alpha1_nexthopgroup.yaml.
```

```
apiVersion: ncm.nokia.com/v1alpha1
kind: NextHopGroup
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: nexthopgroup-sample
spec:
  nodeSelector:
    node: edge # Nodes providing gateway/egress function.
  nextHopTemplate:
    addressFamily: IPv4 # Route address family. It must be the same as the one in
    EgressGateway. Options: IPv4, IPv6.
    bidirectionalForwardingDetectionInterface: eth1 # Interface as BFD local peer on gateway node,
    gateway for worker nodes.
    sourceNetworkAddressTranslation: true # Enable SNAT for outgoing traffic
    sourceNetworkAddressTranslationInterfaces: # Interfaces on which SNAT is done. Usually, these are
    interfaces via which traffic goes to the external.
    - eth0
```

2. (Optional) Apply the NextHopGroup.

```
kubectl apply -f ncm_v1alpha1_nexthopgroup.yaml
```

3. Create an EgressGateway configuration file. For example: ncm_v1alpha1_egressgateway.yaml.

```
apiVersion: ncm.nokia.com/v1alpha1
kind: EgressGateway
metadata:
  name: egressgateway-sample
spec:
  nodeSelector:
    node: worker # Worker nodes to configure the egress function
  nodeEgressGatewayTemplate:
    addressFamily: IPv4 # Route address family
    interface: eth1 # Interface used as local BFD peer on the node
    nextHopGroup: nexthopgroup-sample # Next hop group which provides the gateway function and
    the remote BFD peer
    nextHop:
      - 192.168.10.1 # Ignored, if nextHopGroup is specified.
```

4. Apply the EgressGateway.

```
kubectl apply -f ncm_v1alpha1_egressgateway.yaml
```

10.5.3 Manage cluster egress using CLI commands

1. Access the current egress settings using the command:

```
ncs network egress get
```

Example output:

```
{"egress_port": "eth0", "egress_status": "workeronly", "egress_worker_gateway": "172.16.1.30"}
```

2. Access and define the current egress settings using the command:

```
ncs network egress set
```

Define the following settings:

- **egress_gateway_interface** the interface name, for example: eth0
- **ip_type** egress ip type: ipv6, ipv4 (default)
- **egress_mode** options available are: controlworker, controlonly, workeronly (default).
- **egress_worker_gateway** this is the Worker's egress gateway. If the egress mode is controlworker or workeronly, set this value to a private IP address on the edge node.
- **egress_control_gateway** this is the Control's egress gateway. If the egress mode is controlworker, or controlonly, set this value to a private IP address on the edge node.

Example:

```
ncs network egress set --egress_gateway_interface eth0 --egress_worker_gateway 172.16.1.30
```

Example output:

```
{"task-status": "pending"}  
{"task-status": "running"}  
.....  
{"task-status": "done"}
```

10.5.4 Reset cluster egress configuration settings

1. Display the current egress configuration for a cluster using the command:

```
ncs network egress get
```

2. Remove the current egress configuration for a cluster using the command:

```
ncs network egress reset
```

Example output:

```
{"task-status": "pending"}  
{"task-status": "running"}  
.....
```

```
{"task-status": "done"}
```

Refer to Istio Traffic Management document to find other information about Ingress and Egress requests
<https://istio.io/v1.7/docs/tasks/traffic-management>

10.6 Istio

Istio is a service mesh infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a cloud native application. In practice, the service mesh is implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware and provides:

- Traffic Management
- Policy Enforcement
- Metrics, Logs and Traces
- Security



Note: Current version for NCS20 FP2 is 1.7.



Note: We currently support upgrade from version 1.6 to 1.7. For versions prior to 1.6, the user needs to upgrade to 1.6 first and then to 1.7.

For more information, refer to <https://istio.io/docs/concepts/what-is-istio/>

Istio is provided as an independent module and can be installed after NCS deployment. For further information see the installation guides listed in *Document list for Nokia Container Services (NCS)* on page 19.

10.6.1 Istio in Nokia Container Services

The Istio service mesh is delivered as an independent module of NCS. All life cycle events can be managed with Helm.

NCS does some enhancement to the official Istio chart to meet the requirement of NCS. Official plugins were split into different packages. Prometheus or Grafana are provided by CSF CPRO to leverage the full function of components. Jaeger is provided as a separate package bcmt-jaeger. Kiali stays in NCS Istio, as it is dedicated to Istio.

To provide customized root cert/key for mTLS in the mesh, you must generate the secret with root CA/key/CA chains for Citadel.

You can choose to enable auto injection or not by specifying it in the values of the Helm installation.

NCS adds a component named Bosun to support the backup and restore function. It can be enabled or disabled in Istio chart. Backup and restore are done via CSF Container Backup and Restore along with Bosun. Container Backup and Restore performs the backup or restore function when triggered by user

action or a periodic job defined in BrPolicy. Bosun acts as the field for the backup or restore event. When the user executes the backup or restore operation or a periodical job is triggered in Container Backup and Restore, Container Backup and Restore gets into Istio Bosun pod and runs backup and restore actions. Bosun also provides istioctl installation function if enabled.

In this Istio release, there is following new function added.

- **Multi-cluster deployment** From this release, bcmt Istio supports multi-cluster deployment.



Note: There is significant change to the multi-cluster feature in Istio 1.8, use current version (1.7.6) with caution.

- **To mixer users,** Mixer is removed completed in Istio 1.8, prepare for post-Mixer era.

In the previous Istio release (1.7), there are three significant changes on how Istio is deployed.

- **Control Plane Restructuring**

Istiod as a new component is the core of the control plane, and handles configuration and certificate distribution, sidecar injection, and more. Removed the separate Citadel, Sidecar Injector, and Galley deployments and all functionality is moved into Istiod.

- **Istio CRD Installation**

The following table shows the CRD changes.

Table 30: CRDs changes

add	delete
peerauthentications.security.istio.io requestauthentications.security.istio.io workloadentries.networking.istio.io	meshpolicies.authentication.istio.io policies. authentication.istio.io

- **Support ended for v1alpha1 security policy**

Istio 1.7 no longer supports the following security policy APIs:

- v1alpha1 authentication policy
- v1alpha1 RBAC policy

Istio 1.7 replaces the v1alpha1 authentication policy with the following APIs:

- The v1beta1 request authentication policy
- The v1beta1 peer authentication policy

Istio 1.7 replaced the v1alpha1 RBAC policy APIs with the v1beta1 authorization policy APIs.

The details info refer to <https://istio.io/latest/news/releases/1.6.x/announcing-1.6/upgrade-notes/#support-ended-for-v1alpha1-security-policy>

- **Secret Discovery Service (SDS)**

From this release, SDS is enabled by default and Node Agent deployment is removed which functionality exists in the Envoy sidecar.

- **Global mTLS**

From this release, configuration global.mtls.enabled is removed in istio values.yaml due to upstream change (Meshpolicy API is removed). Instead, manually configure a peer authentication policy to enable strict mTLS. Refer to the upgrade section for Istio upgrading.

- **Telemetry V2 (experimental)**

In this release, Istio supports telemetry V2, which uses an in-proxy extension mechanism to collect telemetry. Mixer Telemetry is deprecated now and is supported until Istio 1.7. There are some gaps in function between telemetry V2 and Mixer Telemetry, for details refer to <https://istio.io/latest/news/releases/1.5.x/announcing-1.5/upgrade-notes/#feature-gaps-between-telemetry-v2-and-mixer-telemetry>.

10.6.2 CITM usage with Istio

As CITM/Nginx Ingress controller provides features which are not available in Istio ingress gateway(Envoy), users can still use CITM/Nginx Ingress controller with Istio deployment.

The following are the steps to make it work.

- Install Istio from Addons package using NCS Portal - please, refer to *Addson ISTIO*
- Create namespace and label it for auto-injection.

```
#kubectl create namespace citm
#kubectl label namespace citm istio-injection=enabled
```

- Install CITM from Addons package using NCS Portal - please, refer to *Addons CITM*

Use the following values.yaml for the deployment (save as /opt/bcmt/app-2.0/CITM/profile/values.yaml and proceed with CITM deployment using NCS Portal):

values.yaml

```
citm-ingress:
  controller:
    hostNetwork: false
    podAnnotations:
      traffic.sidecar.istio.io/includeInboundPorts: ''
    service:
      nodePorts:
        http: 32080
        https: 32443
      type: NodePort
  default404:
```

```
istio:
  enabled: true
```

In following `citm-values.yaml`, `hostNetwork` must be set to `false`, because Istio sidecar does not work on host network by design. In addition, `podAnnotations traffic.sidecar.istio.io/includeInboundPorts: ""` is required to avoid redirection of Nginx inbound ports.

Create Ingress resources for Services.

```
kubectl apply -f ingress.yaml -n tomcat
```



Note: In the Ingress resource, annotation `service-upstream` and `upstream-vhost` are required.

ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tomcat-ingress
  namespace: tomcat
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/service-upstream: "true"
    nginx.ingress.kubernetes.io/upstream-vhost: demo-casf-
tomcat.tomcat.svc.cluster.local
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: demo-tomcat
            port:
              number: 8080
```

Check the service can be reached.

```
curl https://your-node-ip:32443 -v -L -k
```

10.6.3 Components in NCS Istio

Table 31: NCS Istio Components

Components	Description	Version	Included In NCS Istio Package	Note
Gateway	Works as ingress/egress/... gateways	1.76	Y	
CoreDNS	For multicluster DNS, see https://istio.io/v1.7/docs/setup/install/multicluster/gateways/#setup-dns	1.76		Y
Istiod	Works with sidecar for traffic management https://istio.io/latest/docs/ops/deployment/architecture/#istiod	1.76	Y	
Kiali	Works an observability console for Istio with service mesh configuration capabilities https://kiali.io/	v1.22.1	Y	
Mixer	For policy and telemetry https://istio.io/faq/mixer/#why-mixer	1.76	Y	
Bosun	Added by NCS for housekeeping	1.76	Y	If you need to back up Istio CRs or install Istioctl, you need to enable Bosun. See bosun section in values.yaml for more information.

Table 31: NCS Istio Components (continued)

Components	Description	Version	Included In NCS Istio Package	Note
Prometheus	Addon. Used to store metrics generated by Istio components	cpro-prometheus 2.11.0	N	Use CSF CPRO
Grafana	Addon. Displays metrics collected by Prometheus	cpro-grafana 3.16.1	N	Use CSF CPRO Grafana
Jaeger	Addon. Used for tracing application inside the mesh	v1.21.0	N	Released as standalone NCS package
Cert Manager	Addon. Used for generating certificates for ingress gateway	v0.15.2	N	Included in main NCS package

10.6.3.1 Kiali

Kiali works with Istio to visualize the service mesh topology, features like circuit breakers or request rates.

Kiali is provided as sub chart of Istio chart. Kiali image is included in Istio image bundle. User can enable it in Istio chart values. Kiali requires Prometheus deployment to work. Jaeger as external services link intergrated into kiali dashboard is optional.

Deployment

- Set Kiali parameters in Istio chart values and deploy Istio.



Note: Kiali use version v1.14.0 for NCS Istio 20.12. Use '--set kiali.tag=v1.14.0' to sepcify kiali version for helm installation or update it in chart values.yaml.

```
jaegerCollectorURL: jaeger.istio-system:9411
kiali:
  enabled: true
  tag: v1.14.0
  dashboard:
    auth:
      strategy: login
```

```

    secretName: kiali
    prometheusURL: "" # this URL must be accessible from the Kiali pod.
    prometheusCustomMetricsURL: "http://istio-prometheus-cpro-server:80"

```

- If the strategy is login, you must create a secret with the same name as **secretName**. The following example use admin as username and password.

```

echo -n admin |base64
YWRtaW4=

```

```

apiVersion: v1
kind: Secret
metadata:
  name: kiali
  namespace: istio-system
  labels:
    app: kiali
  type: Opaque
data:
  username: YWRtaW4=
  passphrase: YWRtaW4=

```

- (Optional) Configuring ingress using Istio gateway to access kiali dashboard UI.

```

apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: kiali-gateway
  namespace: istio-system
spec:
  selector:
    app: istio-ingressgateway
  servers:
  - port:
      number: 30090
      name: http-kiali
      protocol: HTTP
    hosts:
    - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: kiali-virtual-service
  namespace: istio-system

```

```

spec:
  hosts:
  - "*"
  gateways:
  - kiali-gateway
  http:
  - route:
    - destination:
      host: kiali
      port:
        number: 20001

---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: kiali-destinationrule-istio-system
  namespace: istio-system
spec:
  host: kiali.istio-system.svc.cluster.local
  trafficPolicy:
    tls:
      mode: DISABLE

```

Cert-manager

cert-manager is a Kubernetes add-on to automate the management and issuance of TLS certificates from various issuing sources. Cert-manager can be used to obtain certificates using an arbitrary signing key pair stored in a Kubernetes Secret resource. This secret is used for securing istio gateways with HTTPS.

Cert-manage is installed by default in this release, unless it is purposely removed from the ncm_app_list. A cluster issuer will be created using the ca and ca-key in /etc/openssl/, its name is: ncms-ca-issuer.

10.6.4 Life Cycle Events

Install or Uninstall

You must use Helm to manage the Istio life cycle.

Install

1. Prepare Istio images and chart, which are included in NCS Istio package bundle.



Note: If you just want to try Istio and install it from artifactory without download, you can skip this step and continue. Download package bcmt-istio-20.12.0-centos7.tgz for platform base on RHEL7/centos7 or bcmt-istio-20.12.0-centos8.tgz for paltform base on RHEL8/centos8.

- a. Download the bundle from *Artifactory repo* and decompress the package bcmt-istio-20.12.0-centos7.tgz.

- b. Load Istio images into NCS private registry and Istio chart into NCS private chart repo with *ncs*.

```
ncs app-resource chart add --file_name app-2.0/charts/bcmt-istio-init-1.5.0.tgz
ncs app-resource chart add --file_name app-2.0/charts/bcmt-istio-1.5.0.tgz
for i in `ls app-2.0/images/*`;
do ncs app-resource image add --image_location local --image_path $i;
done
```

2. Create namespace istio-system.

```
kubectl create namespace istio-system
```

3. (Optional) Create CA secret for Citadel, if self-signed root cert is not applicable.

- Manually create a secret with the specific name **cacerts**.



Note: In the below example, all file and secret names must be used exactly as listed below, as they are hard-coded and used by Istio internally.

Table 32: File and secret names

File	Description
ca-cert.pem	Certificate used by Citadel to sign certificate.
ca-key.pem	Certificate private key used by Citadel to sign certificate.
root-cert.pem	Root certificate, which is used to sign ca-cert.pem or the same as ca-cert.pem.
cert-chain.pem	The certificate chains. Can be empty, if root-cert.pem is the same as ca-cert.pem.

```
kubectl create secret generic cacerts -n istio-system --from-file=ca-cert.pem \
--from-file=ca-key.pem \
--from-file=root-cert.pem \
--from-file=cert-chain.pem
```

- Provide ca cert via chart values.
 - Set *istio-discovery.polit.cacerts.enabled* to *true*.
 - Encode them with base64 and then update them into *values.yaml*.

```
cat ca-cert.pem |base64 -w 0

# Plugging in existing CA Certificates
cacerts:
  enabled: true
  # Certificate used by istiod to sign certificate.
  cacert: ""
  # Certificate private key used by istiod to sign certificate.
  cakey: ""
  # Root certificate, which is used to sign ca-cert.pem or the same
  as ca-cert.pem.
  rootcert: ""
  # The certificate chains. Can be empty, if root-cert.pem is the
  same as ca-cert.pem.
  certchain: ""
```

4. Install Istio Custom Resource Definition (CRD) with *istio-init* chart.

```
helm install -n istio-crd-install --namespace istio-system stable/bcm-istio-init
```

5. (Optional) Customize *values.yaml* in the chart. If you do customization then you must generate the tls certificate and map to secret, refer to the following example:

- a. openssl genrsa -out istio_gw.key 4096 && openssl req -x509 -new -days 7300 -nodes -key istio_gw.key -out istio_gw.crt -subj "/CN=csd.nokia.com/O=Nokia" -extensions san -config <(
echo "[req]"; echo "distinguished_name=req"; echo "x509_extensions=v3_ca";echo "[san]";echo "subjectAltName=DNS:*.nokia.com")
- b. kubectl create -n istio-system secret tls istio-ingressgateway-certs --key istio_gw.key --cert istio_gw.crt && kubectl create -n istio-system secret tls istio-egressgateway-certs --key istio_gw.key --cert istio_gw.crt



Note: The istio-ingressgateway and istio-egressgateway pods must be restarted.

6. (Optional) Enable istio-cni.

Set *istiocni.enabled* to *true*.

Set *istiocni.cniConfDir* to */etc/cni/net.d*.

7. (Optional) Enable third-party-jwt.

a. Check if cluster support third-party-jwt.

```
$ kubectl get --raw /api/v1 | jq '.resources[] | select(.name | index("serviceaccounts/token"))'
{
  "name": "serviceaccounts/token",
  "singularName": "",
  "namespaced": true,
  "group": "authentication.k8s.io",
  "version": "v1",
  "kind": "TokenRequest",
  "verbs": [
    "create"
  ]
}
```

b. To enable third-party-jwt, a few Kubernetes options need to be enabled before Istio deployment.



Note: The value of api-audiences must be the same as **global.sds.token.aud**. Default, istio-ca.

Enable Kubernetes options with KubeSetting operator.

```
kubectl apply -f KubeSetting.yaml  
kube-setting.yaml
```

```
apiVersion: setting.nokia.com/v1
kind: KubeSetting
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: sample01
spec:
  kube_api_config:
    - --api-audiences=istio-ca
    - --service-account-issuer=BCMT
    - --service-account-signing-key-file=/etc/kubernetes/ssl/
      serviceaccount-key.pem
    - --service-account-key-file=/etc/kubernetes/ssl/serviceaccount-
      key.pem
```

8. (Optional) Enable Mixer telemetry or telemetry V2.

- Defalut value (enable telemetry V2 without WASM)

```
telemetry:
  enabled: true
```

```

v1:
  # Set true to enable Mixer based telemetry
  enabled: false

v2:
  # For Null VM case now. If enabled, will set disableMixerHttpReports
  to true and not define mixerReportServer
  # This also enables metadata exchange.
  enabled: true
  metadataExchange:
    # Indicates whether to enable WebAssembly runtime for metadata
    exchange filter.
    wasmEnabled: false
  # Indicate if prometheus stats filter is enabled or not
  prometheus:
    enabled: true
    # Indicates whether to enable WebAssembly runtime for stats filter.
    wasmEnabled: false

```

- Enable Mixer telemetry set ***istio-discovery.telemetry.enabled*** to ***true***. set ***istio-discovery.telemetry.v1.enabled*** to ***true***. set ***istio-discovery.telemetry.v2.enabled*** to ***false***. set ***mixer-telemetry.enabled*** to ***true***.
- Enable telemetry V2 set ***istio-discovery.telemetry.enabled*** to ***true***. set ***istio-discovery.telemetry.v1.enabled*** to ***false***. set ***istio-discovery.telemetry.v2.enabled*** to ***true***. set ***mixer-telemetry.enabled*** to ***false***. To enable telemetry generation with the Wasm runtime: set ***istio-discovery.telemetry.v2.metadataExchange.wasmEnabled*** to ***true***. set ***istio-discovery.telemetry.v2.prometheus.wasmEnabled*** to ***true***.

9. Create Istio release with Helm.

```
helm install --namespace istio-system -n bcmt-istio stable/bcmt-istio
```

Uninstall

User can use following steps to uninstall Istio from NCS.

1. Delete release from Helm.

```
helm delete bcmt-istio --purge
```

2. Delete CRD installation release.



Note: This doesn't remove installed CRDs.

```
helm delete istio-crd-install --purge
```

3. Delete CRDs created by istio-crd-install.

```
kubectl delete crd $(kubectl get crd | grep istio.io | cut -d ' ' -f1)
```

4. Delete Istio namespace.

```
kubectl delete namespace istio-system
```

Upgrade or Rollback



Note:

- Refer to the NCS20FP1 upgrade/rollback guide if your current istio version is older than 1.6.
- Upgrade to the latest version from NCS20FP1(istio 1.6.8).
- As there have been some options changed in the new version, you may need to add/modify options with --set or -f along with the upgrade.

```
helm upgrade istio-crd-install stable/bcmt-istio-init  
helm upgrade bcmt-istio stable/bcmt-istio
```

After the pods are running normally, application pods with Istio sidecar injected need to be restarted to upgrade the sidecar.

- Rollback to the previous version.

```
helm rollback bcmt-istio 1
```

After the pods are running normally, application pods with Istio sidecar injected need to be restarted to roll back the sidecar.

Backup/Restore¶

1. To execute backup/restore manually, run **helm backup/restore**.

```
helm backup -t bcmt-istio -a none
```

```
# Backup name can be get from Backup History of kubectl describe brpolicy  
helm restore -t bcmt-istio -b 20191020081426_e02_LOCAL_istio-system_istio-  
bosun
```

2. Query current backup status by describing **BrPolicy istio-bosun**.

```
kubectl describe brpolicy istio-bosun -n istio-system
```

3. You can configure ***backupSchedule*** in Istio chart value to define periodical job for automatic backup of Istio configuration. Enable/disable the job by running *helm backup*.

```
helm backup -t bcmt-istio -a enable
helm backup -t bcmt-istio -a disable
```

10.6.5 Operations

Manage Istio Resources with CRD

Istio resources are defined in the Kubernetes Custom Resource Definition (CRD) block. You can create, modify and delete CRDs to control how Istio works. For example, to create a gateway in Istio, you can create a yaml file by running kubectl, see the following:

```
kubectl apply -f gateway.yaml
```

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway      # Istio default ingress gateway pod
  servers:
  - port:
    number: 80
    name: http
    protocol: HTTP
  hosts:
  - "/*"
```

Use **istioctl** to Diagnose Deployed Cluster

Istioctl can be installed by setting ***bosun.enabled*** and ***bosun.installIstioctl*** to *true* in section *bosun* of *bcmt-istio values.yaml*.

 **Note:** Istioctl should only be used for diagnosing purpose. Deployment or upgrade of istio with istioctl is not supported by bcmt-istio.

istioctl proxy-status			CDS	LDS
NAME	RDS	PILOT	VERSION	
EDS				
cinfo-867b88f7d4-6j4dp.cinfo			SYNCED	SYNCED
SYNCED	SYNCED	istiod-68f7d7cf5f-8qvr9	1.6.8	

cinfo1-0.cinfo			SYNCED	SYNCED
SYNCED	SYNCED	istiod-68f7d7cf5f-8qvr9	1.6.8	
istio-egressgateway-7d9c4fc965-drdgw.istio-system			SYNCED	SYNCED
SYNCED	NOT SENT	istiod-68f7d7cf5f-8qvr9	1.6.8	
istio-ingressgateway-5dff7954bb-mhclm.istio-system			SYNCED	SYNCED
SYNCED	NOT SENT	istiod-68f7d7cf5f-8qvr9	1.6.8	

Istio provides great examples in its documentation. Users are recommended to try it with the bookinfo example. Details about Bookinfo can be found at <https://istio.io/docs/examples/bookinfo/>.

HTTP Server and Tomcat Example

This is an example of how to adopt HTTP Server and Tomcat in Istio. The following steps must be done to make HTTP Server and Tomcat work with Istio.

 **Note:** The following repository URLs are provided as examples, the user should add their own.

1. Create additional RBAC/PSP for HTTP Server and Tomcat. In the following example, namespace tomcat is used and the namespace tomcat is labelled for auto injection by Istio.

```
# create namespace
kubectl create namespace tomcat
# label for auto injection
kubectl label namespace tomcat istio-injection=enabled
```

2. In the current Istio release, NET_ADMIN and NET_RAW privilege are required to do iptables operations in init-container. Thus, NET_ADMIN and NET_RAW must be listed in the PodSecurityPolicy.

```
kubectl apply -f tomcat-rbac.yaml -n tomcat
```

```
# tomcat-rbac.yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  name: tomcat-psp
spec:
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
```

```

rule: RunAsAny
allowedCapabilities:
- 'NET_ADMIN'
- 'NET_RAW'
requiredDropCapabilities:
- ALL
volumes:
- '*'
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: tomcat-role
  namespace: tomcat
rules:
- apiGroups:
  - extensions
resourceNames:
- tomcat-psp
resources:
- podsecuritypolicies
verbs:
- use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: tomcat-rolebinding
  namespace: tomcat
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: tomcat-role
subjects:
- kind: ServiceAccount
  name: default
  namespace: tomcat

```

3. When mTLS is enabled, the liveness/readiness probe provided by kubelet cannot work normally. Either alternative liveness/readiness probe or making permissive Policy is required.



Note: Making permissive Policy causes the service to not be protected by TLS any more, as the service will be accessible via both TLS and plain traffic. If pilot rewrite is enabled, this does not apply.

Here is how to create a permissive Policy.

```
kubectl apply -f tomcat-svc-permissive.yaml -n tomcat
```

```
# tomcat-svc-permissive.yaml
apiVersion: "security.istio.io/v1beta1"
kind: "PeerAuthentication"
metadata:
  name: "demo-casf-tomcat-permissive"
  namespace: tomcat
spec:
  - mtls:
    mode: PERMISSIVE
```

4. Create a gateway (optional) and a VirtualService for tomcat.

```
kubectl apply -f casf-tomcat-gateway.yaml -n tomcat
```

```
# casf-tomcat-gateway.yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: demo-gateway
spec:
  selector:
    istio: ingressgateway # Use Istio default ingress gateway pod
  servers:
  - port:
    number: 80
    name: http
    protocol: HTTP
    hosts:
    - "tomcat.io"
  ---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: demo
spec:
  hosts:
  - "tomcat.io"
  gateways:
  - demo-gateway
  http:
  - route:
```

```

- destination:
  host: demo-casf-tomcat
  port:
    number: 8080

```

5. After these tweaks, you can create HTTP Server and Tomcat release via Helm.

```

helm install --name demo --namespace tomcat \
--set secure=false \
--set admin.enabled=true,admin.password=password \
http://repo.lab.pl.alcatel-lucent.com/csf-helm-releases-local/
casf-tomcat-1.2.13.tgz

```

6. When tomcat pod is running normally, the you can reach the tomcat server with this command:

```

curl --resolve tomcat.io:31380:your-node-ip tomcat.io:31380 -v -H "Host:
tomcat.io"

```

7. Clean up.

```

helm delete demo --purge
kubectl delete -f casf-tomcat-gateway.yaml -n tomcat
kubectl delete -f tomcat-svc-permissive.yaml -n tomcat
kubectl delete -f tomcat-rbac.yaml -n tomcat
kubectl delete namespace tomcat

```

Tips

- **Exception for autoinjection** If autoinjection is enabled in namespace, Istio will inject sidecar into pods in the namespace automatically. For pods that are not supposed to be injected with sidecars by Istio, user can add annotation `sidecar.istio.io/inject: "false"` in the pod spec to disable the autoinjection.
- **Headless service** Previously Istio does not generate any route for headless service. Now Istio by default generates configuration for headless service with some limitation. Istio only generates route for port in Service and the target pod must be the same as the port.

```

spec:
  clusterIP: None
  ports:
  - name: tcp
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: cinfo2
  sessionAffinity: None

```

```
type: ClusterIP
status:
loadBalancer: {}
```

ServiceEntry also can be used to register pod IP and port in the Istio service registry, if the port is not defined in Service.

```
# ServiceEntry example
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: rmq
spec:
  hosts:
    - rmq-0.rabbitmq.default.svc.cluster.local
    - rmq-1.rabbitmq.default.svc.cluster.local
    - rmq-2.rabbitmq.default.svc.cluster.local
  ports:
    - number: 4369
      name: epmd
      protocol: TCP
    - number: 5672
      name: amqp
      protocol: TCP
    - number: 15672
      name: http
      protocol: HTTP
    - number: 25672
      name: inter-node
      protocol: TCP
```

- **Transparent proxy**

Istio sidecar doesn't support transparent proxy for source address, which means application cannot see the original IP address of the request. Instead, application only sees the request is from 127.0.0.1. In the TPROXY mode, only envoy/mixer see the original source address and destination address of downstream. Although envoy has support for this now, Istio has no plan or workaround to support it yet. For more information, see <https://github.com/istio/istio/issues/5679> and <https://github.com/istio/istio/issues/23369>

- **Stop Sidecar** The sidecar container can be stopped by sending request to 15020 now. This can help when pod termination is required, such as, test and job. To stop the sidecar, send a request in the pod where sidecar is injected with

```
curl http://127.0.0.1:15020/quitquitquit -X POST
OK
```

- **Service Port Naming** As Istio identifies service type based on the service port name, user must define the name correctly. Or else, services will be taken as tcp traffic by default. See **Named service ports** in <https://istio.io/docs/ops/prep/requirements/>.

CITM usage with Istio

As CITM/Nginx Ingress controller provides features which are not available in Istio ingress gateway (Envoy), users can still use CITM/Nginx Ingress controller with Istio deployment.

Here are the steps to make it work.

1. Create namespace and label it for auto-injection.

```
kubectl create namespace citm
kubectl label namespace citm istio-injection=enabled
```

2. Create CITM release with Helm.



Note: Make sure istio v1alpha1 API has been migrated to v1beta1 for citm chart.

```
helm install --namespace citm -n citm -f citm-values.yaml file path/csf-helm-releases-local/citm-ingress-1.18.3.tgz
```



Note: In the following citm-values.yaml, hostNetwork must be set to false, because Istio sidecar doesn't work on host network by design. And podAnnotations traffic.sidecar.istio.io/includeInboundPorts: " " is required to avoid redirection of Nginx inbound ports.

```
# citm-values.yaml
default404:
  istio:
    enabled: true
    version: 1.6
  controller:
    hostNetwork: false
    podAnnotations:
      traffic.sidecar.istio.io/includeInboundPorts: " "
  service:
    type: NodePort
    nodePorts:
      http: 32080
      https: 32443
```

3. Create Ingress resources for services.

```
kubectl apply -f ingress.yaml -n tomcat
```



Note: In the following Ingress resource, annotation **service-upstream** and **upsteam-vhost** are required

```
# ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: tomcat-ingress
  namespace: tomcat
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/service-upstream: "true"
    nginx.ingress.kubernetes.io/upstream-vhost: demo-casf-tomcat.tomcat.svc.cluster.local
spec:
  rules:
  - http:
    paths:
    - path: /
      backend:
        serviceName: demo-casf-tomcat
        servicePort: 8080
```

4. Check the service can be reached.

```
curl https://your-node-ip:32443 -v -L -k
```

Istio gateway with host interface

Istio gateway pod with daemonset can use hostNetwork or attach host interface via CNI (multus or danm).

1. Gateway with hostNetwork



Note: Make sure there is only one gateway pod with daemonset on one node to avoid port conflicts.

Set **istio-ingress.istio-ingressgateway.daemonSet.enabled** to **true** and **istio-ingress.istio-ingressgateway.daemonset.hostNetwork** to **true** in istio values.yaml.

2. Gateway with host interface



Note: This is an experimental feature and needs more test to verify different scenarios.



Note: If DANM is not the only DEFAULT cni in the cluster, for example there are some nodes with Multus and some nodes with DANM, or some nodes with Calico as default cni and some nodes with DANM.

a. Enable danm or multus.

Refer to: [Enable multus using a configuration file](#) or [Enable DANM using a configuration file](#)

b. Create NetworkAttachmentDefinition for multus.

```
# danmnet.yaml
---
apiVersion: danm.k8s.io/v1
kind: DanmNet
metadata:
  name: oam
  namespace: istio-system
spec:
  NetworkID: oam
  NetworkType: ipvlan
  Options:
    host_device: eth3
    cidr: 172.16.10.0/24
    allocation_pool:
      start: 172.16.10.110
      end: 172.16.10.115

# ipvlan-net1.yaml
---
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ipvlan-net1
spec:
  config: '{
    "cniVersion": "0.3.0",
    "name": "ipvlan-net1",
    "type": "ipvlan",
    "master": "eth1",
    "ipam": {
      "type": "whereabouts",
      "range": "172.18.110.10-172.18.110.50/24"
    }
  }'
```

```
kubectl apply -f danmnet.yaml
```

```
kubectl apply -f ipvlan-net1.yaml
```

For more information about multus, refer to: [MULTUS](#) and [DANM](#)

- c. (Optional) If the cloud provider is openstack, add allowed address pairs for CIDR.
- d. Set *istio-ingress.istio-ingressgateway.daemonSet.enabled* to *true* and *istio-ingress.istio-ingressgateway.daemonSet.hostNetwork* to *false* in istio values.yaml.
- e. Add cnis annotations *podAnnotations* for istio-ingressgateway in istio values.yaml.

```
# danm podAnnotations
podAnnotations:
  danm.k8s.io/interfaces: |
    [ {"network": "default", "ip": "dynamic"}, {"network": "oam", "ip": "dynamic"} ]
---

# multus podAnnotations
podAnnotations:
  k8s.v1.cni.cncf.io/networks: istio-system/ipvlan-net1
```

- f. (Optional) If you want to completely bypass Istio for a specific IP range, you can configure the Envoy sidecars to prevent them from intercepting external requests.

```
# istio egress capture whitelist
# example: includeIPRanges: "172.30.0.0/16,172.20.0.0/16"
# would only capture egress traffic on those two IP Ranges, all other
# outbound traffic would be allowed by the sidecar
includeIPRanges: "*"
excludeIPRanges: ""
excludeOutboundPorts: ""
```

- g. (Optional) If istiocni is enabled, change *istiocni.cniConfDir* to appropriate cni dir.

```
# Multus
cniConfDir: /etc/cni/net.d/
# Or DANM
cniConfDir: /etc/cni/danm/
```

- h. Install bcmt-istio via helm.

Istio Egress

Istio Egress introduction

There are three ways to send traffic from the application inside a cluster to an external service:

- Scenario 1: Through PassthroughCluster to external service

Traffic flow: app ----> PassthroughCluster ----> external service

- Scenario 2: Through the sidecar with service entry

Traffic flow: app ----> istio-proxy ----> external service

- Scenario 3: Egress gateway

Traffic flow: app ----> istio-proxy ----> egress gateway pod ----> external service

Istio egress gateway allows user to define exit points from the mesh that all outbound traffic flows through. Egress gateways also allow user to apply Istio features, for example, monitoring and route rules, to traffic exiting the mesh. Usually, NCS cluster Control/Worker nodes can not directly access the external network. Instead, Edge nodes provide interfaces to the external network as well as provide access control and load balancing as a gateway. Based on NCS cluster network topology, scheduling Istio egress gateway pod on edge nodes is recommended.

Usage

Enable Egress with Istio Egress Gateway

Here are steps to enable egress gateway in NCS.

1. Enable egress gateway deployment on edge node

Set **istio-egress.istio-egressgateway.enabled** to **true** in istio values.yaml to enable egress gateway.

Set **istio-egress.istio-egressgateway.nodeSelector** to **is_edge: "true"** in values.yaml to select edge node.

(Optional) Set **global.proxy.accessLogFile** to **"/dev/stdout"** to enable access log. This is for debugging to show traffic log.

2. Set **global.outboundTrafficPolicy.mode** to **REGISTRY_ONLY**.

Istio proxy blocks any traffic without service entry defined in istio.

3. Create ServiceEntry to allow traffic to external.

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: artifactory-repo
spec:
  hosts:
  - repo.lab.pl.alcatel-lucent.com
  ports:
  - number: 443
    name: https
    protocol: HTTPS
  - number: 80
    name: http-port
    protocol: HTTP
```

```
resolution: DNS
```

4. Create a Gateway and a destination rule for traffic directed to the egress gateway.



Note: Traffic does not pass through egress gateway without a destination rule.

```
---
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-egressgateway
spec:
  selector:
    istio: egressgateway
  servers:
  - port:
      number: 443
      name: tls
      protocol: TLS
    hosts:
    - repo.lab.pl.alcatel-lucent.com
    tls:
      mode: PASSTHROUGH
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: egressgateway-for-cinfo
spec:
  host: istio-egressgateway.istio-system.svc.cluster.local
  subsets:
  - name: cinfo
```

5. Create a virtual service to direct traffic from the sidecars to the egress gateway and from the egress gateway to the external service.

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: direct-repo-through-egress-gateway
spec:
  hosts:
  - repo.lab.pl.alcatel-lucent.com
  gateways:
  - mesh
```

```

- istio-egressgateway
tls:
- match:
  - gateways:
    - mesh
    port: 443
    sniHosts:
    - repo.lab.pl.alcatel-lucent.com
route:
- destination:
  host: istio-egressgateway.istio-system.svc.cluster.local
  subset: cinfo
  port:
    number: 443
- match:
  - gateways:
    - istio-egressgateway
    port: 443
    sniHosts:
    - repo.lab.pl.alcatel-lucent.com
route:
- destination:
  host: repo.lab.pl.alcatel-lucent.com
  port:
    number: 443
  weight: 100

```

6. Deploy cinfo pod and inject Istio proxy.

```

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cinfo
  labels:
    app: cinfo
spec:
  selector:
    matchLabels:
      app: cinfo
  replicas: 1
  template:
    metadata:
      labels:
        app: cinfo

```

```

spec:
  containers:
  - name: cinfo
    image: registry1-docker-io.repo.lab.pl.alcatel-lucent.com/
governmentpaas/curl-ssl:latest
    command: [ "/bin/sleep", "3650d" ]
  ports:
  - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: cinfo
spec:
  ports:
  - name: http-cinfo
    protocol: TCP
    port: 8000
    targetPort: 80
  selector:
    app: cinfo

```

7. Verify traffic through egress gateway pod

curl repo.lab.pl.alcatel-lucent.com from istio-proxy injected pod and check egress gateway logs:

```

$ date && curl https://repo.lab.pl.alcatel-lucent.com -v
Tue Mar 17 07:41:56 UTC 2020

$ kubectl logs istio-egressgateway-f7ccd7c9-9qwj2 -n istio-system
[2020-03-17T07:41:56.858Z] " - - - " 0 - " - " 797 3818 17 - " - "
" - " " - " "10.75.55.30:443" outbound|443|repo.lab.pl.alcatel-
lucent.com 192.168.1.52:48140 192.168.1.52:443 192.168.1.142:44990
repo.lab.pl.alcatel-lucent.com -

```

Additional security enhancement Cluster administrator can create a Kubernetes network policy to prevent bypassing of the egress gateway. Refer to [apply-kubernetes-network-policies](#)

Enable Egress without Egress Gateway

There are two ways to send traffic to external directly.

1. Send traffic to external service bypass sidecar. See *Scenario 1*.

- Configure Envoy proxy to allow access to any external service. This approach is default mode and the default value of **global.outboundTrafficPolicy.mode** is **ALLOW_ANY**. This mode means all outbound traffic to unknown destinations will be allowed.

- Completely bypass the Envoy proxy for a specific range of IPs. This approach bypasses the Istio sidecar proxy, giving your services direct access to any external server. You can configure them in `Istio values.yaml`.

```
# istio egress capture whitelist
# https://istio.io/docs/tasks/traffic-management/egress.html#calling-
external-services-directly
# example: includeIPRanges: "172.30.0.0/16,172.20.0.0/16"
# would only capture egress traffic on those two IP Ranges, all other
# outbound traffic would
# be allowed by the sidecar
includeIPRanges: "*"
excludeIPRanges: ""
excludeOutboundPorts: ""
```



Note: In the two approach, you will lose the capability of monitoring traffic to external services.

2. Send traffic to external service through sidecar with service entry. See *Scenario 2*.

- Set `global.outboundTrafficPolicy.mode` to `REGISTRY_ONLY`.
- Create a service entry to allow access to an external service.

```
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: cinfo
spec:
  hosts:
    - repo.lab.pl.alcatel-lucent.com
  ports:
    - number: 443
      name: https
      protocol: HTTPS
    - number: 80
      name: http-port
      protocol: HTTP
  resolution: DNS
```

10.6.5.1 Egress Gateways with TLS Origination

For istio egress TLS origination with SDS, refer to *Egress Gateways with TLS Origination (SDS)*.

For istio egress TLS origination with file mount, refer to *Egress Gateways with TLS Origination (File Mount)*.

10.6.6 Limitation

RBAC requirement to application



Note: If Istio CNI is enabled, NET_ADMIN and NET_RAW are not required and this limitation doesn't apply anymore.

As init container injected by Istio shares the service account with application pod, user needs to provision some additional psp/role/rolebinding to make init container work. From this release, Init container will drop ALL capabilities for defense in depth. In order to do iptables operations for non-root user , the init-container must satisfy the following psp:

- NET_RAW and NET_ADMIN are required. NET_ADMIN is used to inject iptables rule in the namespace of application and NET_RAW is used to save iptable updates.
- **allowPrivilegeEscalation** must be **true** for non-root user can escalate privilege to do iptables operations.

For TPROXY mode, it uses iptables TPROXY to redirect to Envoy. This mode preserves both the source and destination IP addresses and ports. So it requires privileged is ture and capability NET_ADMIN and root-user.

Here is an example to show how to provision the psp/role/rolebinding.



Note: seLinux, runAsUser, supplementalGroups, fsGroup, volumes listed here are only for completion purpose and not required by Istio. User can edit it to based on the actual requirements. And user should replace the namespace **default** and service account **default** with the ones used by application.

```
# istio-rolebinding.yaml
---
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  name: istio-psp
spec:
  seLinux:
    rule: RunAsAny
  runAsUser:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  volumes:
  - '*'
  allowPrivilegeEscalation: true
  allowedCapabilities:
```

```
- 'NET_ADMIN'  
- 'NET_RAW'  
requiredDropCapabilities:  
- ALL  
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: Role  
metadata:  
  name: istio-role  
  namespace: default  
rules:  
- apiGroups:  
  - extensions  
  resourceNames:  
  - istio-psp  
  resources:  
  - podsecuritypolicies  
  verbs:  
  - use  
---  
apiVersion: rbac.authorization.k8s.io/v1  
kind: RoleBinding  
metadata:  
  name: istio-rolebinding  
  namespace: default  
roleRef:  
  apiGroup: rbac.authorization.k8s.io  
  kind: Role  
  name: istio-role  
subjects:  
- kind: ServiceAccount  
  name: default  
  namespace: default
```

RunAsUser

Istio init-container and Envoy in istio-proxy run with specific user (runAsUser: 1337). User may need to update the RunAsUser in PSP to allow the specific user.

Pod DnsPolicy

Envoy and other components talk with each other via DNS. Pod DnsPolicy may affect the resolution and communication. ClusterFirst is recommended.

Liveness/readiness probe



Note: If `sidecarInjectorWebhook.rewriteAppHTTPProbe` is set to true, application PodSpec readiness/liveness probe is re-written by Istio, such that the probe request will be sent to Pilot agent. Pilot agent then redirects the request to application, and strips the response body only returning the response code. See <https://istio.io/docs/ops/setup/app-health-check/#probe-rewrite> for more information.

HTTP(S) liveness/readiness probe provided by kubelet cannot be completed when mTLS is enabled, as kubelet doesn't have the valid client certificate. Customized script is one way for liveness/readiness probe (such as curl). Making permissive Policy for the service could also be an option but will make service accessible in both TLS and plain.

Limited protocol support

The current release has a limit support in the protocol range. For example, there is no UDP support yet (This might not be an issue as Istio/Envoy evolves. See <https://github.com/istio/istio/issues/1430>)

Application requirement

Kubernetes pods and services must satisfy the following requirements: refer to link: [required-pod-capabilities](#)

IPv6 Only and IPv4/IPv6 dualstack

Upstream Istio support for IPv6 only cluster is now on Phase Alpha. As for the NCS Istio deployment, gateways can not be deployed with host network.

For IPv4/IPv6 dualstack, istio will only work with IPv4 traffic and IPv6 traffic will not be affected at all.

10.6.7 Plugins

Prometheus/Grafana

Istio metrics can be collected by Prometheus and virtualized in Grafana. CSF CPRO provides Prometheus/Grafana components. Prometheus pull metrics data from the endpoints exposed by Istio components and Grafana displays them.

Configure CPRO/Prometheus in cpro-values.yaml to discover endpoints exposed by Istio. Refer to [Appendix: Example for cpro-values.yaml files](#) on page 810.



Note: In the example yaml file, only `scrape_interval` and `scrape_configs` in section `prometheus.yml` are specific to Istio. The `scrape_interval` should be set to 15s for istio 1m scrape period. All the rest configuration is shown as example, and user can customize them based his/her needs.

Configure CPRO/Grafana Istio dashboards in grafana-values.yaml. Refer to [Appendix: Example for grafana-values.yaml files](#) on page 870.



Note: In the example yaml file, only dashboards are specific to Istio, which includes Istio customized dashboards to show Istio metrics. Please keep json part in each dashboard as it is shown here exactly.

Deploy CPRO/Prometheus/Grafana via helm

```
helm install --namespace istio-system -f cpro-values.yaml -n istio-prometheus
cpro-2.11.0.tgz
```

```
helm install --namespace istio-system -f grafana-values.yaml -n istio-grafana
cpro-grafana-3.16.1.tgz
```

For other detail configuration of Prometheus and Grafana, please refer to CSF [CPRO](#).

Jaeger

Jaeger is a distributed tracing system, which can be used for monitoring microservices-based distributed systems.

Jaeger can provide:

- Distributed context propagation
- Distributed transaction monitoring
- Root cause analysis
- Service dependency analysis
- Performance / latency optimization

Jaeger can work with Istio to trace application traffic. Envoy proxy generates traffic information and sends to Jaeger collector. User can view traffic trace via Jaeger portal. The sampling rate is controlled by the traceSampling in Istio chart values.

Jaeger is provided as an independent package in NCS. The current Jaeger release in NCS is an all-in-one version, which stores all tracing data in memory and will lose data if the pod is restarted. For more information regarding Jaeger installation, please refer to the [Jaeger](#) on page 225 guide.

Jaeger Deployment

The following example assumes istio namespace istio-system is used.

 **Note:** Download package bcmt-jaeger-20.03.0-centos7.tgz for platform base on RHEL7/centos7 or bcmt-jaeger-20.03.0-centos8.tgz for paltform base on RHEL8/centos8.

1. Download the bundle bcmt-jaeger-20.03.0-centos7.tgz from [Artifactory repo](#), decompress the package bcmt-jaeger-20.03.0-centos7.tgz and load the chart and images into the NCS registry and chart repo.

```
ncs app-resource image add --image_location local --image_path app-2.0/
images/bcmt-jaeger-all-in-one-v1.17.0.tar
ncs app-resource chart add --file_name app-2.0/charts/bcmt-
jaeger-1.5.0.tgz
```

2. Configure *jaegerCollectorURL* with the Jaeger service URL in Istio chart values.

```
jaegerCollectorURL: jaeger.istio-system:9411
```

3. Create Jaeger release with helm

```
helm install --namespace istio-system -n bcmt-jaeger stable/bcmt-jaeger
```

You can skip step 1 and install it from Artifactory directly.

```
# platform RHEL7/Centos7
helm install --namespace istio-system -n bcmt-jaeger https://
repo.lab.pl.alcatel-lucent.com/csf-generic-delivered/BCMT/
jaeger-charts/bcmt-jaeger-1.5.0.tgz --set global.hub=csf-docker-
delivered.repo.lab.pl.alcatel-lucent.com/bcmt --set global.tag=v1.17.0-
centos7
# platform RHEL8/Centos8
helm install --namespace istio-system -n bcmt-jaeger https://
repo.lab.pl.alcatel-lucent.com/csf-generic-delivered/BCMT/
jaeger-charts/bcmt-jaeger-1.5.0.tgz --set global.hub=csf-docker-
delivered.repo.lab.pl.alcatel-lucent.com/bcmt --set global.tag=v1.17.0-
centos8
```

4. (Optional) If the gateway is disabled in jaeger chart value, you can configure ingress using Istio gateway to access Jaeger dashboard UI.

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: jaeger-gateway-default
  namespace: default
spec:
  selector:
    istio: ingressgateway
  servers:
    - port:
        number: 15033
        name: http-jaeger
        protocol: HTTP
        hosts:
          - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: jaeger-virtual-service-default
```

```
namespace: default
spec:
hosts:
- "*"
gateways:
- jaeger-gateway-default
http:
- match:
  - port: 15033
route:
- destination:
  host: jaeger-query.default.svc.cluster.local
  port:
    number: 16686
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
name: jaeger-destinationrule-default
namespace: default
spec:
host: jaeger-query.default.svc.cluster.local
trafficPolicy:
  tls:
    mode: DISABLE
```

Uninstall Jaeger

Use the following command to uninstall jaeger from bcmt cluster.

```
helm delete bcmt-jaeger --purge
```

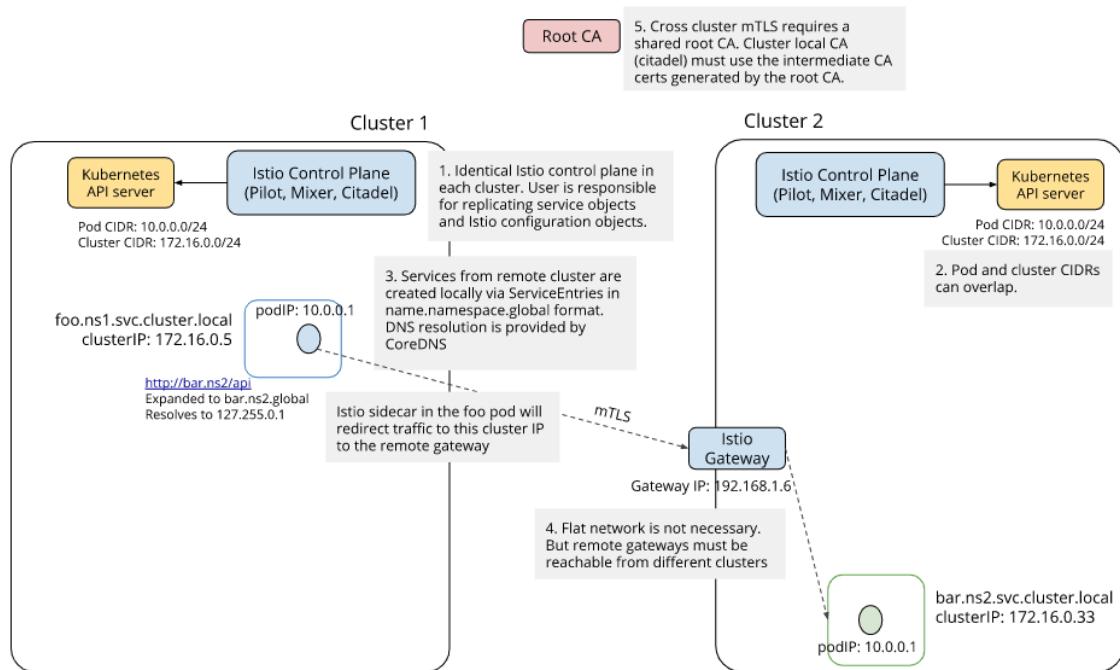
Please refer to the [Jaeger](#) on page 225 guide for more information.

10.6.8 Multicloud deployment

Follow this guide to install an Istio multicloud deployment on both **cluster1** and **cluster2**, each with its own replicated control plane, using gateways to connect services across clusters. In this configuration each cluster has its own Istio control plane installation, each managing its own endpoints. All of the clusters are under a shared administrative control for the purposes of policy enforcement and security.

As the following figure shows, a single Istio service mesh across the clusters is achieved by replicating shared services and namespaces and using a common root CA in all of the clusters. Cross-cluster communication occurs over the Istio gateways of the respective clusters.

Figure 9: Istio Multicloud



10.6.8.1 Prerequisites

- The IP address of the `istio-ingressgateway` service in each cluster must be accessible from istio egress gateway of other clusters.
- A Root CA. Cross cluster communication requires mutual TLS connection between services. To enable mutual TLS communication across clusters, each cluster's Istio CA will be configured with intermediate CA credentials generated by a shared root CA.

10.6.8.2 Deploy the Istio control plane in each cluster

- If self-signed root CA is used, the same self-signed root CA cert file must be used by all the clusters. If intermediate CA is used, all the generated intermediate CA certificates for each cluster's Istio CA must be signed by the same root CA. The shared root CA enables mutual TLS communication across different clusters.
- Run the following commands in every cluster to deploy an identical Istio control plane configuration in all of them.
 - Create a Kubernetes secret for your generated CA certificates using a command similar to the following (don't change the file name):

```
kubectl create namespace istio-system
kubectl create secret generic cacerts -n istio-system \
--from-file=ca-cert.pem \
--from-file=ca-key.pem \
```

```
--from-file=root-cert.pem \
--from-file=cert-chain.pem
```

- Set the following multicluster related parameters in Istio chart values and deploy NCS Istio.

```
istio-egress:
  istio-egressgateway:
    enabled: true
    env:
      # Needed to route traffic via egress gateway if desired. the name
      "external" is referenced in service
    entry
      ISTIO_META_REQUESTED_NETWORK_VIEW: "external"

  global:
    multiCluster:
      enabled: true
      globalDomainSuffix: global
      includeEnvoyFilter: true
      controlPlaneSecurityEnabled: true
```

- Setup the DNS for domain name *.global

To access the services in remote clusters, we need to create fake service name in the format <name>.<namespace>.global. Coredns server deployed by Istio provides DNS resolution for these services. To make NCS forward these DNS requests to the istio coredns server, use following NCS command to configure .global on each istio cluster.

For example:

```
ncs service dns update --external_dns "YOUR_DEFAULT_DNS_SERVER_LIST,global/
$(kubectl get svc -n istio-system
istiocoredns -o jsonpath={.spec.clusterIP})"
```

10.6.8.3 Configure application services

To access a service in the remote cluster, a servive entry is required in the local cluster. The service name used in the service entry should be of the form <name>.<namespace>.global where name and namespace correspond to the service's name and namespace in the remote cluster respectively.

To demonstrate cross cluster access, configure the sleep service running in **cluster1** to call the httpbin service running in a second **cluster2**.

10.6.8.3.1 Deploy the httpbin service in **cluster2**

```
# httpbin.yaml
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  name: httpbin
---
apiVersion: v1
kind: Service
metadata:
  name: httpbin
  labels:
    app: httpbin
spec:
  ports:
  - name: http
    port: 8000
    targetPort: 80
  selector:
    app: httpbin
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpbin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpbin
      version: v1
  template:
    metadata:
      labels:
        app: httpbin
        version: v1
    spec:
      serviceAccountName: httpbin
      containers:
      - image: registry1-docker-io.repo.lab.pl.alcatel-lucent.com/
kennethreitz/httpbin
        imagePullPolicy: IfNotPresent
        name: httpbin
        ports:
        - containerPort: 80
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

```

metadata:
  name: httpbin-role
  namespace: bar
rules:
- apiGroups:
  - extensions
  resourceNames:
  - privileged
  resources:
  - podsecuritypolicies
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: istio-rolebinding
  namespace: bar
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: httpbin-role
subjects:
- kind: ServiceAccount
  name: httpbin
  namespace: bar

```

```

kubectl create namespace bar
kubectl label namespace bar istio-injection=enabled
kubectl apply -n bar -f httpbin.yaml

```

10.6.8.3.2 Get the information of *cluster2* gateway

The *cluster2* gateway address should be an IP address on the edge nodes through which the `istio-ingressgateway` can be accessed.

It is necessary to get the service entry endpoint port from the corresponding nodePort in *cluster2*.

```

kubectl get svc -n istio-system istio-ingressgateway -o=jsonpath=".spec.ports[?(@.port==15443)].nodePort"

```

10.6.8.3.3 Create a service entry for the httpbin service in cluster1

 **Note:** The traffic with the following configuration will be sent out from the pods directly to remote ingress gateway without passing the egress gateway in local cluster.

To allow **cluster1** to access httpbin in **cluster2**, we need to create a service entry for it. The host name of the service entry should be of the form <name>.<namespace>.global where name and namespace correspond to the remote service's name and namespace respectively.

For DNS resolution for services under the *.global domain, assign these services an IP address.



Note: Each service (in the .global DNS domain) must have a unique IP within the cluster.

It is suggested to use IPs from the class E addresses range 240.0.0.0/4. Application traffic for these IPs will be captured by the sidecar and routed to the appropriate remote service.



Note: Multicast addresses (224.0.0.0 ~ 239.255.255.255) should not be used because there is no route to them by default. Loopback addresses (127.0.0.0/8) should also not be used because traffic sent to them may be redirected to the sidecar inbound listener.

Specify the endpoints parameters as what was received in cluster2.

```
# ServiceEntry.yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: httpbin-bar
spec:
  hosts:
    # must be of form name.namespace.global
    - httpbin.bar.global
  # Treat remote cluster services as part of the service mesh
  # as all clusters in the service mesh share the same root of trust.
  location: MESH_INTERNAL
  ports:
    - name: http1
      number: 8000
      protocol: http
  resolution: DNS
  addresses:
    # the IP address to which httpbin.bar.global will resolve to
    # must be unique for each remote service, within a given cluster.
    # This address need not be routable. Traffic for this IP will be captured
    # by the sidecar and routed appropriately.
    - 240.0.0.2
  endpoints:
    # This is the routable address of the ingress gateway in cluster2 that
    # sits in front of httpbin.bar service. Traffic from the sidecar will be
    # routed to this address.
    - address: ${CLUSTER2_GW_ADDR} # Should be specified as a cluster2 gateway
      address
  ports:
```

```
http1: xxxx # Should be specified as the nodeport of an ingress gateway, which is fetched in the last step.
```

```
kubectl create namespace foo
kubectl label namespace foo istio-injection=enabled
kubectl apply -f ServiceEntry.yaml -n foo
```

10.6.8.3.4 Verify that httpbin is accessible from the cluster1

- Deploy the sleep service in *cluster1*.

```
# sleep.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sleep
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sleep
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sleep
  template:
    metadata:
      labels:
        app: sleep
  spec:
    serviceAccountName: sleep
    containers:
      - name: sleep
        image: registry1-docker-io.repo.lab.pl.alcatel-lucent.com/
governmentpaas/curl-ssl
        command: ["/bin/sleep", "3650d"]
        imagePullPolicy: IfNotPresent
    volumeMounts:
      - mountPath: /etc/sleep/tls
        name: secret-volume
    volumes:
      - name: secret-volume
        secret:
          secretName: sleep-secret
```

```

    optional: true
    ---

kubectl apply -n foo -f sleep.yaml

• Verify that httpbin is accessible from the sleep service.

kubectl exec $(kubectl get -n foo pod -l app=sleep -o
jsonpath='{.items..metadata.name}') -n foo -c sleep -- curl -I
httpbin.bar.global:8000/headers

```

10.6.8.3.5 Send remote traffic via an egress gateway

If you want to route traffic from **cluster1** via a dedicated egress gateway, instead of directly from the sidecars, use the following service entry for httpbin.bar instead of the one in the previous section.

 **Note:** The egress gateway used in this configuration cannot also be used for other, non inter-cluster, egress traffic.

- Get the **cluster1** egress gateway cluster IP:

```
kubectl get svc --selector=app=istio-egressgateway \
-n istio-system -o yaml -o jsonpath='{.items[0].spec.clusterIP}'
```

- Apply the httpbin-bar service entry:

```

# ServiceEntry.yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: httpbin-bar
spec:
  hosts:
    # must be of form name.namespace.global
    - httpbin.bar.global
  location: MESH_INTERNAL
  ports:
    - name: http1
      number: 8000
      protocol: http
  resolution: STATIC
  addresses:
    - 240.0.0.2
  endpoints:
    - address: ${CLUSTER2_GW_ADDR} # Should be specified as a cluster2
      gateway address
    network: external

```

```

  ports:
    http1: xxxx # Should be specified as the nodeport of an ingress
    gateway.
    - address: ${CLUSTER1_EGW_CLUSTER_IP} # Should be specified as the
      egress gateway address, which is fetched in the last step.
  ports:
    http1: 15443 # Do not change this port value

```

If \$CLUSTER2_GW_ADDR is a hostname, the following service entry can be used for httpbin.bar.

```

# ServiceEntry.yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: httpbin-bar
spec:
  hosts:
    # must be of form name.namespace.global
    - httpbin.bar.global
  location: MESH_INTERNAL
  ports:
    - name: http1
      number: 8000
      protocol: http
  resolution: DNS
  addresses:
    - 240.0.0.2
  endpoints:
    - address: ${CLUSTER2_GW_ADDR} # Should be specified as a cluster2
      gateway hostname
      network: external
  ports:
    http1: xxxx # Should be specified as the nodeport of an ingress
    gateway.
    - address: istio-egressgateway.istio-system.svc.cluster.local
  ports:
    http1: 15443 # Do not change this port value

```

10.7 Jaeger

Jaeger is a distributed tracing system, which can be used for monitoring microservices-based distributed systems.

Jaeger can provide:

- Distributed context propagation
- Distributed transaction monitoring
- Root cause analysis
- Service dependency analysis
- Performance / latency optimization

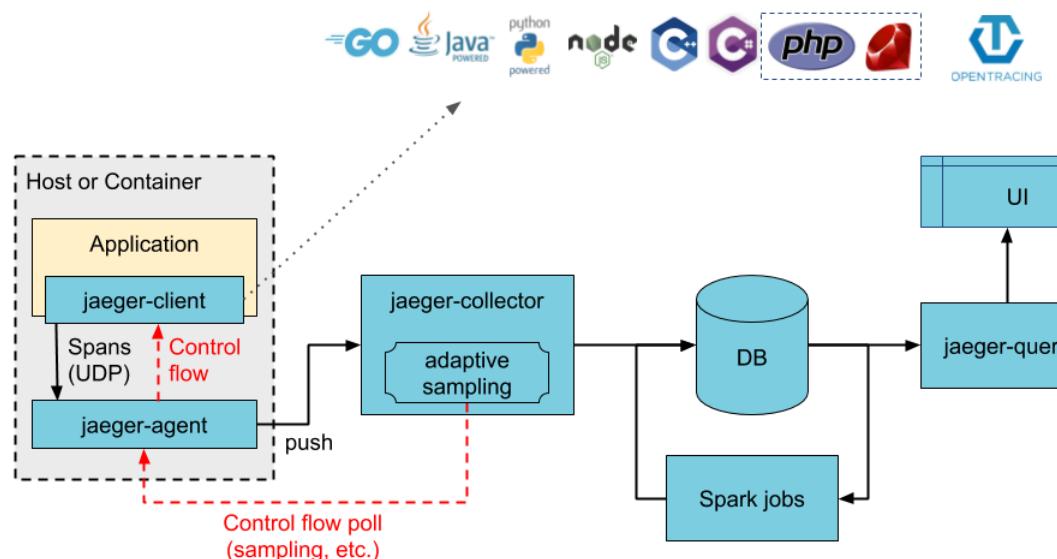
Jaeger is provided as an independent package in NCS. The current Jaeger release in NCS can support all-in-one and production deployment. All-in-one deployment stores all the tracing data in memory and loses the data if the pod is restarted. Production deployment stores all the tracing data in DB (Cassandra) which supports data persistence.

Jaeger clients are language specific implementations of the OpenTracing API. They can be used to instrument applications for distributed tracing either manually or with a variety of existing open source frameworks, such as Flask, Dropwizard, gRPC, and many more, that are already integrated with OpenTracing. For more information on usages of client libraries of different languages, please refer to [client-features](#).

Jaeger can also work with Istio to trace application traffic. Envoy proxy generates traffic information and sends to Jaeger collector. User can view traffic trace via Jaeger portal. The sampling rate is controlled by the **traceSampling** in Istio chart values.

The following graph shows Jaeger direct-to-storage architecture.

Figure 10: Jaeger direct-to-storage architecture



10.7.1 Jaeger in NCS

Table 33: Components in NCS Jaeger

Components	Description	Version	Included In BCMT Jaeger Package	Note
client libraries	Jaeger clients are language specific implementations of the Open Tracing API	1.21.0	Y	
agent	Network daemon that listens for spans sent over UDP	1.21.0	Y	
collector	receives traces from Jaeger agents and store them to DB	1.21.0	Y	
query	retrieves traces from storage and hosts a UI to display them	1.21.0	Y	
spark-dependencies	extract the service graph data from traces	1.21.0	Y	
Apache-cassandra	CCAS - Apache Cassandra based	CCAS - Cassandra 20.10	N	provided by CSF-CCAS

10.7.2 Life Cycle Events

10.7.3 Install/Uninstall

 **Note:** Download the NCS20FP2-BASE-ADDONS.zip package from the Nokia support portal <https://customer.nokia.com/support/s/>.

From this release, Jaeger chart will support all-in-one or production installation.

10.7.3.1 Install Jaeger

Prerequisite Apache Cassandra (CCAS) is deployed. For more instructions on Cassandra, please refer to *Apache Cassandra Distribution*. For details on application life cycle, refer to the *Helm application life cycle* event guide. Get Cassandra parameters required by Jaeger, such as username, password, service URL etc. Image (ccas_apache:3.11.9.663.el8) from Apache Cassandra is available in NCS registry or other accessible repository.

1. Prepare Jaeger images and chart, which are included in the bcmt-addons-20.12.0.tgz bundle of the NCS20FP2-BASE-ADDONS.zip package.
 - a. Download the NCS20FP2-BASE-ADDONS.zip package from the Nokia support portal <https://customer.nokia.com/support/s/> and unzip the package. Then untar the bcmt-addons-20.12.0.tgz which contains the Jaeger images and chart.
 - b. Load Jaeger images into NCS private registry and Jaeger chart into NCS private chart repo with NCS.

```
ncs app-resource chart add --file_name app-2.0/charts/bcmt-
jaeger-1.5.0.tgz
for i in `ls app-2.0/images/*`;
do ncs app-resource image add --image_location local --image_path $i;
done
```

2. Configure storage section in values.yaml. Depending on whether TLS is enabled or not on Cassandra, choose one of the following steps.

- Apache-cassandra TLS disabled:

```
storage:
# currently supported cassandra
type: cassandra
cassandra:
  host: ccas-apa-ccas-apache.ccas.svc.cluster.local
  port: 9042
  tls:
    enabled: false
    secretName: cassandra-tls-secret
    user: suadmindcass
    usePassword: true
    password: suadmindcass
```

```
keyspace: jaeger
datacenter: MyCenter
```

- Apache-cassandra TLS enabled:

 **Note:** spark-dependencies can not work with TLS. Thus, Spark should be disabled. For more information, refer to [issue-294](#).

```
jaeger-production:
  enabled: true
  storage:
    # currently supported cassandra
    type: cassandra
    cassandra:
      host: ccas-apa-ccas-apache.ccas.svc.cluster.local
      port: 9042
      tls:
        enabled: true
        secretName: cassandra-tls-secret
      user: suadmincass
      usePassword: true
      password: suadmincass
      keyspace: jaeger
      datacenter: MyCenter

  spark:
    enabled: false
    certManager:
      enabled: true
```

3. Update repo url with registry where the image ccas_apache:3.11.9.663.el8 is available.



Note: The following registry is provided as examples.

```
schema:
  # The schema image will reuse Apache Cassandra
  hub: csf-docker-delivered.[appropriate repository].com
  image: ccas_apache
  tag: 3.11.9.663.el8
```

4. Install bcmt-jaeger with helm.

```
helm install --namespace jaeger -n bcmt-jaeger stable/bcmt-jaeger
```



Note: As Jaeger is able to collect sensitive or privacy related data and expose access to this data, user access to Cassandra and Jaeger needs to be security hardened and permitted to authorized users only.

10.7.3.1.1 Install all-in-one



Note: All-in-one deployment is not recommended for production, as tracing data is saved in memory and will be lost if pod restart happens.

1. Download the bundle *NCS20FP2-BASE-ADDONS.zip* from the Nokia SW delivery service, in the Nokia Support Portal, and untar the package *bcmt-addons-20.12.0.tgz* and load the chart and images into NCS registry and chart repo.

```
ncs app-resource image add --image_location local --image_path
app-2.0/images/bcmt-jaeger-all-in-one-v1.21.0.tar
ncs app-resource chart add --file_name app-2.0/charts/bcmt-
jaeger-1.5.0.tgz
```

2. Set *jaeger-allinone* to *true* Set *jaeger-production* to *false*

```
jaeger-all-in-one:
  enabled: true
jaeger-production:
  enabled: false
```

3. Create Jaeger release with helm

```
helm install --namespace istio-system -n bcmt-jaeger stable/bcmt-
jaeger
```

10.7.3.1.1 Uninstall

User can follow this step to uninstall Jaeger from NCS.

1. Delete bcmt-jaeger release from helm.

```
helm del bcmt-jaeger --purge
```

10.7.4 Upgrade/Rollback

Upgrade from all-in-one to non all-in-one deployment is not supported.

10.7.5 Operations

Jaeger can be used with Istio or client libraries integrated in application.

10.7.5.1 With Istio

1. Set tracer.jaeger.address to jaeger-service:port in istio values.yaml before Istio installation.

```
tracer:  
  jaeger:  
    # External Jaeger collector service URL. Envoy will be configured with  
    # this URL to send tracing data.  
    # This is also required by Kiali.  
    # address: jaeger.istio-system:9411  
    address: bcmt-jaeger-jaeger-production-collector.jaeger.svc:9411
```

Set the istio-discovery.pilot.traceSampling before Istio installation.

```
istio-discovery:  
  pilot:  
    traceSampling: 100
```

2. (Optional) The user can configure ingress using Istio gateway to access Jaeger dashboard UI.

```
apiVersion: networking.istio.io/v1alpha3  
kind: Gateway  
metadata:  
name: jaeger-gateway-default  
namespace: default  
spec:  
  selector:  
    istio: ingressgateway  
  servers:  
    - port:  
        number: 15033  
        name: http-jaeger  
        protocol: HTTP  
      hosts:  
        - "*"---  
apiVersion: networking.istio.io/v1alpha3  
kind: VirtualService  
metadata:  
name: jaeger-virtual-service-default  
namespace: default  
spec:  
  hosts:  
    - "*"  
  gateways:
```

```

- jaeger-gateway-default
http:
- match:
- port: 15033
route:
- destination:
  host: jaeger-query.default.svc.cluster.local
  port:
  number: 16686
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
name: jaeger-destinationrule-default
namespace: default
spec:
host: jaeger-query.default.svc.cluster.local
trafficPolicy:
tls:
mode: DISABLE

```

10.7.5.1.1 With Jaeger Client Libraries

Jaeger client libraries can be integrated with application based on requirement. In the following example, two servers (test-server-1 and test-server-2) developed on python flask framework are started. On user requests, the test-server-1 will create span and propagate the span via request to test-server-2; and the test-server-2 will use this span as parent to create child spans.

Copy `jaeger-client-test1.yaml` and `jaeger-client-test2.yaml` to NCS cluster and then start services:

 **Note:** Update `JAAGER_AGENT` with proper jaeger agent service exposed by Jaeger deployment. In the following example, `JAAGER_AGENT=jaeger-agent.istio-system.svc`.

```

kubectl apply -f jaeger-client-test1.yaml
kubectl apply -f jaeger-client-test2.yaml

```

`jaeger-client-test1.yaml`

 **Note:** The following repository URLs are provided as examples, the user should add their own.

```

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:

```

```
name: jaeger-client-test1-rolebinding
namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:privileged
subjects:
- kind: ServiceAccount
  name: default
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jaeger-client-test1
  namespace: default
  labels:
    app: jaeger-client-test1
spec:
  selector:
    matchLabels:
      app: jaeger-client-test1
  replicas: 1
  template:
    metadata:
      labels:
        app: jaeger-client-test1
    spec:
      containers:
      - name: jaeger-client-test1
        image: csf-docker-candidates.[appropriate repository].com/bcmt/bcmt-jaeger-client:0.1.0
        imagePullPolicy: Always
        ports:
        - containerPort: 5006
          command: ["/bin/bash", "-c", "FLASK_APP=/opt/jaeger-client-test1.py JAEGER_AGENT=jaeger-agent.istio-system.svc flask run --host=0.0.0.0 --port=5006"]
apiVersion: v1
kind: Service
metadata:
  name: jaeger-client-test1
  namespace: default
spec:
  ports:
  - name: http-jaeger-client-test1
```

```
protocol: TCP
port: 5006
targetPort: 5006
selector:
  app: jaeger-client-test1

jaeger-client-test2.yaml

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: jaeger-client-test2-rolebinding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:privileged
subjects:
- kind: ServiceAccount
  name: default
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jaeger-client-test2
  namespace: default
  labels:
    app: jaeger-client-test2
spec:
  selector:
    matchLabels:
      app: jaeger-client-test2
  replicas: 1
  template:
    metadata:
      labels:
        app: jaeger-client-test2
    spec:
      containers:
        - name: jaeger-client-test2
          image: csf-docker-candidates.[appropriate repository]/bcmt/bcmt-jaeger-client:0.1.0
          imagePullPolicy: Always
      ports:
```

```

    - containerPort: 5004
      command: ["/bin/bash", "-c", "FLASK_APP=/opt/jaeger-client-test2.py JAEGER_AGENT=jaeger-agent.istio-system.svc flask run --host=0.0.0.0 --port=5004"]
    ---
  apiVersion: v1
  kind: Service
  metadata:
    name: jaeger-client-test2
    namespace: default
  spec:
    ports:
      - name: http-jaege-client-test2
        protocol: TCP
        port: 5004
        targetPort: 5004
    selector:
      app: jaeger-client-test2

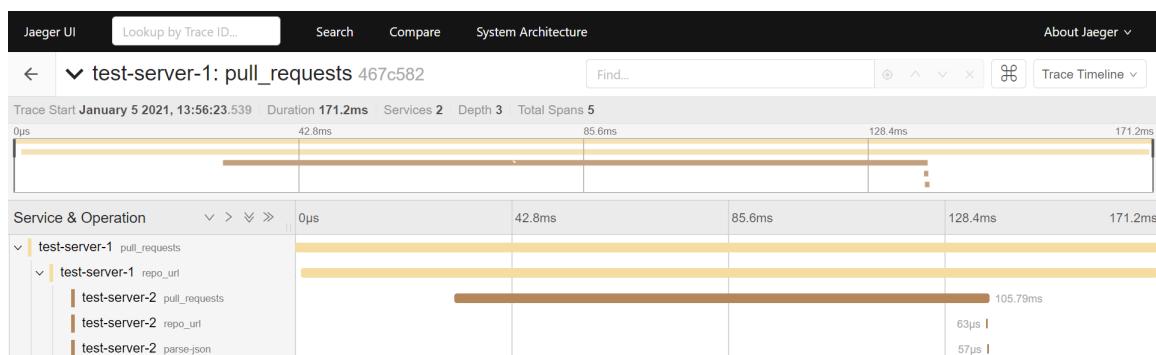
```

After the two services are running, the user can send traffic to jaeger-client-test1 and then find the trace and spans on Jaeger portal.

```
curl jaeger-client-test1.default.svc:5006
```

The result should be like:

Figure 11: Trace and spans on Jaeger portal



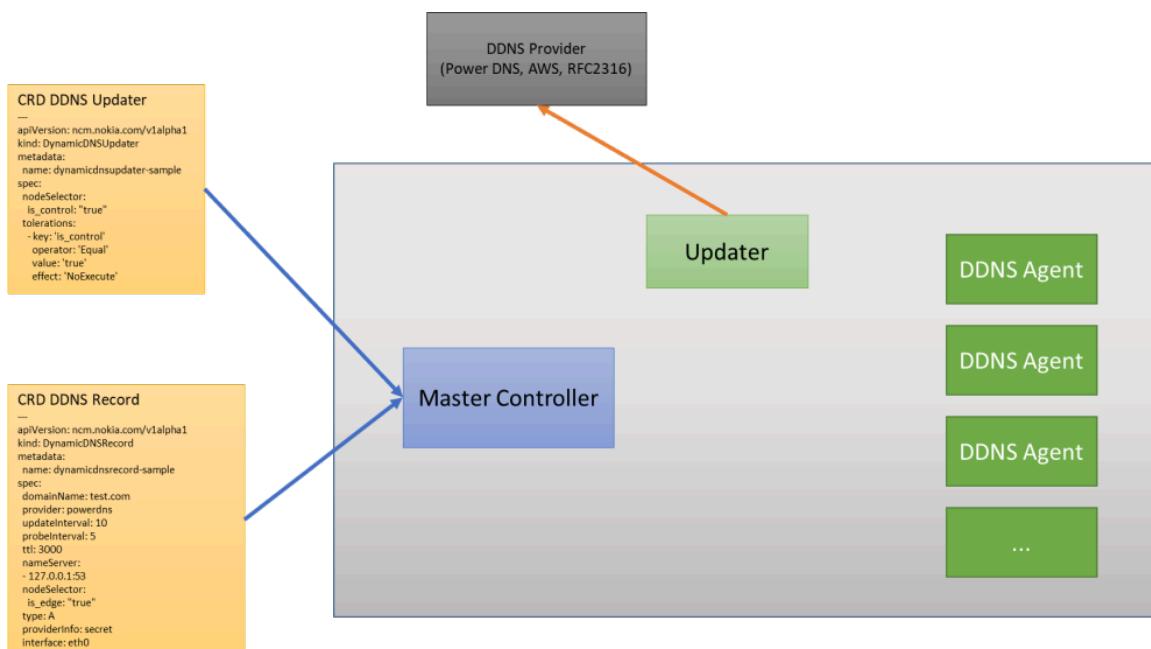
10.8 Dynamic DNS

Dynamic DNS (DDNS or DynDNS) is a method of automatically updating a name server in the Domain Name System (DNS) with the active DDNS configuration of its configured hostnames, addresses or other information. NCS supports DDNS with Kubernetes CustomResourceDefinition (CRD) and a customized controller. Users can dynamically update domain names via NCS CRD. The NCS DDNS controller will automatically discover the addresses of desired interfaces on specific nodes and register them in the

target DNS service provider. The controller periodically queries the service provider to make sure the registration status is correct.

The NCS DDNS controller consists of three parts: a master controller, an updater, and an agent.

- The master controller running on a Control node is responsible for custom resource (CR) scheduling and management of updater.
- The updater performs an automatic update action on behalf of the user when there is a change of addresses on the specific nodes. It also checks the registration status against the provider periodically. The updater service is created by the user or the master controller.
- The agent monitors interfaces on the nodes. When there are events such as node scaling-in/scaling-out, interfaces up/down on a node, the agent updates the status in DynamicDNSRecord CR.



Currently the providers supported by the controller are AWS and most other providers that are compatible with PowerDNS Restful API and RFC2316. DNS record types supported by the controller are A, AAAA, and CNAME.

10.8.1 Usage

1. (Optional) Create an updater.

By default, an updater named `default` is created. Updater creation is required if the default one cannot meet user's requirement, e.g. user requires the updater to be located on some node with external connection to the DDNS service provider.

Example: "dynamicdnsupdater-sample.yaml"

```
apiVersion: ncm.nokia.com/v1alpha1
```

```

kind: DynamicDNSUpdater
metadata:
  name: dynamicdnsupdate-sample
spec:
  nodeSelector:           # Specify where the updater is located
    is_control: "true"
  tolerations:           # Specify tolerations the updater can
  - key: 'is_control'    # tolerate
    operator: 'Equal'
    value: 'true'
    effect: 'NoExecute'

```

Apply the updater:

```
kubectl apply -f dynamicdnsupdate-sample.yaml
```

2. Create a secret with provider information.

Different providers require different information to be provided. For a list of information in the secret for different providers, please see [Provider Information in Secret](#) on page 238.



Note: All values in the secret must be base64 encrypted. Use -n to avoid echo appends newline in the end: echo -n URL | base64 -w 0.

Example for provider PowerDNS: "secret-sample.yaml"

```

apiVersion: v1
kind: Secret
metadata:
  name: secret
type: Opaque
data:
  url: aHR0cDovLzEyNy4wLjAuMT04MDgx
  insecure: ZmFsc2U=
  zone: dGVzdC5jb20u
  apiKey: c2VjcmV0
  account: bXlhY2NvdW50
  serverID: bG9jYWxob3N0
  # insecure: false
  # zone: test.com.
  # apiKey: secret
  # account: myaccount
  # serverID: localhost

```

Apply the secret:

```
kubectl apply -f secret-sample.yaml
```

3. Create Dynamic DNS record.

Example: "dynamicdnsrecord-sample.yaml"

```

apiVersion: ncm.nokia.com/v1alpha1
kind: DynamicDNSRecord
metadata:
  name: dynamicdnsrecord-sample
spec:
  domainName: abc.test.com      # Domain name to be updated
  provider: powerdns            # Provider name. (powerdns, aws, rfc2316)
  updateInterval: 10            # Backoff interval for update operation
                                # to avoid too frequent update request to
                                # DNS provider API server
  probeInterval: 5              # Interval to check the registration
                                # status
  ttl: 3000                     # DNS record TTL
  nameServer:                  # DNS servers used to check the resolution
                                # status by standard DNS query. Only used
                                # when the provider doesn't have a query
                                # API. Currently only provider rfc2316
                                # requires this.
  - 127.0.0.1:53
  nodeSelector:                # Node selector to specify the nodes
                                # to be monitored.
    is_edge: "true"
  type: A                      # DNS record type. (A, AAAA, CNAME)
  CNAMETarget:                 # Target value. Only used when record
                                # type is CNAME
  - test.com
  providerInfo: secret         # The name of secret that stores provider
                                # information. The secret shall reside in
                                # the same namespace as the
                                # DynamicDNSRecord record does.
  interface: eth0              # Interface to be monitored.
  updater: default             # Select updater to do the update. If not
                                # provided, updater 'default' is used.

```

Apply the DynamicDNSRecord:

```
kubectl apply -f dynamicdnsrecord-sample.yaml
```

10.8.2 Provider Information in Secret

Here is a list of secret templates with fields required by the controller to perform DDNS update action to service provider.

- PowerDNS

Information in secret required to use PowerDNS Restful API to update DDNS records.

```
apiVersion: v1
kind: Secret
metadata:
  name: secret
type: Opaque
data:
  url: aHR0cDovLzEyNy4wLjAuMTo4MDgx # PowerDNS server URL
  insecure: ZmFsc2U= # Disable HTTPS certificate
  # verification. (true, false)
  zone: dGVzdC5jb20u # Zone name
  apiKey: c2VjcmV0 # API key
  account: bXlhY2NvdW50 # Account name in PowerDNS
  serverID: bG9jYWxob3N0 # PowerDNS server ID

  # url: http://127.0.0.1:8081
  # insecure: false
  # zone: test.com.
  # apiKey: secret
  # account: myaccount
  # serverID: localhost
```

- AWS Route53

Information in secret required to update DDNS records in Route53 provided by AWS.



Note: For AWS, user can either provide AWS authentication information in secret or assign an IAM role with Route53 permission to the node where Updater is located. If user chooses to provide authentication info with secret, the session token shall be valid permanently or be refreshed by user before it is expired, because periodical check will use it. Using IAM role is highly recommended.

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-aws
type: Opaque
data:
  AWSHostedZoneID: WlpWS1A3S1YzR0lH
  AWSRegion: dxMtZWFzdC0y
```

```
AWSAccessKeyId: QVNJQVpLUEo1WEtaV01FUUNOMkc=
AWSecretAccessKey:
Zmp2WTRWRWlzaTkvRnNmbUlKTj1xeF14UnQ1Vm12SVBYTUD6d0FEdQ==
AWSSessionToken:
R1FvR1pYSXZWGR6RUgwYURQYUV0ak4xT2dvQ1NGRks4Q0ttQXEzME5UaXJ

4QjZpSDI0dmNOdXJkYVZhdHdvcy9QR0wxK3d2MTg3bTJTSgtxTHQrRUTJbU

5ZZXZtUk9CcmwrNEI4eVRMSmVQeEpQZmZTdm1MWG1LQUx2Y1ltTkC2N0Evd

F1rNmFPV1VRVELrNXMzaWdJWDFCcStOVR0bkJEQzlwT31HaWMrTCttWDZj

NTEzblc2QkcvVktUUThhUVBMNVB4NDhFcmZRd3R4dE1CU29zTEhKQWNDS29

IU3YyS1FnRzRZTjJ1aD1BZXJtVG9wdD1ZczBnRS9TOU15eVJwd3FHVG1CND

FLR3ZKYWFKVzJlbHBFV1NOWUdzakZvU29Ja3NBeXZ2WWI3S1U3c2gvdG9ZY

zR2azVKaWdNSzdDd1ZtUXMranU1T0ZhQkRySDh6cmR1b1R5Q2tLeUN3TmJH

ZFFGL2xCRDkycXNXTkJ0eUgraGdrU045c2xUVEVuOFB3cHhVR0tzc2FaVE1

mboNxMuXOUExR2xvNHNTOHp0aCtQZ2xYNW5NdXBRU21CaGZIa0JRPT0=


# AWSHostedZoneID: ZZVKP7KV3GIG
# AWSRegion: us-east-2
# AWSAccessKeyId: ASIAZKPJ5XKZWMEQCN2G
# AWSecretAccessKey: "fjvY4VEmSi9/FsfmIJN9qxYxRt5VmviPXMGzwADu"
# AWSSessionToken:
"FOqoGZXIvYXdzEH0aDPaEtjn1OgoCSFFK8CKmAq3ONTirxB6iH24vcNurd
    aVGtwos/PGL1+wv187m2SHkqLt+EKImNYevmROBr1
+4B8yTLJePxJPffs
    viLXmKALvbYmNG67A/tR+6aOWUQTIk5s3igIX1Bq
+h9TtnBDC9pOyGic+
    L+mX6c513nW6BG/
VKTQ8aQPL5Px48ErfQwtxtMBSosLHJAcCKoHSv2KQg
    G4YN2uh9AermTopt9Ys0gE/
S9IyyRpwqGTmB41KGvJaaJW2elpEVSNYGY
    jFoSoIksAyvvYb7KU7sh/toYc4vk5JigMK7CwVmQs
+ju50FaBDrh8zrde
    nTyCkKyCwNbGdQF/lBD92qsWNByH
+hgkSN9sLTTEn8PwpwUGKssaZTMf
    nsq1KW9A1Glo4sS8zth+PglX5nMupQSiBhfHkBQ=="
```

- RFC2316

Information in secret required to update DDNS records to provider which supports RFC2316

```

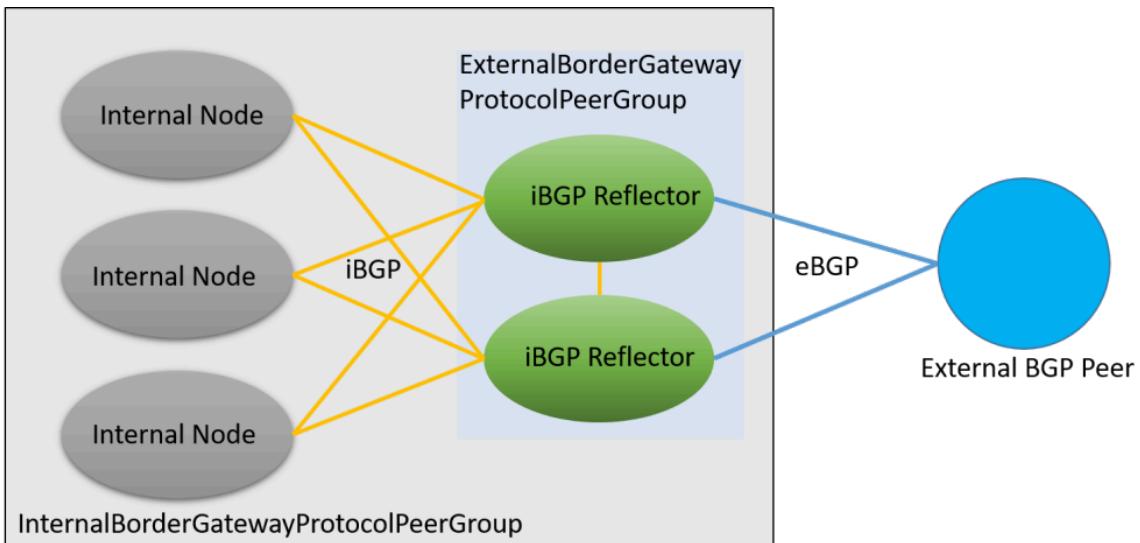
apiVersion: v1
kind: Secret
metadata:
  name: secret-rfc2136
type: Opaque
data:
  zoneName: dGVzdC5jb20=
  DNSServer: MTI3LjAuMC4xOjUz
  TSIGKeyName: dGVzdA==
  TSIGKey:
    R0Mzd1M0Y28yUjhvSHRkbFA1M1dpWGtXbGNlckxaTXJBbk1xaDFycXBhdWV3QWsQ
    2xiOVdHS2pUN0w4a3h1cytVTHdCULN0VHZQOWNMS3VMQ0ZJNWc9PQ==
  HMAC: c2hhMjU2

  # zoneName: test.com
  # DNSServer: 127.0.0.1:53
  # TSIGKeyName: test
  # TSIGKey:
    GC3vS4co2R8oHtd1P53WiXkWlcerLZMrAnIqh1rqpauewAk1C1b9WGKjT7L8kx
    us+ULwBRStTvP9cLKuLCFI5g==
  # HMAC: sha256

```

10.9 Border Gateway Protocol

Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information among autonomous systems (AS). NCS can support BGP interconnection with external peers using CRD. Inside the cluster, iBGP group can be created to include nodes which don't have direct external connection and import route from nodes which have external connection.



Nodes where eBGP are running on are reflectors for iBGP group, which interconnect with external BGP peers and import route from them.

Configuration

To configure BGP interconnection, user needs to create `ExternalBorderGatewayProtocolPeerGroup` and `InternalBorderGatewayProtocolPeerGroup` resource and apply it in Kubernetes.

Requisites: bcmt-ip-man Pods are running normally.

1. Create a `ExternalBorderGatewayProtocolPeerGroup` configuration file. For example:

```
ncm_v1alpha1_externalbordergatewayprotocolpeergroup.yaml.
```

```
apiVersion: ncm.nokia.com/v1alpha1
kind: ExternalBorderGatewayProtocolPeerGroup
metadata:
  name: externalbordergatewayprotocolpeergroup-sample
spec:
  nodeSelector:
    ebpg.networking.ncs.nokia.com/group: out      # Node label
  externalBorderGatewayProtocolPeerTemplate:
    addressFamily: IPv4                          # Address family used
    to connect with peer, used to select local IP address.
                                              # Available options:
    IPv4, IPv6.
    neighbor:                                     # Neighbor information
      autonomousSystemNumber: 64000               # Neighbor Autonomous
      System Number (ASN)
      IPAddress: 172.16.1.16                     # Neighbor address
      port: 1179                                    # Neighbor port
    local:
```

```

        autonomousSystemNumber: 65000          # Local Autonomous
System Number (ASN)
        interface: eth3                      # Interface on node to
connect with neighbor.
                                         # The first non-local
address of addressFamily on the interface is used as the local IP.
                                         # Note, the IP address
to be used will be checked in trackedLocalAddresses list.
                                         # If it is included,
the next one will be used.
        port: 1179                          # Local port on node
to connect with neighbor.
                                         # Note, in current
release, the wellknown BGP port 179 cannot be used due to port
                                         # conflict issue. It
will be fixed in future.
        trackedLocalAddresses:             #
TrackedLocalAddresses is a list of IP addresses, which are monitored on
the node.
                                         # If any one in the
list exists on the node, the BGP local preference will be set
                                         # higher and routes
imported from this node will have high priority.
                                         # It is useful, when
VIP is set on a few nodes and user desire to make internal
                                         # nodes prefer to send
traffic to the node with VIP.
                                         #
TrackedLocalAddresses can be IPv4 or IPv6, which is not limited by
addressFamily
- 172.16.1.100
        advancedOptions:                  # AdvancedOptions are
used to modify BGP interconnection parameters.
                                         # User can find the
list of these parameters in
                                         # https://
bird.network.cz/?get_doc&v=20&f=bird-6.html#ss6.3.
                                         # Note, user should
use with CAUTION, as not all options listed are
                                         # verified and valid
here, such as local, neighbor, interface and etc.
        protocol:                         # Section Protocol
configuration of above link
        # - passive on
        # - keepalive time 60

```

```

    channel:                                # Section Channel
configuration of above link
    # - cost 5

```

2. Apply the ExternalBorderGatewayProtocolPeerGroup.

```
kubectl apply -f ncm_v1alpha1_externalbordergatewayprotocolpeergroup.yaml
```

3. Create InternalBorderGatewayProtocolPeerGroup configuration file. For example:

```
ncm_v1alpha1_internalbordergatewayprotocolpeergroup.yaml.
```

```

apiVersion: ncm.nokia.com/v1alpha1
kind: InternalBorderGatewayProtocolPeerGroup
metadata:
  name: internalbordergatewayprotocolpeergroup-sample
spec:
  nodeGroup: default                      # Node group name.
                                                # A special label
  (ibgp.networking.ncs.nokia.com/group=name) is used
                                                # to define iBGP node
  group.                                     # If group name
                                                # 'default' is used, any node without this label is selected.
                                                # If user defines a
  group other than 'default' (such as, test), # he needs to label
  all the desired nodes with label
                                                #
  'ibgp.networking.ncs.nokia.com/group=test'
  internalBorderGatewayProtocolPeerTemplate:
    port: 1179                               # iBGP interconnection
                                                # Note, in current
    port.                                     # due to port conflict
                                                # release, the wellknown BGP port 179 cannot be used
                                                # issue. It will be fixed in future.
    interface: eth1                           # Interface on node to
                                                # connect with reflector nodes.
                                                # The first non-local
    addressFamily: IPv4                       # Address family.
                                                # The address family of addressFamily on the interface is used.
    autonomousSystemNumber: 65000             # iBGP Autonomous
                                                # System Number (ASN).

```

4. Apply the InternalBorderGatewayProtocolPeerGroup.

```
kubectl apply -f ncm_v1alpha1_internalbordergatewayprotocolpeergroup.yaml
```

5. (Optional) Label desired node for InternalBorderGatewayProtocolPeerGroup.



Note: This is only required when nodeGroup is not 'default'.

```
kubectl label node xxxx ibgp.networking.ncs.nokia.com/group=test
```

Troubleshooting

If it is not working as expected, here are steps to troubleshoot.

1. Check resources are created properly.

```
kubectl get nodeinterfaceaddress -o wide
kubectl describe nodeinterfaceaddress xxxx
kubectl get externalbordergatewayprotocolpeer -o wide
kubectl describe externalbordergatewayprotocolpeer xxxx
kubectl get externalbordergatewayprotocolpeergroup -o wide
kubectl describe externalbordergatewayprotocolpeergroup xxxx

kubectl get internalbordergatewayprotocolpeer -o wide
kubectl describe internalbordergatewayprotocolpeer xxxx
kubectl get internalbordergatewayprotocolpeergroup -o wide
kubectl describe internalbordergatewayprotocolpeergroup xxxx
```

2. Check route table on the node.

```
ip route
ip -6 route
```

3. Check BGP protocol status in bird.

```
kubectl exec -it bcmt-ip-man-agent-xxxx bash
birdcl -s /var/run/ip-man/bird/bird.ctl show route
birdcl -s /var/run/ip-man/bird/bird.ctl show protocol all
```

4. Check the log of ip-man-agent pods.

10.10 Static Route

Some nodes are responsible for communicating with the external networks for exchange of data, we need to maintain the destination-based route table to dynamically add the routes for the external networks.

If the nodes in an iBGP group are configured with static routes, static routes can be exported to internal nodes via iBGP. The nodes with static route configuration can also run as the egress gateway to provide external connection for internal nodes, if there is no iBGP configured. The static routes can be defined in a configmap, which makes it easier to update and maintain. Static route controllers monitor changes in the configmap and add/update/delete static routes accordingly.

A `StaticRouteConfigGroup` defines a group of nodes that connect external networks and maintains the destination-based routes. `StaticRouteConfigGroup` controller discovers nodes and creates `StaticRouteConfig` for each node in the group. IPv6 and IPv4 address family routes can be in the same `StaticRouteConfig`.

A `StaticRouteConfig` defines the static route configuration for a node. `StaticRouteConfig` controller reads the `routeStore` configmap and routes in the spec, and provisions bird configuration for the node.

The `routeStore` configmap is used for store routes, CRD controller can monitor changes from it.

Static Route Configuration To configure Static Route, user needs to create a `StaticRouteConfigGroup` resource and apply it in Kubernetes.

Requisites bcmt-ip-man Pods must be running normally.

A `StaticRouteConfigGroup` defines a group of nodes that connect external networks and maintains the destination-based routes. `StaticRouteConfigGroup` controller discovers nodes and creates `StaticRouteConfig` for each node in the group. IPv6 and IPv4 address family routes can be in the same `StaticRouteConfig`.

1. Create a `StaticRouteConfigGroup` configuration file. For example:

`ncs_v1alpha2_staticrouteconfiggroup.yaml`.

```
apiVersion: ncm.nokia.com/v1alpha2
kind: StaticRouteConfigGroup
metadata:
  name: sample
spec:
  nodeSelector:
    staticroute.networking.ncm.nokia.com/group: edge
  staticRouteConfigTemplate:
    routes:
      - 10.0.0.0/8 via 192.168.1.1 # The
        format of route rules is similar to "ip route add" commands,
      - 2001:db8:1234::/64 dev eth0 # and
        supports multiple nexthops, but can't contain both "via" and "dev" in the
        same rule.
      - 10.10.0.0/16 nexthop via 192.168.1.1 nexthop via 192.168.1.3 # e.g.
        10.10.0.0/16 nexthop via 192.168.1.1 nexthop via 192.168.1.3
    routeStore: routeconfigmap
```

2. Apply the StaticRouteConfigGroup.

```
kubectl apply -f ncm_v1alpha2_staticrouteconfiggroup.yaml
```

3. (Optional) Create a routeStore configmap file. For example: configmap_routeconfig.yaml .

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: routeconfigmap
data:
  routes: |
    172.0.0.0/8 dev eth1
    2001:1::/64 via 2001:1::1
```

4. (Optional) Apply the routeStore configmap. `kubectl apply -f configmap_routeconfig.yaml`.

10.10.1 Dynamic static route management via ConfigMap

StaticRouteConfigGroup & StaticRouteConfig CRDs

Label nodes:

```
kubectl label node bcmt-vc-sut-sandbox1-e-1 \
staticroute.networking.ncm.nokia.com/group=edge
```

Example StaticRouteConfigGroup:

```
apiVersion: ncm.nokia.com/v1alpha2
kind: StaticRouteConfigGroup
metadata:
  name: srcg
spec:
  nodeSelector:
    staticroute.networking.ncm.nokia.com/group: edge
  staticRouteConfigTemplate:
    routes:
      - 100.0.0.0/8 via 10.88.152.177
    routeStore: routeconfigmap
```

Example StaticRouteConfig:

```
apiVersion: ncm.nokia.com/v1alpha2
kind: StaticRouteConfig
metadata:
  name: src
```

```
spec:
  node: bcmt-vc-sut-sandbox1-e-1
  routes:
    - 100.0.0.0/8 via 10.88.152.177
  routeStore: routeconfigmap
```

Example ConfigMap:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: routeconfigmap
data:
  routes: |
    111.0.0.0/8 via 10.88.152.177
    112.0.0.0/8 via 10.88.152.177
```

ip route output on an edge node:

```
default via 10.88.152.225 dev eth0 proto static metric 100
10.20.30.0/24 dev eth1 proto kernel scope link src 10.20.30.41 metric 101
10.88.152.176/28 dev eth2 proto kernel scope link src 10.88.152.188 metric 102
10.88.152.224/27 dev eth0 proto kernel scope link src 10.88.152.253 metric 100
10.150.0.0/16 via 10.88.152.177 dev eth2
100.0.0.0/8 via 10.88.152.177 dev eth2 proto 100 metric 32
111.0.0.0/8 via 10.88.152.177 dev eth2 proto 100 metric 32
112.0.0.0/8 via 10.88.152.177 dev eth2 proto 100 metric 32
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.20.64 dev calid3ce5750ee2 scope link
blackhole 192.168.20.64/26 proto bird
192.168.20.65 dev cali98ae42e5d19 scope link
192.168.20.66 dev calic41a9b699b1 scope link
[...]
```



Note: Either use StaticRouteConfigGroup (SRCG) or StaticRouteConfig (SRC), not together.

Using SRC you can create node specific static routes. But to manage node/host groups together use SRCG instead, which populates SRCs automatically.

A ConfigMap can be referenced which can contain many route definitions. The BCMT IP Man operator watches for these reference ConfigMaps, so whenever you add a new entry into it, it is applied immediately.

Nodes have to be labeled:

```
kubectl label node <nodename> staticroute.networking.ncm.nokia.com/group=edge
```

Formatted example files:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: routeconfigmap
data:
  routes: |
    10.94.193.64/28 via 10.37.76.17
    10.94.243.192/28 via 10.37.76.17
```

```
apiVersion: ncm.nokia.com/v1alpha2
kind: StaticRouteConfigGroup
metadata:
  name: srcg
spec:
  nodeSelector:
    staticroute.networking.ncm.nokia.com/group: edge
  staticRouteConfigTemplate:
    routes:
      routeStore: routeconfigmap
```

11 Services

11.1 NTP

NCS supports chrony modes by default.

NTP configuration parameters are initially set in the bcmt_config.json file and deployed during initial NCS installation. Post-installation, NCS provides CLI commands for the management of configuration settings for NTP services.

11.1.1 Manage NTP using a configuration file

Use this procedure to enable the configuration of NTP settings during NCS deployment.

1. Access the bcmt_config.json file.
2. Specify the node type that will provide access to the public network by setting the bcmt_config.json file parameter **internal_ntpservers** to **control** or **edge**.
3. Specify the external upstream NTP server(s) by listing the IP address for each server in the bcmt_config.json file parameter **external_ntpservers**.
4. This feature is ready for deployment.

Note: If left blank, the NCS cluster will not time sync with an external timing server. Instead, all cluster nodes will self-govern timing using internal timing servers. To avoid single server failure risk, all internal NTP servers act as orphan servers. The server with the smallest reference ID, based on IP addresses, takes the lead role of master and synchronizes all other servers to it. If server failure were to occur, the server with the next smallest reference ID would become the next master.

11.1.2 Manage NTP using CLI commands

Use the **ncs service ntp** command to manage NTP service details.

- Use the following command to update the internal or external NTP server's IP address in **Virtual deployment**.

```
ncs service ntp update --ext_servers --int_servers
```

Example:

```
ncs service ntp update --ext_servers=10.10.10.1,10.10.10.2 --  
int_servers=control
```

- **In case of Baremetal deployment**

To update the external NTP run the following python script:

```
python /usr/lib/python2.7/site-packages/nokia/cmframework/utils/external-
ntp-update.py EXTERNAL_NTP_SERVERS_SEPERATED_BY_COMMA -cn CLUSTER_NAME -pu
NCS_CLUSTER_USER_NAME -pp NCS_CLUSTER_PASSWORD
```

Example:

```
python /usr/lib/python2.7/site-packages/nokia/cmframework/utils/external-
ntp-update.py -ntp 10.20.110.16,1.1.1.1 -cn my-cluster -pu ncs-admin -pp
Password!@
```

Figure 12: Update NTP example

```
Run ncs ntp update and update BM inventories

optional arguments:
  -h, --help            show this help message and exit
  -ntp--external_ntp New_NTP_IP
                        specify one or more ntp spered by comman ','
  -cn cluster name, --cluster_name cluster name
                        specify the full cluster name.
  -pu portal_user, --portal_user portal_user
                        specify portal user name (ncs cluster user name)
  -pp portal_pass, --portal_pass portal_pass
                        specify portal password (ncs cluster password)
```



Note: You should not run ncs service ntp update command, you should only run the script as the command is included.

- Use the following command to display the current NTP settings.

```
ncs service ntp get
```

Example output:

```
{
  "external_ntpservers": "10.10.10.1,10.10.10.2",
  "internal_ntpservers": "control"
}
```



Note: For NCS on BareMetal deployment, you should only run the script. For NCS on Virtualized deployment, you should only run the ncs service ntp update command.

11.1.3 Updating poll interval and open port 123 for localhost

Chronyd uses default poll interval to request ntp servers for time synchronization, the minimum interval between requests sent to the server is 64s and maximum is 1024s. In the NCS cluster, internal ntp clients don't listen for any port. If you want to customize the minpoll and maxpoll interval for time synchronization and open 123 port for localhost in internal ntp clients, you can use bcmt hook script to achieve the function.

Below is the hook usage:

1. After NCS deployment, log in to one control node and create the path:

```
mkdir -p /opt/bcmt/storage/hooks/update-chronyrd
```

2. Go to /opt/bcmt/storage/hooks/update-chronyrd and create file main.yaml as below:

```
- hosts: "{{ run_hosts|default('role_all') }}"
become: yes
gather_facts: "{{ run_gather_facts|default(True) }}"
serial: "{{ run_serial|default('100%') }}"
any_errors_fatal: "{{ run_any_errors_fatal|default(True) }}"
tasks:
  - set_fact:
      var_ntp: '{% if internal_ntpservers %}{{internal_ntpservers}}{% else %}control{%endif%}'

  - set_fact:
      internal_timeservers: '{{ groups["role_" + var_ntp ] }}'

  - name: set minpoll and maxpoll for chronyrd
    become: yes
    replace:
      path: /etc/chrony.conf
      regexp: '(iburst)(.*)'
      replace: 'iburst minpoll 3 maxpoll 3'

  - name: listen localhost for internal client
    replace:
      dest: /etc/chrony.conf
      regexp: "# Allow all NTP clients to access."
      replace: "allow 127.0.0.1\nallow {{ hostvars[inventory_hostname]
['ansible_default_ipv4']['address'] }}"
    when: "ansible_hostname not in internal_timeservers and
network_stack != 'ipv6_only'"

  - name: listen localhost for internal client if ipv6_only
    lineinfile:
      dest: /etc/chrony.conf
      regexp: "# Allow all NTP clients to access."
      replace: "allow ::1\nallow {{ hostvars[inventory_hostname]
['ansible_default_ipv6']['address'] }}"
    when: "ansible_hostname not in internal_timeservers and
network_stack == 'ipv6_only'"
```

```
- name: Restart chronyd service for all nodes
  become: yes
  service:
    name: chronyd
    state: restarted
    enabled: yes
```

Copy the content of below hook script to /opt/bcmt/storage/hooks/update-chronyd/main.yml

```
touch /opt/bcmt/storage/hooks/update-chronyd/main.yml
```



Note: In the hook script, the number after *minpoll/maxpoll* means the minimum/maximum poll internal is defined as a power of 2, so *minpoll 3 maxpoll 3* means both minimum and maximum poll internal are 8 seconds.

3. Execute hook CLI to update chronyd's configuration:

```
ncs tool hook --hook_name update-chronyd
```

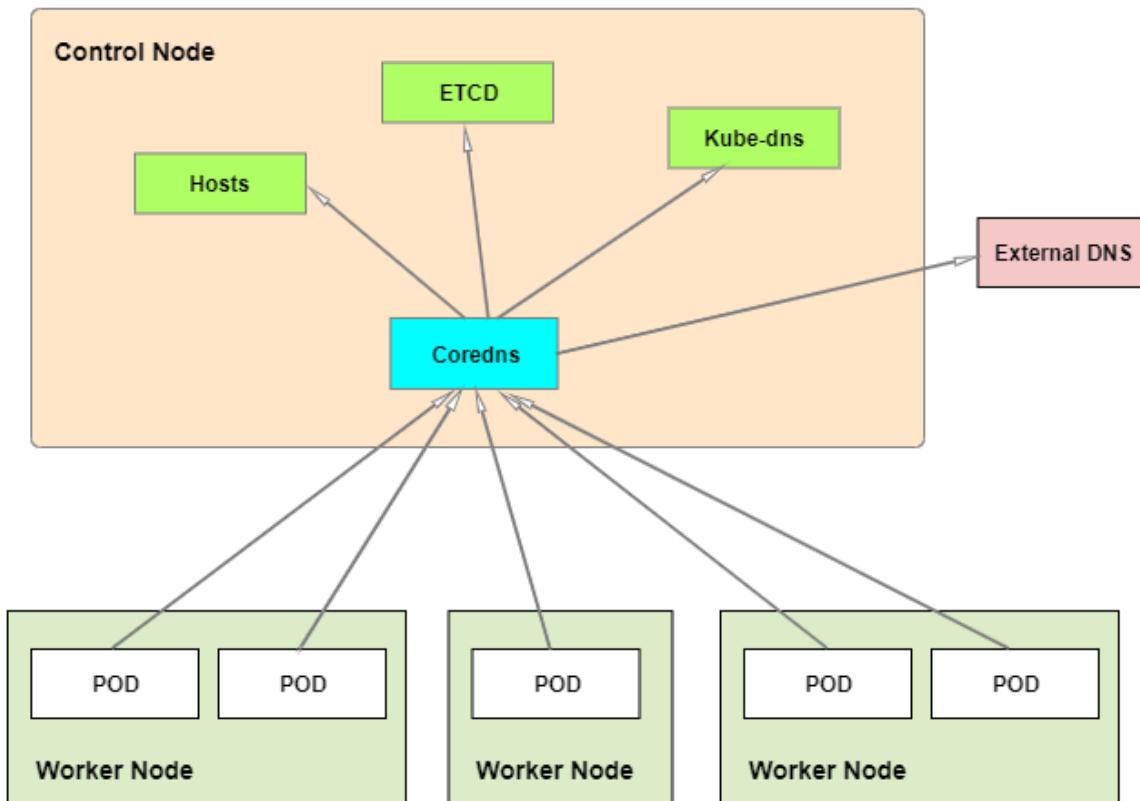
4. After executing cluster scale-out/upgrade, node healing or updating external NTP servers of the cluster, you also need to run above CLI to update chronyd configuration.

11.2 DNS

CoreDNS works as a relay on Control nodes between Worker/Edge nodes. CoreDNS also provides a branch to the Kubernetes add-on, kube-dns, to provide automatically-generated DNS records.

When applications make a DNS request, DNS queries will be sent to CoreDNS in node. Based on the domain suffix, DNS queries will be forwarded to different entities. If the domain suffix is not matched, DNS queries will be sent to external DNS servers, if they are provided in the cluster configuration.

Figure 13: DNS query architecture



11.2.1 Support different DNS server for different zones

Different DNS servers are supported for different zones.

At present, the maximum number of external DNS servers for the same domain is **15**.

Different values are separated by comma, each value can only contain one server which match to one IP.

If one domain wants to match to more IPs, please set different IPs in different values which have the same domain, such as:

```
"external_dns": "abc.com/2.2.2.2, abc.com/3.3.3.3:53"
```

For the value of `external_dns`, the following formats are allowed:

- The value can be NULL
- The value can be IPs(IPv4, IPv6) which don't have domain. Please make sure use '[]' for IPv6 when port used for IPv6, such as:

```
"external_dns": "172.23.102.60,135.239.25.18:53,  
[2001:282:4000:2000::5]:54,2003:282:4000:2000::5"
```

- The value can be domains.

Please make sure the domain name does not start with '/'. Each value can only contain one server which match to one IP. If one domain wants to match to more IPs, please set different IPs in different values which have the same domain, such as:

```
"external_dns": "abc.com/2.2.2.2, abc.com/3.3.3.3:53,ef.com/[2001:282:4000:2000::5]:54,ef.com/2003:282:4000:2000::5"
```

- The value can be a combination of domains and IPs.

For the IP only value, it means the default external DNS server.

For the value with domain, it means the request for the specific domain name will be forward to the related IP, others will be forward to the default external DNS server, such as:

```
"external_dns":  
"10.10.10.10,abc.com/172.23.102.60,ef.com/135.239.25.18:53,2003:282:4000:2000::5"
```



Note:

The following domain formats are not allowed:

- Domain contains '/'
- Multi domains for one external_dns server, such as: ab.com/cd.com/10.10.10.10

11.2.2 Manage DNS using a configuration file

Use this procedure to configure external DNS server IP addresses prior to NCS deployment.

1. Access the bcmt_config.json file.
2. Specify the upstream external DNS server(s) by listing the IP address for each server in the bcmt_config.json file parameter **external_dns**.
3. Deploy the cluster.



Note: During deployment, CoreDNS images are started on Control nodes and kube-dns images are pulled and run by Kubernetes. CoreDNS in node is configured to support DNS query diversion.

4. **End of steps:** this procedure is complete.

11.2.3 Manage DNS using CLI commands

Use the **ncs service dns** command to manage DNS service details.

- Use the following command to update the external DNS server's IP address in **Virtual deployment**.

```
ncs service dns update --external_dns
```

Example:

```
ncs service dns update --external_dns  
"10.10.10.10,baidu.com/172.23.102.60,google.com/135.239.25.18"
```

- In **Virtualized deployment**:

Update external DNS with the following command:

```
ncs service dns update --external_dns LIST_OF_NEW_EXTERNAL_DNS
```

In Baremetal deployment: this is not enough and it does not update inventories, therefore, a script was created in NCS 22 (in later releases it will be an operation).

To update the external DNS run the following python script:

```
python /usr/lib/python2.7/site-packages/nokia/cmframework/utils/external-  
dns-update.py -dns EXTERNAL_DNS_SERVERS_SEPERATED_BY_COMMA -cn CLUSTER_NAME  
-pu NCS_CLUSTER_USER_NAME -pp NCS_CLUSTER_PASSWORD
```

Example:

```
python /usr/lib/python2.7/site-packages/nokia/cmframework/utils/external-  
dns-update.py -dns 10.8.195.1,1.1.1.1 -cn my-cluster -pu ncs-admin -pp  
Password!@
```

Figure 14: Update DNS example

```
Run ncs dns update and update BM inventories

optional arguments:
  -h, --help            show this help message and exit
  -dns--external_dns New_DNS_IP
                        specify one or more dns seperated by comma ','
  -cn cluster_name, --cluster_name cluster_name
                        specify the full cluster name.
  -pu portal_user, --portal_user portal_user
                        specify portal user name (ncs cluster user name)
  -pp portal_pass, --portal_pass portal_pass
                        specify portal password (ncs cluster password)
```



Note: You should not run `ncs service dns update` command, you should only run the script as the command is included.

- Use the following command to display the current DNS settings.

```
ncs service dns list
```

Example output:

```
{
  "external_dns": {
    "abc.com": "135.239.25.18",
    "baidu.com": "172.23.102.60",
    "default": "10.10.10.10"
  }
}
```



Note: For NCS on BareMetal deployment, you should only run the script. For NCS on Virtualized deployment, you should only run the `ncs service dns update` command.

11.2.4 Usage

- *.bcmt

Used for NCS component discovery (services provided by NCS, such as the Docker registry).

Populated to etcd and served by CoreDNS.

- `registry-proxy.bcmt`
- `k8s-apiserver.bcmt`

- *.cluster.local

Used for kubernetes service or pod discovery.

Served by kube-dns. For details, see [dns-pod-service](#).

- Domain name

Used for DNS resolution for specific domain.

Served by external DNS resolver.

- *

Used for public DNS resolution. Served by external DNS resolver.

11.3 High availability

NCS uses Keepalived on Edge nodes to provide high availability. Users can define the ethx interfaces, Virtual IPs (VIP), and Virtual Router Redundancy Protocol Ids (VRRP-ID). Users can also define which ethx interfaces are used for a VIP, the number of VIPs to be used in a cluster, and which networks will provide VIP service.

The following figures show four different Keepalived use cases:

Figure 15: Keepalived use cases (2/1)

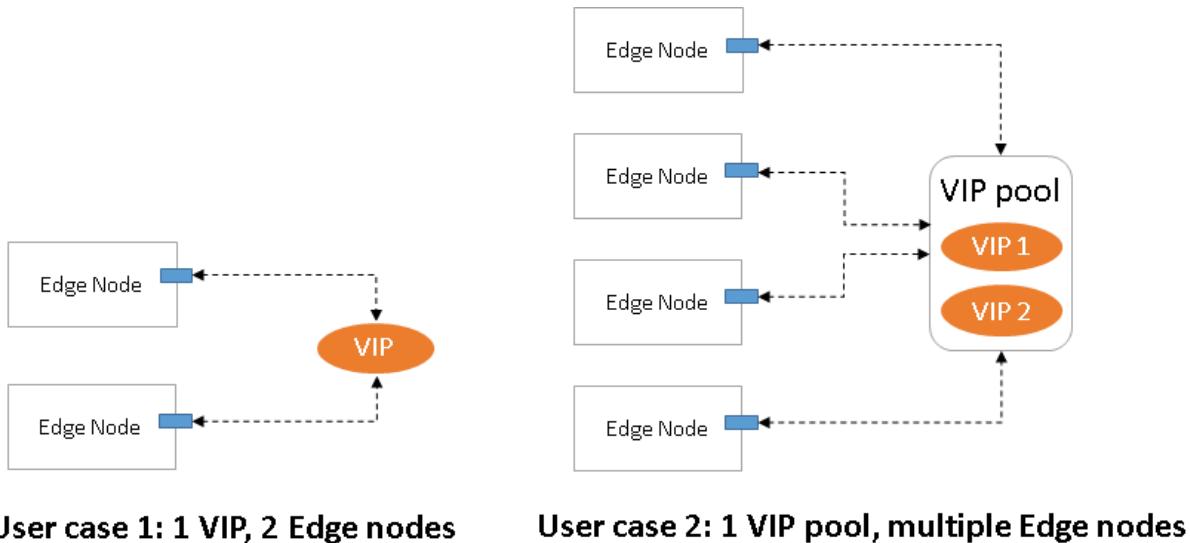
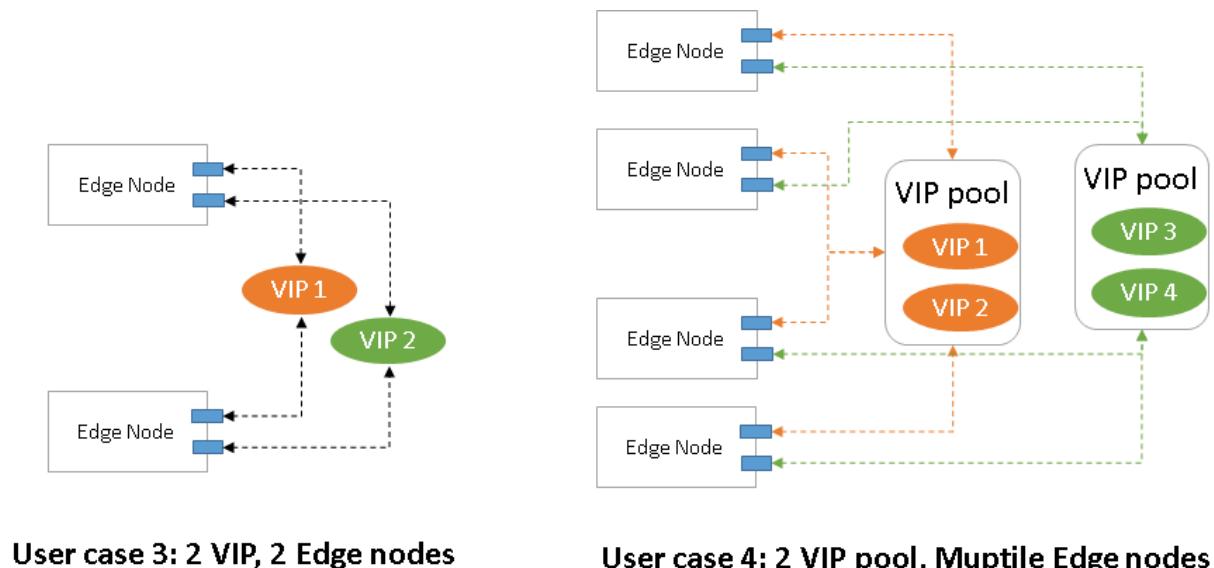


Figure 16: Keepalived use cases (2/2)



11.3.1 Enable high availability using a configuration file

Prerequisites

- By default, keepalived uses multicast and the VRRP protocol. Ensure that multicast is allowed and that the VRRP protocol (IP protocol 112) is allowed as part of the "ingress" security rules. For VIPs, the VRRP protocol must be allowed in external/internal security rules. Ensure that "224.0.0.18" is allowed in iptables.
- For VRRP-ID, ensure that it is a unique integer in the range 1-254. Different VIPs must have different VRRP-ID values.

- The VIP allocation scheme is defined. For example, which ethx interfaces are used to allocate VIPs and how many VIPs are allocated to each interface.
- VIPs are installed on Edge node interfaces
- In an OpenStack environment, the VIP must be added to `allowed_address_pairs`

When NCS is used to create the cluster infrastructure, high availability can be enabled during the software deployment.

1. Access the `user_input.yml` file.

2. Define the following parameters in the `user_input.yml` file:

- Enable VIP by setting the parameter `create_vip` to `true`.
- Identify the networks where VIPs will be created by listing network names in the parameter `network_index`.
- Use the parameter `count_of_vip` to specify the number of VIPs required.
- Support multiple VIPs in multiple networks by listing the values for `count_of_vip`, `network_index`, `allowed_address_pairs` for the required VIPs.

Example:

```
create_vip      : true
vips:
  - network_index : net-02
    count_of_vip  : 2

  - network_index : net-01
    count_of_vip  : 1
```

3. This feature is ready for deployment.

End of steps: this procedure is complete.



Note: This solution does not support the node selector. If you need to assign VIPs for self-select nodes, use the CRD solution.

11.3.2 Enable high availability using a CRD

High availability can be enabled during the software deployment using a CustomResourceDefinition (CRD). This solution has the following advantages:

- Supports unicast addressing. Virtual Router Redundant Protocol (VRRP) is based on multicast addressing, but if that functionality is not available, unicast can be used instead.
- Supports VIP sharing group. Several VIPs can form a sharing group, in which VIPs are assigned with different priorities on different nodes.

- Supports Pod tracking. VIP can be configured to track a Pod's status on a specific node. If the Pod is not in a ready state, the node will drop the VIP until the Pod becomes available.

Prerequisites

bcmt-ip-man Pods must be running normally.

1. Create a VirtualIPClass configuration file that defines global options for VIPs. For example:

ncs_v1alpha1_virtualipclass1.yaml.

```
apiVersion: ncm.nokia.com/v1alpha1
kind: VirtualIPClass
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: virtualipclass-sample1
spec:
  globalConfig:
    vrrpVersion: "2"      # VRRP protocol version. Options: "2", "3". This must be quoted.
```

2. Apply the VirtualIPClass using the following command:

```
kubectl apply -f ncm_v1alpha1_virtualipclass1.yaml
```

3. (Optional) Create a VirtualIPSharingGroup configuration file. For example:

ncm_v1alpha1_virtualipsharinggroup.yaml.

4. (Optional) Apply the VirtualIPSharingGroup using the following command:

```
kubectl apply -f ncm_v1alpha1_virtualipsharinggroup.yaml
```

5. (Optional) Create a VirtualIPTarget configuration file. For example:

ncs_v1alpha1_virtualiptrackingtarget.yaml.

```
apiVersion: ncm.nokia.com/v1alpha1
kind: VirtualIPTarget
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: virtualiptrackingtarget-sample
spec:
  podSelector:
    app: nginx # Labels of pods to be tracked.
```

6. (Optional) Apply the VirtualIPTarget using the following command:

```
kubectl apply -f ncm_v1alpha1_virtualiptrackingtarget.yaml
```

7. Create a VirtualPIstanceGroup configuration file.

```
apiVersion: ncm.nokia.com/v1alpha1
kind: VirtualIPInstanceGroup
metadata:
  labels:
    sharinggroup: sample
    controller-tools.k8s.io: "1.0"
  name: virtualipinstancegroup-sample1
spec:
  nodeSelector:
    node: edge
  unicast: false
  virtualIPInstanceTemplate:
    trackingTargets:
      - virtualiptrackingtarget-sample
    sourceNetworkAddressTranslation: true
    which becomes master
    primaryInterface: "eth1"
    virtualRouterID: 10
    virtualIPAddresses:
      - address: "192.168.100.1/24"
        interface: "eth0"
    primary interface will be used.
      - address: "192.168.200.1/24"
        interface: "eth1"
    virtualRoutes:
      - cidr: 172.19.0.0/16
        via: 192.168.100.2
      - cidr: 172.20.0.0/16
        via: 192.168.200.2
    preempt: true
  service interruption, when failed node with high priority becomes normal again
    virtualIPClass: virtualipclass-sample1
    # Nodes where the VIPs will be floating
    # Whether unicast is used.
    # Enable SNAT for VIPs listed in virtualIPAddresses on node,
    # Interface used to send VRRP messages
    # VRRP-ID
    # All VIPs in virtualIPAddresses switch together
    # Virtual IP address
    # Interface where VIP will be assigned. If not specified,
    # Routes to be added on node, which becomes master.
    # Whether preempt is enabled. Note, enabling preempt may cause
```

8. Apply the VirtualPInstanceGroup using the following command:

```
kubectl apply -f ncm_v1alpha1_virtualipinstancegroup1.yaml
```

9. Verify the status of the VIP using the following kubectl command.

Example command:

```
kubectl describe virtualipinstancegroup.ncs.nokia.com/  
virtualipinstancegroup-sample1
```

End of steps: this procedure is complete.

11.3.3 Manage high availability using CLI commands

Use the **ncs service keepalived** command to manage high availability service details.

- Use the following command to add a VIP for each interface:

```
ncs service keepalived set --virtual_ip4=<virtual ipv4 address> --virtual_ip6=<virtual ipv6 address> --vip_type=<vip type> --interface=<interface> --vrrp_id=<VRRP-ID> --priority=<priority>
```

Example:

```
ncs service keepalived set --virtual_ipv4=172.16.1.124 --vip_type=ipv4 --interface=eth0 --vrrp_id=42
```

- Use the following command to display the cluster's VIP settings.

```
ncs service keepalived get
```



Note: The first string of each set is the **vip_uuid**.

Example output:

```
200 OK
{
    "18a765d0-e9f1-4575-8ef6-7d3c310286d9": {
        "enable_flag": "disable",
        "interface": "eth0",
        "priority": 180,
        "vip": "172.16.1.124",
        "vip_type": "ipv4",
        "vrrpid": null
    },
    "e57b7771-97b8-4f56-ba8f-9a0ddb6650e3": {
        "enable_flag": "disable",
        "interface": "eth0",
        "priority": 180,
        "vip": "2001:db8:1234::",
        "vip_type": "ipv6",
        "vrrpid": 32
    }
}
```

- Use the following command to start the VIP service.

```
ncs service keepalived start --vip_uuid
```

Example:

```
ncs service keepalived start --vip_uuid=18a765d0-e9f1-4575-8ef6-7d3c310286d9
```

- Use the following command to stop the VIP service.

```
ncs service keepalived stop --vip_uuid
```

Example:

```
ncs service keepalived stop --vip_uuid=e57b7771-97b8-4f56-ba8f-9a0ddb6650e3
```

- Use the following command to delete the cluster's VIP settings.

```
ncs service keepalived delete --vip_uid
```

Example:

```
ncs service keepalived delete --vip_uuid=e57b7771-97b8-4f56-ba8f-9a0ddb6650e3
```

- Use the following command to reset a cluster's VIP settings.

```
ncs service keepalived reset --virtual_ipv4 --virtual_ipv6 --vip_type --interface --vrrp_id --priority
```



Note: Ensure the new vip or vrrp_id values are not in use.

Example:

```
ncs service keepalived reset --vip_uuid=18a765d0-e9f1-4575-8ef6-7d3c310286d9 --virtual_ipv4=172.16.1.158 --vip_type=ipv4 --interface=eth0 --vrrp_id=89
```

End of steps: this procedure is complete.



Note: This solution does not support the node selector. If you need to assign VIPs for self-select nodes, use the CRD solution.

11.3.4 Define a default Pod prio class for applications forbidding preemption

Pod priority ensures that high priority pods can be “preempted” (admitted into the cluster even if there is normally not enough resources are freely available). Preemption is prohibited for pods of applications. 'ncs-default-priority-class' is the default priorityclass for applications which have no other dedicated priorityclass. In case of out of memory situation these kind of pods are stopped before system critical infrastructure related ones therefore the infrastructure is kept alive. Non-preemptive priority class made the default to ensure low-priority application pods cannot trigger preemption unless it is explicitly allowed by the operator.

11.3.5 Static VIP address for edges on vCenter

1. The configuration of a keepalived service should be performed manually after BCMT cluster deployment. It is not yet integrated in deployment at that stage.
2. For OpenStack, CLCM, create and reserve the port/IP along with the instance creation and generate one VIP.json file to let the user know the VIP address.

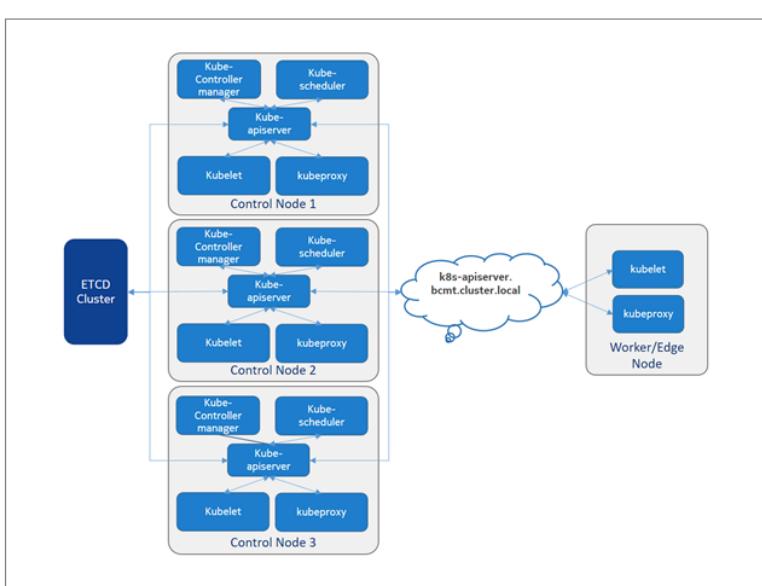
Now the user can proceed to this *High availability* on page 257 section based on these VIP addresses.

Without CLCM, the user could also use OpenStack CLI or the dashboard to create these VIPs before setting the keepalived service.

- Note:** For vSphere vCenter, all the IP addresses including IP used for nodes, for VIP etc. all should be pre-planned ahead of the cluster deployment.
 - Note:** CLCM cannot reserve IPs used for the VIP.
 - Note:** If the setting of the keepalived service is integrated in deployment in later releases, CLCM may align with OpenStack to collect the VIPs, then pass them to BCMT via inventory file.

11.3.6 Kubernetes cluster HA

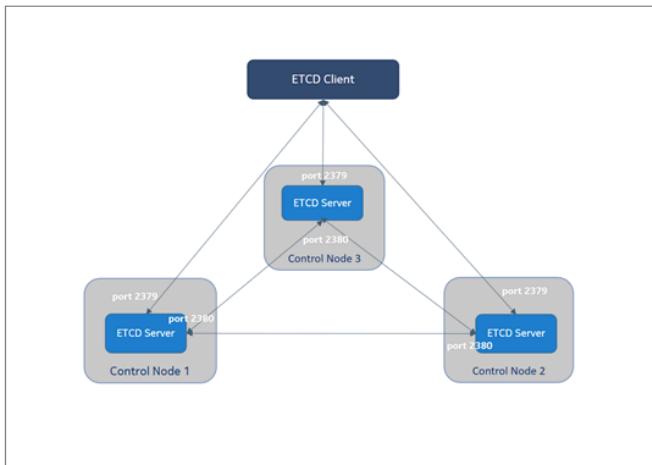
- NCS Control node hosts Kubernetes master services, and Worker and Edge node host Kubernetes node services. To provide high availability, NCS supports 3 control nodes to compose Kubernetes master services, and multiply worker or edge nodes to compose Kubernetes worker machine cluster.
 - The DNS record “k8s-apiserver.bcmt.<customized-domain-name>” points to all available Kubernetes API server end points, which is used for Kubernetes worker machines(worker or edge node) to communicate with Kubernetes API server. (The default is k8s-apiserver.bcmt.cluster.local but this is not user specified).
 - The NCS health check process on each control node keeps monitoring its local Kubernetes API service status, and add/remove its Kubernetes API end point from above DNS record.
 - Kubelet and Kubctl on NCS control node only accesses its local API server end point to avoid cross node traffic, and it also facilitates fast error detection.
 - NCS with One control configuration is also supported, which is only for experimental usage.



11.3.7 ETCD Cluster HA

ETCD is a consistent and highly-available key value store used as Kubernetes's backing store for all cluster data. To provide high available ETCD cluster, NCS set up one ETCD server instance on each control node as systemd service. With 3 control node NCS configuration, each ETCD server could connect to the others to compose one cluster, client could talk to each ETCD server's endpoint.

ETCD systemd unit file: /etc/systemd/system/etcd.service



11.3.8 Edge Node VIP

VIP is optional, it is needed only if the user wants to enable HA for edge nodes. VIP can be enabled during NCS deployment, it can also be added after the deployment. Refer to *High availability* of the *Operations and Administration Manual* for more details.

NCS provides VIP supports on edge nodes using keepalived. In case 2 edge nodes with 1 VIP, the VIP is floating between 2 edge nodes. In case of more than two edge nodes with more VIPs provisioned, all VIPs will distribute on all edge nodes according to scheduling algorithm. As long as there is a living edge node, the VIP is accessible.

11.3.9 API Service HA

NCS runs API servers on three control nodes in active-active mode. Its configuration and data stores in etcd which are accessible from any one of API server instances.

11.3.10 DNS HA

NCS runs 3 DNS service on three control nodes in active-active mode. DNS instances use etcd for data storage and synchronization.

There is one coredns service in each node.

In all non-control nodes, local dns service will forward all requests to control nodes with policy round_robin.

In control node, all requests would be separated to 3 parts:

- Domains end with .bcmt. will be checked in both etcd and hosts file(/opt/bcmt/config/coredns/hosts/hosts in each control node).
- Domains end with dns_domain (.cluster.local as default) will be forward to kube-dns.
- Others will be send to external dns.

Policies are all round_robin for all dns requests.

11.3.11 Registry/chart repo HA

NCS provides local docker registry and chart repository services. They achieve high availability as follows:

- Both services run on three control nodes in active-active mode. It continues provides services as long as one or more control nodes living.
- Both services use the glusterFS as storage backend and glusterFS keeps the data synchronized among different instances.

11.3.12 Living upgrade/rollback support

Leveraging Kubernetes cluster HA architecture, NCS provides living upgrade and rollback without impact to application services.

NCS upgrades control nodes in a round robin mode.

NCS upgrades worker nodes and edge nodes in parallel mode. The number of node to upgrade each time bases on user specified percentages, or user given node list. Kubernetes will reschedule pod resident on node to be upgrades to other nodes.

Rollback follows a similar workflow to upgrade.

11.3.13 Disaster recovery support

Disaster recovery is another level of high availability provided by NCS. NCS provides cluster level backup to a remote server. In case of disaster, the user can retrieve data from the remote server, restore all configuration data and bring up all applications running in clusters before disaster.

11.3.14 NCS heartbeat

NCS runs a bcmt-heartbeat service on three control nodes to monitor cluster health and restore if required.

- The NCS heartbeat process on each control node keeps monitoring its local k8s API service status, and add/remove its k8s API end point from above DNS record.
- The NCS heartbeat process on each control node detects one k8s API service is down, will try to shutdown connected sockets for kubelet.

- The NCS heartbeat process on each cluster node keeps perf info collecting to `/var/log/bcmt/perf.log` based on top command.
- the NCS heartbeat process on each control/storage node starts service to monitor cluster shared volume mount status, such as `/opt/bcmt/storage`. If glusterfs is used for cluster shared volume, heartbeat will restart glusterfs volume when glusterfs transport endpoint is not connected.
- The NCS heartbeat process on each storage node monitors volume (only bcmt defined, such as `bcmt-glusterfs`, `cbr-glusterfs-backup`, `cbr-glusterfs-backup-cluster` and `cbr-glusterfs-repo`) detail status, and restart glusterd service only if 2 bricks 're offline.
- The NCS heartbeat process on each control node keeps monitoring the pods deployed by statefulset on worker nodes, and terminate pods once pod is unknown or terminating status.
- The NCS heartbeat process on each control node keeps monitoring the pods deployed by deployment on worker nodes, and terminate pods once pod is unknown or terminating status.
- The NCS heartbeat process on each cluster node checks root user's account/password validity, and record alarm if the account/password has expired/will expire today/will expire in some days.
- The NCS heartbeat process on each cluster node checks the root ca `/etc/pki/ca-trust/source/anchors/ca.pem` validity, and record alarm if ca has expired/will expire today/will expire in some days.
- The NCS heartbeat process on each control node monitors the bcmt-cmdb-mariadb pods status, and auto heal them if all 3 pods 're not ready.

11.4 Docker registry and chart repository

This section describes the default Docker registry and Chart repo, there is no authentication verification to access them, to set up a security Docker registry and chart repo, please refer to Harbor NCS integration section.

During deployment, the NCS Docker registry is set up automatically and NCS images are loaded into the registry. The Docker registry consists of two parts:

- an internal private Docker registry, and
- a registry proxy

The internal private Docker registry is the repository for all loaded images.

The Registry proxy enables user access to an external Docker registry of images located outside of a NCS cluster environment.

11.4.1 NCS Docker Registry

NCS Docker Registry function includes two parts: internal private Docker registry and registry proxy to external Docker registry outside of NCS cluster. Internal private Docker registry serves as the repository of Docker images for the whole cluster. In the container life cycle, user can push/pull images to/from it. NCS

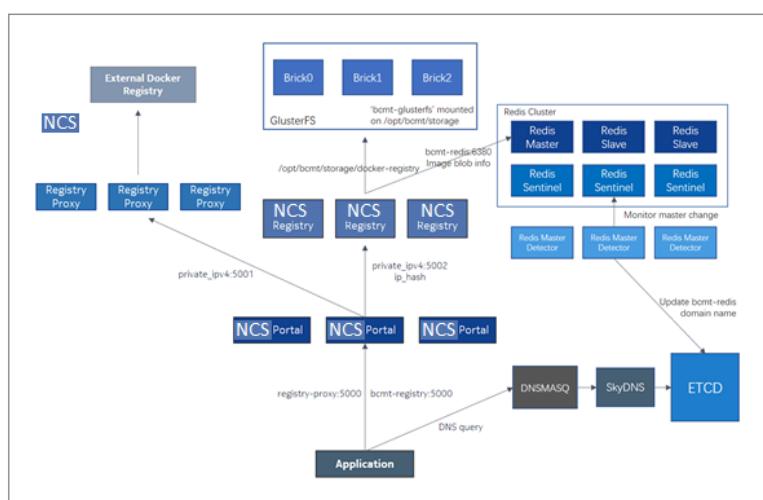
also provides CLI function to manage images in this Registry. Registry proxy provides user a way to access external Docker registry in case the hosts are in isolated environment, where containers are running.

During deployment, NCS Docker registry is set up, and NCS infrastructure images are loaded into it.

NCS Docker Registry contains following entities:

- **Docker Registry/Proxy core function.** Provided by docker-distribution binary for the management of image repository, manifest, blob and etc. It runs three copies on different nodes in load-sharing mode to serve the request of docker image operation.
- **Distributed shared filesystem.** Provides failure-toleration storage of image layer files. In NCS cluster cloud deployment, the distributed filesystem is set by glusterfs, which requires three nodes to form a cluster to prevent data loss in case of node failure. Under BareMetal deployment, the distributed filesystem is replaced with ceph fs shared volumes, which is created and mounted to each node /opt/bcmt/storage directory before Kubernetes deployment.
- **Redis cluster.** Redis cluster provides in-memory storage of image blob information. Docker-distribution saves and fetches blob information in/from the Redis cluster. Redis Cluster works in master/slave mode with three nodes. The cluster is guarded by three Redis Sentinels for the cluster health status. And three copy master monitors are running constantly to provide Docker distribution with correct master address. By pinging master node all the time, master IP address is updated once the master node is down, and Docker distribution will access the new master node.
- **DNS system.** Domain names are used to provide high level redundancy for NCS Docker registry. User uses *bcmt-registry:5000* and *registry-proxy:5000* to access the internal Docker registry and registry proxy separately. These URLs are translated into IP addresses of proper control nodes. Domain name *bcmt-redis* provides IP address of Redis master node and is maintained by NCS Redis master detector.

NCS Portal (nginx). NCS portal serves as the frontend of NCS registry and is running on three nodes in load-sharing mode. Internal Docker registry and registry proxy are anchored on the same port 5000 by the domain name. IP hash function is enabled to avoid Docker daemon PATCH operation is sent to a different backend other than the one by which Docker daemon.



11.4.2 Manage the Docker registry using CLI commands

Use the **ncs service registry** command to manage the Docker registry proxy.

- Use the following command to display details about the remote registry proxy.

```
ncs service registry-proxy get
```

- Use the following command to update the remote registry proxy.

```
ncs service registry-proxy set --url
```

Example:

```
ncs service registry-proxy set --url=https://example.nokia.com
```

If user needs to add a certificate file to the Docker registry, refer to *RegistryCertificate* on page 383.

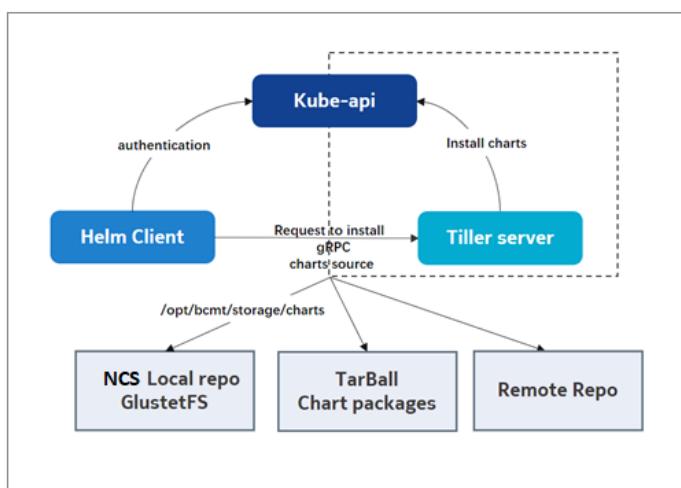
- Use the following command to remove the remote registry proxy.

```
ncs service registry-proxy delete
```

End of steps: this procedure is complete.

11.4.3 NCS Chart Repository

The NCS local chart repo uses `/opt/bcmt/storage/helm_home` as a helm home directory. NCS also provides a CLI function to manage helm repo and charts.



11.5 Log collection

By default, **fluentd** is disabled during deployment of NCS. However, when enabled, **fluentd** integrates with NCS to route log files to an external elasticsearch log server.

11.5.1 Manage fluentd using CLI commands

Use the **ncs service fluentd** command to manage logging functionality.

- Use the following command to obtain the address of an elastic search server.

```
ncs service fluentd get
```

- Use the following command to start fluentd on all nodes. The Docker daemon reboots each node and the Docker log-driver is set to **fluentd**.

```
ncs service fluentd start
```

Example:

```
ncs service fluentd start --fluentd_es_host elasticsearch --  
fluentd_es_port 9200
```

- Use the following command to update fluentd on all nodes..

```
ncs service fluentd update
```

- Use the following command to stop fluentd on all nodes. The Docker daemon reboots each node and the Docker log-driver is set to **journald**.

```
ncs service fluentd stop
```

11.6 Certificate management



Note: The cert-manager API group has been changed to "cert-manager.io/v1alpha2".

NCS integrates cert-manager as an asset for managing certificates in a cluster. The cert-manager is a native Kubernetes controller to automate the management and issuance of TLS certificates from various issuing sources. Documentation for cert-manager can be found at docs.cert-manager.io. By default, one cluster issuer will be created using the ca and ca-key in /etc/openssl/, with the name: **ncms-ca-issuer**.

Applications can create their own certificates using the preloaded cluster issuer. The certificate will be saved in Kubernetes' secret storage.



Note: The default duration for all certificates is 1 year and the default renewal windows is 30 days before expiry. This means that certificates are considered valid for 12 months and renewal will be attempted within one month of expiry.

 **Note:** If duration is not predefined, then cert-manager will default to a duration of 30 days with a renewBefore of 30 days. When setting duration it is recommended to also set renewBefore, if renewBefore is longer than duration you will receive an error.

 **Note:** The *duration* and *renewBefore* parameters must usually be given in the golang format, which understands time duration only in h, m, s, ms, us, or ns. This is not applicable to all incoming certificates. Renewal is triggered (by default) before the certificate expires but can be triggered at will using the kubectl plugin.

Example:

```
apiVersion: v1
kind: Namespace
metadata:
  name: cert-manager-test
---
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: example-com
  namespace: cert-manager-test
spec:
  secretName: example-com-tls
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  commonName: example.com
  dnsNames:
    - example.com
    - www.example.com
  issuerRef:
    name: ncms-ca-issuer
    # We can reference ClusterIssuers by changing the kind here.
    # The default value is Issuer (i.e. a locally namespaced Issuer)
    kind: ClusterIssuer
```

Example of using the certificate in Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    service: {{ .Chart.Name }}
  name: {{ .Chart.Name }}
  namespace: cert-manager-test
spec:
```

```

replicas: 3
selector:
  matchLabels:
    name: {{ .Chart.Name }}
template:
  metadata:
    labels:
      name: {{ .Chart.Name }}
spec:
  containers:
    - name: {{ .Chart.Name }}
      image: {{ .Values.image.registry }}{{ .Values.image.name }}:
{{ .Values.image.tag }}
      imagePullPolicy: {{ .Values.image.pullPolicy }}

    ...
    ...
  volumeMounts:
    - name: cert-test-tls
      mountPath: {{ .Values.volumes.path.tls }}
      readOnly: true
  volumes:
    - name: cert-test-tls
      secret:
        secretName: cert-test-tls

```

Certificate conversion - JKS, PEM or P12 format

In case another TLS certificate format other than CRT is required by the application, the chart owner must use cert-converter as an *init-container* to do the conversion to the required format.

Every application should, during building process, add the cert-converter init-container image into their deployment if conversion to another format is required.

Cert converter source is present in Netguard Base repo and it is built and published, as docker image with all needed tools (e.g. OpenSSL), to Artifactory <http://repo.lab.pl.alcatel-lucent.com/security-docker-releases/common/certs-converter/>

Init container must:

- mount secret under CERT_INPUT_DIR
- mount volume (emptyDir) under CERT_OUTPUT_DIR to transfer certificate change format to desired one (CERT_DESIRED_FORMAT) and put output files in destination place (CERT_OUTPUT_DIR). It will always generate certificates with hardcoded file names:
 - keystore.jks and truststore.jks for JKS
 - tls.p12 for P12

- tls.pem for PEM

Component container must:

- mount the same volume (emptyDir) which is used to transfer certificateuse generated cert + trusted certificate:
 - For JKS: CERT_OUTPUT_DIR/keystore.jks and CERT_OUTPUT_DIR/truststore.jks
 - For PEM: CERT_OUTPUT_DIR/tls.pem
 - For P12: CERT_OUTPUT_DIR/tls.p12

Example of JKS format:

```
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  labels:  
    service: {{ .Chart.Name }}  
  name: {{ .Chart.Name }}  
spec:  
  selector:  
    matchLabels:  
      name: {{ .Chart.Name }}  
  template:  
    metadata:  
      labels:  
        name: {{ .Chart.Name }}  
    spec:  
      initContainers:  
        - name: {{ .Chart.Name }}-init  
          image: {{ .Values.global.registry }}  
{{ .Values.certsConverter.image.name }}:{{ .Values.certsConverter.image.tag }}  
          imagePullPolicy: {{ .Values.image.pullPolicy }}  
          command: ["certs_converter"]  
          args: [--certs-dir={{ .Values.volumes.path.certs }},  
                  --output={{ .Values.volumes.path.tls }},  
                  --format=JKS,  
                  --ks-pass=KEYSTORE_PASSWORD,  
                  --ts-pass=TRUSTSTORE_PASSWORD]  
      volumeMounts:  
        - name: CUSTOM_COMPONENT_NAME-tls  
          mountPath: {{ .Values.volumes.path.certs }}  
          readOnly: true  
        - name: {{ .Chart.Name }}-generated-certs  
          mountPath: {{ .Values.volumes.path.tls }}
```

```
containers:
  - name: {{ .Chart.Name }}
    image: {{ .Values.global.registry }}{{ .Values.image.name }}:
{{ .Values.image.tag }}
    imagePullPolicy: {{ .Values.image.pullPolicy }}

  ...

volumeMounts:
  - name: {{ .Chart.Name }}-generated-certs
    mountPath: {{ .Values.volumes.path.tls }}
    readOnly: true
volumes:
  - name: CUSTOM_COMPONENT_NAME-tls
    secret:
      secretName: CUSTOM_COMPONENT_NAME-tls
  - name: {{ .Chart.Name }}-generated-certs
    emptyDir: {}
```

Example of PEM format:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    service: {{ .Chart.Name }}
  name: {{ .Chart.Name }}
spec:
  selector:
    matchLabels:
      name: {{ .Chart.Name }}
  template:
    metadata:
      labels:
        name: {{ .Chart.Name }}
  spec:
    initContainers:
      - name: {{ .Chart.Name }}-init
        image: {{ .Values.global.registry }}
{{ .Values.certsConverter.image.name }}:{{ .Values.certsConverter.image.tag }}
        imagePullPolicy: {{ .Values.image.pullPolicy }}
        command: ["certs_converter"]
        args: [--certs-dir={{ .Values.volumes.path.certs }},
               --output={{ .Values.volumes.path.tls }}]
```

```

        "--format=PEM" ]

volumeMounts:
  - name: CUSTOM_COMPONENT_NAME-tls
    mountPath: {{ .Values.volumes.path.certs }}
    readOnly: true
  - name: {{ .Chart.Name }}-generated-certs
    mountPath: {{ .Values.volumes.path.tls }}

containers:
  - name: {{ .Chart.Name }}
    image: {{ .Values.global.registry }}{{ .Values.image.name }}:
{{ .Values.image.tag }}
    imagePullPolicy: {{ .Values.image.pullPolicy }}

  ...

volumeMounts:
  - name: {{ .Chart.Name }}-generated-certs
    mountPath: {{ .Values.volumes.path.tls }}
    readOnly: true

volumes:
  - name: CUSTOM_COMPONENT_NAME-tls
    secret:
      secretName: CUSTOM_COMPONENT_NAME-tls
  - name: {{ .Chart.Name }}-generated-certs
    emptyDir: {}

```

Example of P12 format:

```
--8<-- "bcmt_topics_operations/services_certificate_p12_sample.yaml"
```

Istio Ingress Gateway TLS support

It is possible to secure the Istio Ingress gateway with Cert-Manager. Istio documents this procedure at istio.io/docs/examples/advanced-gateways/ingress-certmgr/.

The procedure documented in this link will allow the same TLS certificate to be used for all Ingress resource provided by the Istio gateway. The Cert-manager will manage issuance and renewal of TLS certificates for the Istio Ingress gateway. It will also automatically detect and hot-swapped the certificates using the Istio secrets discovery service (SDS). The SDS server will push certificates to all Envoy instances to ensure certificates are updated and can provide TLS termination at the envoy sidecar injected for each component. This procedure requires setting up the ingress domain name and configuring the Istio Ingress gateway TLS certificates obtained from Cert-manager.

Application Helm charts will need to make sure they can support the domain name provided as part of configuring the SDS Istio domain name. Note the Ingress Domain attributes for dnsNames and domains.

Example:

```

apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: ingress-cert-staging
  namespace: istio-system
spec:
  secretName: ingress-cert-staging
  issuerRef:
    name: ncms-ca-issuer
    kind: ClusterIssuer
  commonName: $INGRESS_DOMAIN
  dnsNames:
  - $INGRESS_DOMAIN
  acme:
    config:
    - http01:
        ingressClass: istio
      domains:
      - $INGRESS_DOMAIN
  ---

```

Please refer to Istio Traffic Management document to find other information about Ingress and Egress requests <https://istio.io/v1.7/docs/tasks/traffic-management>

11.6.1 HA Configuration

Cert-Manager HA Configuration for NCS deployment

In NCS default deployment, Cert-manager has non-HA config. It means 1 cert-manger-controller pod, 1 cert-manager-cainjector pod and 1 cert-manager-webhook pod. During NCS deployment, If customer adds below configuration in bcmt_config.json, NCS will deploy 3 cert-manger-controller pods, 3 cert-manager-cainjector pods and 3 cert-manager-webhook pods.

```

"app": {
  "appv1_enabled": false,
  "chart_enabled": false,
  "chart_repo_url": {},
  "ncm_app_list": [
    "CBUR",
    "APP-API",
    "cert-manager",
    "fluentd"
  ],
  "ncm_app_values": {}
}

```

```

    "cert-manager": {
        "cainjector": {
            "replicaCount": "3",
            "extraArgs": [ "--leader-elect=true" ]
        },
        "webhook": {
            "replicaCount": "3"
        },
        "replicaCount": "3"
    },
    "fluentd": {},
    "CBUR": {},
    "APP-API": {}
}
}

```



Note: Refer only to the configuration related to cert-manager.

Cert-Manager HA Configuration for NCS upgrade

This procedure is only used for below case. Customer has deployed NCS N-1 version or NCS N-2 verison with cert-manager non-HA configuration and wants to use cert-manager HA config after upgrade NCS N version. Before NCS upgrade, customer could edit the file /opt/bcmf/config/csf-charts/cert-manager/all-values.yml on 3 control nodes. The file all-values.yml like below. only refer to the **replicaCount** information.

```

cainjector:
  tolerations: [{"effect": "NoExecute", "key": "is_control", "operator": "Equal",
  "value": "true"}, {"effect": "NoExecute", "key": "is_edge", "operator": "Equal",
  "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal",
  "value": "true"}, {"effect": "NoSchedule", "key": "node.cloudprovider.kubernetes.io/
  uninitialized", "operator": "Equal", "value": "true"}]
  enabled: true
  extraArgs:
  - --leader-elect=true
  replicaCount: '3'
replicaCount: '3'
webhook:
  tolerations: [{"effect": "NoExecute", "key": "is_control", "operator": "Equal",
  "value": "true"}, {"effect": "NoExecute", "key": "is_edge", "operator": "Equal",
  "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal",
  "value": "true"}, {"effect": "NoSchedule", "key": "node.cloudprovider.kubernetes.io/
  uninitialized", "operator": "Equal", "value": "true"}]
  enabled: true
  replicaCount: '3'

```

```
tolerations: [{ "effect": "NoExecute", "key": "is_control", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_edge", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal", "value": "true"}, {"effect": "NoSchedule", "key": "node.cloudprovider.kubernetes.io/uninitialized", "operator": "Equal", "value": "true"}]
```

Known Issues- Not Support Label

The current cert-manager does not have supported labels. After we add some labels in the cert-manager certificate, the cert-manager will not copy these labels in secret. There is a cert-manager feature to track it. [cert-manager Label feature](#)

Below is a clear example.

Certificate:

```
# kubectl describe certificate dc2-ccas-apache-client-cert
Name:           dc2-ccas-apache-client-cert
Namespace:      default
Labels:         app.kubernetes.io/instance=dc2
                app.kubernetes.io/managed-by=Tiller
                app.kubernetes.io/name=dc2-ccas-apache
                helm.sh/chart=ccas-apache-2.5.3-beta.593
Annotations:   <none>
API Version:  cert-manager.io/v1alpha2
Kind:          Certificate
Metadata:
  Creation Timestamp: 2020-06-15T13:44:25Z
  Generation:        1
  Resource Version:  15161399
  Self Link:         /apis/cert-manager.io/v1alpha2/namespaces/default/
                     certificates/dc2-ccas-apache-client-cert
  UID:              69212f0b-ad16-497e-8ed3-60ac8954ad19
Spec:
  Common Name:    dc2-ccas-apache-cassandra
  Duration:       8760h0m0s
  Issuer Ref:
    Kind: ClusterIssuer
    Name: ncms-ca-issuer
  Organization:
    Nokia
  Renew Before:   360h0m0s
  Secret Name:   dc2-ccas-apache-client-cert
  Usages:
    server auth
    client auth
Status:
```

Conditions:

```
Last Transition Time: 2020-06-15T13:44:26Z
Message: Certificate is up to date and has not expired
Reason: Ready
Status: True
Type: Ready
Not After: 2021-06-15T13:44:25Z
Events: <none>
```

Secret:

```
# kubectl describe secret dc2-ccas-apache-client-cert
Name: dc2-ccas-apache-client-cert
Namespace: default
Labels: <none>
Annotations: cert-manager.io/alt-names:
cert-manager.io/certificate-name: dc2-ccas-apache-client-cert
cert-manager.io/common-name: dc2-ccas-apache-cassandra
cert-manager.io/ip-sans:
cert-manager.io/issuer-kind: ClusterIssuer
cert-manager.io/issuer-name: ncms-ca-issuer
cert-manager.io/uri-sans:
Type: kubernetes.io/tls
Data
=====
ca.crt: 1257 bytes
tls.crt: 1229 bytes
tls.key: 1675 bytes
```

11.7 ETCD

ETCD is a consistent and highly-available key value store used by Kubernetes for storing all cluster data. Performance and stability of the cluster is sensitive to network and disk IO. Any resource starvation can lead to heartbeat timeout, causing cluster instability. An unstable etcd indicates that no leader is elected. Under such circumstances, a cluster cannot make any changes to its current state, which implies no new pods can be scheduled.

In the NCS cluster, etcd co-exists on the Control node, which means the node needs additional resources. In most of cases, etcd performance issue is caused by slow disk IO, **fio** could be used to test to the disk, and ETCD provides benchmark tool (<https://github.com/coreos/etcd/blob/master/Documentation/op-guide/performance.md>).

The default etcd storage size is 2GB, configurable with --quota-backend-bytes flag. And 8GB is a suggested maximum size for normal environments and ETCD warns at startup if the configured value exceeds it.

And we set the storage size based on the number of the cluster nodes. When the nodes number is less than 4, set -quota-backend-bytes to 2GB, and the number is between 4 and 10, set quota-backend-bytes to 4Gb, if the number larger than 10, set -quota-backend-bytes to 8GB.

If the storage usage exceeds 70%, an alarm will be fired in bcmt heartbeat. And also alarms generated when ETCD storage NOSPACE and CORRUPT.

ETCD on disk recovery procedure

Normally, there are following two kinds of recovery scenarios according to the unhealthy member numbers.

- If the cluster permanently loses less or equal than $(N-1)/2$ members then need go through scenario I for quick recovery;
- If the cluster permanently loses more than $(N-1)/2$ members then it disastrously fails, irrevocably losing quorum. Once quorum is lost, the cluster cannot reach consensus and therefore cannot continue accepting updates, and need go through scenario II for quick recovery.

The following example command could be referred to check the unhealthy member number and will be used several times during restore procedure, please replace endpoints with the exact value of ETCDCONTROL_ENDPOINTS in /etc/etcd/etcd.client.conf

```
ETCDCONTROL_API=3 etcdctl
  --
endpoints=https://172.16.1.133:2379,https://172.16.1.134:2379,https://172.16.1.135:2379
  --cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem
  --key=/etc/etcd/ssl/etcd-client-key.pem endpoint health
```

Scenario I:

For this scenario, need rejoin the etcd via removing the failed member and adding the failed member, and current /opt/bcmt/bin/start-etcd.sh script has covered such logic.

1. Log in to the failure node with root permission, check the current ETCD status.

Example command:

```
systemctl status etcd
```

2. Stop the ETCD service.

Example command:

```
systemctl stop etcd
```

3. Start the ETCD service.

Example command:

```
systemctl start etcd
```

4. Check the endpoints health status to make sure the previous failed member with healthy status now.

5. Repeat above steps on other remaining failure nodes.

Scenario II:

To recover from disastrous failure, etcd v3 provides snapshot backup and restore facilities to recreate the cluster without v3 key data loss.

1. Snapshotting the keyspace Recovering a cluster first needs a snapshot of the keyspace from an etcd member. A snapshot may either be taken from a live member with the etcdctl snapshot save command or by copying the member/snap/db file from an etcd data directory. For example, the following command snapshots the keyspace to the file snapshot.db under current directory, please replace the endpoints with the corresponding value:

Example command:

```
ETCDCTL_API=3 etcdctl --endpoints=https://172.16.1.135:2379 --cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem snapshot save snapshot.db
```

2. Scp the snapshot backup to other nodes for later restore usage, also change the ownership with root:root.

Example command:

```
scp -i 1.pem snapshot.db cloud-user@d8-g1-01:/home/cloud-user/  
chown root:root /home/cloud-user/snapshot.db
```

3. Stop the ETCD service on each node, and make sure every node in cluster is unhealthy status.

Example command:

```
systemctl stop etcd  
systemctl status etcd
```

4. Mount the current ETCD data dir and make the directory backup.

Example command:

```
mount | grep etcd  
umount /dev/sdd  
mount | grep etcd  
cd /data0  
mv etcd/ etcd.bak
```

5. Prepare for current parameters used during snapshot restore.

Example command:

```
egrep "name |data-dir |initial-cluster |initial-advertise-peer-urls" /opt/bcmt/bin/start-etcd.sh
```

6. Perform restore operation.

Example command:

```
ETCDCTL_API=3 etcdctl snapshot restore /home/cloud-user/snapshot.db \
--name d8-g1-03 \
--data-dir /data0/etc \
--initial-cluster d8-g1-01=https://172.16.1.133:2380,d8-
g1-02=https://172.16.1.134:2380,d8-g1-03=https://172.16.1.135:2380 \
--initial-advertise-peer-urls https://172.16.1.135:2380
```

7. Remount the etcd directory.

Example command:

```
mount -a
mount | grep etcd
```

8. Start the ETCD service, and check the endpoint status.

Example command:

```
systemctl start etcd
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/
etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/
etcd-client-key.pem member list
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/
etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/
etcd-client-key.pem endpoint health
```

9. Repeat Step 4 to Step 8 on the remaining nodes one by one.

10. Make sure all etcd members are with healthy status.

11.7.1 RAM ETCD

In a production environment, high performance disk is needed. But in some environment, disk performance is poor, which impact NCS cluster ETCD performance, and causes Kubernetes unstable. And in some poor disk performance development environment, NCS provide an option to put part of cluster ETCD data to memory to improve ETCD performance.

There are two ways to enable it:

- To enable ram etcd prior to NCS deployment, set "etcd_ram_enabled" to "true" in the bcmt_config.json configuration file.

```
"etcd_ram_enabled": true
```

- To enable ram etcd using CLI commands after installation, use the ncs service rametcd command.

```
ncs service rametcd <enable|disable>
```

If get error when run CLI command, we can enable this feature manually, run follow commands in controller node.

```
# docker exec -it `docker ps | grep bcmt-api | grep -v pause | awk '{print $1}'` bash
# python
# follow commands need run in python terminal
from ncms.commands.utils.bcmt import _bcmt_ctx, BCMT, b
_bcmt_ctx.bcmt = b = BCMT(type="task")
from ncms.commands.utils.bcmt import EtcdDict
from ncms.commands.utils.bcmt import raise_exception
b.ansible_run("commands/service/etcd-ram.yml -e run_hosts='role_control' -e operation_type=s2ram", debug=False)
bcmt_config = b.get_bcmt_config()
bcmt_config['cluster_config']['etcd_ram_enabled'] = True
exit()
```

11.7.1.1 Limitation

If RAM ETCD is enabled, the ETCD cluster could recover automatically during one or two control nodes reboot, but in the case of three nodes reboot/shutdown, the ETCD cluster will be down, though there is manual procedure to recover the ETCD cluster, there will be data lost, and the whole Kubernetes cluster may be crashed, therefore this capability is not recommended in production environment.

Below is the procedure to recover ETCD cluster when all control nodes reboot, naming the control nodes with control1(ip1), control2(ip2) and control3(ip3):

1. Backup the existing data directory, for example to “/” directory on each control:

```
cp -r /data0/etcd /etcd-user-backup
```

2. Choose control1 as etcd leader(or someone else), go to control1 /opt/bcmt/bin directory, restore as leader with below command:

```
cd /opt/bcmt/bin
./etcd-wal-util.sh recover_leader
```

Expected output:

```
2019-05-25 10:51:18.848965 I | mvcc: restore compact to 5210
2019-05-25 10:51:18.872803 I | etcdserver/membership: added member
5b8e47864c7f7c0b [https://ip1:2380] to cluster b45765ce9ca80acd
```

```
2019-05-25 10:51:18.872995 I | etcdserver/membership: added member
925179c2f38263a2 [https://ip2:2380] to cluster b45765ce9ca80acd
2019-05-25 10:51:18.873113 I | etcdserver/membership: added member
bcfe05357550d43b [https://ip3:2380] to cluster b45765ce9ca80acd
active
{"host": "control1", "level": "info", "log": {"message": "etcd db file copied to
ip2"}, "system": "daniel3", "systemid": "1234", "time": "2019-05-25T02:51:29.845Z", "type": "INFO"}
{"host": "control1", "level": "info", "log": {"message": "etcd db file copied to
ip3"}, "system": "daniel3", "systemid": "1234", "time": "2019-05-25T02:51:30.360Z", "type": "INFO"}
{"host": "control1", "level": "info", "log": {"message": "ETCD leader
restarted, please recover members on its host to bring up the etcd
cluster!"}, "system": "daniel3", "systemid": "1234", "time": "2019-05-25T02:51:30.372Z", "type": "INFO"}  


```

- 3.** If above step output contents "etcd db file copied", which mean the db file used to restore etcd member is already copied to the target host. Otherwise user need to copy the file /home/cloud-user/db to other control node into /home/cloud-user/ path manually.

- 4.** Go to control2 /opt/bcmt/bin directory, restore etcd member:

```
cd /opt/bcmt/bin
./etcd-wal-util.sh recover_member
```

Expected output:

```
2019-05-25 10:57:33.209809 I | mvcc: restore compact to 5210
2019-05-25 10:57:33.231915 I | etcdserver/membership: added member
5b8e47864c7f7c0b [https://ip1:2380] to cluster b45765ce9ca80acd
2019-05-25 10:57:33.232024 I | etcdserver/membership: added member
925179c2f38263a2 [https://ip2:2380] to cluster b45765ce9ca80acd
2019-05-25 10:57:33.232047 I | etcdserver/membership: added member
bcfe05357550d43b [https://ip3:2380] to cluster b45765ce9ca80acd
{"host": "control2", "level": "info", "log": {"message": "cluster or member is
unhealthy, still
checking"}, "system": "daniel3", "systemid": "1234", "time": "2019-05-25T02:57:36.327Z", "type": "INFO"}
member 5b8e47864c7f7c0b is healthy: got healthy result from https://
ip2:2379
{"host": "control2", "level": "info", "log": {"message": "ETCD member
recovered!"}, "system": "daniel3", "systemid": "1234", "time": "2019-05-25T02:57:46.406Z", "type": "INFO"}  


```

- 5.** If these two controls are restored successful, the left one will restore itself automatically, please check the status using the following command on each control:

```
ETCDCTL_API=3; etcdctl --endpoints=https://ip1:2379,https://
ip2:2379,https://ip3:2379 --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/
etcd/ssl/etcd-client-key.pem endpoint health
```

Expected output:

The output should include "is healthy".

```
https://ip1:2379 is healthy: successfully committed proposal: took =
25.740596ms
https://ip2:2379 is healthy: successfully committed proposal: took =
24.546723ms
https://ip3:2379 is healthy: successfully committed proposal: took =
27.092773ms
```

If the third ETCD is not healthy after serval minutes, please execute `systemctl stop etcd; rm -fr /mnt/etcd-ramdisk/*; systemctl start etcd` on the third control node, then check its endpoint health status again.

6. Clean up backup files on each control node after verify the Kubernetes cluster workable:

```
rm -fr /home/cloud-user/db /data0/etcd/member.bk /etcd-user-backup
```

11.7.2 ETCD backend storage size

ETCD has a flag `--quota-backend-bytes` to set the backend storage quota. When the NCS nodes number is less than 3, this flag will be set to 2GB, and the number is between 3 and 10, set to 4Gb, if the number larger than 10, set `-quota-backend-bytes` to 8GB.

An alarm will be triggered when the ETCD db usage exceeds 70% of the quota.

11.7.3 ETCD on disk recovery procedure

Normally, there are following two kinds of recovery scenarios according to the unhealthy member numbers.

- If the cluster permanently loses less or equal than $(N-1)/2$ members then need go through scenario I for quick recovery;
- If the cluster permanently loses more than $(N-1)/2$ members then it disastrously fails, irrevocably losing quorum. Once quorum is lost, the cluster cannot reach consensus and therefore cannot continue accepting updates, and need go through scenario II for quick recovery.

The following example command could be referred to check the unhealthy member number and will be used several times during restore procedure, please replace endpoints with the exact value of `ETCDCTL_ENDPOINTS` in `/etc/etcd/etcd.client.conf`.

```
ETCDCTL_API=3 etcdctl
--
endpoints=https://172.16.1.133:2379,https://172.16.1.134:2379,https://172.16.1.135:2379
--cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem
--key=/etc/etcd/ssl/etcd-client-key.pem endpoint health
```

Scenario 1

For this scenario, need rejoin the etcd via removing the failed member and adding the failed member, and current /opt/bcmt/bin/start-etcd.sh script has covered such logic.

1. Login to the failure node with root permission, check the current etcd status.

Example command:

```
systemctl status etcd
```

2. Stop etcd service.

Example command:

```
systemctl stop etcd
```

3. Start etcd service.

Example command:

```
systemctl start etcd
```

4. Check the endpoints health status to make sure the previous failed member with healthy status now.

5. Repeat the above steps on other remaining failure nodes.

Scenario 2

To recover from disastrous failure, etcd v3 provides snapshot backup and restore facilities to recreate the cluster without v3 key data loss.

1. Snapshotting the keyspace Recovering a cluster first needs a snapshot of the keyspace from an etcd member. A snapshot may either be taken from a live member with the etcdctl snapshot save command or by copying the member/snap/db file from an etcd data directory. For example, the following command snapshots the keyspace to the file snapshot.db under current directory, please replace the endpoints with corresponding value:

Example command:

```
ETCDCTL_API=3 etcdctl --endpoints=https://172.16.1.135:2379 --cacert=/etc/etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem snapshot save snapshot.db
```

2. scp the snapshot backup to other nodes for later restore usage, also change the ownership with root:root.

Example command:

```
scp -i 1.pem snapshot.db cloud-user@d8-g1-01:/home/cloud-user/
chown root:root /home/cloud-user/snapshot.db
```

3. Stop etcd service on each node, and make sure every node in cluster is unhealthy status.

Example command:

```
systemctl stop etcd  
systemctl status etcd
```

4. Umount the current etcd data dir and make the directory backup.

Example command:

```
mount | grep etcd  
umount /dev/sdd  
mount | grep etcd  
cd /data0  
mv etcd/ etcd.bak
```

5. Prepare for current parameters used during snapshot restore.

Example command:

```
egrep "name |data-dir |initial-cluster |initial-advertise-peer-urls" /opt/  
bcmt/bin/start-etcd.sh
```

6. Perform restore operation.

Example command:

```
ETCDCTL_API=3 etcdctl snapshot restore /home/cloud-user/snapshot.db \  
--name d8-g1-03 \  
--data-dir /data0/etcd \  
--initial-cluster d8-g1-01=https://172.16.1.133:2380,d8-  
g1-02=https://172.16.1.134:2380,d8-g1-03=https://172.16.1.135:2380 \  
--initial-advertise-peer-urls https://172.16.1.135:2380
```

7. Remount the etcd directory.

Example command:

```
mount -a  
mount | grep etcd
```

8. Start etcd service, and check the endpoint status.

Example command:

```
systemctl start etcd  
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/  
etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/  
etcd-client-key.pem member list
```

```
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/
etcd/ssl/ca.pem --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/
etcd-client-key.pem endpoint health
```

9. Repeat Step 4 ~ Step 8 on other remaining nodes one by one.

10. Make sure that all etcd members are in healthy status.

11.8 Registry server

For big size application (docker image size > 2G), it is easy to meet issue and take a lot of time when on-board docker image file.

NCS provides a pre-build registry solution to improve APP on board efficiency.

It is recommended to convert docker image file to docker registry file in advance this way, then merge docker registry file to bcmt-registry or create a registry server.

11.8.1 CLI commands

```
# ncs service registry-server
NAME:
  ncs service registry-server - registry-server commands

USAGE:
  ncs service registry-server command [command options] [arguments...]

COMMANDS:
  list      list all registry-server
  show      show registry-server
  add       add registry-server
  merge     merge registry-server
  delete    delete registry-server

OPTIONS:
  --help, -h  show help
```

12 Storage

1. From the local machine, launch a web browser (at the moment only Chrome has been tested).
2. Browse to the following address. Be sure to use **https** and not regular http.

```
https://<control server IP>:8082/#/home/storage
```

Example:

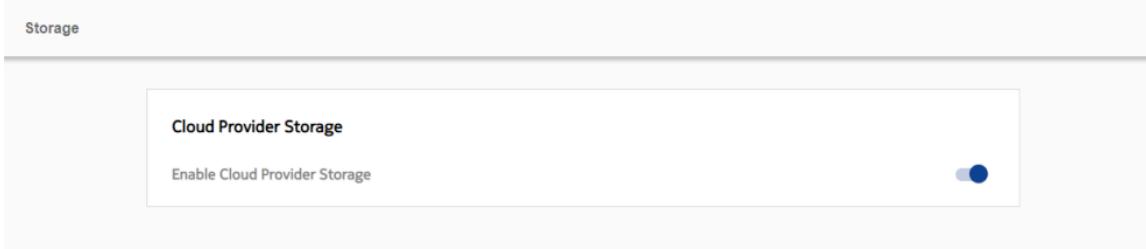
```
https://10.76.53.138:8082/#/home/storage
```



Note: For Baremetal deployment, please use port 8084 instead of 8082.

3. If you see the error "Your connection is not private", click on **Advanced** and choose to **Proceed to 10.76.53.138 (unsafe)**.
4. The Nokia NCS **STORAGE** page will be displayed, starting with the following.

General settings:



Generally, storage has several settings which user could specify configure.

a. **Cloud Provider Storage**, check cloud provider storage configuration.

5. Cloud Provider Storage configuration

General settings:

Cloud Provider Storage

Enable Cloud Provider Storage



In "**Cloud Provider Storage**", switch to enable or disable Cloud Provider Storage.

a. For enable scenario: ncs storage use-cloud-provider-storage enable

b. For disable scenario: ncs storage use-cloud-provider-storage disable

c. For get scenario: ncs storage use-cloud-provider-storage status

12.1 Overview

 **Warning:** Storage must be enabled during the initial NCS deployment. Storage can not be enabled after deployment.

A PersistentVolume (PV) object represents a piece of existing networked storage in the cluster that has been provisioned by an administrator. PVs are defined by a provisioner and StorageClass. PV access modes are:

- ReadWriteOnce (RWO) – the volume can be mounted as read-write by a single node.
- ReadOnlyMany (ROX) – the volume can be mounted read-only by many nodes.
- ReadWriteMany (RWX) – the volume can be mounted as read-write by many nodes.

A StorageClass provides a way for administrator to describe the different grouping of storage offerings. These can map to quality-of-service levels, back policies or any other arbitrary policies determined by the cluster administrator.

The provisioner determines which volume plugin is used for provisioning PVs.

A PersistentVolumeClaim (PVC) object represents a request for storage by a user or application.

NCS provides support for the following persistent storage options:

- Cinder - a volume management service for the OpenStack environment. Cinder provides higher volume transactions per second (TPS) based on infrastructure SLAs. Cinder only supports single pod attachment to the volume claim, and lacks any redundancy; host failure results in no access to the volume claim.
- vSphere - the vSphere Cloud Provider for Kubernetes enables Pods to use enterprise-grade persistent storage.
- Manila - is the OpenStack Shared Filesystems service for providing Shared Filesystems as a service. The method in which the share is provisioned and consumed is determined by the Shared File Systems driver, It can support multiple backend NFS, CIFS, HDFS, GlusterFS, CEPHFS, MAPRFS and other protocols as well. In b6mt, we now support to install nfs and cephfs csi backed driver for manila.
- EBS - AWS Elastic Block Store (EBS) is designed for application workloads that benefit from fine tuning for performance, cost and capacity.
- Rook - an open source cloud-native storage orchestrator that provides the platform and framework. Ceph is a highly scalable, distributed storage solution for block storage, object storage and shared file systems. Rook enables Ceph storage system to run on Kubernetes.
- GlusterFS - a distributed file system that aggregates disk storage resources from numerous environments, including OpenStack, VMware, AWS, Bare metal, or any provider that can supply an additional volume. GlusterFS provides redundancy via replication across multiple hosts, however, this results in a lower TPS. The shared file system allows multiple pods to attach with read/write access to the same volume claim.
- Azure - Azure Disks can be used to create a Kubernetes DataDisk resource. Azure Disks offer multiple performance options, such as Azure Premium storage, backed by high-performance SSDs, or Azure

Standard storage, backed by regular HDDs. Azure Disks are mounted as *ReadWriteOnce*, and are consequently only available for a single node. Azure Files should be used for storage volumes that must be accessed by multiple nodes simultaneously.

- Local storage - a local volume represents a mounted local storage device, such as disk, partition or directory. Local volumes can only be used as a statistically created PersistentVolume. Dynamic provisioning is not currently supported.

Table 34: Default StorageClass

Storage plugin	Platform	RWO	ROX	RWX	Storage type	Storage Class
Cinder	OpenStack	Y	-	-	Block storage	cinder-az-nova
Manila	OpenStack	Y	Y	Y	Shared file system	csi-manila-cephfs/csi-manila-nfs
vSphere	vCenter	Y	-	-	Block storage	vsphere
EBS	AWS	Y	-	-	Block storage	aws-gp2
Rook	Any	Y	Y	N	Block and object storage	rook-ceph-block-rep
GlusterFS	Any	Y	Y	Y	Shared file system	glusterfs-storageclass
Azure	Azure	Y	Y	Y	Azure Disk or Azure Files	azure-disk-premium azure-disk-standard
Local storage	Any	Y	-	-	File system	local-storage
CephCSI	Ceph cluster existence	Y	Y	Y	Block and file storage	csi-cephrbd/csi-cephfs



Note: For trouble shooting, if `df -Th` shows overlay2 volumes high percentage, the user can use the prune command as follows:

```
docker system prune --all --volumes
```

12.2 Software RAID

Software RAID is a form of **RAID** (redundant array of independent disks) performed on the internal server. **RAID** is a data protection method that spreads data on multiple hard disks, balancing overlapping I/O operations, improving performance and increasing the mean time between failures.

Advantages:

Unlike hardware RAID, software RAID uses the processing power of the operating system in which the RAID disks are installed. The cost is lower because no additional hardware RAID controller is required. It also permits users to reconfigure arrays without being restricted by the hardware RAID controller.

Disadvantages:

Software RAID tends to be slightly slower than hardware RAID. Since some processing power is taken by the software, read and write speeds of your RAID configuration, along with other operations carried out on the server can be slowed down by it. Software RAID is often specific to the operating system being used, so it cannot generally be used for partitions that are shared between operating systems.

Replacing a failed disk in the software RAID can be a little more complex. You have to first instruct your system to stop using the disk and then replace the disk.

Summary

Hardware RAID raid is preferable (and if a user has HW RAID it should be used instead of SW RAID).

12.2.1 Software RAID on an operating system device

The goal of this feature is to enable a SW RAID1 (Fake Raid) on the device of the operating system of Baremetal nodes. In SW RAID1, data is stored twice on two physical devices, which enables the continued operation in case of failure of one device.

SW RAID1 can be configured for a management node by changing the configuration text file that is patched to the `bootcd` and can be configured for all other cluster nodes by marking a check box that is shown per host group in the NCS Manager (Baremetal deployment).

The physical devices of the SW RAID1 array are determined by the hardware template files (similarl to the root device) but can be overridden.

In case of a physical device failure, a utility called, `raid_utils_ncs.py`, will help to replace a new device and add it to the SW RAID1 array.

The following is the general guideline for RAID configuration for all nodes:

1. For operating system disks:

- When HW RAID controller is available, it should be used to configure RAID 1 on operating system disks.
- When HW RAID controller does not exist, RAID 1 on operating system disks can be achieved by configuring SW RAID.

2. For storage nodes disks:

- When JBOD controller exists - HW RAID 0 should be configured separately for each individual disk from the JBOD array.

12.2.1.1 Enable Software RAID1 on a management node

To enable SW RAID1 on a management node, edit the configuration text file that is patched to the `bootcd.iso` and add:

```
IS_SW_RAID=true
```

The physical devices from which SW RAID1 are created does not have to be specified. By default, they are determined by the hardware template files.

If there is a need to change the this default – it can be done by setting the physical devices in `RAID_PRIMARY_ROOT_DEVICE` and `RAID_SECONDARY_ROOT_DEVICE` parameters in the patched configuration text file. The overridden device can be given by name (e.g. `/dev/sda`) or by-path (e.g. `/dev/disk/by-path/pci-0000:00:1f.2-ata-3.0`)

For example:

```
RAID_PRIMARY_ROOT_DEVICE= /dev/disk/by-path/pci-0000:00:1f.2-ata-3
RAID_SECONDARY_ROOT_DEVICE= /dev/disk/by-path/pci-0000:00:1f.2-ata-4
```

Example for a patched configuration text file:

```
DEV=ens3f0
VLAN=94
IP=10.75.239.164/27
DGW=10.75.239.161
NAMESERVER=135.239.25.18
ISO_URL="https://repo3.cci.nokia.net/cbis-generic-andidates/
cbis_vlab_repo/20.100.1/cbis/1177/ncs.iso"
IS_SW_RAID=true
```

12.2.1.2 Enable Software RAID1 for other nodes

SW RAID1 can be enabled per host group from baremetal gui manager.

To enable raid

- Select the host group for which SW RAID1 is required

- Check "Enable SW RAID1"

On enabling SW raid, "Select RAID1 primary OS root device hint" and "Select RAID1 secondary OS root device hint" text boxes are shown.

If they are left empty, RAID devices are determined by default from the hardware template files.

If you want to override these defaults, you can fill the required devices, where the device may be given by name (e.g. /dev/sda) or by-path (e.g. /dev/disk/by-path/pci-0000:00:1f.2-ata-3.0)

12.2.1.2.1 SW RAID1 devices should not conflict with other settings

Conflict Resolution

1. When enabling SW RAID1 on host group with "storage" role- make sure that SW RAID1 devices are not used for OSDs.
2. When enabling SW RAID1 on host group with "worker" role- make sure that SW RAID1 devices are not used for local persistent storage devices.
3. When setting **ELK** Disk (in "Resources" tab, "ELK Optional Configuration" section): if elk device conflicts with one of the SW RAID1 devices then elk will go to the root partition.

Example 1:

On a 'Master' role node, *sda* and *sdb* compose the SW RAID1 array, md127, and the root partition is located on md127p3. If the user also decides to locate elk on *sdb*, then here, elk will go to the root partition, md127p3, and not to *sdb* because *sdb* is part of the SW RAID1 array.

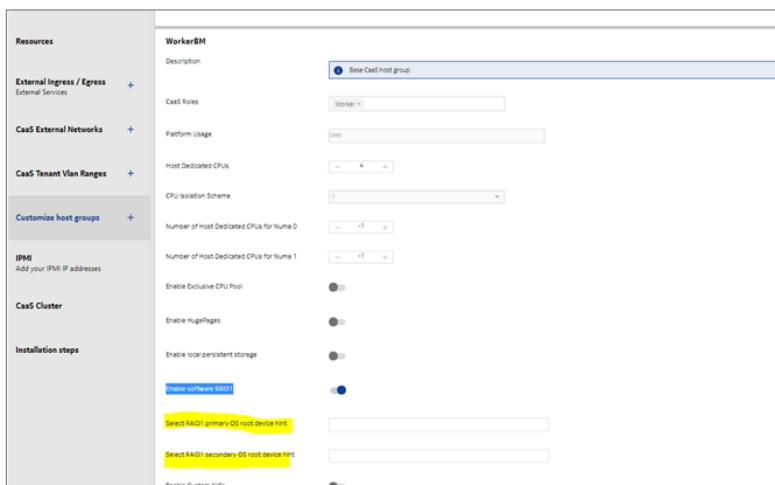
When setting the **Etc_d** partition (the **Etc_d** partition is set per host group with "Master" role): if the **Etc_d** partition conflicts with one of the SW RAID1 devices then the **Etc_d** will go to the 10G partition of the SW RAID1 device.

Example 2:

If on a 'Master' role node, *sda* and *sdb* compose the SW RAID1 array, md127, and the 10G partition is located on md127p1. If the user also decides to locate elk on *sdb*, then here the **Etc_d** will go to md127p1 partition, and not to *sdb* because *sdb* is part of the SW RAID1 array.

SW RAID1 installed on all nodes of the WorkerBM host group

Here, the SW RAID1 will be installed on all nodes of the WorkerBM host group and the physical devices will be determined according to the appropriate hardware template file, as shown here in this picture.



See [Nokia Container Services Manager Reference Guide for Bare Metal implementation](#).

12.2.1.3 Post deployment of Software RAID

After deployment, the `lsblk` output looks for an example as follows (Note that md device name may be different):

```
[root@cluster-manager-0 demo]# lsblk
NAME      MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
sdb        8:16   0 447.1G  0 disk
└─sdb2     8:18   0 447.1G  0 part
  └─md2     9:2    0 447G   0 raid1
    ├─md2p3  259:2  0 337G   0 md   /
    ├─md2p1  259:0  0 10G    0 md
    └─md2p2  259:1  0 100G   0 md
      └─vg_root-_data0 253:0  0 100G   0 lvm   /data0
└─sdb1     8:17   0 1M    0 part
sr0       11:0   1 137.7M  0 rom
sda        8:0    0 447.1G  0 disk
└─sda2     8:2    0 447.1G  0 part
  └─md2     9:2    0 447G   0 raid1
    ├─md2p3  259:2  0 337G   0 md   /
    ├─md2p1  259:0  0 10G    0 md
    └─md2p2  259:1  0 100G   0 md
      └─vg_root-_data0 253:0  0 100G   0 lvm   /data0
└─sda1     8:1    0 1M    0 part
sr1       11:1   1 1024M  0 rom
```

Implementation note:

Each disk is partitioned into 2 parts. The first is the boot partition and is not part of the RAID array and the second partition of each one of the 2 devices is part of the SW RAID1 array.

12.2.1.4 Replace OS drive when Software RAID1 is configured

12.2.1.4.1 Identify a failed drive

Drive failure may be identified by I/O error messages in `/var/log/messages`, or `/var/log/syslog`.

For example Buffer I/O error on device sda, logical block 1

You can use the `- smartctl` utility to check device health.



Note: `cat /proc/mdstat` may show that RAID1 is in degraded state.

12.2.1.4.2 Replacing a failed drive

Use this procedure to recover the M.2 drive which hosts the operating system. It is a software RAID and is configured on the DL380 storage node.

1. Log in to the host with the failed drive as root.
2. The following script utility helps to remove a failed drive from an existing SW RAID1 array and to add a new replaced drive to the SW RAID1 array - `/usr/lib/python2.7/site-packages/cbis_common/cbis_storage/tools/raid_utils_ncs.py`.
3. To see the utility options, run:

```
python /usr/lib/python2.7/site-packages/cbis_common/cbis_storage/tools/raid_utils_ncs.py -h
```

4. Remove the failed drive from the SW RAID1 array by using the `action_remove_raid_disk` option.
 - Give the drive you want to remove as a parameter.

In the following example `sdb` is the failed drive that should be replaced.

```
[root@cluster-manager-0 ~]# python /usr/lib/python2.7/site-packages/cbis_common/cbis_storage/tools/raid_utils_ncs.py action_remove_raid_disk /dev/sdb
-----
action_remove_raid_disk
Are you sure you want to remove /dev/sdb2 from /dev/md2? (y/n) y
Successfully removed /dev/sdb2 from /dev/md2
```

5. After removing the failed drive from the array, `cat /proc/mdstat` should show a degraded RAID1 array.

Example:

```
[root@cluster-manager-0 ~]# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 sda2[0]
        468716544 blocks super 1.2 [2/1] [U_]
```

```
bitmap: 3/4 pages [12KB], 65536KB chunk
```

```
unused devices: <none>
```

6. Physically extract the failed drive.
7. If the failed drive was `/dev/sda`, then move the working drive at `/dev/sdb` to `/dev/sda` where the faulty drive was.
8. Insert a new empty drive of the same size in the socket now empty.
9. Although the new drive is probably already clean, you can use the `action_erase_disk` option from the `raid_utils_ncs.py` utility to clean the device if it is by chance not cleaned.

Example:

```
[root@cluster-manager-0 ~]# python /usr/lib/python2.7/site-packages/cbis_common/
cbis_storage/tools/raid_utils_ncs.py action_erase_disk /dev/sdb
-----
action_erase_disk
All data on /dev/sdb will be deleted. Are you sure? (y/n)y

/dev/sdb was erased.
```

10. Add a new device to the existing RAID1 array by using the `action_add_raid_disk` option and give the device a name that should be added as parameter.



Note:

- This action takes time for synchronizing the devices.
- It is important to run the action from a persistent session like tmux and not to stop it in the middle.

Example:

```
[root@cluster-manager-0 ~]# python /usr/lib/python2.7/site-packages/cbis_common/
cbis_storage/tools/raid_utils_ncs.py action_add_raid_disk /dev/sdb
-----
action_add_raid_disk
Adding a new disk to RAID array takes time and requires the disks to finish syncing.
Are you sure you run on a persistent session like 'tmux'? (y/n)y
Start adding /dev/sdb to RAID array /dev/md2 (existing disk: /dev/sda)
Cleaning /dev/sdb
Preparing new disk partitions of /dev/sdb
Adding /dev/sdb to /dev/md2
Syncing /dev/md2 started. This will take a while...
Grub install on /dev/sdb
Grub install on /dev/sda
Successfully added /dev/sdb to RAID array
```

11. After successfully adding a new device, /etc/mdstat looks for example as follows:

```
[root@cluster-manager-0 ~]# cat /proc/mdstat
Personalities : [raid1]
md2 : active raid1 sdb2[2] sda2[0]
        468716544 blocks super 1.2 [2/2] [UU]
          bitmap: 3/4 pages [12KB], 65536KB chunk
```

12.3 Cinder

In an OpenStack environment, Cinder provides volume management services. A user can request volumes from it for storage purposes via the Cinder API. If NCS is deployed on VMs in OpenStack, pods in Kubernetes can request Cinder volumes to store data. Data stored on the volume can be persistent during the Pod's life cycle.

Cinder CSI Migration is enabled in the kubernetes feature gate. CSI Migration enables the replacement of existing in-tree storage plugins kubernetes.io/cinder with the cinder csi driver. Kubernetes end users don't notice a difference, which means the volume claimed by in-tree storageclass will be handled by CSI plugin transparently.

Limitation: The cinder CSI doesn't support the cinder API v2. For cinder v2, we still use the in-tree cloud provider plugin.

Use this procedure to configure Cinder prior to NCS deployment.

1. Access the `user_input.yml` file and specify all of the OpenStack environmental variables. For more information, see [Appendix user_input.yml](#).
2. Access the `bcmt_config.json` file.
3. Set the `bcmt_config.json` file parameter **provider** to **openstack**.
4. Set the `bcmt_config.json` file parameter **use_cloud_provider_storage** to **true**.

12.3.1 Cinder usage example

1. Create a Kubernetes yaml file to define the StorageClass.

Example file:

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: cinder-storageclass
provisioner: kubernetes.io/cinder
```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
---  
kind: PersistentVolumeClaim  
apiVersion: v1  
metadata:  
  name: nginx-pv-claim  
spec:  
  storageClassName: cinder-storageclass  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 1Gi
```

3. Create a Kubernetes yaml file to define a pod, and mount the PVC in the definition of the pod.

Example file:

```
---  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: nginx  
  namespace: default  
  labels:  
    app: nginx  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 10%  
      maxUnavailable: 0  
  selector:  
    matchLabels:  
      app: nginx  
  replicas: 1  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      volumes:  
        - name: nginx-pv-storage  
          persistentVolumeClaim:  
            claimName: nginx-pv-claim
```

```
containers:
- name: nginx
  image: registry-proxy:5000/library/nginx:alpine
  imagePullPolicy: Always
  volumeMounts:
  - mountPath: "/mnt/cinder"
    name: nginx-pv-storage
  dnsPolicy: Default
```

12.3.2 PVC resize

 **Note:** Kubernetes only support offline PVC resize for Cinder. Kubernetes do not support expand PVC size by edit StatefulSet.volumeClaimTemplates.

The procedure below describes offline resizing.

1. Create a Kubernetes yaml file to define the StorageClass with **allowVolumeExpansion** is true.

Example file:

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: cinder-az-nova
provisioner: kubernetes.io/cinder
parameters:
  availability: nova
allowVolumeExpansion: true
```

2. Define the PersistentVolumeClaim and pods, see *Cinder usage example* on page 298.

3. Scale Kubernetes deployment to 0 replicas using the following command:

```
kubectl scale deployment <deployment> --replicas 0
```

4. Edit PVC object and specify a larger size.

```
kubectl edit pvc <pvc_name\>
```

5. Re-restart Pods to finish PVC resizing.

Command:

```
kubectl scale deployment <deployment> --replicas <replicas\>
```

Example command:

```
kubectl scale deployment nginx-deployment --replicas 3
```

12.3.3 Snapshot create and store

Snapshot is supported when cinder csi is installed and snapshot is enabled by default. As cinder csi not support cinder API2, so Snapshot not supported in Cinder API 2.

After deployment default VolumeSnapshotClass is created.

```
# kubectl get VolumeSnapshotClass
NAME          AGE
cinder-csi-snapclass  11d
```

For Snapshot Creation and Volume Restore, please follow below steps:

1. Create the PVC and pods using the cinder storageclass.
2. Verify that pvc is bounded. The volume to create a snapshot must be in available status.

```
# kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY      ACCESS   MODES   STORAGECLASS      AGE
cinderpvc-cinfo-cinder-0   Bound    pvc-ba40f3cc-3586-4b2a-a8df-
f872d7bd0f52   2Gi     RWO        cinder-az-nova   4m
```

3. Create a snapshot for the PVC.

Example file:

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: snapshot-test
spec:
  snapshotClassName: cinder-csi-snapclass
  source:
    name: cinderpvc-cinfo-cinder-0
    kind: PersistentVolumeClaim
```

4. Apply the snapshot:

```
# kubectl apply -f snapshotpvc.yml
```

5. Verify that snapshot is created.

```
# kubectl get volumesnapshot
NAME          AGE
snapshot-test  2m54s
```

6. Restore the volume from the snapshot.

Example file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: snapshot-test-restore
spec:
  storageClassName: cinder-az-nova
  dataSource:
    name: snapshot-test
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

7. Apply the PVC:

```
# kubectl apply -f snapshotrestore.yml
```

8. Verify the volume from snapshot is created.

```
# kubectl get pvc
NAME                  STATUS   VOLUME
CAPACITY  ACCESS MODES  STORAGECLASS   AGE
cinderpvc-cinfo-cinder-0  Bound    pvc-ba40f3cc-3586-4b2a-a8df-
f872d7bd0f52  2Gi     RWO        cinder-az-nova  20m
snapshot-test-restore       Bound    pvc-db10e969-1b14-4da9-
afba-516b7b642e48  2Gi     RWO        cinder-az-nova  3s
```

12.4 Manila

The CSI Manila driver is able to create and mount OpenStack Manila shares. Snapshots and recovering shares from snapshots is supported as well (support for CephFS snapshots will be added soon). CSI Manila does not care about the origin of a share. As long as the share protocol is supported.

The CSI Manila driver deals with the Manila service only. All node-related operations (attachments, mounts) are performed by a dedicated CSI Node Plugin, to which all Node Service RPCs are forwarded. This means that the operator is expected to already have a working deployment of that dedicated CSI Node Plugin.

Currently we integrated two kinds of CSI Node Plugins, NFS and Cephfs. User can choose to install one or both of them by enabling different configs.

Use this procedure to configure Manila prior to NCS deployment.

1. Access the `user_input.yml` file and specify all of the OpenStack environmental variables. For more information, see *Appendix: user_input.yml* in the *Nokia Container Services OpenStack Installation Guide*.
2. Access the `bcmt_config.json` file.
3. Set the `bcmt_config.json` file parameter `provider` to **openstack**.
4. If the Openstack manila backend driver is Cephfs, set
`storage_csi_config.manila_cephfs_enabled` to true, and set the `share_type` in
`storage_csi_config.manila_cephfs_config`.

If the backend driver is NFS, set `storage_csi_config.manila_nfs_enabled` to true, and set the `share_type` in `storage_csi_config.manila_nfs_config`.

```
"storage_csi_config": {
    "manila_cephfs_enabled": true,
    "manila_cephfs_config": {
        "share_type": "cephfsnativetype"
    },
    "manila_nfs_enabled": true
    "manila_nfs_config": {
        "share_type": "cephfsnfstype"
    },
}
```

12.4.1 Manila usage example

1. After the deployment, storageclass `csi-manila-cephfs` is created when `manila_cephfs_enabled` is enabled. And storageclass `csi-manila-nfs` is created when `manila_nfs_enabled` is enabled.

```
# kubectl get sc
NAME                      PROVISIONER
RECLAIMPOLICY  VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
csi-manila-cephfs          cephfs.manila.csi.openstack.org Delete
                           Immediate           false                  23h
csi-manila-nfs             nfs.manila.csi.openstack.org  Delete
                           Immediate           false                  23h
```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim, taken nfs for example.

Example file:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nginx-pv-claim
spec:
  storageClassName: csi-manila-nfs
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

3. Create a Kubernetes yaml file to define a pod, mount the PVC in the definition of the pod.

Example file:

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 0
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      volumes:
        - name: nginx-pv-storage
          persistentVolumeClaim:
            claimName: nginx-pv-claim
```

```

containers:
- name: nginx
  image: registry-proxy:5000/library/nginx:alpine
  imagePullPolicy: Always
  volumeMounts:
    - mountPath: "/mnt/nfs-manila"
      name: nginx-pv-storage
  dnsPolicy: Default

```

12.5 vSphere

In an VMware environment, vSphere provides volume management services. A user can request volumes from it for storage purposes via the vSphere cloud provider API. If NCS is deployed on VMs in vCenter, pods in Kubernetes can request vSphere volumes to store data. Data stored on the volume can be persistent during the Pod's life cycle.

Use this procedure to configure vSphere prior to NCS deployment.

1. Access the `user_input.yaml` file and specify all of the VMware environmental variables. For more information, see [Appendix user_input.yaml](#).
2. Access the `bcmt_config.json` file.
3. Set the `bcmt_config.json` file parameter `provider` to `vcenter`.
4. Set the `bcmt_config.json` file parameter `use_cloud_provider_storage` to `true`.

12.5.1 Permissions

- The vSphere user designated to the vSphere Cloud Provider should have the permissions defined in [vsphere-storage-for-kubernetes: Permissions](#)
- From kubernetes 1.12.3, get zones is supported, this need the vcenter account has read privilege on the input project, refer to [cis tagging tag association: list attached tags](#)

12.5.2 vSphere usage example

1. Create a Kubernetes yaml file to define the StorageClass.

Example file:

```

---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: vsphere-storageclass
provisioner: kubernetes.io/vsphere-volume

```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
---  
kind: PersistentVolumeClaim  
apiVersion: v1  
metadata:  
  name: nginx-pv-claim  
spec:  
  storageClassName: vsphere-storageclass  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 1Gi
```

3. Create a Kubernetes yaml file to define a pod, and mount the PVC in the definition of the pod.

Example file:

```
---  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: nginx  
  namespace: default  
  labels:  
    app: nginx  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 10%  
      maxUnavailable: 0  
  selector:  
    matchLabels:  
      app: nginx  
  replicas: 1  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      volumes:  
        - name: nginx-pv-storage  
          persistentVolumeClaim:  
            claimName: nginx-pv-claim
```

```

containers:
- name: nginx
  image: registry-proxy:5000/library/nginx:alpine
  imagePullPolicy: Always
  volumeMounts:
    - mountPath: "/mnt/vsphere"
      name: nginx-pv-storage
  dnsPolicy: Default

```

12.5.3 Known issues and troubleshooting

During an upgrade, if a pod does not come up due to Multi-Attach error for RWO (Block) volume, do the following:

1. Find the Node VM in the vCenter Inventory. Make sure the correct VM associated with the Node is used for further instructions.
2. Detach all Persistent Volumes Disk attached to this Node VM.

 **Note:** Do not detach Primary disks used by the Guest OS.

 - a. Right-click a virtual machine in the inventory and select Edit Settings.
 - b. From Virtual Hardware find all Hard Disks for Persistent Volumes and remove them. (Do not select: Delete files from datastore)
 - c. Click on OK to reconfigure VM to detach all Persistent Volumes disks from shutdown/poweroff Node VM.
3. Execute `kubectl get volumeattachments` and find all volumeattachments objects associated with shutdown Node VM.
4. Edit volumeattachment object with `kubectl edit volumeattachments <volumeattachments-object-name>` and remove finalizers.
5. Check if the volumeattachment object is deleted by Kubernetes. If this object remains on the system, you can safely delete this with `kubectl delete volumeattachments <volumeattachments-object-name>`.
6. Wait for some time for pod to come up on a new node.

12.6 EBS

Amazon Elastic Block Store provides persistent block storage volumes for use with Amazon EC2 instances in the AWS environment.

AWS CSI Migration is enabled in the kubernetes feature gate. CSI Migration enables the replacement of existing in-tree storage plugins `kubernetes.io/aws-ebs` with the `aws disk csi` driver. Kubernetes end users

don't notice a difference, which means the volume claimed by in-tree storageclass will be handled by CSI plugin transparently.

Use this procedure to configure EBS prior to NCS deployment.

1. Access the `user_input.yml` file and specify all of the AWS environmental variables. For more information, see [Appendix user_input.yml](#).
2. Access the `bcmt_config.json` file.
3. Set the `bcmt_config.json` file parameter **provider** to **aws**.
4. Set the `bcmt_config.json` file parameter **use_cloud_provider_storage** to **true**.

12.6.1 EBS usage example

1. Create a Kubernetes yaml file to define the StorageClass.

Example file:

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: aws-gp2
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  provisioner: kubernetes.io/aws-ebs
  volumeBindingMode: WaitForFirstConsumer
parameters:
  fsType: xfs
  type: gp2
```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvcsc001
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  storageClassName: aws-gp2
```

3. Create a Kubernetes yaml file to define a pod, and mount the PVC in the definition of the pod.

Exmple file:

```
---  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: nginx  
  namespace: default  
  labels:  
    app: nginx  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 10%  
      maxUnavailable: 0  
  selector:  
    matchLabels:  
      app: nginx  
  replicas: 1  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      volumes:  
      - name: nginx-pv-storage  
        persistentVolumeClaim:  
          claimName: pvcsc001  
      containers:  
      - name: nginx  
        image: registry-proxy:5000/library/nginx:alpine  
        imagePullPolicy: Always  
        volumeMounts:  
        - mountPath: "/mnt/ebs"  
          name: nginx-pv-storage  
      dnsPolicy: Default
```

12.7 Azure Disk

In an Azure environment, Azure Disks can be used to create a Kubernetes DataDisk resource. Azure Disks offer multiple performance options, such as Azure Premium storage, backed by high-performance SSDs, or Azure Standard storage, backed by regular HDDs. For most production and development workloads,

Azure Premium storage should be used. Azure Disks are mounted as *ReadWriteOnce*, and are consequently only available for a single node. Azure Files should be used for storage volumes that must be accessed by multiple nodes simultaneously.

Azure Disk CSI Migration is enabled in the kubernetes feature gate. CSI Migration enables the replacement of existing in-tree storage plugins kubernetes.io/azure with the azure disk csi driver. Kubernetes end users do not notice a difference, which means the volume claimed by in-tree storageclass will be handled by CSI plugin transparently.

Limitation: Does not support CSI migration.

Use this procedure to configure Azure storage prior to NCS deployment.

1. Access the `user_input.yaml` file and specify the all of the Azure environmental variables.
2. Access the `bcmt_config.json` file.
3. Set `provider` to "azure".
4. Set `use_cloud_provide_storage` to "true".
5. This feature is ready for deployment.

End of steps: this procedure is complete.

12.7.1 Azure Disk usage example

After deployment, two Azure Disk StorageClasses are created by default:

- `azure-disk-premium`
- `azure-disk-standard`

1. Create a Kubernetes yaml file to define the StorageClass.

Example file:

```
---  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: azure-disk-standard  
  annotations:  
    storageclass.kubernetes.io/is-default-class: "true"  
  provisioner: kubernetes.io/azure-disk  
parameters:  
  storageaccounttype: Standard_LRS  
  kind: Shared
```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
--  
kind: PersistentVolumeClaim  
apiVersion: v1  
metadata:  
  name: pvcsc001  
spec:  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 2Gi  
  storageClassName: azure-disk-standard
```

3. Create a Kubernetes yaml file to define a pod, and mount the PVC in the definition of the pod.**Exmple file:**

```
kind: Pod  
apiVersion: v1  
metadata:  
  name: mypod  
spec:  
  containers:  
    - name: mypod  
      image: nginx:1.15.5  
    resources:  
      requests:  
        cpu: 100m  
        memory: 128Mi  
      limits:  
        cpu: 250m  
        memory: 256Mi  
    volumeMounts:  
    - mountPath: "/mnt/azure"  
      name: volume  
  volumes:  
  - name: volume  
    persistentVolumeClaim:  
      claimName: azure-disk-standard
```

12.8 Rook

Rook is an open-source cloud-native storage orchestrator for Kubernetes. Rook uses Ceph to provide file, block, and object storage. Rook has two primary components: a Rook operator and Rook agents. The Rook operator is responsible for the management of the Ceph clusters and creating Rook agents, which are pods that are deployed on each Kubernetes node. The Rook agent handles tasks such as attaching network storage devices, and mounting volumes.

NCS creates a Rook Operator and a Ceph cluster based on the inventory input. As a result, a StorageClass named *rook-ceph-block-rep* is created. Users can reference this StorageClass in their PersistentVolumeClaim.

Use this procedure to configure Rook prior to NCS deployment.

1. Prepare dedicated raw devices on the cluster nodes.
2. Access the `user_input.yaml` file and specify the raw devices using the **`rook_storage_devices`** parameter.



Note: Minimum value of ROOK device is 10 G.

3. Access the `bcmt_config.json` file.

4. Set the **`k8s_rook_ceph`** parameter to *true*.



Note: CSTO ROOK do not support multi-cluster.

5. This feature is ready for deployment.



Warning: Suggest to config at least 3 rook storage nodes when use rook default config. If rook storage node number is less than 3, it may cause deploy failure or ceph cluster `HEALTH_WARN` issue and impact future upgrade.



Note: Since CSTO ROOK uses common device list and do not support define device name per node in cephcluster, suggest to use same rook device name for all rook storage nodes. If user don't use default rook config and wants to specify some rook parameters, please read CSTO user guide and config rook_settings in `bcmt_config`.

12.8.1 Verification

Check rook pods are running in namespace `rook-ceph`, and `cephcluster` status is `HEALTH_OK`:

```
# kubectl get pod -n rook-ceph
  NAME                               READY   STATUS
  RESTARTS   AGE
  ceph-oam-798f85dcf5-6hdty         2/2     Running   0
  57s
```

csi-rbdplugin-28gqf 9m44s	3 / 3	Running	0
csi-rbdplugin-6wrvx 9m44s	3 / 3	Running	0
csi-rbdplugin-c6zcr 9m44s	3 / 3	Running	0
csi-rbdplugin-fbk78 9m44s	3 / 3	Running	0
csi-rbdplugin-gdnjp 9m44s	3 / 3	Running	0
csi-rbdplugin-provisioner-78476f6574-jr8wc 9m44s	5 / 5	Running	0
csi-rbdplugin-provisioner-78476f6574-t8vlx 9m44s	5 / 5	Running	0
csi-rbdplugin-vvsgz 9m44	3 / 3	Running	0
rook-ceph-mgr-a-57fc7589fc-n65f7 7m36s	1 / 1	Running	0
rook-ceph-mon-a-db5f8c4d7-nngdc 8m44s	1 / 1	Running	0
rook-ceph-mon-b-6bd98779f5-vbqgj 8m24s	1 / 1	Running	0
rook-ceph-mon-c-7d86b964fc-chd7b 8m6s	1 / 1	Running	0
rook-ceph-operator-6d6dfffc576-zkrrv 9m59s	1 / 1	Running	0
rook-ceph-osd-0-7656f658d6-nvp46 42s	1 / 1	Running	0
rook-ceph-osd-1-59df7cd7f5-v25k5 5m51s	1 / 1	Running	0
rook-ceph-osd-2-6967c458b5-8b6tj 5m46s	1 / 1	Running	0
rook-ceph-osd-3-7f9755c9cf-7c52q 70s	1 / 1	Running	0
rook-ceph-osd-4-554b8bbbb8-kk7dx 4m15s	1 / 1	Running	0
rook-ceph-osd-5-85fd455b76-vrhfc 3m45s	1 / 1	Running	0
rook-ceph-osd-prepare-dennis-1912-control-01-6r5d7 96s	0 / 1	Completed	0
rook-ceph-osd-prepare-dennis-1912-control-02-xfjr4 94s	0 / 1	Completed	0
rook-ceph-osd-prepare-dennis-1912-control-03-48tbs 92s	0 / 1	Completed	0

```

rook-ceph-osd-prepare-dennis-1912-edge-01-mgjh6      0/1    Completed   0
          89s
rook-ceph-osd-prepare-dennis-1912-worker-01-zhxhf    0/1    Completed   0
          87s
rook-ceph-osd-prepare-dennis-1912-worker-02-6b7rr    0/1    Completed   0
          85s
rook-ceph-rgw-rook-ceph-store-a-55b49c478f-clq4p     1/1    Running    0
          102s
rook-discover-56z27                                  1/1    Running    0
          9m54s
rook-discover-6zpn5                                  1/1    Running    0
          9m54s
rook-discover-7hdhd                                 1/1    Running    0
          9m54s
rook-discover-h75kv                                 1/1    Running    0
          9m54s
rook-discover-pwnxf                                1/1    Running    0
          9m54s
rook-discover-vddvc                                1/1    Running    0
          9m54s
# kubectl -n rook-ceph exec ceph-oam-798f85dcf5-6hdtv -c ceph-oam -- ceph
status
  cluster:
    id:      be9dc5e4-ca06-43cd-abaa-d3a624d8c190
    health:  HEALTH_OK
  services:
    mon: 3 daemons, quorum a,b,d (age 3h)
    mgr: a(active, since 3h)
    osd: 5 osds: 5 up (since 3h), 5 in (since 23h)
    rgw: 1 daemon active (rook.ceph.store.a)
  data:
    pools: 8 pools, 184 pgs
    objects: 292 objects, 118 MiB
    usage:  5.4 GiB used, 40 GiB / 45 GiB avail
    pgs:    184 active+clean

```

12.8.2 Rook usage example

1. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```

---
kind: PersistentVolumeClaim
apiVersion: v1

```

```
metadata:  
  name: nginx-pv-claim  
spec:  
  storageClassName: rook-ceph-block-rep  
  accessModes:  
    - ReadWriteOnce  
resources:  
  requests:  
    storage: 1Gi
```

2. Create a Kubernetes yaml file to define a pod, and mount the PVC in the definition of the pod.

Example file:

```
---  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
  name: nginx  
  namespace: default  
  labels:  
    app: nginx  
spec:  
  strategy:  
    rollingUpdate:  
      maxSurge: 10%  
      maxUnavailable: 0  
  selector:  
    matchLabels:  
      app: nginx  
  replicas: 1  
  template:  
    metadata:  
      labels:  
        app: nginx  
    spec:  
      volumes:  
        - name: nginx-pv-storage  
          persistentVolumeClaim:  
            claimName: pvcsc001  
      containers:  
        - name: nginx  
          image: registry-proxy:5000/library/nginx:alpine  
          imagePullPolicy: Always  
          volumeMounts:  
            - mountPath: "/mnt/rook"
```

```
name: nginx-pv-storage
dnsPolicy: Default
```

12.8.3 Rook troubleshooting

When an application that is using Rook storage is deployed on a Worker node, it's possible that the node will become unreachable after a reboot.

- The node must be drained before the planned reboot. For details, see <https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/>
- The node must be marked as schedulable after the reboot, using the following command:

```
kubectl uncordon <node>
```

In the event of an unplanned reboot, nodes can be recovered using the following methods:

- For OpenStack, use a "Soft Reboot Instance" or "Hard Reboot Instance" to bring the node back up.
- For VMware, power off and then power on the node.
- For AWS, stop then start the node.

For further information, see <https://rook.io/docs/rook/master/common-issues.html#node-hangs-after-reboot>.

12.9 GlusterFS

-  **Warning:** GlusterFS is not being used for Baremetal deployent (NCS20 release) and alternative is to use cephfs (see section of cephfs *Ceph CSI* on page 322).
-  **Warning:** Support for GlusterFS must be completed before deploying NCS. The storage function cannot be setup once deployment begins.
-  **Warning:** All partitions/drives must be dedicated, in a raw state without any file system, and not be a part of an LVM group. Failure to comply with these requirements may result in a failed deployment of NCS.

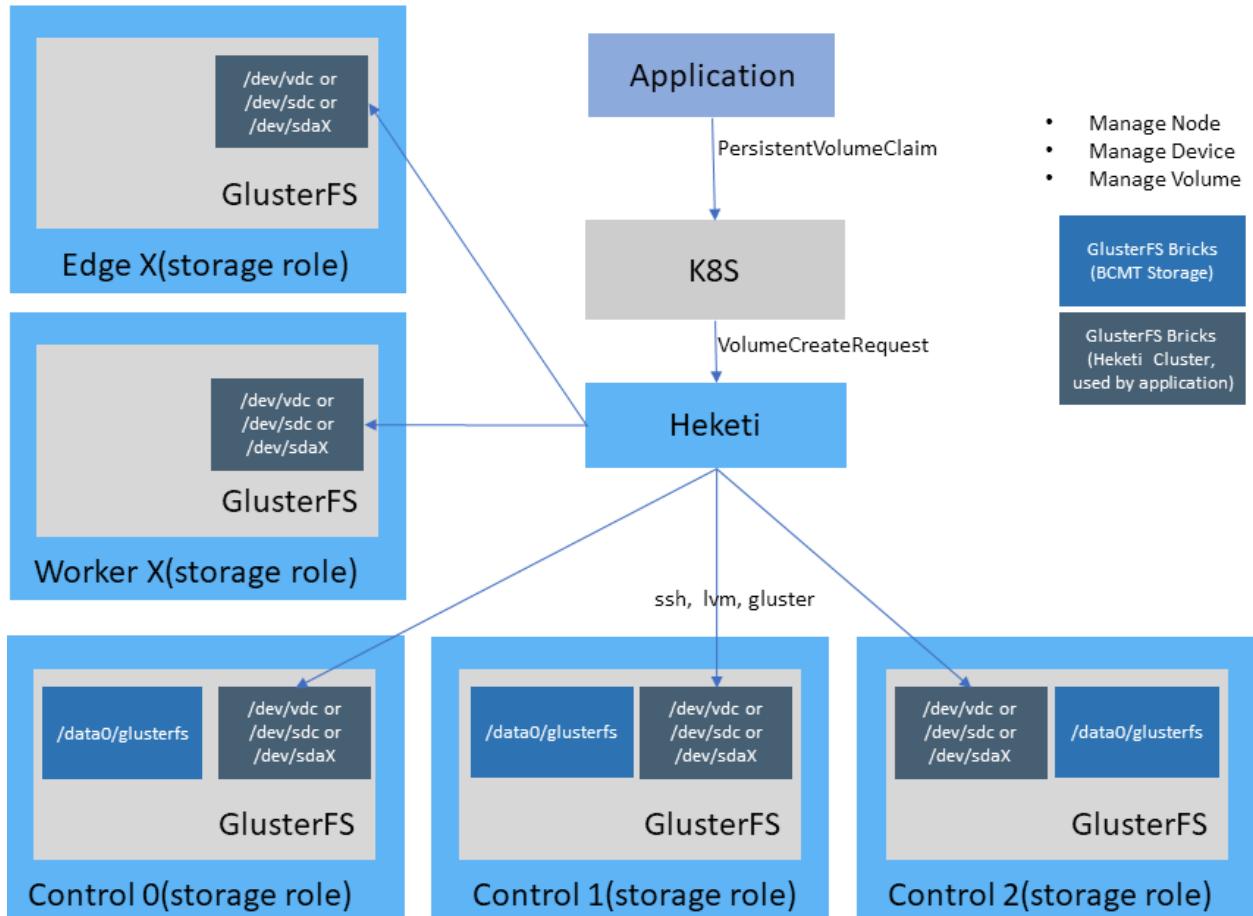
NCS supports GlusterFS to provide application containers with persistent volume and data sharing functions. The NCS storage function is deployed on Storage nodes, which can be combined with Control, Worker or Edge nodes. For replicated GlusterFS volumes, the number of storage nodes must be greater than the GlusterFS brick count to support LCM events. In a 3 node configuration, StorageClass glusterfs-storageclass is used with 2 replicated bricks, and for 4 or more nodes, StorageClass glusterfs-storageclass-arbiter is used with 3 replicated bricks, including 2 real bricks and one arbiter. It is recommended to use 4 Storage nodes to fully support node heal/recover.

When an application container applies for a volume via a volume claim, NCS tries to allocate the required volumes from the pool and create a GlusterFS cluster with them.

NCS includes Heketi to provide a management API to Kubernetes and to perform volume allocation, creation, deletion, healing and other operations.

After deployment, NCS creates StorageClass glusterfs-storageclass that is referenced by the PersistentVolumeClaim.

Figure 17: NCS support for GlusterFS



Use this procedure to configure GlusterFS prior to NCS deployment.

1. Prepare a minimum of three Storage nodes with dedicated, raw partitions or drives.
2. Access the NCS `node_inventory.json` file, and under "storage", declare the raw devices.
3. Access the `bcmt_config.json` file and set the parameter `k8s_shared_storage` to `true`.

12.9.1 GlusterFS usage example

1. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
kind: PersistentVolumeClaim
```

```

apiVersion: v1
metadata:
  name: cinfo-pv-claim
spec:
  storageClassName: glusterfs-storageclass
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

```

2. Create a Kubernetes yaml file to define a pod.

Example file:

```

---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx
  namespace: default
  labels:
    app: nginx
spec:
  strategy:
    rollingUpdate:
      maxSurge: 10%
      maxUnavailable: 0
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
  spec:
    volumes:
      - name: cinfo-pv-storage
        persistentVolumeClaim:
          claimName: cinfo-pv-claim
    containers:
      - name: cinfo
        image: registry-proxy:5000/cinfo-storage:1.0
        imagePullPolicy: Always
        volumeMounts:

```

```

      - mountPath: "/mnt/glusterfs"
        name: cinfo-pv-storage
      dnsPolicy: Default
    
```

3. Mount the PVC in the definition of the pod.

12.9.1.1 PVC resize

Network attached file system, such as GlusterFS, can be expanded without having to restart the referenced Pod.

- Configure the glusterfs-storageclass to set allowVolumeExpansion to "true" and enable the volumeoptions "features.shard". By default "features.shard" is set to disabled.

Example file:

```

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: glusterfs-storageclass
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://10.254.0.101:8080"
  clusterid: "9f5f0a68f7243382b3b90ce96cf25d1b"
  restauthenabled: "true"
  restuser: "admin"
  secretNamespace: "ncms"
  secretName: "heketi-secret"
  gidMin: "40000"
  gidMax: "50000"
  volumetype: "replicate:3"
  volumeoptions: "features.shard enable"
allowVolumeExpansion: true
  
```

2. Create a PVC using the modified glusterfs-storageclass.

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim-expand
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    
```

```

    requests:
      storage: 2Gi
    storageClassName: glusterfs-storageclass
  
```

3. Create a Pod that references the PVC.



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/
            nginx:1.14.0-3.4
          ports:
            - containerPort: 80
          volumes:
            - name: data
          persistentVolumeClaim:
            claimName: myclaim-expand
  
```

4. To request a larger volume, edit the PVC object and specify a larger size.

Example command:

```
kubectl edit persistentvolumeclaim/myclaim-expand
```

5. Verify the PVC status using the following command:

```
kubectl describe pvc myclaim-expand
```

12.10 Local storage

NCS has integrated the external static provisioner to help simplify local storage management once the local volumes are configured. Note that the local storage provisioner is different from most provisioners and does not support dynamic provisioning. Instead, it requires that administrators pre-configure the local volumes on each node. For further information on the Kubernetes external static provisioner, see <https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner>.

The provisioner will manage the volumes under the discovery directories by creating and cleaning up PersistentVolumes for each volume.

Use this procedure to configure local storage prior to NCS deployment.

1. Prepare dedicated raw devices on the cluster nodes.
2. Access the `user_input.yaml` file and specify the `local_storage_devices` parameter.

12.10.1 Local storage usage example

1. After deployment, use the `ncs tool local-storage add` command to add a local storage PV per node.

Example:

```
ncs tool local-storage add --node_name=bcmt-worker-01 --volume_device '{"/dev/vdc": "ext4"}'
```

2. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: example-local-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  storageClassName: local-storage
```

3. Create a Kubernetes yaml file to define a pod.

Exmple file:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
```

```

name: mydeployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: local-storage
  template:
    metadata:
      name: example-local-pod
      labels:
        app: local-storage
    spec:
      containers:
        - name: example-local-pod
          image: bcmt-registry:5000/bcmt-portal:0.3
          volumeMounts:
            - name: data
              mountPath: /usr/test
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: example-local-claim

```

4. Mount the PVC in the definition of the pod

12.11 Ceph CSI

Ceph CSI plugins implement an interface between CSI enabled Container Orchestrator (CO) and Ceph cluster. It allows dynamically provisioning Ceph volumes and attaching them to workloads.

Independent CSI plugins are provided to support RBD and CephFS backed volumes, installation procedure is same, below procedure use CephRBD as the example.

RBD need ceph cluster version \geq Mimic, and CephFS need ceph cluster version Nautilus (\geq 14.2.2). On the time of this asset delivery, NCS cannot support Ceph version 14.2.2, please use Ceph RBD only if ceph cluster is provided by NCS.

12.11.1 Precondition

- The CBCS(cloud band container service) end point is used to retrieve ceph user credential for ceph rbd csi, that end point should be accessible from bcmt control node.
- Since all the bcmt cluster node need to be able to access the ceph monitor address to attach volume, that address should be accessible from every bcmt cluster node

12.11.2 Installation procedure

When `csi-cephrbd/csi-cephfs` is enabled, need to set `csi-cephrbd/csi-cephcsi` access information in `bcmt_config.json` file as below example before NCS deployment. `bcmt_config.json` file example, refer to "storage_csi_config" section as below:

```
{
    "app": {
        "appv1_enabled": false,
        "chart_enabled": false,
        "chart_repo_url": {}
    },
    "cluster_config": {
        "cluster_mode": "kubernetes",
        "cluster_name": "bcmt-01",
        "cluster_id": "123456789",
        .....
    },
    "storage_csi_config": {
        "cephfs_enabled": true,
        "cephfs_config": {
            "clusterID": "7671882a-3c85-44de-bbba-7ffffe3c573b",
            "monitors": ["10.10.10.10:6789", "10.10.10.11:6789",
"10.10.10.12:6789"],
            "pool": "cephfs_pool",
            "adminID": "YWRtaW4=",
            "adminKey": "QVFEbmE3NWNUQzJ2T2hBQWNFa01GMGNpZm9TOVM0Mzh6VTBIM1E9PQ=="
        },
        "cephrbd_enabled": true,
        "cephrbd_config": {
            "clusterID": "7671882a-3c85-44de-bbba-7ffffe3c573b",
            "monitors": ["10.10.10.10:6789", "10.10.10.11:6789",
"10.10.10.12:6789"],
            "pool": "cephrbd_pool",
            "userID": "YWRtaW4=",
            "userKey": "QVFEbmE3NWNUQzJ2T2hBQWNFa01GMGNpZm9TOVM0Mzh6VTBIM1E9PQ=="
        }
    }
}
```

The value of parameter `userID`, `userKey`, `adminID` and `adminKey` should be base64 encoded. Chart `cephrbd/cephfs` is installed automatically in NCS deployment.

12.11.3 Verification

- Check the pod is running on bcmt control node, chart files are installed, storage class is created:

```
# kubectl get po -n ncms | grep cephfs

csi-cephfs-nodeplugin-2tdn7           2/2     Running   0
  23h
csi-cephfs-nodeplugin-fjqc9          2/2     Running   0
  23h
csi-cephfs-nodeplugin-hjrdr          2/2     Running   0
  23h
csi-cephfs-nodeplugin-jztgz          2/2     Running   0
  23h
csi-cephfs-nodeplugin-l2sfh          2/2     Running   0
  23h
csi-cephfs-nodeplugin-p6jvm          2/2     Running   0
  91m
csi-cephfs-provisioner-7f7bf4f9b9-5fc2x 4/4     Running   0
  23h
csi-cephfs-provisioner-7f7bf4f9b9-kq6gd 4/4     Running   0
  23h
csi-cephfs-provisioner-7f7bf4f9b9-xbbr8 4/4     Running   0
  100m

# kubectl get po -n ncms | grep cephrrbd
csi-cephrrbd-nodeplugin-9wwt2         2/2     Running   0
  23h
csi-cephrrbd-nodeplugin-m4tpj         2/2     Running   0
  23h
csi-cephrrbd-nodeplugin-rch97         2/2     Running   0
  23h
csi-cephrrbd-nodeplugin-stcfj        2/2     Running   0
  93m
csi-cephrrbd-nodeplugin-xpwlw         2/2     Running   0
  23h
csi-cephrrbd-nodeplugin-zc5xw         2/2     Running   0
  23h
csi-cephrrbd-provisioner-5f897fcf7-2bsw6 5/5     Running   0
  102m
csi-cephrrbd-provisioner-5f897fcf7-m4bh5 5/5     Running   0
  23h
csi-cephrrbd-provisioner-5f897fcf7-vb6js 5/5     Running   0
  23h

# helm list | grep cephrrbd
```

```

csi-cephrbd      1           Wed Mar 25 08:00:40 2020      DEPLOYED
                  csi-cephrbd-1.0.7    2.0.1                   ncms
# helm list | grep cephfs
csi-cephfs       1           Wed Mar 25 08:00:46 2020      DEPLOYED
                  csi-cephfs-1.0.5    2.0.1                   ncms

# kubectl get sc | grep cephrbd
csi-cephrbd          rbd.csi.ceph.com        Delete
  Immediate          false                23h
# kubectl get sc | grep cephfs
csi-cephfs          cephfs.csi.ceph.com      Delete
  Immediate          false                23h

```

- Create a test volume using this storage class.

1. Create a Kubernetes yaml file to define the PersistentVolumeClaim.

Example file: pvc-test.yaml

```

---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: rbd-pvc-test
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-cephrbd

```

2. Apply the PersistentVolumeClaim yaml file.

```

# kubectl apply -f pvc-test.yaml

persistentvolumeclaim/rbd-pvc-test created

# kubectl get pvc
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
rbd-pvc-test Bound pvc-39d76228-cdeb-11e9-b010-fa163eb033a5 1Gi RWO
  csi-cephrbd 18s

```

3. Create a Kubernetes yaml file to define a pod.



Note: The following repository/registry URLs are provided as examples, the user should add their own.

Example file: pod.yaml

```

---
apiVersion: v1
kind: Pod
metadata:
  name: csi-rbd-demo-pod
spec:
  containers:
    - name: demo-cinfo
      image: csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
        cinfo:3.0
  volumeMounts:
    - mountPath: /etc/localtime
      name: host-timezone
      readOnly: true
    - mountPath: /mnt
      name: mypvc
      dnsPolicy: Default
  volumes:
    - name: host-timezone
      hostPath:
        path: /etc/localtime
    - name: mypvc
      persistentVolumeClaim:
        claimName: rbd-pvc-test
        readOnly: false

```

4. Apply the pod yaml file

```

# kubectl apply -f  pod.yaml

# kubectl get po
NAME READY STATUS RESTARTS AGE
csi-rbd-demo-pod 1/1 Running 0 3m41s

```

12.11.4 Upgrade of cephrrbd-csi

Use case: On NCS20FP2 when resizing a cephrrbd PVC (probably applies for cephfs as well) the new volume size is not automatically at the file system level and the pod has to be recreated for the volume resize to take affect. Upgrading the cephrrbd csi solves the issue.

Scope: The below procedure shows how to upgrade the cephrrbd csi in NCS20FP2 with the one used in NCS22.

Environment:

- Version 20.100.1-1310 + PP6 + PP7 + PP8 + PP9
- username = cbis-admin

The procedure:

1. Download the new chart and add to the repo

- Download csi-cephrbd version 1.1.10
- Add to help repo:

```
$ sudo ncs app-resource chart add --file_name csi-cephrbd-1.1.10.tgz

$ sudo --preserve-env helm repo update ## The --preserve-env is
needed due to the sudo

$ ## Verify by
$ sudo --preserve-env helm search |grep csi-cephrbd
```

2. Download the required images and push to bcmt registry



Note: The following repository and registries are provided as examples.

```
## download images from Nokia repo

sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-provisioner:v2.0.4
sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-resizer:v1.0.1
sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-snapshotter:v3.0.2
sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-attacher:v3.0.2
sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/node-driver-registrar:v2.0.1
sudo docker pull csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/cephcsi:v3.3.1

## Tag

sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/
bcmt/csi/csi-provisioner:v2.0.4 bcmt-registry:5000/bcmt/csi/csi-
provisioner:v2.0.4
sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-resizer:v1.0.1 bcmt-registry:5000/bcmt/csi/csi-resizer:v1.0.1
```

```

sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/
bcmt/csi/csi-snapshotter:v3.0.2 bcmt-registry:5000/bcmt/csi/csi-
snapshotter:v3.0.2
sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/csi-attacher:v3.0.2 bcmt-registry:5000/bcmt/csi/csi-attacher:v3.0.2
sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/node-driver-registrar:v2.0.1 bcmt-registry:5000/bcmt/csi/node-driver-
registrar:v2.0.1
sudo docker tag csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/bcmt/
csi/cephcsi:v3.3.1 bcmt-registry:5000/bcmt/csi/cephcsi:v3.3.1

# Push

sudo docker push bcmt-registry:5000/bcmt/csi/csi-provisioner:v2.0.4
sudo docker push bcmt-registry:5000/bcmt/csi/csi-resizer:v1.0.1
sudo docker push bcmt-registry:5000/bcmt/csi/csi-snapshotter:v3.0.2
sudo docker push bcmt-registry:5000/bcmt/csi/csi-attacher:v3.0.2
sudo docker push bcmt-registry:5000/bcmt/csi/node-driver-registrar:v2.0.1
sudo docker push bcmt-registry:5000/bcmt/csi/cephcsi:v3.3.1

```

3. Adapt the NCS22 all-values.yaml to the local environment

- Save the original all-value file:

```
$ sudo cp /opt/bcmt/config/csf-charts/csi-cephrbd/all-values.yml /opt/
bcmt/config/csf-charts/csi-cephrbd/all-values.yml-orig
```

- Copy an all-value from an ncs22 setup and download the following file.

```
$ sudo scp <all-value file copied from ncs22> /opt/bcmt/config/csf-
charts/csi-cephrbd/all-values.yml
```

- Edit the new file: Copy the following values from the original all-values.yaml file to the new one:

- clusterID
- monitors
- pool
- userID
- userKey

4. Helm upgrade

```

sudo --preserve-env /usr/local/bin/helm upgrade csi-cephrbd stable/csi-
cephrbd \
--version 1.1.10 \
--reset-values \
--recreate-pods \

```

```
--force \
-f /opt/bcmt/config/csf-charts/csi-cephrbd/all-values.yaml
```

5. Verify by running following commands:

```
$ sudo --preserve-env helm list | grep csi-cephrbd $ sudo kubectl get sc |
grep csi-cephrbd
```

ncs22 all-values.yaml example

```
---
clusterID: "70732d29-aa00-47cc-b65c-78de8cb66f77"
monitors: '[ "172.17.4.2,172.17.4.6,172.17.4.7"]'
storageclass:
  is_default: true
  pool: "volumes"
  userID: "Y2VwaHJiZA=="
  userKey: "QVFBy1gwdGhHNXJqTmhBQVhYOEZBSmVUY3d5bERhcUdURzdWQkE9PQ=="
  secret:
    create: true
provisioner:
  tolerations:
    [ {"effect": "NoExecute", "key": "is_edge", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_control", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal", "value": "true"}, {"operator": "Exists", "effect": "NoSchedule"} ]
  replicaCount: 1
  provisioner:
    image:
      repository: "bcmt/csi/csi-provisioner"
      tag: "v2.0.4"
    resources:
      limits:
        cpu: 1000m
        memory: 8Gi
      requests:
        cpu: 10m
        memory: 30Mi
  resizer:
    image:
      repository: "bcmt/csi/csi-resizer"
      tag: "v1.0.1"
    resources:
      limits:
        cpu: 1000m
        memory: 8Gi
      requests:
```

```
cpu: 10m
memory: 30Mi
snapshotter:
  image:
    repository: "bcmt/csi/csi-snapshotter"
    tag: "v3.0.2"
  resources:
    limits:
      cpu: 1000m
      memory: 8Gi
    requests:
      cpu: 10m
      memory: 30Mi
attacher:
  image:
    repository: "bcmt/csi/csi-attacher"
    tag: "v3.0.2"
  resources:
    limits:
      cpu: 1000m
      memory: 8Gi
    requests:
      cpu: 10m
      memory: 30Mi
nodeplugin:
  tolerations:
    [{"effect": "NoExecute", "key": "is_edge", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_control", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal", "value": "true"}, {"operator": "Exists", "effect": "NoSchedule"}]
  registrar:
    image:
      repository: bcmt/csi/node-driver-registrar
      tag: v2.0.1
    resources:
      limits:
        cpu: 1000m
        memory: 8Gi
      requests:
        cpu: 10m
        memory: 30Mi
  plugin:
    image:
      repository: bcmt/csi/cephcsi
      tag: v3.3.1
```

```

resources:
  limits:
    cpu: 1000m
    memory: 8Gi
  requests:
    cpu: 10m
    memory: 30Mi
tolerations: [ {"effect": "NoExecute", "key": "is_edge", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_control", "operator": "Equal", "value": "true"}, {"effect": "NoExecute", "key": "is_storage", "operator": "Equal", "value": "true"}, {"operator": "Exists", "effect": "NoSchedule"} ]

```

12.12 Dynamic NFS

When there is an existing configured NFS server and user wants to use dynamic way to create nfs volumes. User can enable this feature to do the dynamic provisioner.

12.13 Implementation of S3 on NCS

12.13.1 S3 Support

S3 object storage provider can be accessed using s3 protocol from the pods. The Kubernetes CSI way of using s3-csi is still experimental, therefore not recommended for production although it is described in the following.

Basic prerequisites are:

1. Target S3 provider credentials.
2. IP access from the worker on which the app pod runs and the S3 provider.

Note that this is not provided by default through NCS deployment, that's why in the example below "node nodeSelector: is_control: true" was used.

12.13.1.1 How to use S3 from pod

There are two variations for using S3 from a pod:

1. S3 towards the local radosgw
2. S3 towards a remote provider, like AWS S3.

Summarizing the steps needed:

Table 35: Steps

Step	S3 towards the local radosgw - CN-B	S3 towards the local radosgw - CN-A	S3 towards e.g. AWS S3-CN-A&B
Step	S3 towards the local radosgw - CN-B	S3 towards the local radosgw - CN-A	S3 towards e.g. AWS S3 - CN-A&B
Step	S3 towards the local radosgw - CN-B	S3 towards the local radosgw - CN-A	S3 towards e.g. AWS S3 - CN-A&B
IP access from the worker on which the s3 client pod runs	Access to https://<local radosgw external VIP>: 13808	Same as for CN-B	External IP access
Provider info & Credential needed	<p>Create a crd of kind radosgw, simply with the requested:</p> <ul style="list-style-type: none"> • s3 username • quota • secret name & namespace <p>No credential needed. The k8s secret is automatically created when applying the crd.</p>	<p>Create a k8s secret with:</p> <ul style="list-style-type: none"> • Access Key ID • Secret Access Key • Endpoint 	<p>Create a k8s secret with:</p> <ul style="list-style-type: none"> • Access Key ID • Secret Access Key • Aws session token • Region
Pod	Uses the secret (mounted or via environment variables)	same as for CN-B	Same as for the local radosgw case

12.13.1.1.1 S3 towards the local radosgw s3

For NCS-BM (or CN-B), using the radosgw allows for some automation: The user can apply a crd that would trigger creation of a new user in the local radosgw along with a k8s secret containing the s3 provider endpoint and credentials.

For CN-A one has to "manually" prepare the secret file (i.e. rather than using a crd as for the CN-B case), which is similar to the s3 access to a remote provider described in the next section.

1. s3 client code and its pod

Applies for both CN-A and CN-B.

Following is a python partial code example for accessing local radosgw s3. This code has been dockerized and used in the pod description that follows it.

S3 client - Python example

```
import boto3

# REF: https://github.com/ronaldddilley/ceph-s3-examples
# For ssl see: https://boto3.amazonaws.com/v1/documentation/api/latest/
reference/core/session.html

class S3client:
    def __init__(self, endpoint_url, secret_key, access_key,
ca_cert_path):
        self.endpoint_url = endpoint_url
        self.secret_key = secret_key
        self.access_key = access_key

    def connect(self):
        self.s3 = boto3.resource('s3',
                               endpoint_url=self.endpoint_url,
                               aws_access_key_id=self.access_key,
                               aws_secret_access_key=self.secret_key,
                               verify=False) # no ssl, assuming the user
doesn't have the CA
        return

    def bucket_create(self, bucket_name):
        self.bucket = self.s3.Bucket(bucket_name)
        self.bucket.create()
        return
```

Pod secret mounting example

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: s3secret
          secret:
            secretName: s3-access-secret-st
      containers:
```

```

- name: s3podif-example
  volumeMounts:
    - name: s3secret
      mountPath: "/s3secrets"
      readOnly: true

```

2. Running the pod towards the local radosgw - CN-B

For CN-B only, user can apply a crd that triggers creation of the secret with the radosgw credentials:

radosgw-test1-crd.yaml

```

apiVersion: storage.ncm.nokia.com/v1
kind: Radosgw
metadata:
  name: radosgw-example
spec:
  # Add fields here
  user_name: "usera"
  user_quota: "7G"
  access_credential_ns: "default"
  access_credential_name: "s3-local-secret-st"

```

Explaining the above crd fields:

Table 36: crd fields explanation

Name	Value format	Function
access_credential_name	A string	The secretName used on the pod manifest (see pod example above)
access_credential_ns	A string	The created secret namespace, on which also the pod will run on
user_name	A string	The username created on the radosgw
user_quota	A string of a number and a single character M/G/T presenting Mega/Giga/Tera	Total quota for that radosgw user

Testing example:

```
# kubectl apply -f radosgw-test1-crd.yaml

# kubectl get secrets

# kubectl apply -f

# apply -f s3pod-deploy.yaml

# kubectl get pods
```



Note: the secret's fields appear as files in the pod foder, as non base64-encoded values:

```
# kubectl --namespace smoke-test exec -it $pod -- bash

bash-5.1$ ls -l /s3secrets/
total 0
lrwxrwxrwx 1 root worker 18 Jun 20 11:39 accessKeyID -> ..data/
accessKeyID
lrwxrwxrwx 1 root worker 23 Jun 20 11:39 cluster_username -
> ..data/cluster_username
lrwxrwxrwx 1 root worker 15 Jun 20 11:39 endpoint -> ..data/
endpoint
lrwxrwxrwx 1 root worker 22 Jun 20 11:39 secretAccessKey -> ..data/
secretAccessKey
lrwxrwxrwx 1 root worker 17 Jun 20 11:39 user_quota -> ..data/
user_quota
lrwxrwxrwx 1 root worker 15 Jun 20 11:39 username -> ..data/
username

bash-5.1$ cat /s3secrets/endpoint
https://10.22.98.81:13808
```

3. CN-A accessing the radosgw s3 provider

Comparing to the CN-B case described previously, the main difference here is that the secret file has to be "manually created" by the user/admin, i.e. rather than using the radosgw crd shown previously.

An example of an s3 target fields:

Table 37: s3 target fields

Field	Value	base64 encoded
endpoint	https://10.22.98.81:13808	aHR0cHM6Ly8xMC4yMi45OC44MToxMzgwOA==
access KeyID	4DU89D8E0R2NW71M1P0J	NERVOD1EOEUwUjJOVzcxtTFQMEo=
secret Access Key	v57C3LmBTom35zQ0noXRBboBIEIXgmMpEBExNwCm	djU3QzNMbUJUb20zNXpRMG5vWFJCYm9CSUVJWGdtTXBFQkV4TndDbQ==

The secret information is base64 encoded as in the following example:

manual-secret.yaml

```
apiVersion: v1
data:
  accessKeyID: NERVOD1EOEUwUjJOVzcxtTFQMEo=
  endpoint: aHR0cHM6Ly8xMC4yMi45OC44MToxMzgwOA==
  secretAccessKey:
    djU3QzNMbUJUb20zNXpRMG5vWFJCYm9CSUVJWGdtTXBFQkV4TndDbQ==
kind: Secret
metadata:
  name: s3-secret-manual
  namespace: default
  type: Opaque
```

```
# kubectl apply -f manual-secret.yaml
```

12.13.1.1.2 Accessing AWS S3

1. Step 1: Get your user credentials

In the following example a user has been created by:

```
aws-nokia-login --username agideon --duration-seconds 43200 --profile g-s3-test
```

And the result is the file: .aws/credentials

```
cat .aws/credentials
```

```
[g-s3-test]

aws_access_key_id =
aws_secret_access_key =
aws_session_token =
```

Some CLI use examples (not used in the test below):

```
aws --profile g-s3-test s3api create-bucket --bucket g-s3-test2 --region
eu-central-1 --create-bucket-configuration LocationConstraint=eu-
central-1

aws --profile g-s3-test s3api delete-bucket --bucket g-s3-test2
```

2. Step 2: run the application

Following is a python script example for accessing aws s3. The script could be dockerized and put into a pod as in the previous example.

aws s3 client - a python example

```
from boto3.session import Session

class S3client:
    def create_session(self):
        # Per https://boto3.amazonaws.com/v1/documentation/api/latest/
reference/core/session.html
        session = Session(aws_access_key_id=AWS_ACCESS_KEY_ID,
                          aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
                          aws_session_token=AWS_SESSION_TOKEN,
                          region_name=REGION)
        self.s3 = session.resource('s3')
        self.s3cl = session.client('s3')

    def bucket_create(self, bucket_name):
        try:
            self.s3.create_bucket(Bucket=bucket_name,
CreateBucketConfiguration={'LocationConstraint': REGION})
            self.bucket = self.s3.Bucket(bucket_name)
            return 0
        except Exception as e:
            return 1
```

```

def bucket_delete(self, bucket_name):
    try:
        self.bucket = self.s3.Bucket(bucket_name)
        self.bucket.delete()
        print("bucket {} deleted.".format(bucket_name))
    except Exception as e:
        print("Error in {} bucket deletion: {}".format(bucket_name,
e))
    return

```

12.14 S3cmd to access s3 server

When using the CBCS solution, a s3 server is setup. NCS provides a Kubernetes operator to get the radosgw user information from S3 server. It is by applying a crd and then a secret is created including the s3 endpoint, access key, secret key, quota and so on.

After the secret is created, user can get endpoint and key information from it and use s3cmd to put or retrieve data from the s3 bucket. Or user can use csi-s3 to dynamically allocate buckets and mount them via a fuse mount into any container.

S3cmd to access s3 server

1. Before the deployment, the following configurations must be set in the bcmt_config.json.

- Set **controller_storage** to true in **cluster_config** section.
- Set **radosgw_enabled** to true in **storage_csi_config** section.
- Config endpoint, cert, username and password in the **cbs_ceph_mgr_config** section.

A deployment pod starting with storage-controller will be created in kube-system namespace after deployment. Use the following command to get the pod.

Example command:

```
# kubectl get po -n kube-system storage-controller-9c999dc9f-rthjx
```

Expected output:

NAME	READY	STATUS	RESTARTS	AGE
storage-controller-9c999dc9f-rthjx	1/1	Running	0	2d2

2. User can apply a crd.yml file in following format. An explanation of the parameters in spec is provided:

- **user_name**: the radosgw username to be created.
- **user_quota**: the quota to be allocated for the user.
- **access_credential_name**: the secret name will be created to store the endpoint and Accesskey related info.
- **access_credential_ns**: the namespace of the secret.

Example file:

```

apiVersion: storage.ncm.nokia.com/v1
kind: Radosgw
metadata:
  name: radosgw-test1
spec:
  # Add fields here
  user_name: "usera"
  user_quota: "7G"
  access_credential_ns: "default"
  access_credential_name: "s3-access-secret1"

```

Use the following command to apply the crd:

```
# kubectl apply -f crd.yml
```

3. If the operation is successful, a secret named s3-access-secret1 will be created automatically in the default namespace. If after 2 minutes, the secret is still not created, the storage controller pod log can be checked to debug. Use the following command:

```
# kubectl logs -n kube-system storage-controller-9c999dc9f-rthjx
```

Use following command to get the created secret:

```
# kubectl get secret
```

Expected output:

NAME	TYPE	DATA	AGE
s3-access-secret1	Opaque	6	2d1h

4. After the secret is created, user can get the s3 endpoint information from the secret and then use s3cmd to do operations.

Example command:

```
# kubectl get secret s3-access-secret1 -o yaml
```

Expected output:

```

apiVersion: v1
data:
  accessKeyID: MTFBR0g3NlM5QjE3WFNCTkM3NEg=
  cluster_username: dw5pdHRlc3QtdXNlcil1bA==
  endpoint: aHR0cHM6Ly8xMC4xMC4xMC4xMDoxMzgwOA==
  secretAccessKey:
    a3pFNWJvTkE1MTRVTUJQdjRtdzB0eDZ5ZzJVWnd6SnpnMXdkc3BOQw==
  user_quota: N0c=
  username: dXNlcmlE=

```

```

kind: Secret
metadata:
  creationTimestamp: "2020-06-01T07:08:01Z"
  name: s3-access-secret1
  namespace: default
  ownerReferences:
    apiVersion: storage.ncm.nokia.com/v1
    blockOwnerDeletion: true
    controller: true
    kind: Radosgw
    name: radosgw-test1
    uid: 0bdac374-41b2-4e9d-bf71-17733d950158
  resourceVersion: "34347"
  selfLink: /api/v1/namespaces/default/secrets/s3-access-secret1
  uid: 1aa3c59e-2609-4ff8-89c0-b5234435f22c
type: Opaque

```

5. Decode the strings in the secret before using.

```

# echo "aHR0cHM6Ly8xMC4xMC4xMC4xMDoxMzgwOA==" | base64 -d
https://10.10.10.10:13808
# echo "MTFBR0g3NlM5QjE3WFNCTkM3NEg=" | base64 -d
11AGH76S9B17XSBNC74H
# echo "a3pFNWJvTkE1MTRVTUJQdjRtdzB0eDZ5ZzJVWnd6SnpnMXdkc3BOQw==" | base64 -
d
kzE5boNA514UMBPv4mw0tx6yg2UZwzJzglwdspNC

```

6. Use s3cmd cli to access s3 bucket. s3cmd must be installed in the pods that wants to use s3cmd to access s3 endpoint.

Following example is using no ssl mode:

```

# s3cmd --host=10.10.10.10:13808 --ssl --no-check-certificate \
--secret_key="kzE5boNA514UMBPv4mw0tx6yg2UZwzJzglwdspNC" \
--access_key="11AGH76S9B17XSBNC74H" put run_cbc_ceph_mgr.sh s3://bucket0

```

Expected output:

```

upload: 'run_cbc_ceph_mgr.sh' -> 's3://bucket0/run_cbc_ceph_mgr.sh' [1
of 1]
492 of 492 100% in 0s 9.31 kB/s done

```

The cert for the access is stored in /etc/pki/ca-trust/source/anchors/ca.crt.pem in the ncs cluster host nodes. And it also stored in the secret named cbc-ca-file in kube-system namespace. It is necessary to copy the cert in the pods which use the s3cmd.

Example command:

```
# s3cmd --host=10.10.10.10:13808 --ssl --no-check-hostname \
--ca-certs /etc/cbcs-ceph-agent/pki/ca-trust/source/anchors/ca.crt.pem \
--secret_key="VCjqVTnGNQqdAMLQp1sCvwgopiZPpPbpOcQFDPhM" \
--access_key="NO192GB4C55FQAJHC2NY" put /hello_s3cmd s3://bucket0
```

Expected output:

```
upload: '/hello_s3cmd' -> 's3://bucket0/hello_s3cmd' [1 of 1]
9 of 9 100% in 0s 184.74 B/s done
```

csi-s3

With the configuration in **S3cmd to access s3 serve** section, csi-s3 is also deployed. Following csi-s3 pods are created.

Limitation: The csi-s3 cannot set quota for the PV. That is if creating a PV and the size of PVC is 1Gi, but it can write unlimited data until the quota for the s3 user is full.

Example command:

```
# kubectl get po -n kube-system|grep csi-s3
```

Expected output:

csi-s3-controllerplugin-0 2d3h	3/3	Running	0
csi-s3-controllerplugin-1 2d3h	3/3	Running	0
csi-s3-nodeplugin-7q2v9 2d3h	2/2	Running	0
csi-s3-nodeplugin-b9zdz 2d3h	2/2	Running	0
csi-s3-nodeplugin-195b7 2d3h	2/2	Running	0
csi-s3-nodeplugin-pmhtr 2d3h	2/2	Running	0
csi-s3-nodeplugin-r992f 2d3h	2/2	Running	0
csi-s3-nodeplugin-wpfdp 2d3h	2/2	Running	0

Dynamically allocate s3 bucket with csi-s3 storageclass

1. Create the csi-s3 storage class, use the steps in s3cmd section to create a secret. And input the secret name and namespace in the storagclass yaml file.

Example file:

```
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-s3
provisioner: ch.ctrox.csi.s3-driver
parameters:
  # specify which mounter to use
  # can be set to rclone, s3fs or goofys
  mounter: rclone
  csi.storage.k8s.io/provisioner-secret-name: s3-access-secret1
  csi.storage.k8s.io/provisioner-secret-namespace: default
  csi.storage.k8s.io/controller-publish-secret-name: s3-access-secret1
  csi.storage.k8s.io/controller-publish-secret-namespace: default
  csi.storage.k8s.io/node-stage-secret-name: s3-access-secret1
  csi.storage.k8s.io/node-stage-secret-namespace: default
  csi.storage.k8s.io/node-publish-secret-name: s3-access-secret1
  csi.storage.k8s.io/node-publish-secret-namespace: default
```

2. Create a pvc using the new storage class.

Note: that the requested storage size in the PVC shall be less than the user_quota applied in the crd.

Example file:

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-s3-pvc
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: csi-s3
```

3. Create a test pod which mounts the volume.

```
---
apiVersion: v1
```

```

kind: Pod
metadata:
  name: csi-s3-test-nginx
spec:
  containers:
    - name: csi-s3-test-nginx
      image: nginx
      volumeMounts:
        - mountPath: /var/lib/www/html
          name: test
  volumes:
    - name: test
      persistentVolumeClaim:
        claimName: csi-s3-pvc
        readOnly: false

```

12.15 Docker

12.15.1 Project quota functionality

Project quota was introduced to monitor the ephemeral storage usage, and it will be enabled for following specific file systems of XFS format on CentOS7 and Ext4 format on CentOS8 with default soft-limit and hard-limit.

- /data0: (85%, 95%)
- /data0/docker: (80%, 90%)
- /data0/log/bcmt/kube-apiserver: (800M, 1200M)
- /data0/etc: (85%, 95%)

The different project-ids are hardcoded for each specific file system, and corresponding soft-limit and hard-limit values are configured base on the real partition size. If the usage of those FS is between soft-limit and hard-limit, there is no impact within default grace time 7 days, but will be blocked after default grace time; if the usage is above hard-limit, there is no write permission with “No space left on device” error message.

There is the parameter named prjquota_enabled in bcmt_config.json to control this functionality with default false, and the feature could be only enabled during cluster deployment.

Container storage quota limit

Setting this prevents any single container from filling up free storage space in Docker's root directory (default: /data0/docker) shared by every container, risking other containerized services to crash from insufficient storage space.

When `prjquota_enabled` is set, docker service will restrict the overlay storage quota for every container in its root directory. This limit is 1GB for each container. This means that any container that writes data to a path on its container root filesystem that is not mapped to a persistent volume/host/memory backed mount will be restricted by this 1GB quota. Any process in the container trying to create or add objects to this container root filesystem will get denied with a Quota Exceeded message, after this storage quota limit has been reached.

12.16 Storage Classes

A **StorageClass** provides a way for administrators to describe the “classes” of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators. Kubernetes itself is unopinionated about what classes represent. This concept is sometimes called “profiles” in other storage systems.

Example

```
kind: StorageClass
apiVersion: storage.Kubernetes.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
volumeBindingMode: Immediate
```

Parameters in the `bcmt_config.json` file specify which storage options are enabled. Users can select one or more volume types within the same cluster.

Default storage class

Table 38: Default storage class

Storage Type	Feature Flag	Feature Flag	Feature Flag	Config	Default StorageClass
	<code>use_cloud_provider_storage</code>	<code>k8s_shared_storage</code>	<code>k8s_rook_ceph</code>	provider	
Cinder	true	NA	NA	openstack	cinder-az-nova

Table 38: Default storage class (continued)

Storage Type	Feature Flag	Feature Flag	Feature Flag	Config	Default StorageClass
Vsphere	true	NA	NA	vcenter	vsphere
AWSElastic BlockStore	true	NA	NA	aws	aws-gp2
Glusterfs	NA	true	NA	NA	glusterfs-storageclass
Rook	NA	NA	true	NA	rook-ceph-block-rep
Local-Storage	NA	NA	NA	NA	local-storage

Storage class name

When a storage class is deployed, it gets a `storageClass` name that has to be used by the application manifest that requires it. Names used are listed here:

Table 39: Storage class name

Storage class	storageClass name
Ceph RBD (default)	csi-cephrbd
Ceph RBF fast pool	csi-cephrbd-volumes-fast
Ceph FS	csi-cephfs
Local storage – static	local-storage
Local storage – Dynamic	dynamic-local-storage

Provisioner

Storage classes have a provisioner that determines what volume plugin is used for provisioning PVs. This field must be specified.

Reclaim Policy

Persistent Volumes that are dynamically created by a storage class will have the reclaim policy specified in the `reclaimPolicy` field of the class, which can be either `Delete` or `Retain`. If no `reclaimPolicy` is specified when a `StorageClass` object is created, it will default to `Delete`.

Mount Options

Persistent Volumes that are dynamically created by a storage class will have the mount options specified in the `mountOptions` field of the class.



Note: It is possible to create a S3 storage class. Refer to section *Implementation of S3 on NCS* on page 331 for more details.

12.17 Resize partition size

12.17.1 OpenStack Environment

Generally, vdb is specified for /data0 storage, if the size of the volume is too small, and user want to resize the volume without re-installation. Following is a procedure to apply the change.



Note: To extend your data0 block device, make sure the `data0_device` configured in CLCM or in `node_inventory.json` file during deployment.

1. Check that all nodes are in Ready status.

```
# kubectl get nodes
NAME        STATUS    ROLES     AGE      VERSION
test-control-01   Ready    <none>   7h48m   v1.16.4
test-control-02   Ready    <none>   7h40m   v1.16.4
test-control-03   Ready    <none>   7h28m   v1.16.4
test-worker-ege-01   Ready    <none>   8h      v1.16.4
test-worker-ege-02   Ready    <none>   8h      v1.16.4
```

2. Check all nodes' /data0 volume avail space size, detailed requirement of /data0 size refer to *Cluster node requirements* on page 24.

Check if it is too small, like 10G, or usage is above 85% (85%, Kubernetes will evict pod, cluster will run into issue).

```
# df -h | grep /data0
Filesystem  Size  Used  Avail Use% Mounted on
/dev/vdb    50G   19G   28G  41%  /data0
```

3. Resize the volume size from OpenStack dashboard or OpenStack CLI. Following is an example on the dashboard.

- a. Log in to the OpenStack dashboard.
- b. From the **Project** menu, expand the **Compute** menu, click **Instances**.
- c. Choose on node, take test-control-01 for example, from the drop-down menu of **Actions** column, select **Shut off instance**.
- d. Go to **Volumes** → **Volumes** tab, filter /dev/vdb volume on instance test-control-01, from the drop-down menu of **Actions** column, click **Mange Attachments**.

**Note:**

/dev/vdb is /data0's filesystem path, adjust device name depending on your real condition.

- e. Check the instance and click **Detach Volume**.
- f. Filter the volumes, from the drop-down menu of **Actions** column, select **Extend Volume**.
- g. Extend the volume to the size you want, like 60G, 100G, then click the **Extend Volume** button.
- h. Go to the volume, from the drop-down menu of **Actions** column, click **Mange Attachments**.
 - For **Attach to Instance**, choose the node test-control-01 again.
 - For **Device Name**, fill /dev/vdb.
 - Click the **Attach Volume** button.
- i. Go to Compute -> **Instances** tab, click **start instance**.

4. After the node is running, run below command to make the extend volume take into effect.

```
# fsadm resize /dev/vdb
Device does not exist.
Command failed
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/vdb is mounted on /data0; on-line resizing required
old_desc_blocks = 4, new_desc_blocks = 5
The filesystem on /dev/vdb is now 9175040 blocks long.
```

5. Check the node and pod status, all nodes shall be Ready and all pod shall be Running before the operation on next node.

Check that the disk size changed.

```
# df -h | grep data0
```

Check that the node status is *Ready*.

```
# kubectl get nodes
```

Check that the pod status is *Running*.

```
# kubectl get pod -o wide --all-namespaces | grep -v Running
```

12.18 Rack Enablement for CaaS

The division of nodes into logical racks that reflect their physical order, for example in terms of physical proximity, a shared power source or a shared network - prevents data loss in the event of correlated node failures.

When deploying NCS baremetal with racks – each Storage node must be assigned to a rack, both on deployment phase and when scaling out a new storage node.

For more information see **Nokia Container Services Manager Reference Guide for Bare Metal implementation..**

Related information:

- Section: IPMI
- Section: Defining Racks for Storage Node Assignment
- Section: Assigning IPMI Addresses and Availability Zones to Node Groups

12.19 Storage Node Disk Failure and Replacement

This section refers both to **Nokia RM**, **Nokia OR** and **HPGen10 DL380** setups with storage nodes.

The following steps describe how to replace the disk and recreate the OSD which was installed on the disk that failed and required replacement.

When the disk fails or malfunctions, the Ceph OSD which is installed on this disk may be down, and the disk which is located on the JBOD or in HP Storage Node requires a replacement.

 **Note:** AirFrame RM and OR both have JBOD Arrays. It is recommended not to configure JBOD mode but to follow the JBODs array configuration. HP Storage Nodes, for example: HPGen10 DL380 Storage does not use JBODs and it is recommended to have each device configured with virtual drive with Raid 0.

12.19.1 Replacing the Storage Node Disk

This section is relevant for:

- **AirFrame JBOD Disks that failed**
- **HP Storage Nodes.**
- **Dell R740xd Storage Nodes**

To make the replacement, identify the problem disk and identify the OSD # (also known as OSD id) which relates to this physical disk.



Note: When replacing a disk that has malfunctioned, it is important to ensure that the replacement disk is the same size as the one that malfunctioned. This is because the weight of all the disks in the JBOD should be the same, to achieve a uniform split of data among the OSDs.

1. To make the replacement, identify the problematic disk, the OSD # (also known as OSD ID) which relates to this physical disk and also the Storage Controller ID.

```
[cbis-admin@overcloud-controller-0 ~]$ sudo ceph osd tree | grep -i down
#An example output:      0    hdd    5.45799          osd.0
                           down  1.
00000 1.00000
#identifying the Storage Controller ID
[cbis-admin@overcloud-controller-0 ~]$ sudo ceph osd tree
#An example output, in this example we can see that
# host common-overcloud-storage-0, which is overcloud-storage-0
# that has the failed osd with Disk that need replacement
[cbis-admin@overcloud-controller-0 ~]$ sudo ceph osd tree
ID CLASS WEIGHT      TYPE NAME                      STATUS
REWEIGHT PRI-AFF
-7           1.15820 root fast
-19          0.57910    host fast-overcloud-storage-0  24   nvme   0.28955
                           osd.24                  up  1.
00000 1.00000  26   nvme   0.28955                  osd.26
                           up  1.
00000 1.00000
-22          0.57910    host fast-overcloud-storage-1  25   nvme   0.28955
                           osd.25                  up  1.
00000 1.00000  27   nvme   0.28955                  osd.27
                           up  1.
00000 1.00000

-1           131.00198 root common
-5           65.50099   host common-overcloud-storage-0  0    hdd
                           5.45799    osd.0                  down 1.
00000 1.00000  3    hdd    5.45799                  osd.3
                           up  1.
00000 1.00000  5    hdd    5.45799                  osd.5
                           up  1.
00000 1.00000  7    hdd    5.45799                  osd.7
                           up  1.
00000 1.00000  9    hdd    5.45799                  osd.9
                           up  1.
```

```

00000 1.00000 11 hdd 5.45799 osd.11
  up 1.
00000 1.00000 13 hdd 5.45799 osd.13
  up 1.
00000 1.00000 15 hdd 5.45799 osd.15
  up 1.
00000 1.00000 17 hdd 5.45799 osd.17
  up 1.
00000 1.00000 19 hdd 5.45799 osd.19
  up 1.
00000 1.00000 21 hdd 5.45799 osd.21
  up 1.
00000 1.00000 23 hdd 5.45799 osd.23
  up 1.
00000 1.00000
-3      65.50099      host common-overcloud-storage-1 1 hdd
  5.45799      osd.1
  up 1.
00000 1.00000 2 hdd 5.45799 osd.2
  up 1.
00000 1.00000 4 hdd 5.45799 osd.4
  up 1.
00000 1.00000 6 hdd 5.45799 osd.6
  up 1.
00000 1.00000 8 hdd 5.45799 osd.8
  up 1.
00000 1.00000 10 hdd 5.45799 osd.10
  up 1.
00000 1.00000 12 hdd 5.45799 osd.12
  up 1.
00000 1.00000 14 hdd 5.45799 osd.14
  up 1.
00000 1.00000 16 hdd 5.45799 osd.16
  up 1.
00000 1.00000 18 hdd 5.45799 osd.18
  up 1.
00000 1.00000 20 hdd 5.45799 osd.20
  up 1.
00000 1.00000
22 hdd 5.45799 osd.22
  up 1.
00000 1.00000

```

2. Go to the corresponding storage node and retrieve the OSDs device#.

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf
osd.<OSD#> OSD_DEVICE | sed "s/p*[0-9]*$//"
```

Example

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf  
osd.0 OSD_DEVICE | sed "s/p*[0-9]*$/ /"
```

Sample Output

```
sdb
```

3. Retrieve the journal corresponding to this #osd which is down.

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf  
osd.<OSD#> OSD_BLUESTORE_BLOCK_WAL
```

Example

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf  
osd.0 OSD_BLUESTORE_BLOCK_WAL
```

Sample Output

```
/dev/nvme0n1p2
```

4. Identify to which crush map root the OSD fails to belong. This observation is important when creating a new OSD after replacing the disk, because if you are using a fast pool, you will need to add a -t flag specifying the root name of where the OSD will be added (in the section *Creating New OSD on the new Disk*). The system usually identifies this automatically but for you to keep reference, it is better to save the data.

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf  
osd.<OSD#> 'crush_location' | awk '{print $1}' | cut -d '=' -f 2
```

Example

```
[cbis-admin@overcloud-cephstorage-0 ~]# crudini --get /etc/ceph/ceph.conf  
osd.0 'crush_location' | awk '{print $1}' | cut -d '=' -f 2
```

Sample Output

```
common
```



Note: Keep these outputs for *Creating a New OSD on the New Disks*. If your actions have been successful, you will just need to confirm that the `ceph-create-osd-pacific.sh` script is using the correct parameter queried here, otherwise it can be used as an alternative option.

12.19.1.1 Disabling and stopping the failed OSD service

To disable and stop the service, run the following commands and replace <osd_device> with the device that was retrieved in the previous step.

Run the commands as root from the storage node with the failed disk. An example is shown here.

```
[root@overcloud-storage-0 ~]# systemctl stop ceph-osd@<osd_device>.service
[root@overcloud-storage-0 ~]# systemctl disable ceph-osd@<osd_device>.service
```

12.19.2 Locating the disk that must be replaced (AirFrame OR/RM products)

The storcli utility is used to identify the JBOD configuration and identify the location of the disk that needs to be replaced.

1. Connect to the storage node where the disk needs to be replaced.

2. Run the following example command:

```
[root@overcloud-cephstorage-0 ~]# for i in {0..28}; do /usr/share/cbis/utils/mega_raid_util.sh -v $i; done
```



Note: This will identify the Virtual ID of the Disk to be replaced. The example command will identify the Virtual ID of the Disk to be replaced. **If the disk is labeled as device sdg, with OSD ID 7, we need to look for the disk labeled as sdg.** This was identified to have Virtual Device ID: 6

Expected Output

```
Device with Virtual id: 0 is identified by the OS as /dev/sda
Device with Virtual id: 1 is identified by the OS as /dev/sdb
Device with Virtual id: 2 is identified by the OS as /dev/sdc
Device with Virtual id: 3 is identified by the OS as /dev/sdd
Device with Virtual id: 4 is identified by the OS as /dev/sde
Device with Virtual id: 5 is identified by the OS as /dev/sdf
Device with Virtual id: 6 is identified by the OS as /dev/sdg
Device with Virtual id: 7 is identified by the OS as /dev/sdh
Device with Virtual id: 8 is identified by the OS as /dev/sdi
Device with Virtual id: 9 is identified by the OS as /dev/sdj
Device with Virtual id: 10 is identified by the OS as /dev/sdk
Device with Virtual id: 11 is identified by the OS as /dev/sdl
Device with Virtual id: 12 is identified by the OS as /dev/sdm
Device with Virtual id: 13 is identified by the OS as /dev/sdn
Device with Virtual id: 14 is identified by the OS as /dev/sdo
Device with Virtual id: 15 is identified by the OS as /dev/sdp
Device with Virtual id: 16 is identified by the OS as /dev/sdq
Device with Virtual id: 17 is identified by the OS as /dev/sdr
```

```
Device with Virtual iD: 18 is identified by the OS as /dev/sds
Device with Virtual iD: 19 is identified by the OS as /dev/sdt
Device with Virtual iD: 20 is identified by the OS as /dev/sdu
Device with Virtual iD: 21 is identified by the OS as /dev/sdv
Device with Virtual iD: 22 is identified by the OS as /dev/sdw
Device with Virtual iD: 23 is identified by the OS as /dev/sdx
Device with Virtual iD: 24 is identified by the OS as /dev/sdy
Device with Virtual iD: 25 is identified by the OS as /dev/sdz
Device with Virtual iD: 26 is identified by the OS as /dev/sdaa
Device with Virtual iD: 27 is identified by the OS as /dev/sdab
Device with Virtual iD: 28 is identified by the OS as /dev/sdac
```

3. Run the following command, which maps the Virtual Device ID to find the virtual location:

```
[root@overcloud-storage-1 ~]# /opt/MegaRAID/storcli/storcli64 /c0/v6 show
all
```

Expected Output



Note: The outcome below shows that the Virtual device 6 is mapped to Enclosure ID 8 and slot 5.

```
Controller = 0 Status = Success Description = None /c0/v6 :
=====
-----
DG/VD TYPE State Access Consist Cache sCC      Size Name
-----
6/6   RAID 0 Optl  RW     Yes      RWTD  -    5.457 TB
-----PDs for VD 6 :
=====
-----EID:Slt DID State DG      Size Intf Med SED PI SeSz
Model
Sp -----
-----8:5       17 Onln   6 5.457 TB SAS   HDD N   N  512B
ST6000NM0095      U
```

4. Run the following command to identify the location of the disk that will activate the drive LED:



Note: c = controller, e = enclosure and s= slot.

The info could be taken from the storcli64 /cX/vX show all in the previous step.

The command `start locate` will make the disk power LED blink until you later issue the `stop locate` command.

```
[root@overcloud-cephstorage-0 ~]# /opt/MegaRAID/storcli/
storcli64 /c0/ e8/s5 start locate
```

12.19.3 Locating the disk that must be replaced (HP Storage Nodes)

For locating the disk for HP DLXXX nodes, you need to use the `ssaci` utility instead of the `storcli megaraid` utility which is used for AirFrame storage solutions. `ssaci` tools can be downloaded from the following site: https://downloads.linux.hpe.com/SDR/repo/mcp/centos/7/x86_64/current/.

```
[stack@undercloud (stackrc) ~]$ wget
https://downloads.linux.hpe.com/SDR/
repo/mcp/centos/7/x86_64/current/ssa-3.40-3.0.x86_64.rpm
[stack@undercloud (stackrc) ~]$ wget
https://downloads.linux.hpe.com/SDR/
repo/mcp/centos/7/x86_64/current/ssaci-3.40-3.0.x86_64.rpm [stack@undercloud
(stackrc) ~]$ wget
https://downloads.linux.hpe.com/SDR/ repo/mcp/centos/7/x86_64/current/
ssaducli-3.40-3.0.x86_64.rpm
```

After downloading the 3 utilities, copy them to the storage nodes.

```
[stack@undercloud (stackrc) ~]$ scp ssa*.rpm cbis-admin@172.31.0.13:
```

Now install the utility in the Storage Nodes (perform this step for each storage node):

```
[cbis-admin@overcloud-storage-0 ~]$ pwd
/home/cbis-admin [cbis-admin@overcloud-storage-0 ~]
$ sudo rpm -i ssa-3.40-3.0.x86_64.rpm ssaci-3.40-3.0.x86_64.rpm
ssaducli-3.40-3.0.x86_64.rpm
warning: ssa-3.40-3.0.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID
26c2b797: NOKEY
```

If, for example, the disk that failed is identified by the operating system with the label `sdb` (e.g: `/dev/ sdb`), you will need to find the array physical location.

1. Run the following command to find the array index where the disk is located.

```
[root@overcloud-storage-0 ssaci]# for ldisk in {1..30};
do ssaci controller slot=0 logicaldrive
$ldisk show |grep -e "Disk Name" -e "Logical Drive" -e Array;
done
```

Expected Output

```
HPE Smart Array P408i-a SR Gen10 in Slot 0 (Embedded)
  Array A
    Logical Drive: 1
      Disk Name: /dev/sda
      Logical Drive Label: Logical Drive 1 HPE Smart Array P408i-a SR
      Gen10 in Slot 0 (Embedded)
  Array B
    Logical Drive: 2
      Disk Name: /dev/sdb      ===== HPE Smart Array P408i-a SR
      Gen10 in Slot 0 (Embedded)
  Array C
    Logical Drive: 3
      Disk Name: /dev/sdc HPE Smart Array P408i-a SR Gen10 in Slot 0
      (Embedded)
  Array D
    Logical Drive: 4
      Disk Name: /dev/sdd
```

2. Run the following command, which outputs the physical location of a disk.

```
[root@overcloud-storage-0 ssacli]# ssacli controller slot=0 physicaldrive
  all show | grep -A2 "Array B" Array B
  physicaldrive 1I:1:2 (port 1I:box 1:bay 2, SAS HDD, 1.8 TB, OK)
```

3. Run the following commands to activate the drive LED.

```
[root@overcloud-storage-0 ssacli]# ssacli controller slot=0 physicaldrive
  1I:1:2 modify led=on
```

After identifying the disk in the Physical machine and replacing it with a new disk you can use the following command to turn off the led.

```
[root@overcloud-storage-0 ssacli]# ssacli controller slot=0 physicaldrive
  1I:1:2 modify led=off
```

12.19.4 Locating the disk that must be replaced (DELL PowerEdge R740xd with HP740P RAID Controller)

To locate the disk for Dell PowerEdge R740xd nodes, you need to use the `perccli` utility.

1. First, check if the tool is installed in your storage node, as follows:

```
[cbis-admin@overcloud-storage-0 ~]$ sudo rpm -qa | grep perccli
```

2. If there are no results, download the *Linux PERCCLI* utility from the Dell support site.
3. Select your system, then Drivers & Downloads and filter by category **SAS RAID** or by using the keyword **PERCCLI**.
4. You can check your system information with these commands:

```
[cbis-admin@overcloud-storage-0 ~]$ sudo ipmitools fru print | grep
"Product Name"
Product Name : PowerEdge R740xd
```

and

```
[root@overcloud-storage-cbis-0 ~]# uname -a
Linux overcloud-storage-cbis-0
3.10.0-957.27.2.el7.x86_64 #1 SMP Mon
Jul 29 17:46:05 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

5. Once you have obtained the correct version, you can download it directly.

```
[stack@undercloud (stackrc) ~]$ wget -O Perccli_7.1020.tar.gz https://
dl.dell.com/FOLDER05802279M/1/Perccli_7.1020.0000_Linux.tar.gz
[stack@undercloud (stackrc) ~]$ tar -xvf Perccli_7.1020.tar.gz
```

6. After downloading the utility, copy the utilities to the storage nodes as follows:

```
[stack@undercloud (stackrc) ~]$ cd Linux-7.1020; scp perccli*.rpm cbis-
admin@172.31.0.13:
```

7. Now install the utility in the Storage Nodes, (perform this step for each storage node):

```
[cbis-admin@overcloud-storage-0 ~]$ pwd /home/cbis-admin
[cbis-admin@overcloud-storage-0 ~]$ sudo rpm -i
perccli-007.1020.0000.0000-1.noarch.rpm
```

Identifying the Virtual ID

1. It is now necessary to identify the Virtual ID of the Disk to be replaced.
2. The example command shown below, identifies the Virtual ID of the Disk to be replaced. If the disk is labeled as device *sdh*, with OSD ID 7, you need to look for the disk labeled by the OS as *sdh*.
3. This was identified to have a Virtual Device ID: 6.
4. Run the following example command:

```
[root@overcloud-storage-cbis-0 ~]# sed -i -e '/MegaCommand=/s/storcli/
perccli/g' /usr/share/cbis/utils/mega_raid_util.sh
[root@overcloud-cephstorage-0 ~]# for i in
{0..28}; do /usr/share/ cbis/utils/mega_raid_util.sh -v $i; done
```

Expected Output

```

Device with Virtual id: 0 is identified by the OS as /dev/sdb
Device type: SSD Device with Virtual id: 1 is identified by the OS as /
dev/sdc
Device type: SSD Device with Virtual id: 2 is identified by the OS as /
dev/sdd
Device type: SSD Device with Virtual id: 3 is identified by the OS as /
dev/sde
Device type: SSD Device with Virtual id: 4 is identified by the OS as /
dev/sdf
Device type: SSD Device with Virtual id: 5 is identified by the OS as /
dev/sdg
Device type: SSD Device with Virtual id: 6 is identified by the OS as /
dev/sdh
Device type: SSD Device with Virtual id: 7 is identified by the OS as /
dev/sdi
Device type: SSD Device with Virtual id: 8 is identified by the OS as /
dev/sdj
Device type: SSD Device with Virtual id: 9 is identified by the OS as /
dev/sdk
Device type: SSD Device with Virtual id: 10 is identified by the OS as /
dev/sdl
Device type: SSD Device with Virtual id: 11 is identified by the OS as /
dev/sdm
Device type: SSD
Device with Virtual id 12 was not found
Device with Virtual id 13 was not found
Device with Virtual id 14 was not found
Device with Virtual id 15 was not found

```

5. Run the following command, which maps the Virtual Device ID to find the virtual location:

```
[root@overcloud-storage-1 ~]# /opt/MegaRAID/perccli/perccli64 /c0/v6 show
all
```

Expected Output



Note: The outcome below shows that the Virtual device 6 is mapped to Enclosure ID 64 and slot 6.

```

PDs for VD 6 :
=====
-----
```

-----	EID:Sl	DID	State	DG	Size	Intf	Med	SED	PI	SeSz	Model
Sp	Type										
-----	64:6	6	Onln	6	1.745 TB	SATA	SSD	N	N	512B	MZ7LH1T9HMLT0D3
U											

6. Run the following command to identify the location of the disk that will activate the drive LED:



Note: c = controller , e = Enclosure , s = slot. The info could be taken from the `perccli64 / cx/vX show all` in the previous step. The `start locate` command will make the disk power LED blink until you issue the `stop locate` command.

```
[root@overcloud-cephstorage-0 ~]# /opt/MegaRAID/perccli/
perccli64 /c0/ e8/s5 start locate
```

12.19.5 Locating the Physical Storage Node and its JBOD AirFrame Storage Nodes



Note: This step is relevant only for AirFrame OR , as in this system (unlike AirFrame RM), the `start locate` command will not cause the disk power LED to blink in a manner that you can find on the disk. For AirFrame RM, look for the blinking LED on the disk.

Run the following command to find the Storage Node location controller:

```
[cbis-admin@overcloud-cephstorage-0 ~]$ sudo ipmitool chassis identify
255{{}}
```

Expected Output

```
Chassis identify interval: 255 seconds
```

You now have approximately 4 minutes to find the storage node where the identification LED is blinking. Once the storage controller is found, follow the SAS controller cable from the storage controller to the JBOD. This will lead you to the corresponding JBOD.

12.19.6 Replacing the Physical Disk

We assume that the admin identified the JBOD and the slot of the problematic disk, and that the malfunctioning disk has been physically removed and that a new disk has been attached into the JBOD to the same slot. For more information, see *Locating the disk that must be replaced (AirFrame OR/RM products)* on page 352 and *Locating the disk that must be replaced (HP Storage Nodes)* on page 354.

12.19.6.1 Nokia AirFrame Products (RM/OR) and Dell PowerEdge 740xd Support

 **Note:** When working on a Dell PowerEdge R740xd, for every `storcli` command mentioned in this manual, use instead the `perccli` command.

HDD indexes are also described in the JBODs chart. To view the chart, the JBOD must be pulled out.

HDD14s are located in the middle part of the JBOD.



The graphic below is an example of a chart which shows where a virtual device can be mapped to a physical disk location.

Front side of JBOD Chassis	Rear side of JBOD Chassis	Virtual ID:	Physical Location	Disks label on Storage Node					
				SSD	Write Policy: Write Through	Boot disk	RAID1	RAID0	
P1:01: 0	P0:01: 0	0	HDD0	sda	X	X	X	X	
	P0:01: 3	1	HDD3	sdb					X
	P0:01: 4	2	HDD4	sdc					X
	P0:01: 5	3	HDD5	sdd					X
	P0:01: 6	4	HDD6	sde					X
	P0:01: 7	5	HDD7	sdf					X
	P0:01: 8	6	HDD8	sdg					X
	P0:01: 9	7	HDD9	sdh					X
	P0:01: 10	8	HDD10	sdj					X
	P0:01: 11	9	HDD11	sdj					X
	P0:01: 12	10	HDD12	sdk					X
	P0:01: 13	11	HDD13	sdl					X
	P0:01: 14	12	HDD14	sdm					X
P1:01: 3		13	HDD3	sdn					X
P1:01: 4		14	HDD4	sdo					X
P1:01: 5		15	HDD5	sdp					X
P1:01: 6		16	HDD6	sdq					X
P1:01: 7		17	HDD7	sdr					X
P1:01: 8		18	HDD8	sds					X
P1:01: 9		19	HDD9	sdt					X
P1:01: 10		20	HDD10	sdu					X
P1:01: 11		21	HDD11	sdv					X
P1:01: 12		22	HDD12	sdw					X
P1:01: 13		23	HDD13	sdx					X
P1:01: 14		24	HDD14	sdy					X
	P0:01: 1	25	HDD2	sdz	X	X			X
	P0:01: 2	26	HDD3	sdaa	X	X			X
P1:01: 1		27	HDD2	sdab	X	X			X



Note: This is **NOT** applicable for AirFrame RM where the locate will identify the disk devices.

Therefore, there is no need for the mapping table below which shows where a virtual device can be mapped to a physical disk location.

- Configure the new disk back in JBODConfigure using the `perccli` command while connecting to the storage node itself. The disk will probably be identified as Foreign.
- To add it back, run the following command by specifying the Physical location, where 24 is the enclosure and 6 is the slot id (per our example of course).

```
/opt/MegaRAID/perccli/perccli64 /c0 add vd type=r0 drives=24
```

Example: Disk shows UnGood State

You can check by viewing the status of the disk using the following command for example:

```
[root@overcloud-storage-0 ~]# /opt/MegaRAID/perccli/perccli64 /c0/ e8/s5
show Controller = 0 Status = Success Description = Show Drive Information
Succeeded
Drive Information:
-----
EID:Slt DID State DG Size Intf Med SED PI SeSz Model Sp
-----
8.5 6 UGood F 5,457 TB SAS HDD N N 512B ST6000NM0095 U
-----
```

Then run the following command to clear the cache data of this RAID configuration:

```
[root@overcloud-storage-0 ~]# /opt/MegaRAID/perccli/perccli64 /c0
/fall import Controller = 0
Status = Success
Description = Successfully imported foreign configuration
```

Verify the status of the HDD in JBOD:

```
[root@overcloud-storage-0 ~]# /opt/MegaRAID/perccli/perccli64 /
c0/ e8/s5 show Controller = 0 Status = Success Description = Show
Drive Information Succeeded.
Drive Information:
-----
EID:Slt DID State DG Size Intf Med SED PI SeSz Model Sp
-----
8.5 6 Onln 12 5,457 TB SAS HDD N N 512B ST6000NM0095 U
-----
```



Note: If the node showed as Ubad and NOT as UGood. 252:4 2 UBad - 372.093 GB SATA SSD N N 512B ST6000NM0095 U, execute the following command:

```
/opt/MegaRAID/perccli/perccli64 /c0 /e252/s5 set good /opt/MegaRAID/perccli/perccli64 /c0 /fall import
```

The HDD comes back online and will be identified again in the operating system, with the same label that the operating system identified it previously (before the replacement).

12.19.6.2 HP Storage Nodes Support

After replacing the Disk there will be a Failed indication in the OA (Onboard administration).

There will be also an indication by the OS of a failure in the Logical volume.

```
[root@overcloud-storage-0 ~]# ssaci ctrl slot=0 show config HPE Smart Array P204i-b SR Gen10 in Slot 0 (Embedded) (sn: PEYHF0KLMF007) Internal Drive Cage at Port 1I, Box 1, OK
Port Name: 1I (Mixed) Array A (SAS, Unused Space: 0 MB) logicaldrive 1
(1.64 TB, RAID 0, OK)
physicaldrive 1I:1:1 (port 1I:box 1:bay 1, SAS HDD, 1.8 TB, OK) Array B (SAS,
Unused Space: 0 MB) logicaldrive 2 (1.64 TB, RAID 0, Failed)
physicaldrive 1I:1:2 (port 1I:box 1:bay 2, SAS HDD, 1.8 TB, OK) SEP (Vendor ID
HPE, Model Smart Adapter) 379 (WWID: 500143804171008C,
Port: Unknown)

The disk label will also disappear
[root@overcloud-storage-0 ~]# lsblk
NAME      MAJ:MIN RM    SIZE RO TYPE  MOUNTPOINT
sda        8:0      0   447.1G  0 disk  ##
sda1       8:1      0      4M  0 part
sda2       8:2      0   447.1G  0 part  /
# #the label of sdb disappears here
sdc        8:32     0      3G  0 disk
```



Note: This is just an illustration, we do not list all devices here.

To fix this state of the logical volume run the following command:

```
[root@overcloud-storage-0 ~]# ssaci ctrl slot=0 ld 2 modify reenable
```



Warning: Any previous existing data on the logical drive may not be valid or recoverable.
Continue? (y/n) y.

After that, the logical volume should be changing its status to OK and you can verify it by running the `show config` command again:

```
[root@overcloud-storage-0 ~]# ssaci ctrl slot=0 show config HPE Smart Array P204i-b SR
Gen10 in Slot 0 (Embedded) (sn: PEYHF0KLMBF007)
Internal Drive Cage at Port 1I, Box 1, OK
Port Name: 1I (Mixed)      Array A (SAS, Unused Space: 0 MB)
logicaldrive 1 (1.64 TB, RAID 0, OK)
physicaldrive 1I:1:1 (port 1I:box 1:bay 1, SAS HDD, 1.8 TB, OK)      Array B (SAS,
Unused Space: 0 MB)      logicaldrive 2 (1.64 TB, RAID 0, OK)
physicaldrive 1I:1:2 (port 1I:box 1:bay 2, SAS HDD, 1.8 TB, OK)      SEP (Vendor ID
HPE, Model Smart Adapter) 379 (WWID: 500143804171008C, Port: Unknown)
```

The OA (onboard administration) indication will also disappear at this stage. The OS disk label will reappear.

```
[root@overcloud-ovscompute-1 ~]# lsblk | head -10
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda     8:0     0 447.1G  0 disk ##
sda1    8:1     0    4M  0 part ##
sda2    8:2     0 447.1G  0 part /
sdb     8:16    0 1.7T  0 disk
```

 **Note:** The above is listed just for illustration, we do not list all devices here.

At this stage we are ready to continue to [Creating a new OSD on the New Disks](#) on page 362.

12.19.7 Creating a new OSD on the New Disks

1. Connect to the storage node as the user root.
2. Run a script that will show how to re-create the disk.
3. Use the OSD device that was retrieved earlier.

```
[root@overcloud-cephstorage-0 ~]# /usr/share/cbis/undercloud/
tools/ceph-create-osd-pacific.sh -d /dev/<osd_device> -m
<cluster_manager_provision_ip>
```

Example

```
[root@overcloud-cephstorage-0 ~]# /usr/share/cbis/undercloud/tools/ceph-
create-osd-pacific.sh -d /dev/sdb
```

Sample Output

```
[root@overcloud-storage-0 ~]# bash /usr/share/cbis/undercloud/tools/ ceph-create-osd.sh -d /dev/sdb
```

OSD 0 is configured on the selected device /dev/sdb , stop and cleanup the osd as follows:

1. Mark out the osd from the cluster `ceph osd out osd.0`
2. Stop the OSD service using `systemctl stop ceph-osd@sdb`



Important: Wait for Ceph to rebalance the data; (until rebalancing finishes and there remain no degraded objects). You can run the command below to watch: `/usr/share/cbis/undercloud/tools/ceph-wait-for-healt-warn.sh`

3. Then continue with the following commands: `ceph auth del osd.0`, `ceph osd crush rm osd.0` and `ceph osd rm osd.0`.
4. Zap the device and create a GPT label `sgdisk -Z /dev/sdb` `sgdisk -g /dev/sdb` `sgdisk -Z /dev/sdb2` `sgdisk -Z /dev/nvme0n1p1` `sgdisk -Z /dev/nvme0n1p2`
5. Delete the docker prepare operation `docker rm ceph-osd-prepare-overcloud-storage-0-sdb`
6. #Run the script again `bash /usr/share/cbis/undercloud/tools/ceph-create-osd.sh -d /dev/sdb j /dev/nvme0n1p2 -t common.`
7. Execute: `sudo crudini --get --format=ini /etc/ceph/ceph.conf osd.3`



Note:

1. `-m` flag is mandatory.
2. To recreate an osd on a disk that **already has partitions for other osd's block.db and journal**, you need to specify the disk partitions for: **-d** the block device disk **-b** the block.db and **-j** the journal.
3. If the recreation of osd is for an empty disk, specify **-d** and put the disk name.
4. The **-t** option is relevant only on setups with ceph fast pool. When you recreate the osd, specify if its for "common" pool or "fast" pool. (**-t** has only two options)

12.20 Creating and Recreating a Specific OSD

12.20.1 Symptoms of Unhealthy OSDs and when to Recreate them

Usually, when a `ceph -s` command is issued, it returns a Non health status of Ceph.

You may want to recreate specific OSDs for various reasons, due to the reporting of unhealthy OSDs, for example:

1. OSD is down and fails to restart.



Note: Restart the OSD command example for OSD with ID 1.

- To identify the service command, use the helper script in the compute where this OSD resides:

Example

```
[cbis-admin@overcloud-storage-0 ~]$ sudo crudini --get --format=ini /etc/ceph/ceph.conf osd.18
```

Sample Output

```
[osd.18] OSD_BLUESTORE_BLOCK_DB = /dev/nvme1n1p11
OSD_DEVICE = sdm1 OSD_BLUESTORE_BLOCK_WAL = /dev/nvme1n1p12
BLOCK_DEVICE = /dev/sdm2
SERVICE_NAME = ceph-osd@sdm.service HOSTNAME = common-overcloud-
storage-0
osd crush location = root=common host=common-overcloud-storage-0
```

2. The disk on which this OSD is installed has been replaced. In this case, the data is lost and rebalanced among the other Ceph OSDs. This OSD must be recreated.
3. There are some computes that do not have the same disk layout as all the other computes. These computers must be specifically handled.
4. The OSD refuses to get back to healthy state and no other way was found to recover.

12.20.2 Creating and Recreating an OSD- Overview

In case of a failed/replaced disk due to one of the symptoms described above, its OSD can be recreated by using the `ceph-create-osd-pacific.sh` script, located in `/usr/share/cbis/undercloud/tools/` from the node with the damaged OSD.

`ceph-create-osd-pacific.sh` is used to easily create and recreate OSDs. This provides detailed steps of how to delete an existing OSD before creating a new one. For detailed information about all available commands, execute:

```
sudo bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -h
```

Expected Output

```
[root@f1-803-hp-bm01-storagebm-0 ~]# sudo bash /usr/share/cbis/undercloud/tools/ceph-create-osd-nautilus.sh -h
Usage: ceph-create-osd-nautilus.sh -d [device name] -j [journal name] -c -w -a | -h
  -d device_name:
    This is the device name where the osd will be installed.
    Usually the utility assumes that the device is empty zapped with sgdisk and marked with gpt partition
    before it starts installing the OSD.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdb
    If there are existing partitions on the device, and you want the utility to create additional
    partition for the OSD installation then specify -c flag without specifying device name after it.
  -j journal_name: Optional
    The device name of the journal to be used for the installed OSD.
    If not specified, the journal will reside on the same device where the osd is installed upon.
    If specified, the osd will create the journal on the device specified.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdb -j /dev/sdj
    If you specify an existing partition in a device it will be used and not created as oppose to the above.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdb -j /dev/sdj
  -b blockdb_name: Optional
    The device name of the blockdb (metadata) to be used for the installed OSD.
    If specified, the OSD will create the blockdb on the device specified.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdb -b /dev/sdb
    If you specify an existing partition in a device, it will be used and not created as oppose to the above.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdb -b /dev/sdb
  -c create_partition: default false
    if this flag is set then the the device specified where osd is to be installed with,
    will have an additional partition created (as a last partition) only if there is extra space
    for additional partition.
    When this flag is NOT set the osd will be installed on the an existing disk.
    System will create a partition for the OSD and for the journal if resides on the same device.
    If a journal is specified then only one partition will be created for the osd assuming the journal
    is on another device.
    Example: ceph-create-osd-nautilus.sh -d /dev/sdj -c
    In the example above on device sdj if extra space is available a partition will be created
    and an osd will be created on this created partition.
  -w wait_for_health: default false
    if you set this flag then after creation of the osd there will be a consequent query
    to wait for ceph status to change to HEALTH OK.
```

12.20.3 Example 1 - Recreating an OSD (block.db and journal partitions reside on the same disk)

i Note: If a backup has not been performed, do so now. It is important to perform a backup before any major (potentially harmful) change, as detailed later in this section. The backup ensures that there is a rollback option in case of failure.

i Note: As an existing OSD has been removed, make sure that the number of replicas is greater than 1 so that data will not be lost.

12.20.3.1 Preparation Phase for Example 1

- To see the all existing OSDs and their status, run `ceph osd tree`:

ID	CLASS	WEIGHT	TYPE	NAME	STATUS	REWEIGHT	PRI-AFF
9		4.39062	root	fast			
-25		2.39062	rack	Rack1-fast			
-19		2.39062	host	fast-f1-803-hp-bm01-storagebm-0			
2	ssd	1.39062		osd.2	up	1.00000	1.00000
36	ssd	1.00000		osd.36	up	1.00000	1.00000
-15		2.00000	rack	Rack2-fast			
-21		2.00000	host	fast-f1-803-hp-bm01-storagebm-1			
37	ssd	1.00000		osd.37	up	1.00000	1.00000
41	ssd	1.00000		osd.41	up	1.00000	1.00000
-1		24.00000	root	common			
-27		12.00000	rack	Rack1-common			
3		12.00000	host	common-f1-803-hp-bm01-storagebm-0			
0	ssd	1.00000		osd.0	up	1.00000	1.00000
3	ssd	1.00000		osd.3	up	1.00000	1.00000
6	ssd	1.00000		osd.6	up	1.00000	1.00000
10	ssd	1.00000		osd.10	up	1.00000	1.00000
13	ssd	1.00000		osd.13	up	1.00000	1.00000
15	ssd	1.00000		osd.15	up	1.00000	1.00000
18	ssd	1.00000		osd.18	up	1.00000	1.00000
21	ssd	1.00000		osd.21	up	1.00000	1.00000
24	ssd	1.00000		osd.24	up	1.00000	1.00000
27	ssd	1.00000		osd.27	up	1.00000	1.00000
30	ssd	1.00000		osd.30	up	1.00000	1.00000
33	ssd	1.00000		osd.33	up	1.00000	1.00000
-17		12.00000	rack	Rack2-common			
-7		12.00000	host	common-f1-803-hp-bm01-storagebm-1			
1	ssd	1.00000		osd.1	up	1.00000	1.00000
4	ssd	1.00000		osd.4	up	1.00000	1.00000
7	ssd	1.00000		osd.7	up	1.00000	1.00000
9	ssd	1.00000		osd.9	up	1.00000	1.00000
12	ssd	1.00000		osd.12	up	1.00000	1.00000
16	ssd	1.00000		osd.16	up	1.00000	1.00000
19	ssd	1.00000		osd.19	up	1.00000	1.00000
22	ssd	1.00000		osd.22	up	1.00000	1.00000
26	ssd	1.00000		osd.26	up	1.00000	1.00000
29	ssd	1.00000		osd.29	up	1.00000	1.00000
32	ssd	1.00000		osd.32	up	1.00000	1.00000
35	ssd	1.00000		osd.35	up	1.00000	1.00000

- Run `lsblk` from the node with a damaged OSD to see the disk hardware layout.

Sample Output

sd*	8:15	0	447.1G	0	disk
-ceph-45cea223-0f12-44dc-b27a-22add9aa11b-osd	-block--2b3bbc0f-93ba-4300-8a13-9605124a6217	23:19	0	3.5T	0 disk
-ceph-45d102e-5fd1-4a51-a3a8-d8cccf249ead-osd	-block--eb48ab7-876f-4b7d-9d9-09cccd9dceea	253:7	0	3.5T	0 lvm
-ceph-4a362722-c44f-47a5-a37b-adefb6ed6ed5-osd	-block--7f1b0ff0d-a19d-4646-a52d-1fb1842184ed	253:8	0	3.5T	0 disk
-ceph-ac611d2c-8ef0-4659-ace5-a6f1f4fd5385-osd	-block--e6bdad7a-2ec9-4656-beef-59015909e549	253:9	0	3.5T	0 lvm
-ceph-269a07fa-2079-4498-a34f-b7bfe583272-osd	-block--e9bd79df-02e5-4e79-9d08-4ab592bffffd5	253:10	0	3.5T	0 lvm
-ceph-4b7d70ac-3055-4b27-85b3-813f0f289bb0-osd	-block--e62ab2c0-1c76-40e2-b20f-d94bccc039bf	253:11	0	3.5T	0 lvm
-ceph-c854f338-8c7c-46e4-ae8d-8b29cd4524f4-osd	-block--06a0b591-c08d-40e4-9b3f-2ac7fb3e814b	253:12	0	3.5T	0 lvm
-nvme0n1p1		8:176	0	3.5T	0 disk
-nvme0n1p2		8:177	0	3.5T	0 disk
-nvme0n1p3		259:13	0	10G	0 part
-nvme0n1p4		259:7	0	10G	0 part
-nvme0n1p5		259:25	0	10G	0 part
-nvme0n1p6		259:27	0	10G	0 part
-nvme0n1p7		259:21	0	1G	0 part
-nvme0n1p8		259:29	0	10G	0 part
-nvme0n1p9		259:28	0	10G	0 part
-nvme0n1p10		259:10	0	1G	0 part
-nvme0n1p11		259:13	0	1G	0 part
-nvme0n1p12		259:25	0	10G	0 part
-nvme0n1p13		259:27	0	10G	0 part
-ceph-54867213-6595-40ea-82c6-6e03aa3ad0fb-osd	-block--6465231e-5666-40cf-beed-4eb7fd4b36da	253:14	0	1.4T	0 lvm
-nvme0n1p14		259:22	0	10G	0 part
-nvme0n1p15		259:20	0	10G	0 part
-nvme0n1p16		8:0	0	447.1G	0 disk
-sda1		8:1	0	1M	0 part
-sda2		8:2	0	10G	0 part
-sda5		8:5	0	337.1G	0 part
-sda6		8:6	0	10G	0 part
-ceph-8a01e677-bbac-4505-b666-6de34e18ae5-osd	-block--724d9a9b-dfb1-4512-8986-ff59d362fd5c	253:9	0	100G	0 part /data0
-ceph-4c48f6b5-de99-45a1-88fc-77279690b042-osd	-block--eff7557bc-b9f9-41e2-8b25-bd901500408b	253:10	0	3.5T	0 disk
-ceph-269a07fa-2079-4498-a34f-b7bfe583272-osd	-block--e9bd79df-02e5-4e79-9d08-4ab592bffffd5	253:11	0	3.5T	0 disk

3. Assuming that we want to recreate osd.36 in storage 0 then proceed to the following steps.

4. Find the disk that osd.36 is used on:

```
sudo crudini --get --format=ini /etc/ceph/ceph.conf osd.36
```

Expected Output

In this output, osd.36 is located on a nvme0n1p13 disk.

5. Run the command using /dev/nvme0n1p13 as the disk parameter:

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/nvme0n1p13
```

There are 2 possible outcomes:

- An existing OSD has been detected by the ceph-create-osd-pacific.sh on disk, in our case, nvme0n1p13. See [Scenario 1 - Preparation Phase for Example 1](#) on page 366.
- No OSD has been detected on the nvme0n1p13 disk, and OSD creation is being performed. See [Scenario 2 - Preparation Phase for Example 1](#) on page 368.

12.20.3.1.1 Scenario 1 - Preparation Phase for Example 1

Conditions

- If an existing OSD has been detected by the ceph-create-osd-pacific.sh on the nvme0n1 disk.
- If an existing OSD was present (the OSD to be recreated), the ceph-create-osd-pacific.sh script will print a detailed procedure of how to delete the current OSD existing on disk (for our example, nvme0n1p13).
- Follow the procedure described in the expected output shown here:

```

bash /usr/share/cbis/undercloud/tools/ceph-create-osd-nautilus.sh -d /dev/nvme0n1p13 -t fast -m 172.31.7.254
# ..... End copy block .....#
[root@fi-803-hp-bm01-storagebm-0 ~]# c
[root@fi-803-hp-bm01-storagebm-0 ~]# bash /usr/share/cbis/undercloud/tools/ceph-create-osd-nautilus.sh -d /dev/nvme0n1p13
OSD 36 is running on the selected device /dev/nvme0n1p13, stop and cleanup osd. Copy & paste the following steps:
# ..... Start first copy block .....#
# Mark out the osd from the cluster
ceph osd out osd.36
# stop the OSD service
systemctl stop ceph-osd@36
# ..... End first copy block, copy this command: .....#
systemctl disable ceph-osd@36
# ..... Start second copy block .....#
# Important: wait for ceph to rebalance the data.
# until rebalancing finishes and no deleted objects, you can run the command below to watch
/usr/share/cbis/undercloud/tools/ceph-wait-for-healt-warn.sh
# Then continue with the following commands
ceph osd destroy 36 -yes-i-really-mean-it
ceph osd crush rm osd.36
ceph osd rm osd.36
# Zap the device
vgremove ceph-ac57bc25-7a5b-4e55-b9ea-b6252c13ae9b -f
# Run the script again
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-nautilus.sh -d /dev/nvme0n1p13 -t fast -m 172.31.7.254
# ..... End copy block .....#
[root@fi-803-hp-bm01-storagebm-0 ~]#

```

Actions

Execute the following commands(in this specific order):

1. Take out the OSD:

```
ceph osd out osd.36
```

2. Stop and disable the OSD process:

```
systemctl stop ceph-osd@36
systemctl disable ceph-osd@36
```

3. Run the following script and wait until rebalancing of Ceph is finished.

```
/usr/share/cbis/undercloud/tools/ceph-wait-for-healt-warn.sh
```

4. Destroy the OSD:

```
ceph osd destroy 36 -yes-ireally-mean-it
```

5. Remove OSD from crush:

```
ceph osd crush rm osd.36
```

6. Remove OSD from Ceph:

```
ceph osd rm osd.36
```

7. Remove the Volume Group which was created with the Physical Volume that's on nvme0n1p13:

```
vgremove ceph-ac57bc25-7a5b-4e55-b9ea-b6252c13ae9b -f
```

8. Remove the Physical Volume that was created with nvme0n1p13:

```
pvremove /dev/nvme0n1p13 -f -f
```

By executing all the steps detailed above, the selected OSD will be safely removed and zap all related partitions will be zapped from nvme0n1p13 disk.



Note: These steps are dynamic and may differ for a different OSD and its location. Always use the `ceph-create-osd-pacific.sh` script to generate the deletion procedure.

- Execute last command in the procedure, to create the disk:

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/nvme0n1p13 -t fast -m 172.31.7.254
```

12.20.3.1.2 Scenario 2 - Preparation Phase for Example 1

Conditions

- No OSD has been detected on the nvme0n1 disk and OSD creation is being performed on a selected disk.

Actions

- In this case, no older OSD was detected on the nvme0n1 disk and therefore you will be able to see that the OSD is being created as expected.
- Wait until the procedure has completed.

Expected Output

```
OSD id (osd.3) is up
osd_id after activate is: 3
osd_size: 222800876 for osd_id: 3
calculated_weight: 0.207499 for osd_id: 3
current_weight: 0.207489 for osd_id: 3
current weight is in the acceptable range (+/- 5%)
Setting read permissions on /etc/ceph/ceph.client.admin.keyring
Working on container ceph-osd-overcloud-ovscompute-1-sdb:
  OSD_DEVICE: sdb1
  OSD_PATH: /var/lib/ceph/osd/ceph-3
  OSD_ID: 3
  OSD_BLUESTORE_BLOCK_DB: /dev/sdb3
  OSD_BLUESTORE_BLOCK_WAL: /dev/sdb4
  BLOCK_DEVICE: /dev/sdb2
  SERVICE_NAME: ceph-osd@sdb.service
Working on container ceph-osd-overcloud-ovscompute-1-sda2:
  OSD_DEVICE: sda2
  OSD_PATH: /var/lib/ceph/osd/ceph-1
  OSD_ID: 1
  OSD_BLUESTORE_BLOCK_DB: /dev/sda4
  OSD_BLUESTORE_BLOCK_WAL: /dev/sda5
  BLOCK_DEVICE: /dev/sda3
  SERVICE_NAME: ceph-osd@sda2.service
Running Ceph post install Finished Successfully
[root@overcloud-ovscompute-1 ~]#
```

Results

The OSD that has been recreated successfully on the nvme0n1 disk is highlighted with the red marker above with details on all OSDs, their partitions and locations.

Validation

Now, make sure that the osd has been recreated as follows:

1. Run `ceph osd tree` again and check if the newly-created osd is present.
2. Ensure that the `ceph -s` command shows an **healthy ceph environment**.
3. Run `lsblk` to see if all osd partitions exist on sda.

12.20.4 Example 2 – Recreating an OSD that uses other disks for Journal and block.dbd

If the OSDs Journal (`OSD_BLUESTORE_BLOCK_WAL`) and the OSDs metadata

(`OSD_BLUESTORE_BLOCK_DB`) sit on another drive (NVMe or SSD), you need to specify them when running the `ceph-create-osd-pacific.sh` script.

In this example, we will recreate osd.1 on a storage node with NVMe disks.

- ### **1. Find the osd.1 physical disk:**

```
sudo crudini --get --format=ini /etc/ceph/ceph.conf osd.3
```

Expected Output

```
[root@fi-803-hp-bm01-storagegbm-0 ~]# sudo crudini --get --format=inini /etc/ceph/ceph.conf osd.3
[osd.3]
crush_location = root=common rack=Rack1-common host=common-fi-803-hp-bm01-storagegbm-0
OSD_DEVICE = sdd
BLOCK_DEVICE = /dev/ceph-af5aeca8-d21c-4cbe-888e-f8e1fea4eca3/osd-block-fab23326-5918-4eaе-a68d-c00e32a81cb0
OSD_BLUESTORE_BLOCK_DB = /dev/vme0np1p3
OSD_BLUESTORE_BLOCK_WAL = /dev/vme0np1p4
SERVNAME_NAME = ceph-osd.3.servname
HOSTNAME = common-fi-803-hp-bm01-storagegbm-0
[osd.4]
osd_id = 1
osd_weight = 1
osd_max_size = 1000000000000000000
```



Note: We can see that OSD_BLUESTORE_BLOCK_DB and OSD_BLUESTORE_BLOCK_WAL are on an NVMe disk (not on sdd), but the OSD_DEVICE is on the sdd disk.

2. Run the command using '/dev/sdd' as the disk parameter (same as in Example 1):

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/sdd
```

Expected Output

```

root@BB-50-bmc:~# storagepool -l /var/share/cbs/undercloud/tools/ceph-create-osd-nautilus.sh -d /dev/sdd
# 3 is running on the selected device /dev/sdd, stop and cleanup osd. Copy & paste the following steps:
#----- start first copy block -----
#----- put the osd from the cluster
ceph osd rm 3
#----- stop the OSD service
systemctl stop ceph-osd@3
#----- first copy block, copy this command: -----
systemctl disable ceph-osd@3
#----- start second copy block -----
#----- wait until rebalancing finishes and no degraded objects, you can run the command below to watch
#----- until rebalancing finishes and no degraded objects, you can run the command below to watch
#----- e.g. until rebalancing finishes and no degraded objects, you can run the command below to watch
#----- /usr/share/cbs/undercloud/tools/ceph-wait-for-health-warn.sh
#----- run the following command
ceph osd destroy 3 -yes-i-really-mean-it
ceph osd crush rm 3
#----- Zap the device
sgdisk -Z /dev/nvme0nlp3
sgdisk -Z /dev/nvme0nlp3
sgdisk -Z /dev/nvme0nlp4
sgdisk -g /dev/nvme0nlp4
sgdisk --zap-all
#----- destroy the device
fdisk -l /dev/nvme0nlp3 >/dev/nvme0nlp3 --destroy
#----- Run the script again
cash /usr/share/cbs/undercloud/tools/ceph-create-osd-nautilus.sh -d /dev/sdd -r /dev/nvme0nlp4 -b /dev/nvme0nlp3 -t common -m 172.31.7.254

```

Results

As we can see, the script located all the OSD partitions and printed detailed steps of how to remove the existing osd.

Actions

Run all the procedure that is specified in the output:

```
ceph osd out osd.3
systemctl stop ceph-osd@3
systemctl disable ceph-osd@3
/usr/share/cbis/undercloud/tools/ceph-wait-for-health-warn.sh
ceph osd destroy 3 --yes-i-really-mean-it
```

```
ceph osd crush rm osd.3
ceph osd rm osd.3
# Zap the device
sgdisk -Z /dev/nvme0n1p3
sgdisk -g /dev/nvme0n1p3
sgdisk -Z /dev/nvme0n1p4
sgdisk -g /dev/nvme0n1p4
ceph-volume lvm zap /dev/sdd --destroy
```

Run the script again as follows:

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/sdd -j /dev/nvme0n1p4 -b /dev/nvme0n1p3 -t common -m 172.31.7.254
```

Further Actions

Ensure that the osd has been recreated:

- a. Run `ceph osd tree` again and check if the newly-created osd is present.
- b. Ensure that the `ceph -s` command shows an **healthy ceph environment**.
- c. Run `lsblk` to see if osd partitions exist on sda and on NVMe Disks.

12.20.5 Example 3 - Creating an OSD on a Root Disk

Recreating a specific OSD on a root disk



Note: The OSD on the root disk is not relevant for AirFrame HD nor AirFrame OR storage nodes and their respective setups.



Note: OSDs can only be recreated. New OSDs cannot be created.

Condition

Root is on a disk named "sda" and stores the root folder and file-system.

Action

Run `lsblk`.

Expected Output

```
[root@overcloud-ovscompute-1 ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda     8:0    0 223.6G  0 disk
|---sda1 8:1    0   1M  0 part
|---sda2 8:2    0 100M  0 part --> OSD's device
|---sda3 8:3    0 149.5G 0 part --> OSD's Block device
|---sda4 8:4    0   1G  0 part --> OSD's Block DB (Metadata)
|---sda5 8:5    0   10G 0 part --> OSD's Journal
|---sda6 8:6    0   4M  0 part
|---sda7 8:7    0   63G 0 part /
sdb     8:16   0 223.6G 0 disk
|---sdb1 8:17   0 100M 0 part
|---sdb2 8:18   0 212.5G 0 part
|---sdb3 8:19   0   1G 0 part
|---sdb4 8:20   0   10G 0 part
[root@overcloud-ovscompute-1 ~]#
```

Results

As we can see, sda7 is our root partition and should not be touched. Therefore, in this case we **do not** want to zap the entire sda disk, but only the 4 partitions related to the osd that is installed on sda.

Further Actions

Now run the `ceph-create-osd.sh` script on `/dev/sda2` (where the OSD device is installed) and not on the entire `/dev/sda` disk:

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/sda2
```

Follow the procedure described in the output (as described in *Example 1 - Recreating an OSD (block.db and journal partitions reside on the same disk)* on page 365.

Expected Output

```
[root@overcloud-ssdcompute-noam-0 ~]# bash /usr/share/cbis/undercloud/tools/ceph-create-osd.sh -d /dev/sda2
/dev/sda2
OSD 0 is running on the selected device /dev/sda2 , stop and cleanup osd follow the following steps:
# ..... Start copy block .....
# Mark out the osd from the cluster
ceph osd out osd.0
#stop the OSD service
systemctl stop ceph_osd@osd.0
#Important! wait ceph to rebalance the data..
#After rebalancing finishes and no degraded objects, you can run the command below to watch
/usr/share/cbis/underCloud/tools/ceph-wait-for-health-warn.sh
#Then continue with the following commands
ceph auth del osd.0
ceph osd crush rm osd.0
ceph osd rm osd.0
#zap the device and create a GPT label
sgdisk -Z /dev/sda3
sgdisk -g /dev/sda3
sgdisk -Z /dev/sda4
sgdisk -g /dev/sda4
sgdisk -Z /dev/sda5
sgdisk -g /dev/sda5
sgdisk -Z /dev/sda2
sgdisk -g /dev/sda2
partprobe /dev/sda
#Delete the device prepare operation
dmcrypt-prepare-overcloud-ssdcompute-noam-0-sda2
#run the script again
bash /usr/share/cbis/undercloud/tools/ceph-create-osd.sh -d /dev/sda2 -j /dev/sda5 -b /dev/sda4
# ..... End copy block .....
[root@overcloud-ssdcompute-noam-0 ~]#
```

More Actions

After executing the steps described above, rerun the script:

```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/sda2
```

Or,(this case, same results):

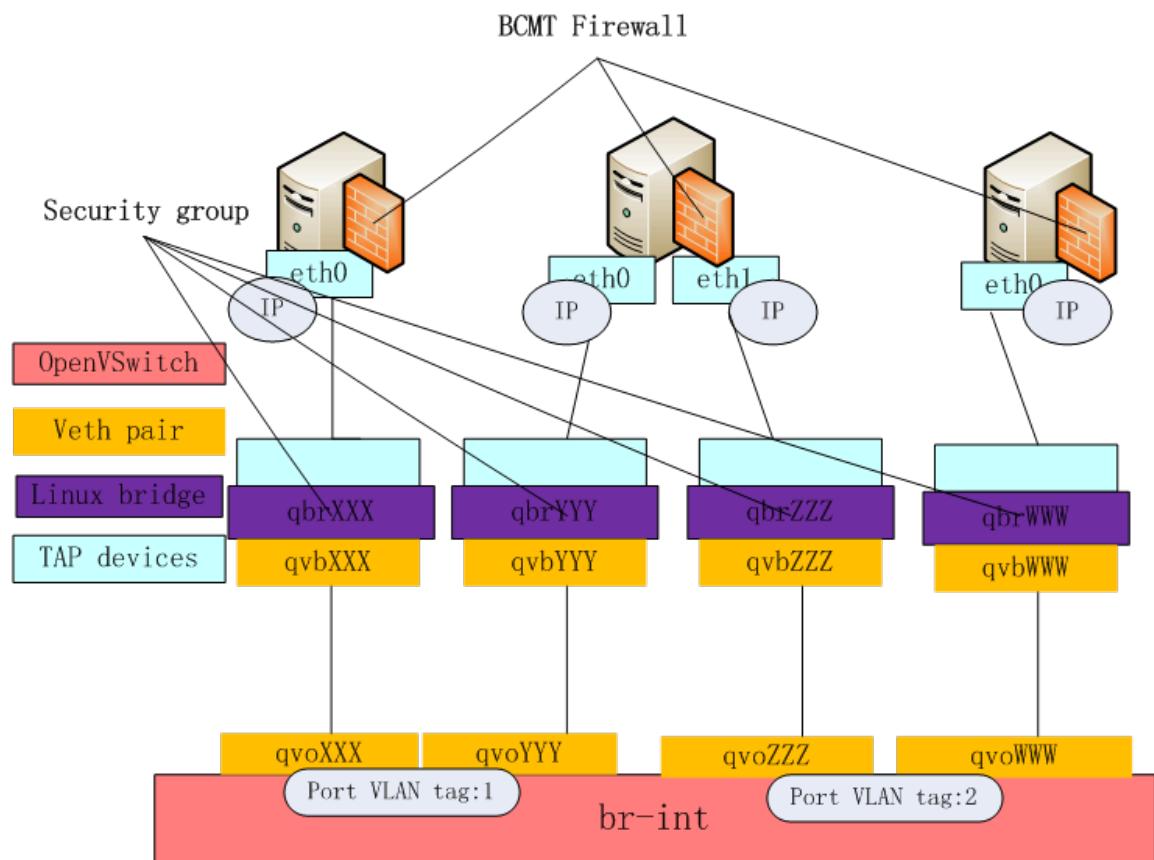
```
bash /usr/share/cbis/undercloud/tools/ceph-create-osd-pacific.sh -d /dev/sda2
-j /dev/sda5 -b /dev/sda4
```

After installation of the OSD is complete, make sure that the osd was recreated:

1. Run `ceph osd tree` again and check if the newly-created osd is present.
2. Ensure that the `ceph -s` command shows an healthy ceph environment.
3. Run `lsblk` to see if all OSD partitions exist on sda.

13 Firewall management

Security groups in OpenStack and AWS environments function as access control firewalls between VMs in a cloud environment. The NCS firewall service implements iptables to focus on cluster security inside a VM. The firewall will filter packets that access the host in the cluster, while allowing the Kubernetes NAT behavior. The firewalld service on every node in the cluster will be disabled, and iptables will be used as an alternative, to ensure compatibility with Kubernetes. During the NCS deployment, all trusted IPs (such as the internal and external IP address of each node in the cluster) are added into a whitelist. Users can manually add trusted IP addresses. If the source IP address of a package is in the whitelist, the firewall will allow it to access internal resources. NCS supports iptable rules in the form of five tuples.



13.1 Enable firewall

About this task

To enable firewall prior to NCS deployment, set `firewall_enabled` and `firewall_whitelist` in the `bcmt_config.json` configuration file..

```
"firewall_enabled": true,
```

```
"firewall_whitelist": "10.10.10.10,10.10.10.11"
```

1. Set *firewall_enabled* to true.
2. Set *firewall_whitelist*, providing IP addresses separated by commas, IPs in the list are not filtered by the firewall.

To enable the firewall using CLI commands after installation, use the **ncs network firewall** command to manage the firewall service.

```
NAME:  
ncs network firewall - firewall commands  
  
USAGE:  
ncs network firewall command [command options] [arguments...]  
  
COMMANDS:  
get      get firewall status and user rules list  
enable   enable firewall service  
disable  disable firewall service  
rules    firewall rules
```

1. Log in to a Control node.
2. Retrieve the status of the firewall and a list of existing rules using the **ncs network firewall get** command.

Sample output:

```
{  
  "firewall_status": "disabled"  
}
```

3. If the firewall is not enabled, use the **ncs network firewall enable** command to enable it.

Sample output:

```
{"task-status": "pending"}  
  
{"task-status": "pending"}  
  
{"task-status": "pending"}  
  
{"task-status": "running"}  
  
.....  
{"task-status": "done"}
```

13.2 Add firewall rules

1. Log in to a Control node.
2. Create an /opt/bcmt/add-rules.json file that describes the firewall rules to be added.

Sample add-rules.json file contents:

```
[  
  {"role": "worker", "destination": "10.9.242.15", "dport": 31486, "protocol": "tcp"},  
  {"role": "worker", "destination": "10.9.242.15", "dport": 31484, "protocol": "tcp"},  
  {"role": "worker", "destination": "10.9.242.15", "dport": 31969, "protocol": "tcp"},  
  {"role": "worker", "destination": "10.9.242.15", "dport": 31485, "protocol": "tcp"},  
  {"role": "worker", "destination": "10.9.242.15", "dport": 32061, "protocol": "tcp"},  
  {"role": "ncs-control-02", "destination": "10.9.242.15", "dport": 30009, "protocol": "udp"},  
  { "role": "edge", "destination": "fd00:bada:135::110", "dport": 6668, "protocol": "tcp",  
    "ip_type": "ipv6"}  
]
```



Note: The role support role name and hostname, *ncs-control-02* is the hostname in sample.

3. Use the **ncs network firewall rule add** command to add the new rules.

```
ncs network firewall rules add --rules_file <rules file>
```

Example:

```
ncs network firewall rules add --rules_file /opt/bcmt/add-rules.json
```

Expected output:

```
{"task-status": "pending"}  
  
{"task-status": "pending"}  
  
{"task-status": "running"}  
  
.....  
{"task-status": "done"}
```

13.2.1 Add Firewall Rules for Multiports

A range of ports can be opened on a NCS firewall by specifying a port range for the dport key.

Example

```
{"role": "worker", "destination": "10.9.242.15", "dport": "32450:32480", "protocol": "tcp"}
```

13.3 Delete firewall rules

1. Log in to a Control node.
2. Create an /opt/bcmt/delete-rules.json file that describes the firewall rules to be deleted.

Sample delete-rules.json file contents:

```
{
  "control": [],
  "all": [],
  "edge": [],
  "worker": [
    " -d 10.9.242.15 -p tcp --dport 31486 -j ACCEPT",
    " -d 10.9.242.15 -p tcp --dport 31484 -j ACCEPT",
    " -d 10.9.242.15 -p tcp --dport 31969 -j ACCEPT",
    " -d 10.9.242.15 -p tcp --dport 31485 -j ACCEPT",
    " -d 10.9.242.15 -p tcp --dport 32061 -j ACCEPT",
    " -d 10.9.242.15 -p tcp --dport 30008 -j ACCEPT"
  ],
  "ncs-control-02": [
    " -d 10.9.242.15 -p udp --dport 30009 -j ACCEPT"
  ]
}
```

If you want to delete rules in a ipv6 firewall, A *ip_type* option is needed.

Sample delete-rules.json file contents:

```
{
  "ip_type": "ipv6",
  "all": [],
  "control": [],
  "edge": [
    " -d fd00:bada:135::110 -p tcp --dport 6668 -j ACCEPT"
  ],
  "worker": []
}
```

3. Use the **ncs network firewall rule delete** command to delete the existing rules.

```
ncs network firewall rules delete --rules_file <rules file>
```

Example:

```
ncs network firewall rules delete --rules_file /opt/bcmt/delete-rules.json
```

Expected output:

```
{
  "task-status": "pending"
}

{
  "task-status": "running"
}

...
{
  "task-status": "done"
}
```

13.4 SELinux enforcement rules

About this task

If the default type enforcement rules that are included with NCS prevent the cluster from deploying successfully, we need to collect and enable the rules.

1. Generate the new enforcement rules that are not covered by the default .te file.

- a) Log in to cluster nodes and change to the root user using the **sudo su -** command.
- b) Change SELinux running mode to Permissive and remove dontaudits from policy on every node:

```
# setenforce 0
# semodule -DB
```

- c) Install applications.

- d) Collect selinux violations and generate corresponding enforcement rules on every node:

```
# ausearch -m AVC | audit2allow -M app-rules
```

Sample output:

```
Email option is specified but /usr/lib/sendmail doesn't seem
executable.
***** IMPORTANT *****
To make this policy package active, execute:

semodule -i app-rules.pp
```

app-rules.te and app-rules.pp files will be created at current directory.

Save app-rules.te file to deployment server or local machine for later use, for example, cluster nodes are recreated.

e) Enable the new policy on every node:

```
# semodule -i app-rules.pp
```

f) Delete the applications that has been installed.

g) Change SELinux running mode to enforcing on every node:

```
# setenforce 1
```

h) Re-install applications, the applications should be deployed successfully.

2. Enable new policy file on all cluster nodes if enforcement rules exist.

a) Copy .te files to the Deployment Server and into the **/opt/bcmt** folder.

b) Log in to deployment server and change to the root user using the **sudo su -** command.

c) Log in to the NCS admin container by using the following command:

```
docker exec -it bcmt-admin bash
```

d) Set the endpoint to point to one of the Control nodes using the **ncs config** command:

```
ncs config set --endpoint=https://<control node IP>:<port>/ncm/api/v1
```

Example command:

```
ncs config set --endpoint=https://10.0.2.1:8082/ncm/api/v1
```



Note: For Baremetal deployment, use port 8084 instead of 8082.

e) Log in in using the **ncs user login** command:

```
ncs user login --username=<username> --password=<password>
```

Example command:

```
ncs user login --username=ncs-admin --password=NewPassword@1234!
```

f) Use the ncs cluster inventory command to retrieve the cluster inventory.

Example command:

```
ncs cluster inventory
```

Example output:

```
*****
inventory tarball has been downloaded in current path.
Name is inventory.tgz
*****
```

g) Unzip the tarball.

Command:

```
tar -zxvf inventory.tgz
```

Expected output:

```
bcmt_var-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
bcmt_inventory-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

h) Change the permissions for the bcmt_vm_key-<uuid>.json file using the chmod command.

Example command:

```
chmod 600 bcmt_vm_key-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json
```

i) Enable new policy file using the following command syntax.

Command syntax:

```
ansible-playbook -i bcmt_inventory-<uuid>.json -e 'file_path=<path
of .te file>' selinux_policy_enable.yml
```

Example command:

```
ansible-playbook -i
bcmt_inventory-7adc99c8-6754-4287-9bf6-2d9b35af6a7f.json -e
'file_path=/opt/bcmt/app-rules.te' selinux_policy_enable.yml
```

The ansible script **selinux_policy_enable.yml** is provided as below:

```
---
- hosts: "{{ run_hosts|default('role_all') }}"
  become: yes
  gather_facts: "{{ run_gather_facts|default(True) }}"
  serial: "{{ run_serial|default('100%') }}"
  any_errors_fatal: "{{ run_any_errors_fatal|default(True) }}"
  tasks:
```

```
- name: copy selinux policy for common
  synchronize:
    src: "{{ file_path }}"
    dest: "/tmp/selinux/"

- name: get all selinux policy files
  shell: ls /tmp/selinux | sed 's/\.\te$//'
  register: tefiles
  ignore_errors: yes

- name: compile policy
  shell: checkmodule -M -m -o {{ item }}.mod /tmp/selinux/
{{ item }}.te; semodule_package -o {{ item }}.pp -m {{ item }}.mod
  with_items:
    - "{{ tefiles.stdout_lines }}"
  when: tefiles is defined

- name: enable policy
  shell: semodule -i {{ item }}.pp
  ignore_errors: yes
  with_items:
    - "{{ tefiles.stdout_lines }}"
  when: tefiles is defined

- name: remove selinux directory
  file:
    path: "/tmp/selinux/"
    state: absent
  ignore_errors: yes
  when: tefiles is defined
```

14 Operators

The NCS operators allow the user to perform system tuning. NCS also support Kubernetes native methods for adjusting system parameters.

14.1 Sysctl

The sysctl operator (kind: Sysctl) is used to modify the `/etc/sysctl.conf` parameters for all of the nodes in a cluster. Users can create a `sysctl.yaml` file to affect changes. The `sysctlRules` section can include any parameter that the OS supports, so long as it fulfills the "key=value" pattern.

Sample sysctl.yaml file:

```
apiVersion: ncm.nokia.com/v1alpha1
kind: Sysctl
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: sysctl-sample
spec:
  sysctlRules:
    - "vm.max_map_count=1048575"
    - "fs.file-max=100000"
    - "vm.swappiness=1"
    - "net.core.rmem_max=16777216"
    - "net.core.wmem_max=16777216"
    - "net.core.rmem_default=16777216"
    - "net.core.wmem_default=16777216"
    - "net.core.optmem_max=40960"
    - "net.ipv4.tcp_rmem=4096 87380 16777216"
    - "net.ipv4.tcp_wmem=4096 65536 16777216"
  nodeSelector:
    is_control: "true"
```

The `nodeSelector` parameter is optional. When specified, all nodes that match the label of the `nodeSelector` value will apply `sysctlRules`. Otherwise, if `nodeSelector` is not specified, all cluster nodes will apply `sysctlRules`.

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f sysctl.yaml
```

- To revert changes, use the **kubectl apply** command:

```
kubectl delete -f sysctl.yaml
```



Note: Ensure that there are no contradictions with the sysctlRules with previously installed Sysctl objects and when configuring multiple Sysctl objects.

14.2 Limit

The limit operator (kind: Limit) is used to modify the `/etc/security/limits.conf` parameters for all of the nodes in a cluster. Users can create a `limits.yaml` file to affect changes. The `limitRules` section can include any parameter that the OS supports.

Sample limits.yaml file:

```
apiVersion: ncm.nokia.com/v1alpha1
kind: Limit
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: limit-sample
spec:
  limitsRules:
    - "* soft nproc 102400"
    - "* hard nproc 102400"
    - "root soft nproc unlimited"
    - "root hard nproc unlimited"
    - "* soft nofile 63536"
    - "* hard nofile 63536"
  nodeSelector:
    is_control: "true"
```

The `nodeSelector` parameter is optional. When specified, all nodes that match the label of the `nodeSelector` value will apply `limitsRules`. Otherwise, if `nodeSelector` is not specified, all cluster nodes will apply `limitsRules`.

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f limits.yaml
```

- To revert changes, use the **kubectl apply** command:

```
kubectl delete -f limits.yaml
```

14.3 Password

The password operator (kind: Password) is used to modify the /etc/default/passwd parameters for all of the nodes in a cluster. Users can create a password.yaml file to affect changes. The **passwordRules** must be parameters that are supported by /etc/default/passwd.

Sample password.yaml file:

```
apiVersion: ncm.nokia.com/v1alpha1
kind: Password
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: password-sample
spec:
  passwordRules:
    - "MAX_SIMULTANEOUS_LOGINS_cloud-user=5"
    - "MAX_SIMULTANEOUS_LOGINS_root=5"
    - "MAX_LOGIN_FAILURES=5"
    - "MAX_LOGIN_FAILURES_cloud-user=5"
  nodeSelector:
    is_control: "true"
```

The **nodeSelector** parameter is optional. When specified, all nodes that match the label of the nodeSelector value will apply passwordRules. Otherwise, if **nodeSelector** is not specified, all cluster nodes will apply passwordRules.

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f password.yaml
```

- To revert changes, use the **kubectl apply** command:

```
kubectl delete -f password.yaml
```

14.4 KernelModule

The KernelModule operator (kind: KernelModule) is used to modify the kernel parameters for all of the nodes in a cluster. Users can create a kernelmodule.yaml file to affect changes. The kernelModuleRules section can include any parameter that the OS supports.

Sample kernelmodule.yaml file:

```
apiVersion: ncm.nokia.com/v1alpha1
kind: KernelModule
metadata:
```

```

labels:
  controller-tools.k8s.io: "1.0"
name: kernelmodule-sample
spec:
  kernelModuleRules:
    - "/sys/kernel/mm/transparent_hugepage/defrag=never"
    - "/sys/kernel/mm/transparent_hugepage/enabled=never"

```

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f kernelmodule.yaml
```

- To revert changes, use the **kubectl apply** command:

```
kubectl delete -f kernelmodule.yaml
```

14.5 RegistryCertificate

The RegistryCertificate operator (`kind: RegistryCertificate`) is used add a certificate file to the Docker registry. Users can create a `registrycertificate.yaml` file to affect changes.



Note: The value for cert should be the base64-encoded content of the cert file.

Sample `registrycertificate.yaml` file:

```

apiVersion: ncm.nokia.com/v1alpha1
kind: RegistryCertificate
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: 172.16.1.13
spec:
  # Add fields here
  Registrydomain: "172.16.1.13:5000"
  cert: |
    LS0tLS1CRUdJTiBDRVJUSUZQ0FURS0tLS0tCk1JSURjakNDQWxxZ0F3SUJBZ01JUXRQQitOSFQ1
    S1F3RFFZSktvWk1odmNOQVFTEJRQXdVREVMTUfrR0ExVUUKQmhNQ1FVRXhDekFKQmdOVkJBZ01B
    a0ZCTVFzd0NRWURWUVFIREFKQ1FURUxNQWtHQTFVRUNd0NRUVU4Q3pBSgpCZ05WQkFzTUFrRkJN
    UTB3Q3dZRFZRUUREQVJDUTAxVU1CNFhEVEU1TURJeE9UQTFNamd5TkZvWERUSTVNRE14Ck5qQTFN
    amd5TkZvd1VERUxNQWtHQTFVRUJoTUNRVUV4Q3pBSkJnT1ZCQWdNQWtGQk1Rc3dDUV1EV1FRSERB
    SkIKUVRFTE1Ba0dbMVVFQ2d3Q1FVRXhDekFKQmdOVkJBc01Ba0ZCTVEwd0N3WURWUVFERFSQ1Ew
    MVVNSU1CSWpBTgpCZ2txaGtpRz13MEJBUUVGQUFPQ0FROEFNSU1CQ2dLQ0FRRUEzS1VOQ3pyVEpK
    OGtvbjlEdXdPa3pEVE4rTG1LCmltS3k2Yk5kKzRXL3d1UkQwODdnbTVxUENWdzd2zzzsTG4wbF15
    TFFkNkZaNVdKeGdtbmNqVmZzd0xsTmlkdFEKbWYvZTZWRGQ0YnFuM3JuK0NmbTRNM3VEW1psYzBY
    bktBvkVxLzc5a3JwN0pnRVJ5Tz1zcWJKZENLZncrExBPQwo2ZGh5Z2ZyblptVXZBYjFOS28rN1pR
    Z0h0TmhxQ041MkF0NXZTYzdiKzMvM0RjRkZm0WhIak1IV0pzQjRkb3VhCm9jV1d4WmdCajhnMGQ3
    WXptM0Y0aFnwRjRiSERUeDvpQmNTVXhqcitQTEUxaFrqSGVLN05DUjVrV29tNkNuajeKSjFIKzha
    bfBUOTRKQWQ2cGgrM3N0WDFIWHRrUUxSQ3U3MzRTd31Ic0tUV1F2cGdoOXQvQWNZdk40UU1EQVFB
    QgpvMUF3VGpBTUJnT1ZIuk1FQ1RBREFRSC9NQjBHQTfvZERnUVdCQ1Mwcm1HUT12dWR1Sy93aVZu
    bwtsSG9PN3NCC1BEQWZCZ05WSFNNRUdEQVdnQ1Mwcm1HUT12dWR1Sy93aVZubWtsSG9PN3NCUERB
    TkJna3Foa21HOXcwQkFRc0YKQUFpQ0FRRUFWN3RnK25NbHY5OHJ1OxhCekRUa21URkVsNnVvd2VT
    eDQ5M09rl3pwaE9yeXFvVUztU0dUSHzvdAphL0d2dHhXSnsJVMmVqMTVhT2ZrTkh0Njm4cGtteFJO
    bfPqrQ1V3UDFYNU9MS1FDQXo2dHznR0t4cTZNSUxWVDZYCkUrcmlRODA3aXZwYkVLVTNIT2JySDD4

```

```
QzN1dE5uOFRZZU1BS09BbmI4VG5hWTc5Q3JrTGNSWEJQdUsrL3BmcWIKaVZ0N1NGYnVxRGtjWk5U  
M29EbHBrTnRNZWFlenB5L2tyUnVMTEFQS0vUTFRNWRxaFkyNktZK2hNeDZYQ1hJWQpyQktZZytx  
R0Y5MkNDYXNFb29zeUduVy9kU2FMV2t0ZkMyNklSwnQrcXluTk5xK0Z0UVNuWWVvVFhNalJ3K0Nm  
CmMvSDBLcGZ5ZUIzOvhvS1hRQ2NzU2wzYU1zZVpzdz09Ci0tLS0tRU5EIENFU1RJRk1DQVRFLS0t  
LSOK  
nodeSelector:  
  is_control: "true"
```

The **nodeSelector** parameter is optional. When specified, all nodes that match the label of the nodeSelector value be impacted. Otherwise, if **nodeSelector** is not specified, all cluster nodes will be impacted.

- To apply changes, use the **kubectl apply** command:

```
kubectl apply -f registrycertificate.yaml
```

- To revert changes, use the **kubectl apply** command:

```
kubectl delete -f registrycertificate.yaml
```

14.6 Application POD TimeZone

Each application container could use its own timezone, if applicaiton container need to use the same time zone with host, need to mount host /etc/localtime to application container, application could update its POD spec, or use PodPreset object to inject the volume mount into container at POD creation time. include any parameter that the OS supports.

Example PodPreset.yaml file:

```
apiVersion: settings.k8s.io/v1alpha1  
kind: PodPreset  
metadata:  
  name: mount-local-time  
  namespace: application-namespace  
spec:  
  volumeMounts:  
    - name: host-timezone  
      mountPath: /etc/localtime  
      readOnly: true  
  volumes:  
    - name: host-timezone  
      hostPath:  
        path: /etc/localtime
```

The namespace parameter is the one that the application will be installed to.

- To apply changes, use the `kubectl apply` command:

```
kubectl apply -f PodPreset.yaml
```

- To revert changes, use the `kubectl apply` command:

```
kubectl delete -f PodPreset.yaml
```

15 Custom installation/NCS hooks

Hooks definition

Hooks are designed as an ansible playbook, NCS also added hooks for upgrade.

- pre-install-app
- post-install-app
- pre-install-cluster
- pre-scale-in
- post-scale-in
- pre-scale-out
- post-scale-out
- pre-upgrade-node
- post-upgrade-node
- post-upgrade-nodes

Execution order when installing NCS:

1. pre-install-cluster
2. deploy NCS
3. pre-install-app
4. install APP
5. post-install-app

Hooks usage

The Hooks work directory is `/opt/bcmt/storage/hooks/`. The path does not exist by default, you must create these files manually.

Copy the following to the deploy node:

- pre-install-cluster
- pre-install-app
- post-install-app

Copy the following to only one controller node:

- pre-scale-in
- post-scale-in

- pre-scale-out
- post-scale-out
- pre-upgrade-node
- post-upgrade-node
- post-upgrade-nodes

Each hook is one whole ansible playbook, it's entry file name should be "main.yml", all hooks' designated location:

- /opt/bcmt/storage/hooks/pre-install-app/main.yml
- /opt/bcmt/storage/hooks/post-install-app/main.yml
- /opt/bcmt/storage/hooks/pre-install-cluster/main.yml
- /opt/bcmt/storage/hooks/pre-scale-in/main.yml
- /opt/bcmt/storage/hooks/post-scale-in/main.yml
- /opt/bcmt/storage/hooks/pre-scale-out/main.yml
- /opt/bcmt/storage/hooks/post-scale-out/main.yml
- /opt/bcmt/storage/hooks/pre-upgrade-node/main.yml
- /opt/bcmt/storage/hooks/post-upgrade-node/main.yml
- /opt/bcmt/storage/hooks/post-upgrade-nodes/main.yml

These hooks are triggered automatically when deploying NCS. If you want to do operations after deployment or debug hooks, refer to [Cluster inventory](#) on page 53.

Hooks demo

Demo of one hook:

```
- hosts: "{{ run_hosts|default('role_all') }}"
become: yes
gather_facts: "{{ run_gather_facts|default(True) }}"
serial: "{{ run_serial|default('100%') }}"
any_errors_fatal: "{{ run_any_errors_fatal|default(True) }}"
tasks:
  - name: change root password
    user:
      name: root
      password: "{{ 'XXXX@1234t' | password_hash('sha512') }}"
```

16 Application management

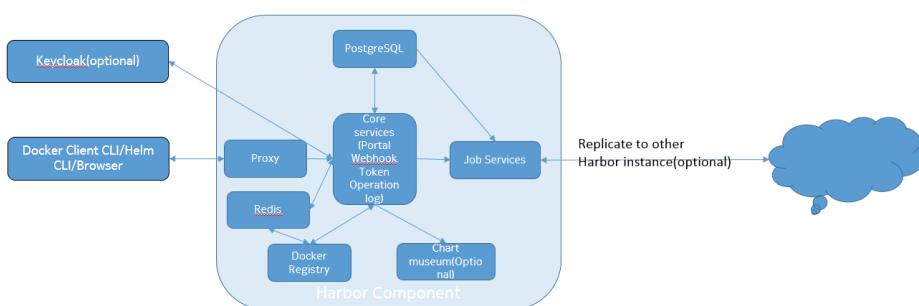
16.1 Harbor

16.1.1 What is Harbor

Project Harbor is an open source trusted cloud native registry project that stores, signs, and scans content. Harbor extends the open source Docker Distribution by adding the functionalities usually required by users such as security, identity and management.

The following figure shows the high level Harbor component architecture.

Figure 18: Harbor architecture



16.1.2 Harbor on NCS

1. There are NCS enhancements on the top of official Harbor to support SSO with keycloak, as that is not supported with community version Harbor.
2. IPV6 and Helm3 installation are supported.
3. Notary functionalities is not supported by this NCS delivery yet, please open new feature request if it is needed.

16.1.3 What's New Harbor NCS FP2

1. Harbor version upgraded to 2.1.0, and Harbor API changed with new endpoint.
2. Trivy is supported as vulnerability scanner for containers. For new installed cluster, it's installed with 5G PVC size as default, user could change "harbor_settings" section in installation configuration file for a different size. For upgrading the cluster, the user could enable that with following command.

```
helm upgrade harbor --recreate-pods --reuse-values --set trivy.enabled=true
--force stable/harbor
```

3. When using CLI to access the Harbor repo from outside NCS cluster, and Harbor exposure type is "nodePort" (it is the case in NCS installation), it is unnecessary to update Harbor's externalURL option.

16.1.4 Installation

NCS deploys Harbor by default when cluster installation: harbor_enable in bcmt_config.json is set to true by default, and you can customize its configuration by change harbor_settings in bcmt_config.json before deployment, refer to the *Nokia Container Services (NCS) Installation guide* for details.

 **Note:** The following repository URLs are provided as examples, the user should add their own.

The following is the example online installation procedure:

1. Install Harbor using the `helm install` command.

Flag `--set` can be used or modify the `values.yaml` to configure the Harbor helm chart during the installation.

Sample of `values.yaml`

 **Note:** The following repository/registry URLs are provided as examples, the user should add their own.

```
realm: bcmt-harbor-realm
uaa_verify_cert: false
# to enable https ca verification, need set uaa_verify_cert to true
and create a secret:
# kubectl create secret generic keycloak-ca --from-file=/path/
keycloak.pem -n namespace
#uaa_verify_cert: true
ca_cert_secret: keycloak-ca
ca_cert_file: keycloak.pem

security:
  enabled: true
  runAsUser: 10000
  runAsGroup: 10000
  fsGroup: 10000

cbur:
  image:
    imageRepo: cbur/cbura
    imageTag: 1.0.3-983
  database:
```

```
enabled: true
mode: "local"
cronSpec: "* * * * */5"
maxiCopy: 3
scheduleEnable: true
resources:
  requests:
    memory: 256Mi
    cpu: 100m
    ephemeral-storage: "200Mi"
  limits:
    ephemeral-storage: "500Mi"
chartmuseum:
  enabled: true
  mode: "local"
  cronSpec: "* * * * */5"
  maxiCopy: 3
  scheduleEnable: true
  resources:
    requests:
      memory: 256Mi
      cpu: 100m
      ephemeral-storage: "200Mi"
    limits:
      ephemeral-storage: "500Mi"
registry:
  enabled: false # disable by default.
  mode: "local"
  cronSpec: "* * * * */5"
  maxiCopy: 1
  scheduleEnable: true
  backupwithpvc: false # mount a pvc as backup tmp directory, default
is false means using tmpdir
  resources:
    requests:
      memory: 256Mi
      cpu: 100m
      ephemeral-storage: "200Mi"
    limits:
      ephemeral-storage: "100G"

expose:
  # Set the way how to expose the service. Set the type as "ingress",
  # "clusterIP" or "nodePort" and fill the information in the
corresponding
```

```
# section
enabled: true
type: ingress
tls:
    # Certification for nginx when expose type is clusterIP/nodePort/
hostPort,
    # if not provide, internal.tls.useCertman must be true
    nginxCert:
ingress:
hosts:
    core: core.harbor.domain
    notary: notary.harbor.domain
# set to the type of ingress controller if it has specific
requirements.
# leave as `default` for most ingress controllers.
# set to `gce` if using the GCE ingress controller
# set to `ncp` if using the NCP (NSX-T Container Plugin) ingress
controller
controller: default
annotations:
    ingress.kubernetes.io/ssl-redirect: "true"
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    ingress.kubernetes.io/proxy-body-size: "0"
    nginx.ingress.kubernetes.io/proxy-body-size: "0"
    ingress.citm.nokia.com/sticky-route-services:
"$cookie_JSESSIONID|JSESSIONID ip_cookie"
clusterIP:
    # Specify cluster IP address as part of a Service creation
request,
    # must be a valid IPv4 or IPv6 address from within the service-
cluster-ip-range CIDR
    # range that is configured for the API server
clusterIP: 10.254.249.249
ports:
    # The service port Harbor listens on when serving with HTTPS
    httpsPort: 3443
    # The service port Notary listens on. Only needed when
notary.enabled
    # is set to true
    notaryPort: 34443
nodePort:
ports:
https:
    # The service port Harbor listens on when serving with HTTPS
    port: 8443
```

```
# The node port Harbor listens on when serving with HTTPS
nodePort: 30003

# Only needed when notary.enabled is set to true
notary:
    # The service port Notary listens on
    port: 4443
    # The node port Notary listens on
    nodePort: 30004

hostPort:
    ports:
        # The node port Harbor listens on when serving with HTTPS
        https: 30003
        # Only needed when notary.enabled is set to true
        notary: 30004

# The external URL for Harbor core service. It is used to
# 1) populate the docker/helm commands showed on portal
# 2) populate the token service URL returned to docker/notary client
# Usually it's harbor core service DNS.
externalURL: https://harbor-harbor-core.default.svc.cluster.local

# The internal component communicate must be tls.
internal:
    tls:
        #useCertman is true, all cert will be generated by it, otherwise,
        need to create them manually.
        useCertman:
            enabled: true
            #commonName: "harbor.domain"
            dnsdomain: "cluster.local"
            apiVersion: "cert-manager.io/v1alpha2"
            issuerRef:
                name: "ncms-ca-issuer"
                kind: "ClusterIssuer"
                # When expose type is clusterIP/nodePort/hostPort, harbor will
                # expose by nginx on some nodes,
                # populate the IP address list and this nginx will be installed,
                for example,
                # all edge nodes external IP list.
                nginxIpList: ["1.2.3.4", "1.2.3.5"]

        #Below XCert only needed when useCertman is false.
        # Trusted CA bundle for all the certifications, generation command
example:
```

```
# kubectl create secret generic ca-tls-secret --from-file=ca-
bundle=ca.pem
caBundle: cert-harbor-svc

# Certification for harbor core, requirement:
# 1. private key for core must be RSA format, could use
# "openssl rsa -in key.pem -out key_rsa.pem" to convert PEM
private key to RSA format
# 2. harbor core service short name and full name must include in
certification, for example, harbor core service:
#       kubectl get svc -l app=harbor |grep core
#       harbor-harbor-core           ClusterIP   XXXX
#       the certification SAN field should like below:
#           X509v3 extensions:
#           X509v3 Subject Alternative Name:
#               IP Address:X.X.X.X, DNS:harbor-harbor-core,
DNS:harbor-harbor-core.default.svc.cluster.local
# 3. create k8s secret with key and cert from the certification
and private key.
#       kubectl create secret generic core-tls-secret --from-
file=key=key_rsa.pem --from-file=cert=cert.pem
coreCert: cert-harbor-svc

# Certification for harbor registry, requirement:
# 1. service short name must include in certification, for
example:
#       kubectl get svc -l app=harbor |grep registry
#       harbor-harbor-registry           ClusterIP   XXXX
#       the certification SAN field should like below:
#           X509v3 extensions:
#           X509v3 Subject Alternative Name:
#               IP Address:X.X.X.X, DNS:harbor-harbor-registry
# 3. create k8s secret with key and cert from the certification
and private key.
#       kubectl create secret generic registry-tls-secret --
from-file=key=XXX --from-file=cert=YYY
# 4. similar with other harbor component
registryCert: cert-harbor-svc
portalCert: cert-harbor-svc
jobserviceCert: cert-harbor-svc
adminserverCert: cert-harbor-svc
clairCert: cert-harbor-svc
chartmuseumCert: cert-harbor-svc
databaseCert: cert-harbor-svc
notaryServerCert: cert-harbor-svc
```

```
notarySignerCert: cert-harbor-svc

global:
  registry1: csf-docker-delivered.repo.lab.pl.alcatel-lucent.com
  imagePullPolicy: IfNotPresent

# The persistence is enabled by default and a default StorageClass
# is needed in the k8s cluster to provision volumes dynamically.
# Specify another StorageClass in the "storageClass" or set
"existingClaim"
  # if you have already existing persistent volumes to use
  #
  # For storing images and charts, you can also use "azure", "gcs",
"s3",
  # "swift" or "oss". Set it in the "imageChartStorage" section
persistence:
  enabled: true
  # Setting it to "keep" to avoid removing PVCs during a helm delete
  # operation. Set it to "delete" will delete PVCs after the chart
deleted
  resourcePolicy: keep
  persistentVolumeClaim:
    registry:
      # Use the existing PVC which must be created manually before
bound
      existingClaim: ""
      # Specify the "storageClass" used to provision the volume. Or
the default
      # StorageClass will be used(the default).
      # Set it to "-" to disable dynamic provisioning
      storageClass: "glusterfs-storageclass"
      subPath: "registry"
      accessMode: ReadWriteMany
      #accessMode: ReadWriteOnce
      size: 5Gi
      pv: "harbor-registry"
    chartmuseum:
      existingClaim: ""
      storageClass: "glusterfs-storageclass"
      subPath: "chartmuseum"
      accessMode: ReadWriteMany
      #accessMode: ReadWriteOnce
      size: 5Gi
      pv: "harbor-chart"
    jobservice:
```

```
existingClaim: ""
storageClass: "glusterfs-storageclass"
subPath: "jobservice"
accessMode: ReadWriteMany
#accessMode: ReadWriteOnce
size: 1Gi
# If external database is used, the following settings for
database will
# be ignored
database:
  existingClaim: ""
  storageClass: ""
  subPath: "database"
  accessMode: ReadWriteOnce
  size: 1Gi
  pv: "harbor-database"
# If external Redis is used, the following settings for Redis will
# be ignored
redis:
  existingClaim: ""
  storageClass: ""
  subPath: "redis"
  accessMode: ReadWriteOnce
  size: 1Gi
  pv: "harbor-redis"
# Define which storage backend is used for registry and chartmuseum
to store
# images and charts. Refer to
# https://github.com/docker/distribution/blob/master/docs/
configuration.md#storage
# for the detail.
imageChartStorage:
  # Specify whether to disable `redirect` for images and chart
storage, for
  # backends which not supported it (such as using minio for `s3`
storage type), please disable
  # it. To disable redirects, simply set `disableredirect` to `true`
instead.
  # Refer to
  # https://github.com/docker/distribution/blob/master/docs/
configuration.md#redirect
  # for the detail.
  disableredirect: false

#additional CA need put into container
```

```
caBundleSecretName:

# Specify the type of storage: "filesystem", "azure", "gcs", "s3",
"swift",
    # "oss" and fill the information needed in the corresponding
section. The type
        # must be "filesystem" if you want to use persistent volumes for
registry
            # and chartmuseum
type: filesystem
filesystem:
    rootdirectory: /storage
    #maxthreads: 100
azure:
    accountname: accountname
    accountkey: base64encodedaccountkey
    container: containername
    #realm: core.windows.net
gcs:
    bucket: bucketname
    # TODO: support the keyfile of gcs
    #keyfile: /path/to/keyfile
    #rootdirectory: /gcs/object/name/prefix
    #chunksize: 5242880
s3:
    region: us-west-1
    bucket: bucketname
    #accesskey: awsaccesskey
    #secretkey: awssecretkey
    #regionendpoint: http://myobjects.local
    #encrypt: false
    #keyid: mykeyid
    #secure: true
    #v4auth: true
    #chunksize: 5242880
    #rootdirectory: /s3/object/name/prefix
    #storageclass: STANDARD
swift:
    authurl: https://storage.myprovider.com/v3/auth
    username: username
    password: password
    container: containername
    #region: fr
    #tenant: tenantname
    #tenantid: tenantid
```

```
#domain: domainname
#domainid: domainid
#trustid: trustid
#insecureskipverify: false
#chunksize: 5M
#prefix:
#secretkey: secretkey
#accesskey: accesskey
#authversion: 3
#endpointtype: public
#tempurlcontainerkey: false
#tempurlmethods:
oss:
    accesskeyid: accesskeyid
    accesskeysecret: accesskeysecret
    region: regionname
    bucket: bucketname
    #endpoint: endpoint
    #internal: false
    #encrypt: false
    #secure: true
    #chunksize: 10M
    #rootdirectory: rootdirectory

logLevel: info
# The initial password of Harbor admin. Change it from portal after
launching Harbor
harborAdminPassword:
# The secret key used for encryption. Must be a string of 16 chars.
secretKey: not-a-secure-key

# The proxy settings for updating clair vulnerabilities from the
Internet and replicating
# artifacts from/to the registries that cannot be reached directly
proxy:
    httpProxy:
    httpsProxy:
    noProxy: 127.0.0.1,localhost,.local,.internal
components:
    - core
    - jobservice
    - clair

# If expose the service via "ingress", the Nginx will not be used
nginx:
```

```
image:
  repository: bcmt/harbor-portal
  tag: v1.10.0-bcmt.1
# Not used when expose by hostPort
replicas: 1
resources:
  requests:
    memory: 256Mi
    cpu: 100m
    ephemeral-storage: "200Mi"
  limits:
    ephemeral-storage: "300Mi"
nodeSelector: {}
tolerations: []
affinity: {}
## Additional annotations
podAnnotations: {}

portal:
  image:
    repository: bcmt/harbor-portal
    tag: v1.10.0-bcmt.1
  replicas: 1
  livenessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  readinessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  resources:
    requests:
      memory: 256Mi
      cpu: 100m
      ephemeral-storage: "200Mi"
    limits:
      ephemeral-storage: "500Mi"
  nodeSelector: {}
  tolerations: []
  affinity: {}
```

```
## Additional deployment annotations
podAnnotations: {}

core:
  image:
    repository: bcmt/harbor-core
    tag: v1.10.1-bcmt.3
  replicas: 1
  livenessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  readinessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  resources:
    requests:
      memory: 256Mi
      cpu: 100m
      ephemeral-storage: "200Mi"
    limits:
      ephemeral-storage: "500Mi"
  nodeSelector: {}
  tolerations: []
  affinity: {}
  podAnnotations: {}
# Secret is used when core server communicates with other
components.
# If a secret key is not specified, Helm will generate one.
# Must be a string of 16 chars.
secret: ""
# The XSRF key. Will be generated automatically if it isn't
specified
xsrfKey: ""

jobservice:
  image:
    repository: bcmt/harbor-jobservice
    tag: v1.10.0-bcmt.1
  replicas: 1
```

```
maxJobWorkers: 10
# The logger for jobs: "file", "database" or "stdout"
jobLogger: database
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
resources:
  requests:
    memory: 256Mi
    cpu: 100m
    ephemeral-storage: "200Mi"
  limits:
    ephemeral-storage: "500Mi"
nodeSelector: {}
tolerations: []
affinity: {}
## Additional deployment annotations
podAnnotations: {}

registry:
  registry:
    image:
      repository: bcmt/harbor-registry
      #tag: 2.7.1
      tag: v2.7.1-patch-2819-2553-v1.10.0-bcmt.1
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 10
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
```

```
failureThreshold: 3
resources:
requests:
  memory: 256Mi
  cpu: 100m
  ephemeral-storage: "200Mi"
limits:
  ephemeral-storage: "500Mi"
controller:
image:
  repository: bcmt/harbor-registryctl
  tag: v1.10.0-bcmt.1
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
resources:
requests:
  memory: 256Mi
  cpu: 100m
  ephemeral-storage: "200Mi"
limits:
  ephemeral-storage: "500Mi"
replicas: 1
tolerations: []
affinity: {}
## Additional deployment annotations
podAnnotations: {}
# Secret is used to secure the upload state from client
# and registry storage backend.
# See: https://github.com/docker/distribution/blob/master/docs/configuration.md#http
# If a secret key is not specified, Helm will generate one.
# Must be a string of 16 chars.
secret: ""
# If true, the registry returns relative URLs in Location headers.
The client is responsible for resolving the correct URL.
```

```
relativeurls: false
middleware:
  enabled: false
  type: cloudFront
  cloudFront:
    baseurl: example.cloudfront.net
    keypairid: KEYPAIRID
    duration: 3000s
    ipfilteredby: none
    # The secret key that should be present is CLOUDFRONT_KEY_DATA,
which should be the encoded private key
    # that allows access to CloudFront
    privateKeySecret: "my-secret"

chartmuseum:
  enabled: true
  # Harbor defaults ChartMuseum to returning relative urls, if you
want using absolute url you should enable it by change the following
value to 'true'
  absoluteUrl: false
  image:
    repository: bcmt/harbor-chartmuseum
    tag: v0.9.0-v1.10.0-bcmt.1
  replicas: 1
  livenessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  readinessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
  resources:
    requests:
      memory: 256Mi
      cpu: 100m
      ephemeral-storage: "200Mi"
    limits:
      ephemeral-storage: "500Mi"
  nodeSelector: {}
  tolerations: []
```

```
affinity: {}

## Additional deployment annotations
podAnnotations: {}

clair:
  enabled: false
  clair:
    image:
      repository: bcmt/harbor-clair
      #tag: v2.1.1-v1.10.0
      tag: v2.1.1-v1.10.0-bcmt.1

    #Disable TLS server's certificate chain and hostname verification
    when pulling layers.
    insecureTls: true

    #additional CA need put into container, create it before
    installation if need:
    #kubectl create secret generic clair-ca --from-file=ca.crt=clair-
    ca.pem
    caBundleSecretName:
    livenessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    readinessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    resources:
      requests:
        memory: 256Mi
        cpu: 100m
        ephemeral-storage: "200Mi"
      limits:
        ephemeral-storage: "500Mi"

    adapter:
      image:
        repository: bcmt/harbor-clair-adapter
        tag: v1.0.1-v1.10.0-bcmt.1
```

```
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
resources:
requests:
  memory: 256Mi
  cpu: 100m
  ephemeral-storage: "200Mi"
limits:
  ephemeral-storage: "500Mi"

replicas: 1
# The interval of clair updaters, the unit is hour, set to 0 to
# disable the updaters
#updatersInterval: 12
updatersInterval: 0
nodeSelector: {}
tolerations: []
affinity: {}
## Additional deployment annotations
podAnnotations: {}

database:
# if external database is used, set "type" to "external"
# and fill the connection informations in "external" section
type: internal
internal:
image:
  repository: bcmt/harbor-db
  tag: v1.7.9-9.6.10
# The initial superuser password for internal database
password: "changeit"
livenessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
```

```
        successThreshold: 1
        failureThreshold: 3
    readinessProbe:
        initialDelaySeconds: 10
        periodSeconds: 20
        timeoutSeconds: 5
        successThreshold: 1
        failureThreshold: 3
    resources:
        requests:
            memory: 256Mi
            cpu: 100m
            ephemeral-storage: "200Mi"
        limits:
            ephemeral-storage: "500Mi"
    nodeSelector: {}
    tolerations: []
    affinity: {}
  external:
    host: "192.168.0.1"
    port: "5432"
    username: "user"
    password: "password"
    coreDatabase: "registry"
    clairDatabase: "clair"
    notaryServerDatabase: "notary_server"
    notarySignerDatabase: "notary_signer"
    # "disable" - No SSL
    # "require" - Always SSL (skip verification)
    # "verify-ca" - Always SSL (verify that the certificate presented
by the
    # server was signed by a trusted CA)
    # "verify-full" - Always SSL (verify that the certification
presented by the
        # server was signed by a trusted CA and the server host name
matches the one
        # in the certificate)
    sslmode: "disable"
    # The maximum number of connections in the idle connection pool.
    # If it <=0, no idle connections are retained.
    maxIdleConns: 50
    # The maximum number of open connections to the database.
    # If it <= 0, then there is no limit on the number of open
connections.
    # Note: the default number of connections is 100 for postgres.
```

```
maxOpenConns: 100
## Additional deployment annotations
podAnnotations: {}

redis:
  # if external Redis is used, set "type" to "external"
  # and fill the connection informations in "external" section
  type: internal
  internal:
    image:
      repository: crdb/redisio
      tag: 2.9-0.1263
    livenessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    readinessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    resources:
      requests:
        memory: 256Mi
        cpu: 100m
        ephemeral-storage: "200Mi"
      limits:
        ephemeral-storage: "500Mi"
    nodeSelector: {}
    tolerations: []
    affinity: {}
  external:
    host: "192.168.0.2"
    port: "6379"
    # The "coreDatabaseIndex" must be "0" as the library Harbor
    # used doesn't support configuring it
    coreDatabaseIndex: "0"
    jobserviceDatabaseIndex: "1"
    registryDatabaseIndex: "2"
    chartmuseumDatabaseIndex: "3"
    clairAdapterIndex: "4"
    password: ""
```

```
## Additional deployment annotations
podAnnotations: {}

kubectl:
  image:
    repository: tools/kubectl
    tag: v1.14.8-nano

notary:
  enabled: false
  server:
    image:
      repository: bcmt/harbor-notary-server
      tag: v0.6.1-v1.10.0-bcmt.1
    replicas: 1
    livenessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    readinessProbe:
      initialDelaySeconds: 10
      periodSeconds: 20
      timeoutSeconds: 5
      successThreshold: 1
      failureThreshold: 3
    resources:
      requests:
        memory: 256Mi
        cpu: 100m
        ephemeral-storage: "200Mi"
      limits:
        ephemeral-storage: "500Mi"
  signer:
    image:
      repository: bcmt/harbor-notary-signer
      tag: v0.6.1-v1.10.0-bcmt.1
  replicas: 1
  livenessProbe:
    initialDelaySeconds: 10
    periodSeconds: 20
    timeoutSeconds: 5
    successThreshold: 1
    failureThreshold: 3
```

```

readinessProbe:
  initialDelaySeconds: 10
  periodSeconds: 20
  timeoutSeconds: 5
  successThreshold: 1
  failureThreshold: 3
resources:
requests:
  memory: 256Mi
  cpu: 100m
  ephemeral-storage: "200Mi"
limits:
  ephemeral-storage: "500Mi"
nodeSelector: {}
tolerations: []
affinity: {}
## Additional deployment annotations
podAnnotations: {}

```

- [Optional]Modify the uaa info in values.yaml with the keycloak endpoint information.

realm: bcmt-harbor-realm

- ExternalURL is the Harbor core SVC DNS. If using a non-default namespace, be sure to change this value.
- The default docker registry is csf-docker-delivered.repo.lab.pl.alcatel-lucent.com, change it if images are loaded to NCS internal registry already, by change this parameter during installation --set global.registry1="bcmt-registry:5000".
- The internal communication within harbor component must be https based, if the corresponding certifications/keys are not provided manually, this chart uses cert-man to create them automatically.
- Docker registry backup/restore is disabled by default, as it may need more space for Container Backup and Restore, turn it on if need during installation by option --set cbur.registry.enabled=true, refer to values.yaml for detail.
- PVC is used for persistence, change the volume size basing on need. The default is for demo, not for production! such as --set registry.volumes.data.size=100Gi.
- Storage class supporting **ReadWriteMany** access mode PVC is needed for the purpose of HA, otherwise, there is no HA, which means POD will be delete/recreate during upgrade, one example of no HA support configuration is as follows:

```

persistence:
  enabled: true
  persistentVolumeClaim:
    registry:

```

```

    ...
    storageClass: ""
    accessMode: ReadWriteOnce
    ...
chartmuseum:
    ...
    storageClass: ""
    accessMode: ReadWriteOnce

```

- Suppose there is a ingress controller already installed in cluster. Harbor could be accessible through ingress with default user name/password *admin/Harbor12345*.
- Default the Harbor chart will bring up single Redis POD, if the deployment need a HA Redis server for production environment.

Example command to install Redis:

```
helm install --name harbor-redis
--set common.password="none" -f values.yaml .
```

Install harbor with the following options:

```
--set redis.type=external
--set redis.external.host=harbor-redis-crdb-
redisio.default.svc.cluster.local
```

Example command to install Harbor:

```
helm repo add stable https://repo.lab.pl.alcatel-lucent.com/csf-helm-stable

helm install stable/harbor --name harbor -f values.yaml
--set imagePullPolicy=Always
--set externalURL=https://harbor-harbor-
core.default.svc.cluster.local
--set expose.ingress.hosts.core="core.harbor.domain"
--set persistence.resourcePolicy="keep"
```

Note the end part of the output:

NOTES:

Please wait for several minutes for Harbor deployment to complete.

2. Check the Harbor chart installed using the `helm list` command.

```
harbor  1
Wed Dec  2 14:58:05 2020      DEPLOYED
harbor-1.0.51                   2.1.0
```

ncms

3. Config harbor to be able to use keycloak as authentication by below API:

```
curl -X PUT -u "admin:admin-password" -H "Content-Type: application/json"  
-ki https://harbor-harbor-core.default.svc/api/configurations  
-d'{"auth_mode": "uaa_auth", "uaa_endpoint": "https://1.2.3.4:8089/auth",  
"uaa_client_id": "harbor-client",  
"uaa_client_secret": "1111111-1111",  
"uaa_verify_cert":false,  
"realm": "harbor-realm"}'````
```

16.1.5 Administration

1. Backup/Restore

- a. Registry backup is disabled by default as it may need more space for CBUR, turn it on if need during installation by option "--set cbur.registry.enabled=true", refer to values.yaml for detail.
- b. CBUR-a sidecar need copy the backup data to its container /tmp location before transfer that between backup location. To avoid this container in registry pod exhaust node resource, could set cbur.registry.backupwithpvc=true, which will request another PVC **harbor-registry-backup** and mount it to CBUR-a sidecar /tmp directory, use cbur.registry.storageSize/storageClass to control this PVC size and storageclass.
- c. By default the CBUR-a sidecar is always brought up in harbor registry pod. If set cbur.registry.inject=true and cbur.registry.backupwithpvc=true, only when execute backup/restore, PVC **harbor-registry-backup** is requested and the CBUR-a sidecar is injected to harbor registry pod, and after finish backup/restore, this PVC and sidecar container will be removed. This behavior saves resource at most time but will cause harbor registry pod remove and recreate. If harbor registry pod is replicate with one only, that will cause service interrupt.
- d. Directory /data0/glusterfs-cbur-repo/repo/data is used by CBUR to store the data, please ensure its space is enough.

Example command:

Backup:

```
helm backup -t harbor -a none
```

Restore:

```
helm restore -t harbor
```

2. Upgrade/Rollback

- a. Upgrade/Rollback chart version between 1.0.55 and above 1.0.55

In case of rolling upgrade/rollback, there will be more than one POD attaching the same PVC of harbor-harbor-chartmuseum/harbor-harbor-registry, that is the reason of their PVC

"accessMode" is ReadWriteMany, if that kind of storage class is not installed, only recreate POD will happen during upgrade/rollback. Refer to [HA section](#) of this document.

Example command:

Upgrade:

```
helm upgrade harbor --version **new_harbor_version** --recreate-pods --reuse-values --debug stable/harbor
```

Rollback:

```
helm rollback harbor 1
```

b. Upgrade/Rollback chart version between 1.0.33 and above 1.0.55

Harbor chart version 1.0.33 integrate Harbor 1.10.1 while chart version above 1.55 integrated Harbor 2.1.0, if upgrade/rollback between them, need special procedure(version old than 1.0.33 must upgrade to 1.0.33 first, refer to the previous release upgrade procedure):

- Upgrade

- i. Backup database files manually on one Control node:

```
mkdir -p /home/cloud-user/harbor_data;
kubectl -n default cp harbor-harbor-database-0:var/lib/
postgresql/data/ -c database
/home/cloud-user harbor_data
```

- ii. Backup Harbor using CBUR:

```
helm backup -t harbor
```

- iii. Get the existing Harbor setting:

```
helm get values harbor > tmp-upgrade.values
```

- iv. Patch Harbor secret: untar the chart package, use the script patch_before_v2.sh under conf directory to patch the configuration, if harbor was installed under "ncms" namespace, release name is "harbor", new version harbor core component image is "bcmt-registry:5000/bcmt/harbor-core:v2.1.0-bcmt.1", and "registry_password123" is a example password, choose random string for that.

```
helm fetch stable/harbor; tar xvf harbor-1.0.*.tgz;
ls harbor/conf/patch_before_v2.sh
bash harbor/conf/patch_before_v2.sh
ncms harbor bcmt-registry:5000/bcmt/harbor-core:v2.1.0-bcmt.1
registry_password123
```

- v. Upgrade Harbor:

```
helm upgrade harbor
--version 1.0.55
```

```
--recreate-pods  
--reset-values  
-f tmp.values stable/harbor
```

- Rollback:

- i. Execute helm rollback:

```
helm rollback harbor 1234 --recreate-pods
```

- ii. Scale in core and database pod to zero, delete database PVC:

```
kubectl scale -n default deployment harbor-harbor-core --  
replicas 0;  
kubectl scale -n default sts harbor-harbor-database --replicas  
0;  
kubectl delete pvc -n default database-data-harbor-harbor-  
database-0
```

- iii. Scale out database pod to one, wait until it is ready:

```
kubectl scale -n default sts harbor-harbor-database  
--replicas 1;kubectl wait po -n default -l app=harbor-harbor -l  
component=database  
--for-condition=Ready
```

- iv. Restore database files by CBUR:

```
helm restore -t harbor;
```

- v. Scale out core pod to one(or other value great than one), wait until it is ready:

```
kubectl scale -n default deployment harbor-harbor-core --  
replicas 1;  
kubectl wait po -n default -l app=harbor-harbor -l  
component=core  
--for-condition=Ready
```

- vi. After a while, all Harbor pods should be running.

```
kubectl wait po -n default -l app=harbor-harbor --  
for-condition=Ready
```

3. Scale

- **Database and internal redis POD** are not able to scale, these are StatefulSet, and Harbor don't support cluster configuration of that, for external Redis, please follow [CRDB instruction](#) to scale.
- **Adminserver/core/portal POD** are stateless, could scale.
- **If jobservice POD** save the log to database(it's default case), it's stateless also, could scale.

- If **chartmuseum/registry** POD use the `ReadWriteMany` access mode PVC, it's able to scale, otherwise, it's not.

Example command:

```
helm scale harbor --set core.replicas=2,portal.replicas=2,
jobservice.replicas=2,registry.replicas=2,chartmuseum.replicas=2

helm scale harbor --set core.replicas=1,portal.replicas=1,
jobservice.replicas=1,registry.replicas=1,chartmuseum.replicas=1
```

4. Uninstall

Command:

```
helm delete harbor;
helm delete --purge harbor
```

5. Heal

Unnecessary at the moment.

16.1.6 Usage

16.1.6.1 Harbor access endpoint

1. Within k8s cluster, user could access by harbor core service DNS

```
Get admin password:
# kubectl get secret harbor-harbor-core -n ncms -o yaml|grep
HARBOR_ADMIN_PASSWORD |head -1 |cut -f2 -d':' |xargs |base64 -d
Zn9VzMedmywVRa51z7

Login to the registry:
# docker login https://harbor-harbor-core.ncms.svc
Username: admin
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/
config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-
store

Login Succeeded
# docker tag bcmt-registry:5000/busybox:latest    harbor-harbor-
core.ncms.svc/library/busybox:latest
```

```
docker push harbor-harbor-core.ncm.svc/library/busybox:latest
The push refers to repository [harbor-harbor-core.ncm.svc/library/busybox]
6c0ea40aef9d: Layer already exists
latest: digest:
sha256:dd97a3fe6d721c5cf03abac0f50e2848dc583f7c4e41bf39102ceb42edfd1808
size: 527
```

2. Outside Kubernetes cluster, user needs to access harbor portal URL, refer to portal exposure section.

16.1.6.2 Docker repository

1. User need to create project, add keycloak user to project first, in a terminal login to get a token, then push/pull image.
2. Use it in Kubernetes: need to associate docker access token credential with service account, or uses the Secret of docker-registry type to authenticate with a container registry to pull a private image.

16.1.6.3 Helm Chart repository

As a helm chart repository, Harbor can work smoothly with Helm CLI. Run command `helm version` to make sure the version of Helm CLI is v2.9.1+.

1. Use `helm repo add` to add Harbor as a **unified chart repository** with specified username, all the project namespaces accessible by that user should be visible to Helm, if user does not have the permission to access all project namespaces, will get "403 Forbidden" error.

```
helm repo add
--username=yyy
--password=xxx myrepo https://xx.xx.xx.xx/chartrepo/
```

2. Use `helm repo add` to add a Harbor project as a **separate chart repository**, only the charts under that project are visible to Helm.

```
helm repo add
--username=yyy
--password=xxx myrepo
https://xx.xx.xx.xx/chartrepo/test_project1
```

3. Use the **push plugin** of the helm CLI to push charts to Harbor.



Note: Helm push plugin is installed on NCS cluster already, but is not enabled. Execute below command on either Control node to enable it.

```
cp -r $HELM_HOME/plugins/helm-plugin-v0.7.1/helm-plugin/*
$HELM_HOME/plugins/helm-plugin-v0.7.1/
helm push example.tgz myrepo
```

4. Use helm install to download the chart from Harbor and install it to the target Kubernetes environment.

5. Other commands like helm search or helm verify are also supported.

16.1.6.4 API

There is no plan from NCS to wrapper the API as this time, refer to Harbor API directly.

16.1.6.5 Harbor portal exposure

Harbor could be exposed by other options:

- 1. clusterIP:** expose by this type means Harbor is only available within k8s cluster.
- 2. nodePort:** Harbor could be accessed from out side k8s cluster by specified port number and each node IP address. In NCS installation from NCS 20FP2SU2, **it is the expose type** when "expose_portal" is enabled(default is enabled) in "harbor_settings" section, and the port number is 30003.

```
"harbor_settings": {
  "expose_portal": true,
  ...
}
```

- 3. hostPort:** By specify a node selector, Harbor nginx POD will installed on selected node, and Harbor could be accessed from out side Kubernetes cluster by specified port number and selected node IP address. Refer to expose.type section in values.yaml file for details.

If portal is not exposed when installation, could expose that with following command:

```
helm upgrade harbor --reuse-values
--set expose.enabled=true --set expose.type="nodePort"
--set expose.nodePort.ports.https.nodePort="30003" stable/harbor
```

If Harbor is exposed after installation and need to use docker CLI to access docker repo remotely, below step are needed to update the certification with correct IP list, Compose a file, "ip-list.yaml" with below content, the "nginxIpList" should be the external IP list of control node:

```
internal:
  tls:
    useCertman:
      nginxIpList: ["10.7.6.5", "10.7.6.4", "10.7.6.3"]
```

Update chart:

```
helm upgrade harbor
--reuse-values
--recreate-pods -f ip-list.yaml
```

```
--force  stable/harbor
```

16.1.6.6 Access Harbor repo from outside NCS cluster

In NCS installation, Harbor portal/API could be exposed to manage Harbor project as below procedure.

1. (Optional, only need the expose type is "ingress") If Harbor portal is accessible from 10.10.10.10:443 already, execute following command on one of the control node:

```
# helm upgrade  harbor
--reuse-values
--recreate-pods
--set  externalURL=https://10.10.10.10:443  stable/harbor
```

2. (Optional, only need the expose type is "ingress") Wait until all the Harbor POD are running:

```
# kubectl get po -n ncms |grep harbor
harbor-harbor-chartmuseum-55858cbd99-b54js    2/2      Running     0
12d
harbor-harbor-core-845574499-7m595          1/1      Running     0
100m
harbor-harbor-database-0                     2/2      Running     0
12d
harbor-harbor-jobservice-77769c54d4-5f7bp   1/1      Running     0
100m
harbor-harbor-portal-68b777bd6f-6zjtb       1/1      Running     0
12d
harbor-harbor-registry-57bd8bf475-m9qdk     3/3      Running     0
99m
```

3. Get the Harbor portal exposure certification from k8s secret, if it's exposed by CITM ingress, the secret name is citm-cert-tls, If it is exposed by nodePort or hostPort, the secret name is "harbor-harbor-nginx-srt". Get the certification content with below command, decode the "ca.crt" part with base64, and copy the decoded content to a file named with ca.crt:

```
# kubectl get secret citm-cert-tls -n ncms -o yaml |grep -e ca.crt -e
tls.crt
        ca.crt: LS0tLS1CRUdJTi.....LQo=
# echo "LS0tLS1CRUdJTi.....LQo=" |base64 -d
```

4. On a client Linux host, config Docker to trust the certification:

```
# mkdir -p /etc/docker/certs.d/10.10.10.10:443;
copy ca.crt /etc/docker/certs.d/10.10.10.10:443;
```

5. Use Docker CLI to login:

```
# docker --debug login 10.10.10.10:443
Username: admin
Password:
...
Login Succeeded
```

16.1.6.7 Add trusted CA to Harbor

Harbor could be used to replicate resources from remote repository. If remote repository is HTTPS based and Harbor is configured to validate remote repository certification, user may need to add trusted CA to Harbor.

1. For example the trusted CA is named with ca.pem, create Kubernetes secret in Harbor installed namespace, for example ncms

```
kubectl create secret generic ca-tls-secret
--from-file=ca-bundle=ca.pem -n ncms
```

2. Update Harbor chart to add the CA:

```
helm upgrade harbor
--reuse-values
--recreate-pods
--set internal.tls.caBundle=ca-tls-secret stable/harbor
```

16.1.7 Design Items

16.1.7.1 DB

Currently Harbor uses PostgreSQL as unified DB, and Harbor needs its own DB, or one external PostgreSQL database.

Migrate multiple DBs to one PostgreSQL DB:

In the previous releases, there are two or three database instances running on one Harbor node, which are MariaDB/MySQL and PostgreSQL. This of course increases the effort to maintain the Harbor system. This new feature makes it possible to merge multiple databases into a single database, making it easier to maintain and possible to enable HA solutions for the future releases.

Main features:

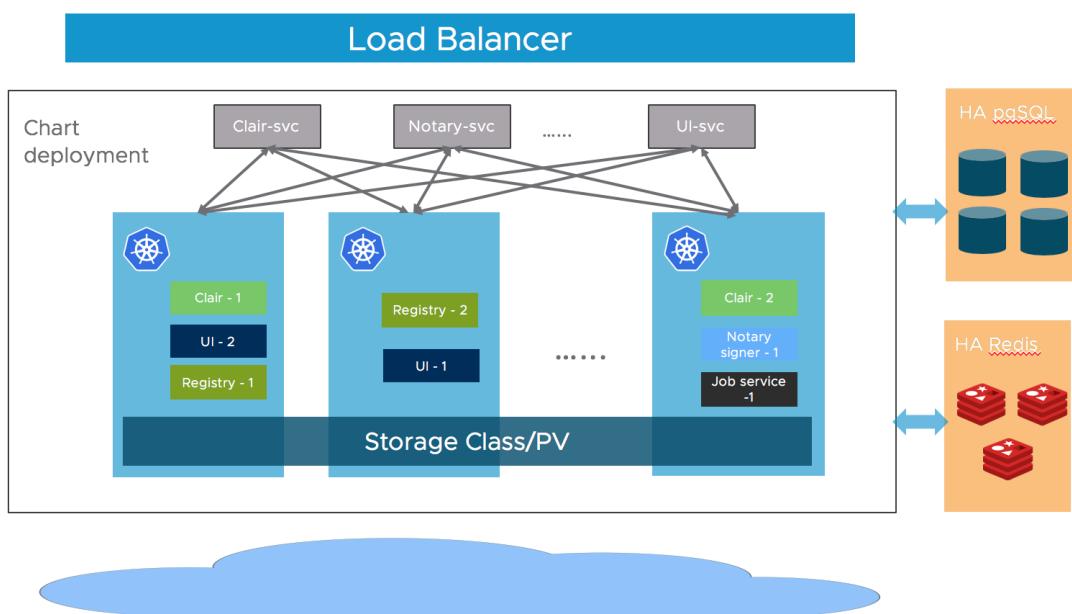
- Migrate Harbor DB to PostgreSQL
- Migrate Notary DB to PostgreSQL
- Redirect Clair DB to Harbor/Notary DB

16.1.7.2 HA

1. Current Situation

- a. From Harbor, most of Harbor's components are stateless now. So we can simply increase the replica of the pods to make sure the components are distributed to multiple worker nodes, and leverage the "Service" mechanism of K8S to ensure the connectivity across pods.
- b. As for storage layer, it is expected that the user provide high available PostgreSQL, Redis cluster for application data and PVCs or object storage for storing images and charts.

Figure 19: Load balancer



2. Solution

- Storage for Docker Images and chart

Harbor's HA solution is based on ReadWriteMany access mode PVC provided shared storage, Glusterfs support that mode, if need HA, please use that kind of storage class.

Additional, Rook-ceph on NCS supports S3 object storage(not filesystem type), which could use for images and charts also as shared storage for HA purpose.

- Redis DB

Set up HA Redis Cluster using [CRDB-redis.io](#) in production environment is recommended(there is HA redis on NCS Control node, but which is for NCS docker registry, there may be conflict to use that for Harbor)

- Harbor Database



Note: No cluster PQ database is provided, the following is provided as an example.

CSF does not provide cluster PQ database, only one instance is installed in the NCS

16.1.7.3 TLS

1. From release 20.03, internal TLS traffic among Harbor components are mandatory. If cert-manager is installed, Harbor could generate certification using it, otherwise user need provide certification to each Harbor components.
2. If expose Harbor with ingress, external TLS will be provided by ingress controller.
3. If expose Harbor with nodePort/hostPort/clusterIP, external TLS will be provided by Harbor nginx component.

Please refer to `internal.tls` and `expose.type` section in `values.yaml` file for details.

16.1.8 FAQ

16.1.8.1 Why can't add user to a harbor project

The reasons could be:

1. The keycloak access information is not correct, or the password is not correct, please double check its accessible. Please use below commands to test keycloak(complete the command with the real keycloak access endpoint information).

```
export TKN=$(curl -k -X POST https://XXX/auth/realms/XXX/protocol/openid-connect/token  
-H "Content-Type: application/x-www-form-urlencoded"  
-d "username=XXXXXX" -d 'password=YYYYYY' -d 'grant_type=password' -d  
'client_id=XXX'  
-d "client_secret=XXX" | jq -r '.access_token')  
echo $TKN
```

2. The service account does not have permission to query users, please use below command to check(complete the command with the real keycloak access endpoint information and the user name need to added into harbor project).

```
export TKN=$(curl -k -X POST https://XXX/auth/realms/XXX/protocol/openid-connect/token  
-H "Content-Type: application/x-www-form-urlencoded"  
-d 'grant_type=client_credentials'  
-d 'client_id=XXX' -d "client_secret=XXX" | jq -r '.access_token')  
echo $TKN  
curl -v -k -H "Accept: application/json"  
-H "Authorization: Bearer $TKN"
```

```
-X GET 'https://XXX/auth/admin/realms/xxx/users?filter=Username+eq+
%27test1%27'
```

If it returns error code 401, check keycloak configuration on the corresponding realm.

16.1.8.2 How Harbor uses Keycloak

1. Authentication:

User could login from Harbor's GUI using the user name/password in Keycloak. Harbor will follow *Client Credentials Grant* flow to authorize its with Keycloak. In NCS deployment, user could access the keycloak GUI via go to NCS portal first, then click "PORTAL", follow "KEYCLOAK Dashboard" button.

2. Project user association management:

With this functionality, after Harbor project creation, it does not need to add user to Harbor project, Harbor could retrieve the project/user association from keycloak, configuration procedure as below:

- a. In Keycloak, defines user group with below attribute and value, the value is the Harbor project/ user association, in this example, this group has developer role on test_project1, guest role on test_project2 and admin role on test_project3. In NCS deployment, the group is "ncms", don't need create new group.
- b. Add user to the group. In NCS deployment, user "ncm-admin" and "ncs-admin" are in group "ncms" already.
- c. Configure Keycloak client ID to **include harbor-project-role** attribute to **token** and **userinfo**. In NCS deployment, the client ID name is "ncm-manager".
- d. Login to Harbor with the keycloak user name/password. In NCS deployment, could use "ncm-admin" and "ncs-admin" to login to Harbor.



Note:

- a. The Harbor project/user relationship could be set on Harbor GUI as well, and the final project role will be the combination from Keycloak and Harbor DB.
- b. If user belong to multiple groups, Keycloak 5.0 introduced mapper parameter: Aggregate attribute values, enable this parameter and enable multivalued parameter, and use "harbor-project-role-list" to carry the multivalued attribute. For example, "harbor-project-role-list":["project_1:admin","project_2:admin"], Harbor could parse this list, and if there is "harbor-project-role" attribute in the access token, Harbor will combine the project role, and override low level permission with high level. For example, project1:guest in one attribute, and project1:admin in another, then admin will take effect instead of guest, whatever is in harbor-project-role-list or harbor-project-role.

16.1.8.3 Is Keycloak necessary

NO, if there is no keycloak available, Harbor could manage the user using its own database, but in NCS deployment, Harbor is configured to connect with NCS keycloak.

16.1.8.4 How to generate Clair offline database

1. Make sure to backup database before try to import to Harbor.
2. From the testing, only need dump the first sql file, then import this file to Harbor.

```
pg_dump -U postgres -a -t feature  
-t keyvalue  
-t namespace -t schema_migrations  
-t vulnerability  
-t vulnerability_fixedin_feature > vulnerability.sql
```

16.1.8.5 How to update Trivy offline vulnerability database

As Vulnerability Scanner, if Trivy is installed(trivy.enabled is true) but proxy.httpsProxy/httpProxy is not set, or trivy.skipUpdate is set to true, manually update the Trivy database on a regular basis is needed, then scanner can detect recently-identified vulnerabilities.

1. Make sure db directory exists in Trivy pod.

```
kubectl exec -n ncms harbor-harbor-trivy-0  
-c trivy  
-- mkdir -p /home/scanner/.cache/trivy/db/
```

2. Download the vulnerability database from official website, then cp to pod.

```
wget https://github.com/aquasecurity/trivy-db/releases/latest/download/  
trivy-offline.db.tgz

kubectl cp trivy-offline.db.tgz -n ncms harbor-harbor-trivy-0:/home/  
scanner/.cache/trivy/db/ -c trivy

kubectl exec -it -n ncms harbor-harbor-trivy-0 -c trivy -- bash

bash-4.4$ cd /home/scanner/.cache/trivy/db/

bash-4.4$ ls
trivy-offline.db.tgz

bash-4.4$ tar xvf trivy-offline.db.tgz
trivy.db
metadata.json

bash-4.4$ rm trivy-offline.db.tgz
```

16.1.8.6 How to increase harbor PVC size

Harbor PVC is just regular Kubernetes PVC, expendable or not depends on the storage solution in NCS platform, and most of storage solution support expand volume, like cinder/vsphere/cephrdb/cephfs/glusterfs, use "kubectl edit" command to increase the volume size.

```
kubectl edit pvc -n ncms pvc_name
```

16.1.9 Vulnerability Scanning

Harbor allows users to perform three main types of vulnerability scans in their artifacts:

- Scan Individual Artifacts
- Scan All Artifacts
- Scheduled Scans

 **Note:** For other updates on Harbor Vulnerability Scans, please refer to their documentation latest version under the menu tab **Docs** on their main website <https://goharbor.io/>

Scan Individual Artifacts

The scanning process on individual artifacts can be performed in the Harbor interface by selecting a project and clicking on the **Scanner tab**, which shows the selected scanner. This task can successfully be completed if the status doesn't show **Queued** or **Scanning**. For detailed step-by-step guidance on how to conduct scan on individual artifacts, please refer to the respective guide in the **Docs** menu tab on Harbor website <https://goharbor.io/>, where information on *Vulnerability scanning for OCI image index* can also be found.

Scan All Artifacts

The Harbor interface allows user to run global scans on all of the artifacts in a Harbor instance, across all projects. This function can be found in the Harbor interface under **Administration > Interrogation Services > Vulnerability tab**, selecting the option **Scan Now**. However, the option is not available if another scanning is being performed at the same moment.

Schedule Scans

The user can set policies to control when vulnerability scanning should run. This function is available in the Harbor interface, under the menu **Administration > Interrogation Services > Vulnerability > Edit > Schedule to scan all**, where the scanning frequency can be set and saved.

For more details on **Harbor Vulnerability Scanning**, please refer to the specific sections under **Harbor Administration** in the menu tab **Docs** on <https://goharbor.io/>

For information on Harbor in NCS, please refer to our *Harbor* on page 388 section in the *OAM Guide*.

16.2 Chart application management

It is the responsibility of the owner of each component to build the docker image for an asset and create configuration charts for the same, following the [Helms official guide](#).

16.3 Integrated NCS application manager

NCS support to enable/disable integrated NCS internal application. CBUR, APP-API and cert-mananger application are enabled by default.

16.3.1 Enable integrated NCS application

To enable integrated NCS application, add two arguments, **ncs_app_list** and **ncs_app_values** under the **app** section in the **bcmt_config.json** configuration file.

Example:

```
"app": {  
    "chart_enabled": true,  
    "chart_repo_url": {},  
    "ncs_app_list": [  
        "CBUR",  
        "APP-API",  
        "cert-manager"  
    ],  
    "ncm_app_values": {  
        "CBUR": {  
            "k8swatcher": {  
                "enabled": true  
            }  
        }  
    }  
}
```

16.3.2 ncs_app_list

Add components under **ncs_app_list**, and these will be installed by NCS.

This key does not show by default:

- if it does not show, all default components will be installed
- if it show up, it must contains all components that user want to install.

16.3.3 ncs_app_values

Users can change configuration of CBUR, cert-manager and app-api before installation. The following configurations can be changed:

- values for CBUR

```
---  
volumeType:  
  glusterfs: true  
  backupEndpoints: glusterfs-cluster  
  backupPath: cbur-glusterfs-backup  
  repoEndpoints: glusterfs-cluster  
  repoPath: cbur-glusterfs-repo  
  backupClusterEndpoints: glusterfs-cluster  
  backupClusterPath: cbur-glusterfs-backup-cluster  
k8swatcher:  
  enabled: false
```

- values for cert-manager

```
---  
webhook:  
  enabled: true  
cainjector:  
  enabled: true
```

- values for app-api

```
---  
env:  
  SECURE: true
```

If the user wants to set parameters not exist in above configuration file, but helm command or app support it, user can set it in `bcmt_config.json` file directly.

Following is a more detailed configuration for `ncs_app_values`.

```
"ncs_app_values": {  
  "CBUR": {  
    "storageRepo": "900Gi",  
    "storageBackupCluster": "240Gi",  
    "storageClass": "glusterfs-storageclass-distributed",  
    "storageBackup": "60Gi",  
    "volumeType": {  
      "glusterfs": false  
    },  
    "k8swatcher": {  
      "enabled": true  
    },  
    "logging": {  
      "level": "INFO",  
      "file": "app.log",  
      "rotation": "100MB",  
      "maxLogs": 10  
    }  
  }  
}
```

```
        "unified_logging": true
    },
    "use_celery": {
        "enabled": true
    }
},
"cert-manager": {
    "webhook": {
        "enabled": true
    },
    "cainjector": {
        "enabled": true
    }
}
"app-api": {
    "env": {
        "SECURE": true
    }
}
}
```

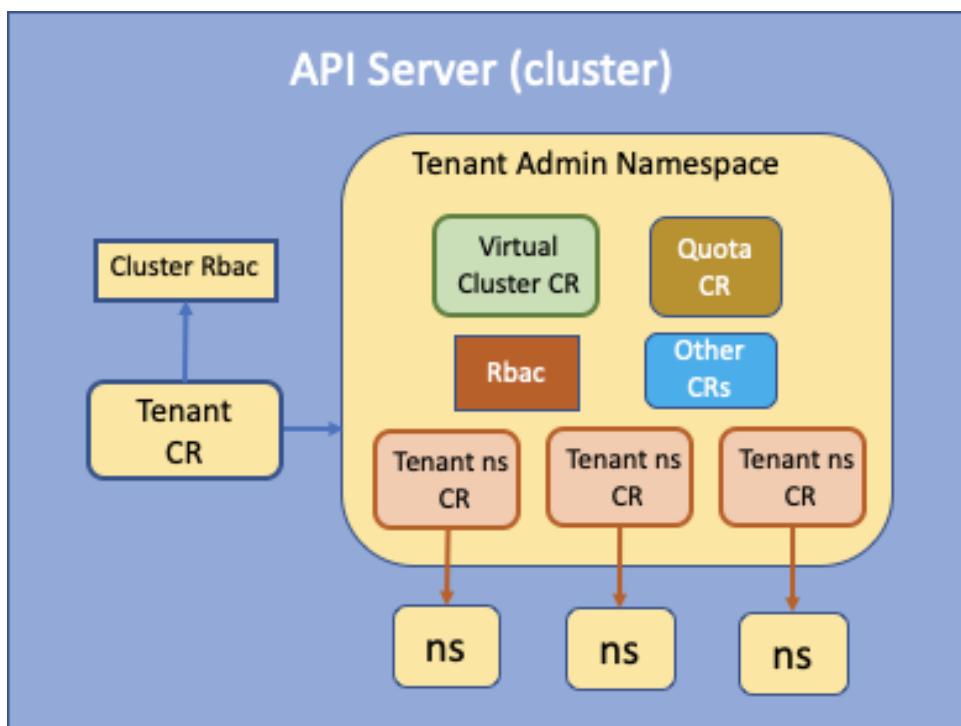
17 Tenant management

17.1 Tenant management overview

17.1.1 Multi-tenancy

Note: All tenant related operations should be executed from the CLI. The usage of Nokia Container Services (NCS) Portal is not supported for tenant management. Multi-tenancy works only with helm3 so all the applications should be installed with helm3.

In case that you have a need to run multiple applications with some level of isolation, multi-tenancy feature can be activated. NCS20 supports soft multi-tenancy. We leverage the multi-tenancy proposals from the community, using Kubernetes namespace as a security boundary: <https://github.com/kubernetes-sigs/multi-tenancy>



Multi-tenancy supporting in NCS is in initial phase, *tenant* is the same as a Kubernetes namespace. The support of multiple namespaces and node level isolation will be added in the future release.

Note: When creating a tenant, the following namespaces will be created:

- <tenant_name>-admin
- <tenant_name>-admin-ns

The application will be stored in <tenant_name>-admin-ns namespace.

17.1.1.1 NCS multi-tenancy limitations in NCS20FP2SU2

1. Installing two charts with the same name in different namespaces is not supported in NCS20FP2SU2 multi-tenancy
2. NCS20FP2SU2 uses Kubernetes v1.19.5, which requires the following app changes in helm charts:
 - a. All resources under *apps/v1beta1* and *apps/v1beta2* should be replaced with *apps/v1*.
 - b. For daemonsets, deployments, replicaset resources under *extensions/v1beta1* use *apps/v1*.
 - c. For networkpolicies resources under *extensions/v1beta1* use *networking.k8s.io/v1*.
 - d. For podsecuritypolicies resources under *extensions/v1beta1* use *policy/v1beta1*.
 - e. The user should modify *apps/v1beta1* to *apps/v1*.
3. Harbor quota needs to be set during tenant creation. This will be required if/when users upload their docker images to tenant harbor. In NCS20FP2SU2 version, there is no way to increase harbor quota after the tenant is created.

17.1.1.2 Workaround for multitenancy + cpu pooler case

 **Note:** The "*priorityClass*" scope cannot be used in "*scopeSelectors*" having *request.storage* in their "*spec.hard*" section. If there are no entries for *requests.storage* in the ResourceQuota of the given Tenant's namespace, the first and the third step can be skipped.

1. Create a second ResourceQuota object in the Tenant's namespace with the value of *request.storage* of the original quota. Create for example *resq.yaml* with similar content (modify the name, namespace and storage as needed):

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: nc0700-admin-ns-rq-sto
  namespace: nc0700-admin-ns
spec:
  hard:
    requests.storage: 3000Gi
```

```
kubectl create -f resq.yaml
```

2. Create PriorityClass. Create an example *prioclass.yaml* with the following content (value should be lower than 10000000000):

```
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
```

```

metadata:
  name: cpu-pooler-priority
  value: 10000
  globalDefault: false
  description: "PriorityClass for cpu pooler usage with multitenancy."

```

```
kubectl create -f prioclass.yaml
```

3. Edit the ResourceQuota created for the given Tenant by default:

```
kubectl edit resourcequotas -n <namespace> <namespace>-rq
```

For example:

```
kubectl edit resourcequotas -n nc0700-admin-ns nc0700-admin-ns-rq
```

Locate the 'spec.hard' section, remove the request.storage line.

4. Add the scopeSelector, so the resulting file will provide the output similar to the following:

```

spec:
  hard:
    limits.cpu: ...
    limits.memory: ...
    requests.cpu: ...
    requests.memory: ...
  scopeSelector:
    matchExpressions:
      - operator: NotIn
        scopeName: PriorityClass
        values:
          - cpu-pooler-priority
  status:
...

```

then save & exit the manifest.

5. The pools provided by cpu pooler can be used now, if the pod.spec contains the following:

```
priorityClassName: cpu-pooler-priority
```

17.1.1.3 Feature Flag

The tenant feature can be only enabled/disabled during the cluster deployment.

A new feature flag is added in the NCS configure file: multi_tenant, the valid value is either "disabled" or "enabled". This field is an optional one, multi-tenancy will be disabled by default if the field is not set in NCS configure file.

17.1.1.4 Dimensioning

When multi-tenancy is enabled in NCS, following new services will be brought up:

- Harbor
- Open Policy Agent Gatekeeper
- Tenant Operator

Harbor is deployed as a container image registry that secures images and helm/charts with role-based access control. It is used by tenant admin and tenant users.

Gatekeeper is used as a validating webhook that enforces CRD-based policies executed by *Open Policy Agent*.

Tenant Operator provide controller to handle the reconciling of "Tenant" CRD and "TenantNamespace" CRD.

17.1.1.4.1 Services List

Table 40: Services list

Service Name	CPU Request	CPU limit	Memory Request	Memory Limit
harbor-harbor-chartmuseum-5d58d77c65-24fwm	200m (2%)	2 (25%)	512Mi (1%)	8Gi (31%)
harbor-harbor-core-848879f796-4zqdm	100m (1%)	1 (12%)	256Mi (0%)	4Gi (15%)
harbor-harbor-jobservice-dfc7f9b96-8wbrj	100m (1%)	1 (12%)	256Mi (0%)	4Gi (15%)
harbor-harbor-portal-8ffbbcc67-2kmcg	100m (1%)	1 (12%)	256Mi (0%)	4Gi (15%)
harbor-harbor-registry-76d6c86fcc-7v554	300m (3%)	3 (37%)	768Mi (2%)	12Gi (47%)

Table 40: Services list (continued)

Service Name	CPU Request	CPU limit	Memory Request	Memory Limit
harbor-harbor-database-0	200m (2%)	2 (25%)	512Mi (1%)	8Gi (31%)
bcmt-mt-tenant-5f68f777b5-ph78q	100m (1%)	200m (2%)	40Mi (0%)	100Mi (0%)
harbor-harbor-redis-0	100m (1%)	1(12%)	256Mi (0%)	4Gi (15%)
harbor-harbor-trivy-0	300m (3%)	2(25%)	768Mi (2%)	8Gi (31%)
gatekeeper-audit	100m	1	256Mi	512Mi
gatekeeper-controller-manager	100m	1	256Mi	512Mi

17.1.1.4.2 Disk Requirements

PVC	Size (minimal default)
harbor-harbor-database-0	1Gi
harbor-chartmuseum	5Gi
harbor-registry	100Gi
harbor-jobservice	1Gi
harbor-redis	1Gi

17.1.1.4.3 Increasing registry size

To scale/grow the PVC size run the following command:

```
kubectl edit pvc pvc-name
```



Note: The cephrrbd-csi version 1.1.10 is required in order to successfully complete the extension of registry volume by aforementioned command. Refer to section *Upgrade of cephrrbd-csi* on page 326 in the *Nokia Container Services (NCS) Operations and Administration Manual* for more details.

17.1.1.5 User & Role

Two new roles are added for tenant.

- Tenant-admin:
 - Manage specific tenant: ns, user
 - Tenant level backup/restore
- Tenant-user:
 - Manage applications within a tenant

Whenever a tenant is created by cluster admin, a default tenant admin user will be created: "tenant-name"-admin. E.g. for tenant "team1", the admin user name will be: team1-admin. The default password for tenant admin is: ncs@CSF_k8s, which is the same as the default password of ncs-admin user. For the first time login, the password must be reset either via cli or management portal.

The roles and newly created tenant specific user accounts with specific RBAC rules are applied to specific Dashboard GUIs (or NCS Manager GUI). Refer to *User roles* in chapter *User management* for default RBAC rules.



Note: Refer to *Onboarding and Managing Applications on Nokia Container Services* and *Dashboard User Guide for Nokia Container Services (NCS)* documents for more information regarding when multi-tenancy (MT), Harbor image and chart registry is used instead of bcmt-registry.

Harbor RBAC and Harbor registry user account mappings i.e., using the following (or additional) Harbor user roles mapped to project users/tenant:

- "Limited Guest",
- "Guest",
- "Developer",
- "Maintainer",
- "ProjectAdmin",
- "Harbor system administrator",
- "Anonymous"

For Harbor RBAC "ncs-admin" is by default used for the Harbor admin tasks and it has by default "Harbor system administrator" role mappings.

17.1.1.6 Kubernetes Multi-tenancy Profile

The following set of Kubernetes built-in admission controllers is enabled:

- PodSecurityPolicy
- AlwaysPullImages
- NodeRestriction
- ServiceAccount
- ResourceQuota
- LimitRanger

OPA Gatekeeper is used to validate the resource request and resource limit must be presented in the POD spec. Application installation will fail if this information is missed.

17.1.1.7 Tenant Level Isolations

Tenants are isolated in different places.



Note: Refer to *Onboarding and Managing Applications on Nokia Container Services* and *Dashboard User Guide for Nokia Container Services (NCS)* documents for more information regarding when multi-tenancy (MT), Harbor image and chart registry is used instead of `bcmnt-registry`.

17.1.1.7.1 UI Changes

The existing user interfaces are enhanced to support tenant management and tenant user management.

17.1.1.7.2 REST API MT

`ncm/api/v1/tenants`: tenant management.

`ncm/api/v1/tenants/{tenant_name}/namespaces`: tenant namespace management.

It is designed for multiple namespaces within one tenant. It should be called directly in the current release.

`ncm/api/v1/tenants/{tenant_name}/users`: tenant user management.

Details on these APIs can be found in the NCS REST API document.

17.1.1.7.3 CLI MT

There is "tenant" sub-command added for NCS. Details on the NCS cli can be found in the *NCS reference guide*.

17.1.1.7.4 Management Portal MT

Two new views have been added: tenant admin view and tenant user view. Details on the portal changes can be found in the appropriate guide.

17.1.1.8 Resource Quota

With resource quotas, cluster administrators can restrict resource consumption and creation on a namespace basis. Within a namespace, a Pod or Container can consume as much CPU and memory as defined by the namespace's resource quota. Resource quota could be configured by cluster admin during the creation of a tenant.

17.1.1.9 Limit Range

A LimitRange is a policy to constrain resource allocations (to Pods or Containers) in a namespace. Limit range could be configured by cluster admin during the creation of a tenant.

17.1.1.10 Network

Deny all ingress by applying the network policy:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: t1-ns1-np-default
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

All Egress is allowed.

17.1.1.11 Container Image Registry

Each tenant has their own container image registry that is hold by Harbor. The tenant private registry can only be accessed with right credentials, the image pull secret will be created in the tenant namespace, and it has been added to the default service account.

17.1.1.12 Chart Repo

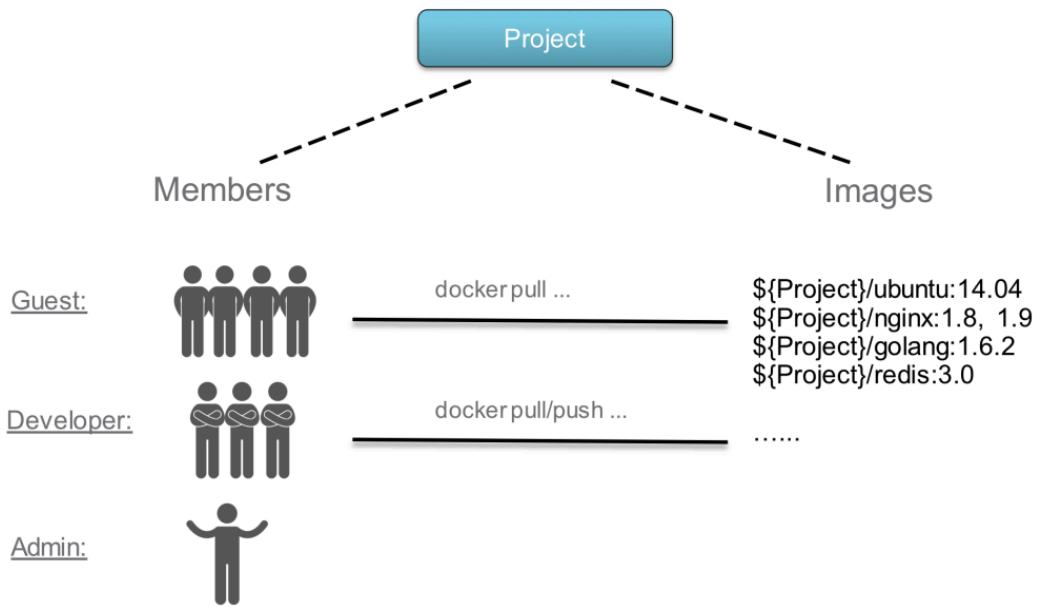
Each tenant has their own chart repo that is hold by Harbor. The tenant private chart repo can only be access with right credentials.

17.1.1.13 Harbor RBAC Roles

Project/tenant level RBAC roles

Harbor manages images through projects. You provide access to these images to users by including the users in projects and assigning one of the following roles to them.

Figure 20: Project/tenant level RBAC roles



- **Limited Guest:** A Limited Guest does not have full read privileges for a project. They can pull images but cannot push, and they cannot see logs or the other members of a project. For example, you can create limited guests for users from different organizations who share access to a project.
- **Guest:** Guest has read-only privilege for a specified project. They can pull and retag images, but cannot push.
- **Developer:** Developer has read and write privileges for a project.
- **Maintainer:** Maintainer has elevated permissions beyond those of ‘Developer’ including the ability to scan images, view replication jobs, and delete images and helm charts.
- **ProjectAdmin:** When creating a new project, you will be assigned the “ProjectAdmin” role to the project. Besides read-write privileges, the “ProjectAdmin” also has some management privileges, such as adding and removing members, starting a vulnerability scan.

System Level Roles

Besides the above roles, there are two system-level roles:

- **Harbor system administrator:** “Harbor system administrator” has the most privileges. In addition to the privileges mentioned above, “Harbor system administrator” can also list all projects, set an ordinary user as administrator, delete users and set vulnerability scan policy for all images. The public project “library” is also owned by the administrator.
- **Anonymous:** When a user is not logged in, the user is considered as an “Anonymous” user. An anonymous user has no access to private projects and has read-only access to public projects.

For full details of the permissions of the different roles, see [User Permissions By Role](#).

If you run Harbor in database authentication mode, you create user accounts directly in the Harbor interface. For information about how to create local user accounts, see [Create User Accounts in Database Mode](#).

If you run Harbor in LDAP/AD or OIDC authentication mode, you create and manage user accounts in your LDAP/AD or OIDC provider. Harbor obtains the users from the LDAP/AD or OIDC server and displays them in the **Users** tab of the Harbor interface.

Assigning the Harbor System Administrator Role

Harbor system administrators can assign the Harbor system administrator role to other users by selecting usernames and clicking **Set as Administrator** in the **Users** tab.

Figure 21: Users

Name	Admin	Email	Registration time
daniel	No	whatever@aaa.com	8/20/2018, 2:39 PM
jack	No	jiangd@vmware.com	8/20/2018, 2:38 PM

To delete users, select a user and click **DELETE**. Deleting users is only supported under database authentication mode.

User permissions by Role

Action	Limited Guest	Guest	Developer	Master	Project Admin
See the project configurations	✓	✓	✓	✓	✓
Edit the project configurations					✓
See a list of project members		✓	✓	✓	✓
Create/edit/delete project members					✓

Action	Limited Guest	Guest	Developer	Master	Project Admin
See a list of project logs		✓	✓	✓	✓
See a list of project replications				✓	✓
See a list of project replication jobs					✓
See a list of project labels				✓	✓
Create/edit/delete project labels				✓	✓
See a list of repositories	✓	✓	✓	✓	✓
Create repositories			✓	✓	✓
Edit/delete repositories				✓	✓
See a list of images	✓	✓	✓	✓	✓
Retag image		✓	✓	✓	✓
Pull image	✓	✓	✓	✓	✓
Push image			✓	✓	✓

Action	Limited Guest	Guest	Developer	Master	Project Admin
Scan/delete image				✓	✓
Add scanners to Harbor					
Edit scanners in projects					✓
See a list of image vulnerabilities	✓	✓	✓	✓	✓
See image build history	✓	✓	✓	✓	✓
Add/Remove labels of image			✓	✓	✓
See a list of helm charts	✓	✓	✓	✓	✓
Download helm charts	✓	✓	✓	✓	✓
Upload helm charts			✓	✓	✓
Delete helm charts				✓	✓
See a list of helm chart versions	✓	✓	✓	✓	✓
Download helm chart versions	✓	✓	✓	✓	✓

Action	Limited Guest	Guest	Developer	Master	Project Admin
Upload helm chart versions			✓	✓	✓
Delete helm chart versions				✓	✓
Add/Remove labels of helm chart version			✓	✓	✓
See a list of project robots				✓	✓
Create/edit/delete project robots					✓
See configured CVE whitelist	✓	✓	✓	✓	✓
Create/edit/remove CVE whitelist					✓
Enable/disable webhooks			✓	✓	✓
Create/delete tag retention rules			✓	✓	✓
Enable/disable tag retention rules			✓	✓	✓
Create/delete tag				✓	✓

Action	Limited Guest	Guest	Developer	Master	Project Admin
immutability rules					
Enable/ disable tag immutability rules				✓	✓
See project quotas	✓	✓	✓	✓	✓

 **Note:** Only the Harbor system administrator can edit project quotas and add new scanners.

NCS specific new RBAC roles created for tenants:

- Tenant-admin
 - Manage specific tenant: ns, user
 - Tenant level backup/restore
- Tenant-user
 - Manage applications within a tenant

 **Note:** During a new tenant creation, both "<tenant-name>-admin" and "<tenant-name>-user" accounts are created with default credentials set to "ncs@CSF_k8s".

 **Note:** "ncs-admin" does not have any (system or project/tenant level) Harbor RBAC roles assigned. This also means that by default "ncs-admin" is not created in Harbor. Check with NOKIA support if it is recommended to create a "ncs-admin" user account for both NCS and Harbor users.

17.1.1.14 Multi-Project and Multi-Tenant Admins

You may also need to create "multi-project" or "multi-tenant" admins who are allowed to manage multiple projects/tenants. These kinds of Harbor users may be needed, e.g., for NCOM integration for CNF orchestration. These types of admin users need to be able to have assigned multiple "ProjectAdmin" and "Tenant-admin" roles for selected tenants.

17.1.1.15 Rest API for Multi tenancy

ncm/api/v1/tenants: tenant management.

ncm/api/v1/tenants/{tenant_name}/namespaces: tenant namespace management.

It is designed for multiple namespaces within one tenant. It should be called directly in the current release.

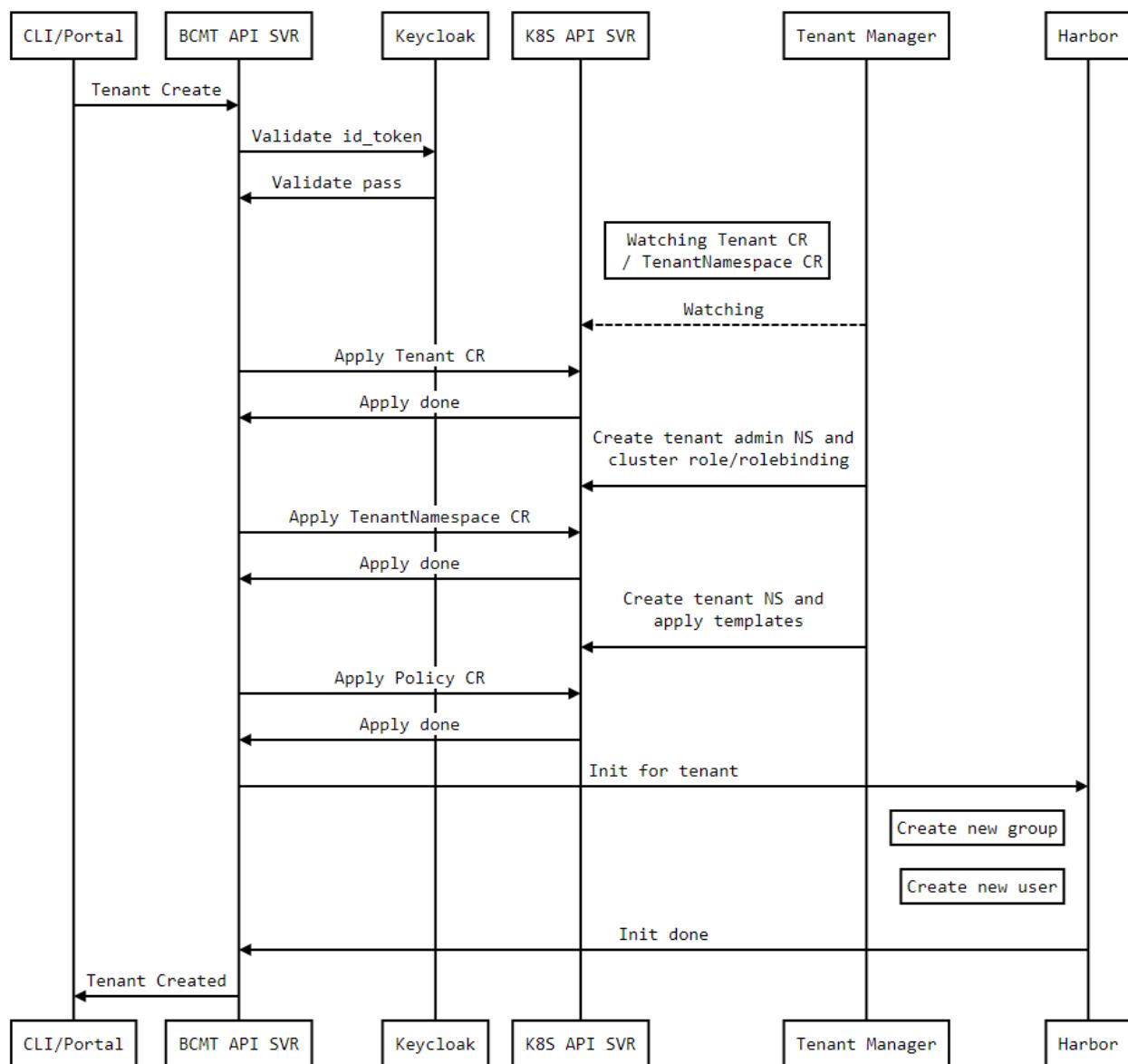
`ncm/api/v1/tenants/{tenant_name}/users`: tenant user management

Details on these APIs can be found in the NCS API document.

17.1.1.16 CLI with Multi tenancy

There is a "tenant" sub-command added for NCS. Details can be found in the NCS CLI guide.

17.1.1.17 Tenant Creation Flow



Note: During a new tenant creation, both “<tenant-name>-admin” and “<tenant-name>-user” accounts are created with default credentials set to "ncs@CSF_k8s".

17.2 Tenant Management Operation

This page provides instructions on how to create multi-tenant for cluster Administrator. How to deploy an APP on NCS in MT mode for Tenant user, refer to Application management.

Cluster Administrator

```
NCS-APP version 1.14.7 or higher  
NCS CLI version 20.12.1111-1.10.3 or higher  
Importer version 1.0.161 or higher
```

Install or upgrade NCS-APP with MT enabled

- Fresh install of the ncs-app with MT enabled:

```
helm upgrade app-api registry-name/ncm-app  
--namespace ncms  
--set env.AUTHORIZED_NS_PATTERN=%s-ns,  
env.ADMIN_AUTHORIZED_NS_PATTERN=ncm-admin,  
env.REJECT_PROJECT_HEADER=true,env.HELML_FAVORITE=3,  
env.HELML_BLACKLIST=2
```



Note: A fresh install of the NCS-APP is performed during the cluster installation. User has to use the update command to blacklist helm2 after the cluster is up and running.



Restriction: Limitations may still exist after updating the NCS-APP with the above command.

Log into the deployer node of the cluster (Cluster Administrator)

```
ncs user login --username ncm-admin --password testPassword
```

Retrieve ncs-admin configuration file for ncs multi tenancy operation

Log into the bcmt-admin container on the deployer node of the cluster to retrieve the configuration file and copy the config file (config.json) from the directory /usr/share

```
docker exec -it bcmt-admin -- bash
```

Create a tenant

The input required to create a tenant is in a JSON format. An example of the JSON snippet is shown below.

Associated to a tenant is a default admin tenant namespace following the format, tenant-name-admin. For example, a tenant netguard-base will have a default admin namespace netguard-base-admin and a default namespace netguard-base-admin-ns.

```
{  
    "name": "netguard-base",  
    "requireNamespacePrefix": false,  
    "harborProjectQuota": 20480,  
    "resourceQuota": {  
        "hard": {  
            "requests.cpu": "1",  
            "requests.memory": "1Gi",  
            "requests.storage": "1Gi",  
            "limits.cpu": "1",  
            "limits.memory": "1Gi"  
        }  
    },  
    "limitRange": {  
        "limits": [  
            {  
                "max": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "min": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "default": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "defaultRequest": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "maxLimitRequestRatio": {  
                    "cpu": "1",  
                    "memory": "1"  
                },  
                "type": "Container"  
            },  
            {  
                "max": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "min": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "default": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "defaultRequest": {  
                    "cpu": "1",  
                    "memory": "1Gi"  
                },  
                "maxLimitRequestRatio": {  
                    "cpu": "1",  
                    "memory": "1"  
                },  
                "type": "Container"  
            }  
        ]  
    }  
}
```

```
        "memory": "1Gi"
    },
    "min": {
        "cpu": "1",
        "memory": "1Gi"
    },
    "type": "Pod"
},
{
    "min": {
        "storage": "1Gi"
    },
    "max": {
        "storage": "1Gi"
    },
    "type": "PersistentVolumeClaim"
}
]
}
```

Create a tenant netguard-base using the previous tenant code snippet and the config file retrieved from the bcmt-admin container

```
ncs tenant create --config create-tenant.json
{
    "task-status": "running"
}
.....
{
    "task-status": "done"
}
```

List Tenants

```
ncs tenant list
{
    "tenants": [
        {
            "id": "dc778d66-db2d-4ff5-8148-d32e1dde0d74",
            "name": "netguard-base",
            "tenantAdminNamespaceName": "netguard-base-admin"
        },
        {
            "id": "c0a42739-c31d-47de-9aca-86b3c9bb92b5",
            "name": "netguard-smc",
            "tenantAdminNamespaceName": "netguard-smc-admin"
        }
    ]
}
```

}

 **Note:** A tenant user is created as -admin, for other tenant operations, refer to [Tenant Management Operation](#) on page 441. For - APP installation in tenant, refer to [Application management](#) on page 388.

Update Tenants

```
ncs tenant update --tenant_name test --config test_tenant.json
```

Where the test_tenant.json contains the modified tenant resource:

```
cat test_tenant.json

{
  "name": "test",
  "requireNamespacePrefix": false,
  "harborProjectQuota": 20480,
  "resourceQuota": {
    "hard": {
      "limits.cpu": "1500m",
      "limits.memory": "5Gi",
      "requests.cpu": "1",
      "requests.memory": "5Gi",
      "requests.storage": "13Gi"
    }
}
```

After the update operation one will see something like this:

```
ncs tenant show --tenant_name test
{
  "uid": "4b84d5b4-f7b3-4d78-aa77-23bb0b369f0a",
  "name": "test-admin-ns",
  "resourceQuota": {
    "hard": {
      "limits.cpu": "1500m",
      "limits.memory": "5Gi",
      "requests.cpu": "1",
      "requests.memory": "5Gi",
      "requests.storage": "13Gi"
    },
    "used": {
      "limits.cpu": "0",
      "limits.memory": "0",
      "requests.cpu": "0",
      "requests.memory": "0",
      "requests.storage": "0"
    },
    "limitRange": {}
  }
}
```

18 Fault Management and Performance Management

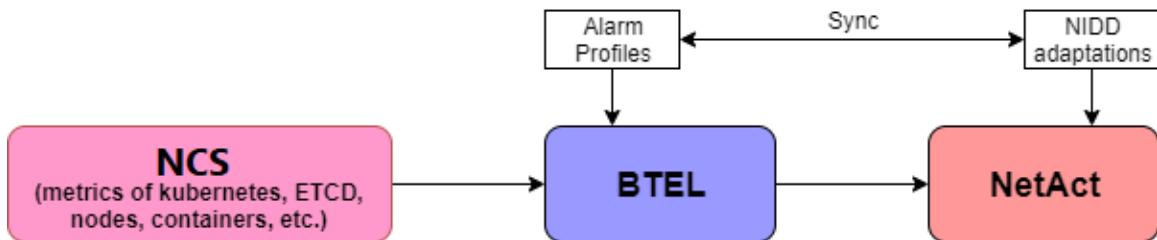
NCS relies on CSF Telemetry Blueprint (BTEL) to interface with NetAct. NCS defined measurements, counters, and alarms will be collected by NetAct.

18.1 Fault Management

Fault Management (FM) cares about alarms. NCS has prepared the Alarm Profiles and synched them to NIDD.

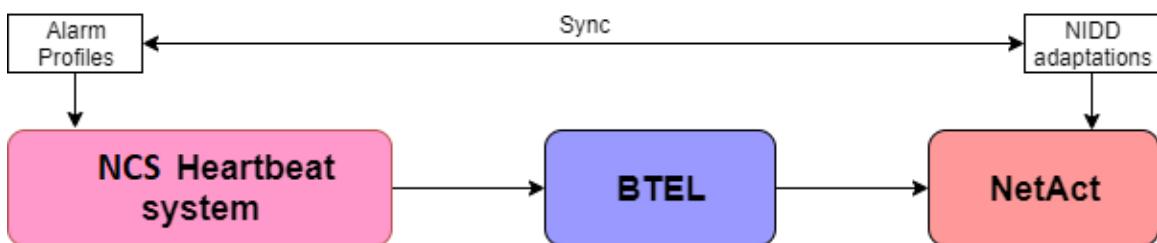
1. Based on the metrics exposed by NCS

NCS exposes metrics of kubernetes, ETCD, nodes, containers, etc. Based on the Alarm Profiles defined, alarms can be triggered and transferred to NetAct. Then the NetAct will do responses based on the NIDD adaptations synced with the Alarm Profiles.



2. Generated by NCS Heartbeat system

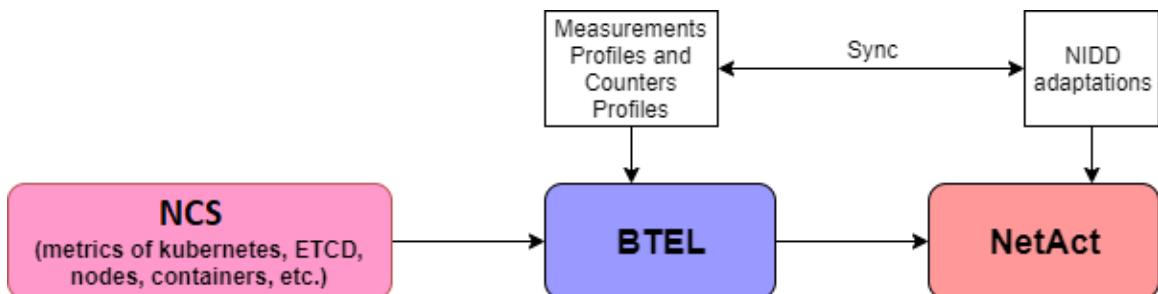
NCS has a self-check system called NCS Heartbeat, which monitors the running status of NCS and triggers alarms based on the Alarm Profiles defined. Alarms will be transferred to NetAct. Then the NetAct will do responses based on the NIDD adaptations synced with the Alarm Profiles.



18.2 Performance Management

Performance Management (PM) cares about measurements and counters. NCS has prepared the Measurements Profiles and Counters Profiles and sync them to NIDD.

Based on the Measurements Profiles and Counters Profiles, measurements and counters will be transferred to NetAct. Then the NetAct will show the data based on the NIDD adaptations synced with the Measurements Profiles and Counters Profiles.



18.3 Configuration

For BTEL configuration and deployment on NCS, see the NCS Addon package installation guide. The detailed Telemetry profile can be found under [Appendix: Example Telemetry profile yaml](#) on page 918 for the FM & PM integration.

18.4 NCS Logging

Server Logs

Change Level of Logs

By default the NCS server is provided with logs configured to WARN level. To activate all logs and modify a level, a log API is provided.

Examples of logs:

- **ncms**: allows you to show a helm command launched behind the REST API. INFO level for logs is sufficient for debugging
- **http.access**: allows to show Launched http request received by the server and returned response. DEBUG level allows to print incoming request PDU

The line below automatically set the log to INFO for the logger ncms:

```
# ncs app log
```

Output Example:

```
OK
200 OK
WARN -> DEBUG
WARNING: Cli API version greater than Server version.
Server upgrade recommended !
```

```
WARNING: SERVER : 1.6.6
WARNING: CLI : 1.6.8
```

Alternatively you can active the DEBUG level:

```
# ncs app log --name ncms -l DEBUG
```

Using this API enable to update value of env.LOGGERS from "ncms=WARN" to "ncms=DEBUG"

Output Example:

```
OK
200 OK
WARN -> DEBUG
WARNING: Cli API version greater then Server version.
Server upgrade recommended !
WARNING: SERVER : 1.6.6
WARNING: CLI : 1.6.8
```

Show Logs

Prints the ncs server logs. These are used by admin only. Further options are available, please check by adding --help. Like filters or tails etc.

By default, logs are configured to WARN level. To activate all logs and modify a level, use a log API listed above.

Examples of:

```
# ncs appli logs
```

Output Example:

```
OK
200 OK
2020-02-26 03:47:37,625 [Processing-
ThreadPool-17654] INFO ncms.applications - WARN -> DEBUG
2020-02-26 03:47:37,658 [Processing-
ThreadPool-17654] DEBUG ncms.filter.authentication - eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJIRThVm1VMkFsS11SOUxqdkQzRHM2eDRjU1p3LWRvY1FkaUp1UVBBMndNIn0.eyJqdGkiOiI3ZjkxOGZkNy01MjdkLTQyZDctOWYyZi1hZGQzMmYyOWFhzjYiLCJleHAioje1ODI2OTI0NTcsIm5iZi6MCwiaWF0IjoxNTgyNjg4ODU3LCJpc3MiOiJodHRwczovL2JjbXQtY2t1eS1ja2V5Lm5jbXMuc3Zjojg0NDMvYXV0aC9yZWFBsbXMvbmnTiiwiYXVkijpbinJ1YWxtLW1hbmFnZW11bnQiLCJhY2NvdW50Il0sInN1YiI6IjFhYTkzMGZmLTvkOTAtNGNhhy05NmNlLWIxYzAwNzIxOTVhMiIsInR5cCI6IkJ1YXJ1ciIsImF6cCI6Im5jbS1tYW5hZ2VyIiwiYXV0aF90aW11IjowLCJzZXNzaW9uX3N0YXR1IjoiZDU2MDA2OWEtMGM4MC00Mjb1LWFmYWYtYzg3NzgzzTEyMTR1IiwiYWNyIjoiMSIsInJ1YWxtX2FjY2Vz
```

```
cyI6eyJyb2xlcyI6WyJhZG1pbmlzdHJhdG9yIiwib2ZmbGluzV9hY2N1c3MiLCJ1b
WFfYXV0aG9yaXphdGlvbiJdfSwicmVzb3VyY2VfYWNjZXNzIjp7InJ1YWxtLW1hbm
FnZW1lbnQiOnsicm9sZXMiOlsibWFuYWd1LXJ1YWxtIiwibWFuYWd1LXVzZXJzIwi
dm11dy11c2VycyIsInZpZXctY2xpZW50cyIsIm1hbmFnZS1hdXR0b3JpemF0aW9uI
iwicXVlcnktY2xpZW50cyIsInF1ZXJ5LWdyb3VwcyIsInF1ZXJ5LXVzZXJzI119LCJ
uY20tbWFuYWd1ciI6eyJyb2xlcyI6WyJhZG1pbmlzdHJhdG9yI119LCJhY2NvdW50I
jp7InJvbGVzIjpBIm1hbmFnZS1hY2NvdW50IiwibWFuYWd1LWFjY291bnQtbgLuA3M
iLCJ2aWV3LXByb2ZpbGUixX19LCJzY29wZSI6Im9wZW5pZCBwcm9maWx1IGVtYWlsI
iwiZW1haWxfdmVyaWzpZWQiOmZhbHN1LCJncm91cHMiOlsiYWRtaW5pc3RyYXRvcij
dLCJwcmVmZXJyZWRfdXN1cm5hbWUiOijY20tYWRtaW4iLCJ1bWFpbCI6Im5jbS1hZ
G1pbkBub2tpYS5jb20ifQ.c2spQ5CylR6y8GhifpGrSKjN08ZZpe8jNFBFUcpP3mi
j6QzV2XrFKMaovMXS2QPvXKdEpVgaElzWvnx6GwpIYkJMC8ludLaPyvmkwT0Kd5f2
tJ13h_uEyhxC4rmbfcXyd4VnHbJgRt5n_jXAhWShpidGVlgt1CuNvvE1Lx-SrD4N56
n1uPg_pjG1wZyNO09cvMhA7EzxFe1mLdEP-XFtfkXVZ8WCx6VHq657IUkc9PKQ2LD
n3-J2j_7r43YZirjvsVAIphQ4aspl1PbieEqkQ6B96nHtpK8OHbJ1xJd8Yh8fHOft0
lsZMRTYobW03Jn45tCXfOvRX1-C10T98fQ
```

```
2020-02-26 03:47:37,664 [Processing-
ThreadPool-17654] DEBUG ncms.filter.authentication - {"jti":"7f918
fd7-527d-42d7-9f2f-add32f29aaf6","exp":1582692457,"nbf":0,"iat":1582688857,"iss":"https://bcmt-ckey-ckey.ncms.svc:8443/auth/realms/ncm","aud":["realm-management","account"],"sub":"1aa930ff-5d90-4cac-96ce-b1c0072195a2","typ":"Bearer","azp":"ncm-manager","auth_time":0,"session_state":"d560069a-0c80-420e-afaf-c87783e1214e","acr":"1","realm_access":{"roles":["administrator","offline_access","uma_authorization"]},"resource_access":{"realm-management":{"roles":["manage-realm","manage-users","view-users","view-clients","manage-authorization","query-clients","query-groups","query-users"]}},"ncm-manager":{"roles":["administrator"]}, "account": {"roles": ["manage-account", "manage-account-links", "view-profile"]}}, "scope": "openid profile email", "email_verified":false, "groups": ["administrator"], "preferred_username": "ncm-admin", "email": "ncm-admin@nokia.com"}
```

```
2020-02-26 03:47:37,688 [Processing-
ThreadPool-17654] WARN ncm.audit - {"facility": "25", "level": "info", "log": {"event-data": {"error-code": "E_SUCCESS", "interface": "NcmCli-Nokia/20.03.553-1.6.8", "operation": "PUT /applications/logger/ncms?l=DEBUG&home=/root/.helm/", "result": "success", "session-id": "151", "user": "\\\\"ncm-admin\\\""}, "event-type": "N_USER_OPER", "message": ":"}, "time": "2020-02-26T03:47:37.684Z", "timezone": "UTC", "type": "log"}
```

```
2020-02-26 03:47:41,498 [Processing-
ThreadPool-17654] DEBUG ncms.filter.pre - change /*/* to /*/*;q=0.2, text/plain;q=0.5, text/uri-list;q=0.8
```

```
2020-02-26 03:47:41,504 [Processing-
ThreadPool-17654] DEBUG ncms.filter.pre - use multi-tenant context: --tiller-namespace kube-system --home [/root/.helm/] with namespaces scoped in null
```

```
2020-02-26 03:47:41,513 [Processing-
ThreadPool-17654] DEBUG ncms.command - executing [1105][30s]:helm
version &>/dev/null
2020-02-26 03:47:41,718 [WaitForProcess-
java.lang.UNIXProcess@60f9c0f6] DEBUG ncms.command - executed [1105][30s][0]
2020-02-26 03:47:47,069 [Processing-
ThreadPool-17655] DEBUG ncms.filter.pre - change /* to */;q=0.2
,text/plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:47:47,069 [Processing-
ThreadPool-17655] DEBUG ncms.filter.pre - use multi-tenant context:
--tiller-namespace kube-system --home [/root/.helm/] with
namespaces scoped in null
2020-02-26 03:47:57,077 [Processing-
ThreadPool-17657] DEBUG ncms.filter.pre - change /* to */;q=0.2
,text/plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:47:57,077 [Processing-
ThreadPool-17657] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:01,473 [Processing-
ThreadPool-17657] DEBUG ncms.filter.pre - change /* to */;q=0.2,text/
plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:48:01,473 [Processing-
ThreadPool-17657] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:01,475 [Processing-
ThreadPool-17657] DEBUG ncms.command - executing [1242][30s]:helm version &>/
dev/null
2020-02-26 03:48:01,622 [WaitForProcess
-java.lang.UNIXProcess@44d17ce6] DEBUG ncms.command - executed [1242][30s][0]
2020-02-26 03:48:07,092 [Processing-
ThreadPool-17659] DEBUG ncms.filter.pre - change /* to */;q=0.2,text/
plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:48:07,093 [Processing-
ThreadPool-17659] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:17,066 [Processing-
ThreadPool-17661] DEBUG ncms.filter.pre - change /* to */;q=0.2,text/
plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:48:17,070 [Processing-
ThreadPool-17661] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:21,478 [Processing-
ThreadPool-17662] DEBUG ncms.filter.pre - change /* to */;q=0.2,text/
plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:48:21,478 [Processing-
```

```
ThreadPool-17662] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:21,480 [Processing-
ThreadPool-17662] DEBUG ncms.command - executing [795][30s]:helm version &>/
dev/null
2020-02-26 03:48:21,641 [WaitForProcess
- java.lang.UNIXProcess@21a39c87] DEBUG ncms.command - executed [795][30s][0]
2020-02-26 03:48:27,073 [Processing-
ThreadPool-17663] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
2020-02-26 03:48:27,074 [Processing-
ThreadPool-17663] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:37,067 [Processing-
ThreadPool-17665] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
2020-02-26 03:48:37,067 [Processing-
ThreadPool-17665] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:41,483 [Processing-
ThreadPool-17665] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
2020-02-26 03:48:41,483 [Processing-
ThreadPool-17665] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:41,484 [Processing-
ThreadPool-17665] DEBUG ncms.command - executing [1490][30s]:helm version &>/
dev/null
2020-02-26 03:48:41,678 [WaitForProcess
- java.lang.UNIXProcess@56135474] DEBUG ncms.command - executed [1490][30s][0]
2020-02-26 03:48:47,073 [Processing-
ThreadPool-17667] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
2020-02-26 03:48:47,074 [Processing-
ThreadPool-17667] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:48:57,064 [Processing-
ThreadPool-17669] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
2020-02-26 03:48:57,066 [Processing-
ThreadPool-17669] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:49:01,480 [Processing-
ThreadPool-17670] DEBUG ncms.filter.pre - change /* to */;q=0.2;text/
plain;q=0.5;text/uri-list;q=0.8
```

```

2020-02-26 03:49:01,481 [Processing-
ThreadPool-17670] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:49:01,483 [Processing-
ThreadPool-17670] DEBUG ncms.command - executing [1922][30s]:helm version &>/dev/null
2020-02-26 03:49:01,544 [Processing-
ThreadPool-17671] DEBUG ncms.filter.pre - change /* to */;q=0.2,text/
plain;q=0.5,text/uri-list;q=0.8
2020-02-26 03:49:01,544 [Processing-
ThreadPool-17671] DEBUG ncms.filter.pre - use multi-tenant context:--tiller-
namespace kube-system --home [/root/.helm/] with namespaces scoped in null
2020-02-26 03:49:01,544 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication.jersey - off
2020-02-26 03:49:01,545 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication.keycloak - begins
2020-02-26 03:49:01,547 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication.keycloak - proceeding...
2020-02-26 03:49:01,547 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication.keycloak - ends
2020-02-26 03:49:01,547 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication - begins
2020-02-26 03:49:01,547 [Processing-
ThreadPool-17671] DEBUG ncms.filter.authentication - ends
2020-02-26 03:49:01,548 [Processing-
ThreadPool-17671] INFO ncms.applications - invoking logs with 100 null

WARNING: Cli API version greater
then Server version. Server upgrade recommended !
WARNING: SERVER : 1.6.6
WARNING: CLI : 1.6.8

```

Show Tiller Logs

Here, you can show logs related to tiller event. Tiller logs are also returned by default for the profile status API.

```

$ ncs appli logsTiller -h
NAME:
ncs application logsTiller - logs the tiller for given namespace
and project related to ncm events

USAGE:
ncs application logsTiller [command options] [arguments...]

OPTIONS:

```

```
--follow          <follow>           Follow the log stream of the pod. Defaults
                           to false (optional)
--limitBytes      <limitBytes>        The number of bytes to read from the
                           server before terminating the log output. (optional)
--NCS-Project    <NCS-Project>       project, required when used in a multi-
                           tenant K8S (optional)
--pretty          <pretty>            If 'true', then the output is pretty
                           printed (optional)
--sinceSeconds   <sinceSeconds>       A relative time in seconds before the
                           current time from which to show logs. (optional)
--tail            <tail>              If set, the number of lines from the end
                           of the logs to show. (optional)
--timeStamps     <timeStamps>         If true, add an RFC3339 or RFC3339Nano
                           timestamp at the beginning of every line of log output. Defaults to false
                           (optional)
```

Output Example:

```
$ ncm appli logsTiller -h
NAME:
ncs application logsTiller - logs the tiller for given namespace and project
related to ncm events

USAGE:
ncs application logsTiller [command options] [arguments...]

OPTIONS:
--follow          <follow>           Follow the log stream of the pod. Defaults
                           to false (optional)
--limitBytes      <limitBytes>        The number of bytes to read from the
                           server before terminating the log output. (optional)
--NCS-Project    <NCS-Project>       project, required when used in a multi-
                           tenant K8S (optional)
--pretty          <pretty>            If 'true', then the output is pretty
                           printed (optional)
--sinceSeconds   <sinceSeconds>       A relative time in seconds before the
                           current time from which to show logs. (optional)
--tail            <tail>              If set, the number of lines from the end
                           of the logs to show. (optional)
--timeStamps     <timeStamps>         If true, add an RFC3339 or RFC3339Nano
                           timestamp at the beginning of every line of log output. Defaults to false
                           (optional)
```

CLI logs

To get the complete debug logs while executing ncs commands, use "--log debug" before each command:

Examples:

```
# ncs --log debug appli list
```

Output Example:

```
DEBUG: Check Token for user : ncs-admin
      DEBUG: Get Token _endpoint : https://10.99.12.200:8082/
ncm/api/v1
      DEBUG: No authentication request sent for user : ncs-
admin with token : true
      DEBUG: Command : gettoken
      DEBUG: Uri : /ncm/users/login
      DEBUG: HttpCommand : post
      DEBUG: Parameters : map[]
      DEBUG: Header parameters : map[]
      DEBUG: Form parameters : map[]
      DEBUG: Form Data param
{"password": "*****", "username": "ncs-admin"}
      DEBUG: Path parameters : map[]
      DEBUG: Boolean parameters : map[]
      DEBUG: Invoke Pass Param https://10.99.12.200:8082/ncm/
api/v1 /ncm/users/login
      DEBUG: formparams : map[]
      DEBUG: Form Data param
{"password": "*****", "username": "ncs-admin"}
      DEBUG: boolparams : map[]
      DEBUG: formparams body len : 0
      DEBUG: content type :
      DEBUG: command : gettoken
      DEBUG: Build request : https://10.99.12.200:8082/ncm/api/
v1 /users/login post
      DEBUG: Request built with content type :
      DEBUG: No content type on the request
      DEBUG: Cli main version : 20.03.553-1.6.8
      DEBUG: Request sent : curl -X 'POST' -d '' -H 'Accept-
Encoding: identity' -H 'Cache-Control: no-cache, no-store, must-revalidate,
proxy-revalidate, max-age=0, s-maxage=0' -H 'Cluster: local' -H 'Content-
Type: application/json' -H 'User-Agent: NcmCli-Nokia/20.03.553-1.6.8'
'https://10.99.12.200:8082/ncm/api/v1/users/login'
      DEBUG: Formparam k : JsonStr
      DEBUG: Formparam contains password: ****
      DEBUG: fparams contains password: ****
      DEBUG: Formparam v contains password: ****
      DEBUG: Adding content type application/json
```

```
DEBUG: For gettoken commands sending request type json
DEBUG: Token received 200
DEBUG: User authentified
DEBUG: Command : list
DEBUG: Uri : /ncm/applications
DEBUG: HttpCommand : get
DEBUG: Parameters : map[fields: flags:]
DEBUG: Header parameters : map[Accept:/*/* NCS-Project:
produces:[text/uri-list, text/plain, application/json]]
DEBUG: Form parameters : map[]
DEBUG: Path parameters : map[]
DEBUG: Boolean parameters : map[resources:false]
DEBUG: Invoke Pass Param https://10.99.12.200:8082/ncm/
api/v1 /ncs/applications
DEBUG: formparams : map[]
DEBUG: boolparams : map[resources:false]
DEBUG: formparams body len : 0
DEBUG: content type :
DEBUG: command : list
DEBUG: Build request : https://10.99.12.200:8082/ncm/api/
v1 /applications get
DEBUG: Request built with content type :
DEBUG: No content type on the request
DEBUG: Cli main version : 20.03.553-1.6.8
DEBUG: Token :
eyJhbGciOiJSUzI1NiIiInR5cCIgOiAiSldUIiwia2lkIiA6ICJIRhtVm1VMkFsS1lSOUxqdkQzRHM2eDRjU1p3
mikuEl2wjmuzH94Mm6itc_4vKSXQoGofsn7hYWRXJf14ABdj3UD_rpFOz3VmWvVDW2RL7PDSIcT5XNXbUi8ujLgR
ug4pb8mEOsJMTVfbYxpR7ZChe94U2RjI7Ma-73zPk-0CYe0Lla8aK5QGhc9iG56xBI-
wm_nIQYdfS01ffDt0ai-
q8ULyIZ1B4PKiEovIVOj26alF_A5HbgaRHRYdwnx4DtwCkfD0VzzErk20CM8hPNibHHwpt1ndW704igo9xXL05--
ojJwEsN-h7t2rmO8PQ
[http] 2020/02/26 03:55:50 HTTP Request: GET /ncm/api/v1/
applications HTTP/1.1
Host: 10.99.12.200:8082
Accept: /*
Accept-Encoding: identity
Authorization: Bearer
eyJhbGciOiJSUzI1NiIiInR5cCIgOiAiSldUIiwia2lkIiA6ICJIRhtVm1VMkFsS1lSOUxqdkQzRHM2eDRjU1p3
mikuEl2wjmuzH94Mm6itc_4vKSXQoGofsn7hYWRXJf14ABdj3UD_rpFOz3VmWvVDW2RL7PDSIcT5XNXbUi8ujLgR
ug4pb8mEOsJMTVfbYxpR7ZChe94U2RjI7Ma-73zPk-0CYe0Lla8aK5QGhc9iG56xBI-
wm_nIQYdfS01ffDt0ai-
q8ULyIZ1B4PKiEovIVOj26alF_A5HbgaRHRYdwnx4DtwCkfD0VzzErk20CM8hPNibHHwpt1ndW704igo9xXL05--
ojJwEsN-h7t2rmO8PQ
Cache-Control: no-cache, no-store, must-revalidate, proxy-
revalidate, max-age=0, s-maxage=0
```

```

Cluster: local
Content-Type: application/json
User-Agent: NcmCli-Nokia/20.03.553-1.6.8

[curl] 2020/02/26 03:55:50 CURL
command line: curl -X 'GET' -d '' -H 'Accept: */*' -H
'Accept-Encoding: identity' -H 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJIRThtVm1VMkFsS1lSOUxqdkQzRHM2eDRjU3
mikuEl2wjmuzH94Mm6itc_4vKSXQoGofsn7hYWRXJf14ABdj3UD_rpFOz3VmWvVDW2RL7PDSIcT5XNbUi8u
ug4pb8mEOsJMTVfbYxpR7ZChe94U2RjI7Ma-73zPk-0CYe0Lla8aK5QGhc9iG56xBI-
wm_nIQYdfS01ffDt0ai-
q8ULyIZ1B4PKiEovIVOj26alF_A5HbgaRHHDwnx4DtwCkfD0VzzErdK20CM8hPNibHHwpt1ndW704igo9xXL05-
ojJwEsN-h7t2rmO8PQ' -H 'Cache-Control: no-cache, no-store, must-revalidate,
proxy-revalidate, max-age=0, s-maxage=0' -H 'Cluster: local' -H 'Content-
Type: application/json' -H 'User-Agent: NcmCli-Nokia/20.03.553-1.6.8'
'https://10.99.12.200:8082/ncm/api/v1/applications'

[curl] 2020/02/26 03:55:50 HTTP Response: HTTP/1.1 200 OK
Connection: close
Transfer-Encoding: chunked
Cache-Control: no-cache, no-store, no-transform, must-
revalidate, proxy-revalidate, max-age=0, s-maxage=0
Content-Security-Policy: default-src 'self' https://
apis.google.com https://fonts.googleapis.com https://fonts.googleapis.com
https://ajax.googleapis.com https://www.google-analytics.com https://
storage.googleapis.com https://pwc.int.net.nokia.com/ https://github.com/
nokia/danm https://repo.lab.pl.alcatel-lucent.com; script-src 'self'; style-
src 'self' 'unsafe-inline'; img-src 'self' https: data:; connect-src 'self';
font-src 'self' https: data:; object-src 'none'; media-src 'self' https:;
form-action 'self'; frame-ancestors 'self';

Content-Type: text/plain
Date: Wed, 26 Feb 2020 03:55:50 GMT
Expires: 0
Pragma: no-cache
Server: nginx
Strict-Transport-Security: max-age=31536000;

includeSubDomains
X-Api-Version: 1.6.6
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block

29e
NAME          UPDATED          CHART
bcmt-cmdb     Fri Feb 21 02:40:28 2020  cmdb-7.5.0
bcmt-ckey      Fri Feb 21 02:43:33 2020  ckey-8.7.6

```

app-1.8.8	app-api	Fri Feb 21 02:43:58 2020	ncm-
manager-v0.12.0	cert-manager	Fri Feb 21 02:44:09 2020	cert-
cinder-1.16.4	cbur-master	Fri Feb 21 02:46:14 2020	cbur-1.3.4
ceph-2.1.2	csi-cinder	Fri Feb 21 02:46:51 2020	csi-
local-0.1.0	rook-ceph	Fri Feb 21 03:23:59 2020	rook-
rook-0.1.0	cinfo-local	Fri Feb 21 06:03:08 2020	cinfo-
storage-0.1.0	cinfo-rook	Fri Feb 21 06:11:59 2020	cinfo-
	cinfo-storage	Fri Feb 21 06:13:31 2020	cinfo-
	0		
	OK		
	200 OK		
	NAME	UPDATED	CHART
app-1.8.8	bcmt-cmdb	Fri Feb 21 02:40:28 2020	cmdb-7.5.0
manager-v0.12.0	bcmt-ckey	Fri Feb 21 02:43:33 2020	ckey-8.7.6
cinder-1.16.4	app-api	Fri Feb 21 02:43:58 2020	ncm-
ceph-2.1.2	cert-manager	Fri Feb 21 02:44:09 2020	cert-
local-0.1.0	cbur-master	Fri Feb 21 02:46:14 2020	cbur-1.3.4
rook-0.1.0	csi-cinder	Fri Feb 21 02:46:51 2020	csi-
storage-0.1.0	rook-ceph	Fri Feb 21 03:23:59 2020	rook-
	cinfo-local	Fri Feb 21 06:03:08 2020	cinfo-
	cinfo-rook	Fri Feb 21 06:11:59 2020	cinfo-
	cinfo-storage	Fri Feb 21 06:13:31 2020	cinfo-
	Server API Version : 1.6.6		
	DEBUG: Response Header: map[Cache-Control:[no-cache, no-store, no-transform, must-revalidate, proxy-revalidate, max-age=0, s-maxage=0] Content-Security-Policy:[default-src 'self' https://apis.google.com https://fonts.googleapis.com https://fonts.googleapis.com https://ajax.googleapis.com https://www.google-analytics.com https://storage.googleapis.com https://pwc.int.net.nokia.com/ https://github.com/nokia/danm https://repo.lab.pl.alcatel-lucent.com; script-src 'self';		

```

style-src 'self' 'unsafe-inline'; img-src 'self' https: data:; connect-src
'self'; font-src 'self' https: data:; object-src 'none'; media-src 'self'
https:; form-action 'self'; frame-ancestors 'self';] Content-Type:[text/
plain] Date:[Wed, 26 Feb 2020 03:55:50 GMT] Expires:[0] Pragma:[no-cache]
Retry-Count:[0] Server:[nginx] Strict-Transport-Security:[max-age=31536000;
includeSubDomains] X-Api-Version:[1.6.6] X-Content-Type-Options:[nosniff] X-
Frame-Options:[SAMEORIGIN] X-Xss-Protection:[1; mode=block]]

        DEBUG: Status of the response: 200 OK
        DEBUG: Body returned : NAME           UPDATED
        CHART
            bcmt-cmdb      Fri Feb 21 02:40:28 2020      cmdb-7.5.0
            bcmt-ckey      Fri Feb 21 02:43:33 2020      ckey-8.7.6
            app-api        Fri Feb 21 02:43:58 2020      ncm-
app-1.8.8
            cert-manager   Fri Feb 21 02:44:09 2020      cert-
manager-v0.12.0
            cbur-master    Fri Feb 21 02:46:14 2020      cbur-1.3.4
            csi-cinder     Fri Feb 21 02:46:51 2020      csi-
cinder-1.16.4
            rook-ceph      Fri Feb 21 03:23:59 2020      rook-
ceph-2.1.2
            cinfo-local    Fri Feb 21 06:03:08 2020      cinfo-
local-0.1.0
            cinfo-rook     Fri Feb 21 06:11:59 2020      cinfo-
rook-0.1.0
            cinfo-storage   Fri Feb 21 06:13:31 2020      cinfo-
storage-0.1.0
            WARNING: Cli API version greater then Server version.
            Server upgrade recommended !
            WARNING: SERVER : 1.6.6
            WARNING: CLI : 1.6.8
            DEBUG : Log file is under /var/log/ncs.log

```

18.5 Logs Location

There are different levels of logs found in NCS.

- OS, systemd logs are located in `/var/log`
- POD/container logs
 - If docker's `log_driver` is **json-file**, logs are in `/var/log/containers`. When pods are deleted by kubelet, only current and last container logs are kept.
 - If docker's `log_driver` is **journald**, logs are in `/var/log`

Docker's log_driver is configured in the bcmt_config.json configuration file.

- Audit logs
 - BCMT API audit logs are in /opt/bcmt/log/audit.log
 - Kubernetes API audit logs are in /data0/log/bcmt/kube-apiserver/audit.log

To enable advance auditing on master node for the Kubernetes API server, set advanced_auditing to "true" in the bcmt_config.json configuration file.

- NCS logs are located in /opt/bcmt/log/
- OneKey client log, generated when run the cmd. Logs are in /var/log
- Directories referenced:

/data0/log
/data0/log/bcmt/
/data0/log/bcmt/kube-apiserver/

About the log collection:

- For Remote centralized log with external EK/BELK server, BCMT fluentd or BELK-fluentd is required.
- Only log collection and analysis locally, inside BELK + fluentd + Kibana are required.

For more information, refer to section: *Logging*.

18.6 Audit

There are a few types of Audit mechanism performed at NCS Cluster.

1. Kubernetes Auditing

Kubernetes auditing can be enabled by advanced-auditing field.

Kubernetes auditing provides a security-relevant chronological set of records documenting the sequence of activities that have affected system by individual users, administrators or other components of the system.

Kube-apiserver performs auditing. Each request at each stage of its execution generates an event, which is then pre-processed according to a certain policy and written to a backend. The audit policy determines what recorded and the backends persist the records. The current backend implementations include Kube-apiserver logs files and fluentd if EFK(Elastic + Fluentd + Kibana) is used.

For more information, see: <https://kubernetes.io/docs/tasks/debug-application-cluster/audit/>



Note: The audit logging feature in Kubernetes increases the memory consumption of the API server because some context required for audigin is stored for each request. Additionally, memory consumption depends on the audit policy configuration. The disk and performance also would be impacted. NCS provides a default audit policy file. Customers can modify this default audit policy file or use their own audit policy file to replace the default

one. Audit policy file can be found in `/etc/kubernetes/audit-policy.yaml` when `advanced_auditing` is true. For usage of **Advanced Auditing** in the NCS Manager, see section: *CaaS Cluster*.

2. Falco

NCS integrates the Falco into NCS Cluster. Users can install Falco using helm.

Falco is a behavioral activity monitor designed to detect anomalous activity in your applications. You can use Falco to monitor run-time security of your Kubernetes applications and internal components. Run the command `kubectl log` to check the audit log.

3. Auditd

`auditd` is the userspace component to the Linux Auditing System. It's responsible for writing audit records to the disk. Viewing the logs is done with the `ausearch` or `aureport` utilities. Configuring the audit rules is done with the `auditctl` utility. During startup, the rules in `/etc/audit/audit.rules` are read by `auditctl`. The audit daemon itself has some configuration options that the admin may wish to customize. They are found in the `auditd.conf` file.

NCS makes use of `auditd` to monitor/audit the OS system.

`auditd` is useful for the docker daemon/service. To audit the files:

```
/usr/bin/docker
/etc/docker
/etc/systemd/system/docker.service
/etc/docker/daemon.json
/var/lib/docker
/usr/bin/docker-containerd
```

`pam_tty_audit.so` are added to `/etc/pam.d/system-auth`, `/etc/pam.d/password-auth`, and `/etc/pam.d/sshd` to audit user commands on NCS nodes through tty.

18.7 Monitoring

Two main tools are used for Monitoring, they are:

[Zabbix](#) on page 545

[Prometheus](#) on page 467



Note:

- For NCS on VMs, Zabbix is not used. BTEL and its components provide NCS Kubernetes and CNFs monitoring.
- For NCS on Bare-metal, BTEL only performs CNFs monitoring and this is optional. There are also 2 CALM and ELK instances (separated for NCS and CNFs). See [Monitoring for Bare-metal](#) on page 545.

For Monitoring Cluster Deployment, see *Monitoring Cluster Deployment* in the *NCS Manager Guide*.

18.7.1 Monitoring for both VM and Bare Metal

18.7.1.1 Node problem detector

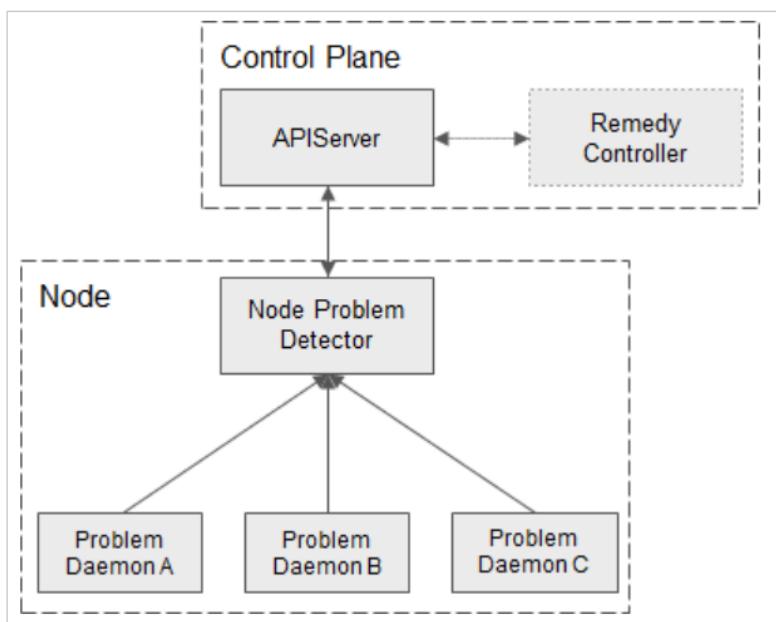
This tool is supported in NCS on VM.

The Node Problem Detector aims to make various node problems visible to the upstream layers in cluster management stack. It is a daemon which runs on each node, detects node problems and reports them to apiserver. Node Problem Detector can either run as a DaemonSet or run standalone.

During early releases, Kubernetes suffered from node problems, (especially docker problems). Sometimes the kernel encountered deadlock, or the docker just hung and nothing would work anymore. However, Kubernetes knew nothing about these problems, so it still committed a series of scheduling pods and fails.

To remedy, it was necessary to make these node problems visible to the control plane, and then try to fix these bad nodes. So Node Problem Detector was introduced in Kubernetes 1.3.

18.7.1.1.1 Node problem detector architecture



Given that users run Kubernetes in different environments, it is impossible to invent a daemon to handle all problems in all these environments. So the Node Problem Detector gathers problems from sub-daemons, or what are called "problem daemons".

Node Problem Detector: A DaemonSet detects node problems and reports them to APIServer.

Problem Daemon: A sub-daemon monitors a specific kind of node problems. It could be a tiny daemon dedicated for kubernetes or an existing solution integrated.

Remedy Controller: (Future Scope in Kubernetes, not implemented yet)

18.7.1.2 User Facing API

Node Problems will be reported as Node Conditions and Events.

NodeCondition: A field in NodeStatus describes the condition of a Running node, such as Ready, OutOfDisk, MemoryPressure etc.

Event: A report of an event somewhere in the cluster.

A Node problem is reported as NodeCondition or Event:

NodeCondition => Permanent problem making the node unavailable

- e.g. KernelDeadlock, DockerHung, BadDisk etc

Event => Temporary problem but informative

- e.g. OOM Kill etc

18.7.1.3 Node problem detection rule

Node problems are detected by rules defined by json file as below:

Kernel Rules:

```
{
  "plugin": "journald",
  "pluginConfig": {
    "source": "kernel"
  },
  "logPath": "/var/log/journal",
  "lookback": "5m",
  "bufferSize": 10,
  "source": "kernel-monitor",
  "conditions": [
    {
      "type": "KernelDeadlock",
      "reason": "KernelHasNoDeadlock",
      "message": "kernel has no deadlock"
    }
  ],
  "rules": [
    {
      "type": "temporary",
      "reason": "OOMKilling",
      "pattern": "Kill process \\\d+ (.+) score \\\d+ or sacrifice child\\nKilled process \\\d+ (.+) total-vm:\\\\d+kB, anon-rss:\\\\d+kB, file-rss:\\\\d+kB"
    },
    {
      "type": "temporary",
      "reason": "MemoryPressure"
    }
  ]
}
```

```

        "reason": "TaskHung",
        "pattern": "task \\S+:\\w+ blocked for more than \\w+ seconds\\."
    },
    {
        "type": "temporary",
        "reason": "UnregisterNetDevice",
        "pattern": "unregister_netdevice: waiting for \\w+ to become free.
Usage count = \\d+"
    },
    {
        "type": "temporary",
        "reason": "KernelOops",
        "pattern": "BUG: unable to handle kernel NULL pointer dereference
at .*"
    },
    {
        "type": "temporary",
        "reason": "KernelOops",
        "pattern": "divide error: 0000 \\#[\\d+]\\] SMP"
    },
    {
        "type": "permanent",
        "condition": "KernelDeadlock",
        "reason": "AUFSUmountHung",
        "pattern": "task umount\\.aufs:\\w+ blocked for more than \\w+
seconds\\."
    },
    {
        "type": "permanent",
        "condition": "KernelDeadlock",
        "reason": "DockerHung",
        "pattern": "task docker:\\w+ blocked for more than \\w+ seconds\\.
."
    }
]
}

```

Docker Rules:

```
{
    "plugin": "journald",
    "pluginConfig": {
        "source": "docker"
    },
    "logPath": "/var/log/journal",
}
```

```

    "lookback": "5m",
    "bufferSize": 10,
    "source": "docker-monitor",
    "conditions": [ ],
    "rules": [
        {
            "type": "temporary",
            "reason": "CorruptDockerImage",
            "pattern": "Error trying v2 registry: failed to register layer:
rename /var/lib/docker/image/(.+) /var/lib/docker/image/(.+): directory not
empty.*"
        }
    ]
}

```

18.7.1.4 Starting and Stopping the Node problem detector

In NCS, the Node Problem Detector can be started and stopped using the following ncs cli commands.

Start Node Problem Detector:

```
ncs service npd start
```

Stop Node Problem Detector:

```
ncs service npd stop
```

18.7.1.5 Node condition reported by Node problem detector

Table 41: NodeCondition

NodeCondition	Description
KernelDeadlock	kernel encounter deadlock or not
ReadonlyFilesystem	filesystem is read-only or not
DiskUtilization	all partition utilizations are ok or not (Disk usage more than 75%)
DiskHealth	all partitions are writable or not

18.7.1.2 Escalator

The Escalator is used to collect node conditions, node events and pods events for nodes in a NCS Cluster. It is started as a deployment in a NCS cluster by default with 1 replica.

It watches changes of node condition and events that can be shown by the "kubectl describe nodes ***" command.

Example

Conditions:

Type	Status	LastHeartbeatTime		
LastTransitionTime			Reason	Message
---	-----	-----		
-----	-----	-----	-----	-----
MemoryPressure	False	Wed, 29 Apr 2020 05:05:33 +0000		Wed, 29 Apr 2020
04:52:21 +0000	KubeletHasSufficientMemory	kubelet has sufficient memory available		
DiskPressure	False	Wed, 29 Apr 2020 05:05:33 +0000		Wed, 29 Apr 2020
04:52:21 +0000	KubeletHasNoDiskPressure	kubelet has no disk pressure		
PIDPressure	False	Wed, 29 Apr 2020 05:05:33 +0000		Wed, 29 Apr 2020
04:52:21 +0000	KubeletHasSufficientPID	kubelet has sufficient PID available		
Ready	True	Wed, 29 Apr 2020 05:05:33 +0000		Wed, 29 Apr 2020
04:52:31 +0000	KubeletReady	kubelet is posting ready status		

Events:

Type	Reason	Age	From
Message			
---	-----	-----	-----
-----	-----	-----	-----
Normal	Starting	13m	kubelet, bcmt-worker-01
	Starting kubelet.		
Normal	NodeHasSufficientMemory	13m (x2 over 13m)	kubelet, bcmt-worker-01
	Node bcmt-worker-01 status is now: NodeHasSufficientMemory		
Normal	NodeHasNoDiskPressure	13m (x2 over 13m)	kubelet, bcmt-worker-01
	Node bcmt-worker-01 status is now: NodeHasNoDiskPressure		
Normal	NodeHasSufficientPID	13m (x2 over 13m)	kubelet, bcmt-worker-01
	Node bcmt-worker-01 status is now: NodeHasSufficientPID		
Normal	NodeAllocatableEnforced	13m	kubelet, bcmt-worker-01
	Updated Node Allocatable limit across pods		
Normal	NodeReady	13m	kubelet, bcmt-worker-01
	Node bcmt-worker-01 status is now: NodeReady		
Normal	Starting	13m	kube-proxy, bcmt-
	worker-01 Starting kube-proxy.		

The Escalator transfers node condition and events to alarms, and redirects those alarms to stdout. Other components including fluentd can grasp those alarms for further analysis.

Example

```
{ "type": "alarm", "level": "error", "time": "2020-04-29T05:02:46.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\SuccessfulCreate\", \"message\": \"Created pod: bcmt-autoscaler-7c6fbfb6bb-kgjgz\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:02:46.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Scheduled \\, \"message\": \"Successfully assigned ncms/bcmt-autoscaler-7c6fbfb6bb-kgjgz to amoscontrol-03\\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:02:47.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Pulling \\, \"message\": \"Pulling image \\\\\\"bcmt-registry:5000/bcmt-autoscaler:v1.0.1\\\\\\\" \\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:02:48.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Pulled\\, \"message\": \"Successfully pulled image \\\\\\"bcmt-registry:5000/bcmt-autoscaler:v1.0.1\\\\\\\" \\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:02:48.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Created \\, \"message\": \"Created container bcmt-autoscaler\\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:02:48.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Started \\, \"message\": \"Started container bcmt-autoscaler\\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:04:56.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-autoscaler-7c6fbfb6bb-kgjgz", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\Pulled\\, \"message\": \"Container image \\\\\\"bcmt-registry:5000/bcmt-autoscaler:v1.0.1\\\\\\\" already present on machine\\\" } } } { "type": "alarm", "level": "error", "time": "2020-04-29T05:06:13.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-fpgnj", "timezone": "", "alarm": { "task": "notify", "name": "node_event", "severity": "minor", "data": { "\reason": "\\\" } } }
```

```
{
  "task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Unhealthy\", \"message\": \"Liveness probe failed: HTTP probe failed with statuscode: 500\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:06:13.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-2z7zj", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Unhealthy\", \"message\": \"Liveness probe failed: HTTP probe failed with statuscode: 500\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:06:18.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-68cbd", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Unhealthy\", \"message\": \"Liveness probe failed: HTTP probe failed with statuscode: 500\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:06:33.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-fpgnj", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Killing\", \"message\": \"Container bcmt-api failed liveness probe, will be restarted\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:06:33.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-2z7zj", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Killing\", \"message\": \"Container bcmt-api failed liveness probe, will be restarted\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:06:38.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-68cbd", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Killing\", \"message\": \"Container bcmt-api failed liveness probe, will be restarted\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:07:03.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-fpgnj", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Pulled\", \"message\": \"Container image \\\\\"bcmt-registry:5000/bcmt-admin:19.12.1\\\\\" already present on machine\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:07:04.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-2z7zj", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Pulled\", \"message\": \"Container image \\\\\"bcmt-registry:5000/bcmt-admin:19.12.1\\\\\" already present on machine\""}},
  {"type": "alarm", "level": "error", "time": "2020-04-29T05:07:09.000000Z", "process": "bcmt-escalator", "service": "", "system": "", "neid": "", "container": "256cffd7c415", "host": "bcmt-api-68cbd", "timezone": "", "alarm": ""},
  {"task": "notify", "name": "node_event", "severity": "minor", "data": {"\"reason\": \"Pulled\", \"message\": \"Container image \\\\\"bcmt-registry:5000/bcmt-admin:19.12.1\\\\\" already present on machine\""}}
```

18.7.1.2.1 Enable the Escalator using a configuration file

Use this procedure to enable the Escalator during NCS deployment.

1. Access the `bcmt_config.json` file.
2. Set the `bcmt_config.json` file parameter `escalator_enabled` to `true`.

18.7.2 Monitoring for VM

18.7.2.1 Prometheus

Prometheus is an open-source systems monitoring and alerting toolkit.

Please check **CPro - Prometheus Guide** to install Prometheus in Nokia Container Services (NCS).

NCS provides Prometheus metrics and alert rules for cluster monitoring. Detailed metrics and alert rules, see *Prometheus metrics* on page 467 and *Prometheus alert rules* on page 534.

- When installing Prometheus, use the `values.yml` provided by NCS.
- If Prometheus has been installed, use the `configmap.yml` to update the configmap to monitor NCS.

18.7.2.1.1 Prometheus metrics

18.7.2.1.1 API-server metrics

Table 42: API-server metrics

Metrics
APIServiceOpenAPIAggregationControllerQueue1_adds
APIServiceOpenAPIAggregationControllerQueue1_depth
APIServiceOpenAPIAggregationControllerQueue1_queue_latency
APIServiceOpenAPIAggregationControllerQueue1_queue_latency_count
APIServiceOpenAPIAggregationControllerQueue1_queue_latency_sum
APIServiceOpenAPIAggregationControllerQueue1_retries

Table 42: API-server metrics (continued)

Metrics
APIServiceOpenAPIAggregationControllerQueue1_work_duration
APIServiceOpenAPIAggregationControllerQueue1_work_duration_count
APIServiceOpenAPIAggregationControllerQueue1_work_duration_sum
APIServiceRegistrationController_adds
APIServiceRegistrationController_depth
APIServiceRegistrationController_queue_latency
APIServiceRegistrationController_queue_latency_count
APIServiceRegistrationController_queue_latency_sum
APIServiceRegistrationController_retries
APIServiceRegistrationController_work_duration
APIServiceRegistrationController_work_duration_count
APIServiceRegistrationController_work_duration_sum
AvailableConditionController_adds
AvailableConditionController_depth
AvailableConditionController_queue_latency
AvailableConditionController_queue_latency_count
AvailableConditionController_queue_latency_sum
AvailableConditionController_retries
AvailableConditionController_work_duration

Table 42: API-server metrics (continued)

Metrics
AvailableConditionController_work_duration_count
AvailableConditionController_work_duration_sum
DiscoveryController_adds
DiscoveryController_depth
DiscoveryController_queue_latency
DiscoveryController_queue_latency_count
DiscoveryController_queue_latency_sum
DiscoveryController_retries
DiscoveryController_work_duration
DiscoveryController_work_duration_count
DiscoveryController_work_duration_sum
admission_quota_controller_adds
admission_quota_controller_depth
admission_quota_controller_queue_latency
admission_quota_controller_queue_latency_count
admission_quota_controller_queue_latency_sum
admission_quota_controller_work_duration
admission_quota_controller_work_duration_count
admission_quota_controller_work_duration_sum

Table 42: API-server metrics (continued)

Metrics
apiserver_admission_controller_admission_latencies_seconds_bucket
apiserver_admission_controller_admission_latencies_seconds_count
apiserver_admission_controller_admission_latencies_seconds_sum
apiserver_admission_step_admission_latencies_seconds_bucket
apiserver_admission_step_admission_latencies_seconds_count
apiserver_admission_step_admission_latencies_seconds_sum
apiserver_admission_step_admission_latencies_seconds_summary
apiserver_admission_step_admission_latencies_seconds_summary_count
apiserver_admission_step_admission_latencies_seconds_summary_sum
apiserver_audit_event_total
apiserver_client_certificate_expiration_seconds_bucket
apiserver_client_certificate_expiration_seconds_count
apiserver_client_certificate_expiration_seconds_sum
apiserver_current_inflight_requests
apiserver_longrunning_gauge
apiserver_registered_watchers
apiserver_request_count
apiserver_request_latencies_bucket
apiserver_request_latencies_count

Table 42: API-server metrics (continued)

Metrics
apiserver_request_latencies_sum
apiserver_request_latencies_summary
apiserver_request_latencies_summary_count
apiserver_request_latencies_summary_sum
apiserver_response_sizes_bucket
apiserver_response_sizes_count
apiserver_response_sizes_sum
apiserver_storage_data_key_generation_failures_total
apiserver_storage_data_key_generation_latencies_microseconds_bucket
apiserver_storage_data_key_generation_latencies_microseconds_count
apiserver_storage_data_key_generation_latencies_microseconds_sum
apiserver_storage_envelope_transformation_cache_misses_total
apiserver_storage_transformation_latencies_microseconds_bucket
apiserver_storage_transformation_latencies_microseconds_count
apiserver_storage_transformation_latencies_microseconds_sum
authenticated_user_requests
autoregister_adds
autoregister_depth
autoregister_queue_latency

Table 42: API-server metrics (continued)

Metrics
autoregister_queue_latency_count
autoregister_queue_latency_sum
autoregister_retries
autoregister_work_duration
autoregister_work_duration_count
autoregister_work_duration_sum
reflector_items_per_list
reflector_items_per_list_count
reflector_items_per_list_sum
reflector_items_per_watch
reflector_items_per_watch_count
reflector_items_per_watch_sum
reflector_last_resource_version
reflector_list_duration_seconds
reflector_list_duration_seconds_count
reflector_list_duration_seconds_sum
reflector_lists_total
reflector_short_watches_total
reflector_watch_duration_seconds

Table 42: API-server metrics (continued)

Metrics
reflector_watch_duration_seconds_count
reflector_watch_duration_seconds_sum
reflector_watches_total
rest_client_request_latency_seconds_bucket
rest_client_request_latency_seconds_count
rest_client_request_latency_seconds_sum
rest_client_requests_total
ssh_tunnel_open_count
ssh_tunnel_open_fail_count
storage_operation_duration_seconds_bucket
storage_operation_duration_seconds_count
storage_operation_duration_seconds_sum
storage_operation_errors_total
volume_manager_total_volumes
crdEstablishing_adds
crdEstablishing_depth
crdEstablishing_queue_latency
crdEstablishing_queue_latency_count
crdEstablishing_queue_latency_sum

Table 42: API-server metrics (continued)

Metrics
crdEstablishing_retries
crdEstablishing_work_duration
crdEstablishing_work_duration_count
crdEstablishing_work_duration_sum

18.7.2.1.1.2 ETCD metrics

Table 43: ETCD metrics

Metrics
etcd_debugging_mvcc_db_compaction_pause_duration_milliseconds_bucket
etcd_debugging_mvcc_db_compaction_pause_duration_milliseconds_count
etcd_debugging_mvcc_db_compaction_pause_duration_milliseconds_sum
etcd_debugging_mvcc_db_compaction_total_duration_milliseconds_bucket
etcd_debugging_mvcc_db_compaction_total_duration_milliseconds_count
etcd_debugging_mvcc_db_compaction_total_duration_milliseconds_sum
etcd_debugging_mvcc_db_total_size_in_bytes
etcd_debugging_mvcc_delete_total
etcd_debugging_mvcc_events_total
etcd_debugging_mvcc_index_compaction_pause_duration_milliseconds_bucket
etcd_debugging_mvcc_index_compaction_pause_duration_milliseconds_count

Table 43: ETCD metrics (continued)

Metrics
etcd_debugging_mvcc_index_compaction_pause_duration_milliseconds_sum
etcd_debugging_mvcc_keys_total
etcd_debugging_mvcc_pending_events_total
etcd_debugging_mvcc_put_total
etcd_debugging_mvcc_range_total
etcd_debugging_mvcc_slow_watcher_total
etcd_debugging_mvcc_txn_total
etcd_debugging_mvcc_watch_stream_total
etcd_debugging_mvcc_watcher_total
etcd_debugging_server_lease_expired_total
etcd_debugging_snap_save_marshalling_duration_seconds_bucket
etcd_debugging_snap_save_marshalling_duration_seconds_count
etcd_debugging_snap_save_marshalling_duration_seconds_sum
etcd_debugging_snap_save_total_duration_seconds_bucket
etcd_debugging_snap_save_total_duration_seconds_count
etcd_debugging_snap_save_total_duration_seconds_sum
etcd_debugging_store_expires_total
etcd_debugging_store_reads_total
etcd_debugging_store_watch_requests_total

Table 43: ETCD metrics (continued)

Metrics
etcd_debugging_store_watchers
etcd_debugging_store_writes_total
etcd_disk_backend_commit_duration_seconds_bucket
etcd_disk_backend_commit_duration_seconds_count
etcd_disk_backend_commit_duration_seconds_sum
etcd_disk_backend_defrag_duration_seconds_bucket
etcd_disk_backend_defrag_duration_seconds_count
etcd_disk_backend_defrag_duration_seconds_sum
etcd_disk_backend_snapshot_duration_seconds_bucket
etcd_disk_backend_snapshot_duration_seconds_count
etcd_disk_backend_snapshot_duration_seconds_sum
etcd_disk_wal_fsync_duration_seconds_bucket
etcd_disk_wal_fsync_duration_seconds_count
etcd_disk_wal_fsync_duration_seconds_sum
etcd_grpc_proxy_cache_hits_total
etcd_grpc_proxy_cache_keys_total
etcd_grpc_proxy_cache_misses_total
etcd_grpc_proxy_events_coalescing_total
etcd_grpc_proxy_watchers_coalescing_total

Table 43: ETCD metrics (continued)

Metrics
etcd_helper_cache_entry_count
etcd_helper_cache_hit_count
etcd_helper_cache_miss_count
etcd_http_failed_total
etcd_http_received_total
etcd_http_successful_duration_seconds_bucket
etcd_http_successful_duration_seconds_count
etcd_http_successful_duration_seconds_sum
etcd_mvcc_db_total_size_in_bytes
etcd_mvcc_db_total_size_in_use_in_bytes
etcd_mvcc_hash_duration_seconds_bucket
etcd_mvcc_hash_duration_seconds_count
etcd_mvcc_hash_duration_seconds_sum
etcd_network_client_grpc_received_bytes_total
etcd_network_client_grpc_sent_bytes_total
etcd_network_peer_received_bytes_total
etcd_network_peer_round_trip_time_seconds_bucket
etcd_network_peer_round_trip_time_seconds_count
etcd_network_peer_round_trip_time_seconds_sum

Table 43: ETCD metrics (continued)

Metrics
etcd_network_peer_sent_bytes_total
etcd_network_peer_sent_failures_total
etcd_object_counts
etcd_request_cache_add_latencies_summary
etcd_request_cache_add_latencies_summary_count
etcd_request_cache_add_latencies_summary_sum
etcd_request_cache_get_latencies_summary
etcd_request_cache_get_latencies_summary_count
etcd_request_cache_get_latencies_summary_sum
etcd_server_go_version
etcd_server_has_leader
etcd_server_heartbeat_send_failures_total
etcd_server_is_leader
etcd_server_leader_changes_seen_total
etcd_server_proposals_applied_total
etcd_server_proposals_committed_total
etcd_server_proposals_failed_total
etcd_server_proposals_pending
etcd_server_quota_backend_bytes

Table 43: ETCD metrics (continued)

Metrics
etcd_server_slow_apply_total
etcd_server_slow_read_indexes_total
etcd_server_version
grpc_client_handled_total
grpc_client_msg_received_total
grpc_client_msg_sent_total
grpc_client_started_total
grpc_server_handled_total
grpc_server_msg_received_total
grpc_server_msg_sent_total
grpc_server_started_total
http_request_duration_microseconds
http_request_duration_microseconds_count
http_request_duration_microseconds_sum
http_request_size_bytes
http_request_size_bytes_count
http_request_size_bytes_sum
http_requests_total
http_response_size_bytes

Table 43: ETCD metrics (continued)

Metrics
http_response_size_bytes_count
http_response_size_bytes_sum
get_token_count
get_token_fail_count

18.7.2.1.1.3 Kubernetes metrics

Table 44: Kubernetes metrics

Metrics
kube_daemonset_created
kube_daemonset_metadata_generation
kube_daemonset_status_current_number_scheduled
kube_daemonset_status_desired_number_scheduled
kube_daemonset_status_number_available
kube_daemonset_status_number_misscheduled
kube_daemonset_status_number_ready
kube_daemonset_status_number_unavailable
kube_daemonset_updated_number_scheduled
kube_deployment_created
kube_deployment_labels

Table 44: Kubernetes metrics (continued)

Metrics
kube_deployment_metadata_generation
kube_deployment_spec_paused
kube_deployment_spec_replicas
kube_deployment_spec_strategy_rollingupdate_max_unavailable
kube_deployment_status_observed_generation
kube_deployment_status_replicas
kube_deployment_status_replicas_available
kube_deployment_status_replicas_unavailable
kube_deployment_status_replicas_updated
kube_endpoint_address_available
kube_endpoint_address_not_ready
kube_endpoint_created
kube_endpoint_info
kube_endpoint_labels
kube_job_complete
kube_job_created
kube_job_info
kube_job_labels
kube_job_spec_completions

Table 44: Kubernetes metrics (continued)

Metrics
kube_job_spec_parallelism
kube_job_status_active
kube_job_status_completion_time
kube_job_status_failed
kube_job_status_start_time
kube_job_status_succeeded
kube_namespace_created
kube_namespace_labels
kube_namespace_status_phase
kube_node_created
kube_node_info
kube_node_labels
kube_node_spec_unschedulable
kube_node_status_allocatable_cpu_cores
kube_node_status_allocatable_memory_bytes
kube_node_status_allocatable_pods
kube_node_status_capacity_cpu_cores
kube_node_status_capacity_memory_bytes
kube_node_status_capacity_pods

Table 44: Kubernetes metrics (continued)

Metrics
kube_node_status_condition
kube_persistentvolume_info
kube_persistentvolume_status_phase
kube_persistentvolumeclaim_info
kube_persistentvolumeclaim_resource_requests_storage_bytes
kube_persistentvolumeclaim_status_phase
kube_pod_container_info
kube_pod_container_resource_limits_cpu_cores
kube_pod_container_resource_limits_memory_bytes
kube_pod_container_resource_requests_cpu_cores
kube_pod_container_resource_requests_memory_bytes
kube_pod_container_status_ready
kube_pod_container_status_restarts_total
kube_pod_container_status_running
kube_pod_container_status_terminated
kube_pod_container_status_terminated_reason
kube_pod_container_status_waiting
kube_pod_container_status_waiting_reason
kube_pod_created

Table 44: Kubernetes metrics (continued)

Metrics
kube_pod_info
kube_pod_labels
kube_pod_owner
kube_pod_start_time
kube_pod_status_phase
kube_pod_status_ready
kube_pod_status_scheduled
kube_replicaset_created
kube_replicaset_metadata_generation
kube_replicaset_spec_replicas
kube_replicaset_status_fully_labeled_replicas
kube_replicaset_status_observed_generation
kube_replicaset_status_ready_replicas
kube_replicaset_status_replicas
kube_service_created
kube_service_info
kube_service_labels
kube_service_spec_type
kubelet_cgroup_manager_latency_microseconds

Table 44: Kubernetes metrics (continued)

Metrics
kubelet_cgroup_manager_latency_microseconds_count
kubelet_cgroup_manager_latency_microseconds_sum
kubelet_containers_per_pod_count
kubelet_containers_per_pod_count_count
kubelet_containers_per_pod_count_sum
kubelet_docker_operations
kubelet_docker_operations_errors
kubelet_docker_operations_latency_microseconds
kubelet_docker_operations_latency_microseconds_count
kubelet_docker_operations_latency_microseconds_sum
kubelet_network_plugin_operations_latency_microseconds
kubelet_network_plugin_operations_latency_microseconds_count
kubelet_network_plugin_operations_latency_microseconds_sum
kubelet_node_config_error
kubelet_pleg_relist_interval_microseconds
kubelet_pleg_relist_interval_microseconds_count
kubelet_pleg_relist_interval_microseconds_sum
kubelet_pleg_relist_latency_microseconds
kubelet_pleg_relist_latency_microseconds_count

Table 44: Kubernetes metrics (continued)

Metrics
kubelet_pong_relist_latency_microseconds_sum
kubelet_pod_start_latency_microseconds
kubelet_pod_start_latency_microseconds_count
kubelet_pod_start_latency_microseconds_sum
kubelet_pod_worker_latency_microseconds
kubelet_pod_worker_latency_microseconds_count
kubelet_pod_worker_latency_microseconds_sum
kubelet_pod_worker_start_latency_microseconds
kubelet_pod_worker_start_latency_microseconds_count
kubelet_pod_worker_start_latency_microseconds_sum
kubelet_running_container_count
kubelet_running_pod_count
kubelet_runtime_operations
kubelet_runtime_operations_errors
kubelet_runtime_operations_latency_microseconds
kubelet_runtime_operations_latency_microseconds_count
kubelet_runtime_operations_latency_microseconds_sum
kubelet_volume_stats_available_bytes
kubelet_volume_stats_capacity_bytes

Table 44: Kubernetes metrics (continued)

Metrics
kubelet_volume_stats_inodes
kubelet_volume_stats_inodes_free
kubelet_volume_stats_inodes_used
kubelet_volume_stats_used_bytes
kubernetes_build_info

18.7.2.1.1.4 cAdviser metrics

Table 45: cAdviser metrics

Metrics
cadvisor_version_info
container_cpu_cfs_periods_total
container_cpu_cfs_throttled_periods_total
container_cpu_cfs_throttled_seconds_total
container_cpu_load_average_10s
container_cpu_system_seconds_total
container_cpu_usage_seconds_total
container_cpu_user_seconds_total
container_fs_inodes_free
container_fs_inodes_total

Table 45: cAdviser metrics (continued)

Metrics
container_fs_io_current
container_fs_io_time_seconds_total
container_fs_io_time_weighted_seconds_total
container_fs_limit_bytes
container_fs_read_seconds_total
container_fs_reads_bytes_total
container_fs_reads_merged_total
container_fs_reads_total
container_fs_sector_reads_total
container_fs_sector_writes_total
container_fs_usage_bytes
container_fs_write_seconds_total
container_fs_writes_bytes_total
container_fs_writes_merged_total
container_fs_writes_total
container_last_seen
container_memory_cache
container_memory_failcnt
container_memory_failures_total

Table 45: cAdviser metrics (continued)

Metrics
container_memory_mapped_file
container_memory_max_usage_bytes
container_memory_rss
container_memory_swap
container_memory_usage_bytes
container_memory_working_set_bytes
container_network_receive_bytes_total
container_network_receive_errors_total
container_network_receive_packets_dropped_total
container_network_receive_packets_total
container_network_transmit_bytes_total
container_network_transmit_errors_total
container_network_transmit_packets_dropped_total
container_network_transmit_packets_total
container_scrape_error
container_spec_cpu_period
container_spec_cpu_quota
container_spec_cpu_shares
container_spec_memory_limit_bytes

Table 45: cAdviser metrics (continued)

Metrics
container_spec_memory_reservation_limit_bytes
container_spec_memory_swap_limit_bytes
container_start_time_seconds
container_tasks_state

18.7.2.1.1.5 Node metrics

Table 46: Node metrics

Metrics
node_arp_entries
node_boot_time_seconds
node_context_switches_total
node_cpu_guest_seconds_total
node_cpu_seconds_total
node_disk_io_now
node_disk_io_time_seconds_total
node_disk_io_time_weighted_seconds_total
node_disk_read_bytes_total
node_disk_read_time_seconds_total
node_disk_reads_completed_total

Table 46: Node metrics (continued)

Metrics
node_disk_reads_merged_total
node_disk_write_time_seconds_total
node_disk_writes_completed_total
node_disk_writes_merged_total
node_disk_written_bytes_total
node_entropy_available_bits
node_exporter_build_info
node_filefd_allocated
node_filefd_maximum
node_filesystem_avail_bytes
node_filesystem_device_error
node_filesystem_files
node_filesystem_files_free
node_filesystem_free_bytes
node_filesystem_READONLY
node_filesystem_size_bytes
node_forks_total
node_intr_total
node_ipvs_connections_total

Table 46: Node metrics (continued)

Metrics
node_ipvs_incoming_bytes_total
node_ipvs_incoming_packets_total
node_ipvs_outgoing_bytes_total
node_ipvs_outgoing_packets_total
node_load1
node_load15
node_load5
node_memory_Active_anon_bytes
node_memory_Active_bytes
node_memory_Active_file_bytes
node_memory_AnonHugePages_bytes
node_memory_AnonPages_bytes
node_memory_Bounce_bytes
node_memory_Buffers_bytes
node_memory_Cached_bytes
node_memory_CommitLimit_bytes
node_memory_Committed_AS_bytes
node_memory_DirectMap1G_bytes
node_memory_DirectMap2M_bytes

Table 46: Node metrics (continued)

Metrics
node_memory_DirectMap4k_bytes
node_memory_Dirty_bytes
node_memory_HardwareCorrupted_bytes
node_memory_HugePages_Free
node_memory_HugePages_Rsvd
node_memory_HugePages_Surp
node_memory_HugePages_Total
node_memory_Hugepagesize_bytes
node_memory_Hugetlb_bytes
node_memory_Inactive_anon_bytes
node_memory_Inactive_bytes
node_memory_Inactive_file_bytes
node_memory_KernelStack_bytes
node_memory_Mapped_bytes
node_memory_MemAvailable_bytes
node_memory_MemFree_bytes
node_memory_MemTotal_bytes
node_memory_Mlocked_bytes
node_memory_NFS_Unstable_bytes

Table 46: Node metrics (continued)

Metrics
node_memory_PageTables_bytes
node_memory_SReclaimable_bytes
node_memory_SUnreclaim_bytes
node_memory_ShmemHugePages_bytes
node_memory_ShmemPmdMapped_bytes
node_memory_Shmem_bytes
node_memory_Slab_bytes
node_memory_SwapCached_bytes
node_memory_SwapFree_bytes
node_memory_SwapTotal_bytes
node_memory_Unevictable_bytes
node_memory_VmallocChunk_bytes
node_memory_VmallocTotal_bytes
node_memory_VmallocUsed_bytes
node_memory_WritebackTmp_bytes
node_memory_Writeback_bytes
node_netstat_Icmp6_InErrors
node_netstat_Icmp6_InMsgs
node_netstat_Icmp6_OutMsgs

Table 46: Node metrics (continued)

Metrics
node_netstat_Icmp_InErrors
node_netstat_Icmp_InMsgs
node_netstat_Icmp_OutMsgs
node_netstat_Ip6_InOctets
node_netstat_Ip6_OutOctets
node_netstat_IpExt_InOctets
node_netstat_IpExt_OutOctets
node_netstat_Ip_Forwarding
node_netstat_TcpExt_ListenDrops
node_netstat_TcpExt_ListenOverflows
node_netstat_TcpExt_SyncookiesFailed
node_netstat_TcpExt_SyncookiesRecv
node_netstat_TcpExt_SyncookiesSent
node_netstat_Tcp_ActiveOpens
node_netstat_Tcp_CurrEstab
node_netstat_Tcp_InErrs
node_netstat_Tcp_PassiveOpens
node_netstat_Tcp_RetransSegs
node_netstat_Udp6_InDatagrams

Table 46: Node metrics (continued)

Metrics
node_netstat_Udp6_InErrors
node_netstat_Udp6_NoPorts
node_netstat_Udp6_OutDatagrams
node_netstat_UdpLite6_InErrors
node_netstat_UdpLite_InErrors
node_netstat_Udp_InDatagrams
node_netstat_Udp_InErrors
node_netstat_Udp_NoPorts
node_netstat_Udp_OutDatagrams
node_network_receive_bytes_total
node_network_receive_compressed_total
node_network_receive_drop_total
node_network_receive_errs_total
node_network_receive_fifo_total
node_network_receive_frame_total
node_network_receive_multicast_total
node_network_receive_packets_total
node_network_transmit_bytes_total
node_network_transmit_carrier_total

Table 46: Node metrics (continued)

Metrics
node_network_transmit_colls_total
node_network_transmit_compressed_total
node_network_transmit_drop_total
node_network_transmit_errs_total
node_network_transmit_fifo_total
node_network_transmit_packets_total
node_nf_conntrack_entries
node_nf_conntrack_entries_limit
node_procs_blocked
node_procs_running
node_scrape_collector_duration_seconds
node_scrape_collector_success
node_sockstat_FRAG_inuse
node_sockstat_FRAG_memory
node_sockstat_RAW_inuse
node_sockstat_TCP_alloc
node_sockstat_TCP_inuse
node_sockstat_TCP_mem
node_sockstat_TCP_mem_bytes

Table 46: Node metrics (continued)

Metrics
node_sockstat_TCP_orphan
node_sockstat_TCP_tw
node_sockstat_UDPLITE_inuse
node_sockstat_UDP_inuse
node_sockstat_UDP_mem
node_sockstat_UDP_mem_bytes
node_sockstat_sockets_used
node_textfile_scrape_error
node_time_seconds
node_timex_estimated_error_seconds
node_timex_frequency_adjustment_ratio
node_timex_loop_time_constant
node_timex_maxerror_seconds
node_timex_offset_seconds
node_timex_pps_calibration_total
node_timex_pps_error_total
node_timex_pps_frequency_hertz
node_timex_pps_jitter_seconds
node_timex_pps_jitter_total

Table 46: Node metrics (continued)

Metrics
node_timex_pps_shift_seconds
node_timex_pps_stability_exceeded_total
node_timex_pps_stability_hertz
node_timex_status
node_timex_sync_status
node_timex_tai_offset_seconds
node_timex_tick_seconds
node_uname_info
node_vmstat_oom_kill
node_vmstat_pgfault
node_vmstat_pgmajfault
node_vmstat_pgpgin
node_vmstat_pgpgout
node_vmstat_pswpin
node_vmstat_pswpout
node_xfs_allocation_btree_comparisons_total
node_xfs_allocation_btree_lookups_total
node_xfs_allocation_btree_records_deleted_total
node_xfs_allocation_btree_records_inserted_total

Table 46: Node metrics (continued)

Metrics
node_xfs_block_map_btree_comparisons_total
node_xfs_block_map_btree_lookups_total
node_xfs_block_map_btree_records_deleted_total
node_xfs_block_map_btree_records_inserted_total
node_xfs_block_mapping_extent_list_comparisons_total
node_xfs_block_mapping_extent_list_deletions_total
node_xfs_block_mapping_extent_list_insertions_total
node_xfs_block_mapping_extent_list_lookups_total
node_xfs_block_mapping_reads_total
node_xfs_block_mapping_unmaps_total
node_xfs_block_mapping_writes_total
node_xfs_extent_allocation_blocks_allocated_total
node_xfs_extent_allocation_blocks_freed_total
node_xfs_extent_allocation_extents_allocated_total
node_xfs_extent_allocation_extents_freed_total

18.7.2.1.1.6 Local_volume metrics

Table 47: Local_volume metrics

Metrics
local_volume_provisioner_apiserver_requests_duration_seconds_bucket
local_volume_provisioner_apiserver_requests_duration_seconds_count
local_volume_provisioner_apiserver_requests_duration_seconds_sum
local_volume_provisioner_apiserver_requests_total
local_volume_provisioner_persistentvolume_discovery_duration_seconds_bucket
local_volume_provisioner_persistentvolume_discovery_duration_seconds_count
local_volume_provisioner_persistentvolume_discovery_duration_seconds_sum
local_volume_provisioner_persistentvolume_discovery_total
local_volume_provisioner_proctable_failed
local_volume_provisioner_proctable_running
local_volume_provisioner_proctable_succeeded
local_volume_provisioner_persistentvolume_delete_duration_seconds_bucket
local_volume_provisioner_persistentvolume_delete_duration_seconds_count
local_volume_provisioner_persistentvolume_delete_duration_seconds_sum
local_volume_provisioner_persistentvolume_delete_total

18.7.2.1.1.7 Ceph metrics

Table 48: Ceph metrics

Metrics
ceph_bluefs_bytes_written_sst
ceph_bluefs_bytes_written_wal
ceph_bluefs_db_total_bytes
ceph_bluefs_db_used_bytes
ceph_bluefs_log_bytes
ceph_bluefs_logged_bytes
ceph_bluefs_num_files
ceph_bluefs_slow_total_bytes
ceph_bluefs_slow_used_bytes
ceph_bluefs_wal_total_bytes
ceph_bluefs_wal_used_bytes
ceph_bluystore_commit_lat_count
ceph_bluystore_commit_lat_sum
ceph_bluystore_kv_flush_lat_count
ceph_bluystore_kv_flush_lat_sum
ceph_bluystore_kv_lat_count
ceph_bluystore_kv_lat_sum
ceph_bluystore_read_lat_count

Table 48: Ceph metrics (continued)

Metrics
ceph_bluystore_read_lat_sum
ceph_bluystore_state_aio_wait_lat_count
ceph_bluystore_state_aio_wait_lat_sum
ceph_bluystore_submit_lat_count
ceph_bluystore_submit_lat_sum
ceph_bluystore_throttle_lat_count
ceph_bluystore_throttle_lat_sum
ceph_cluster_total_bytes
ceph_cluster_total_objects
ceph_cluster_total_used_bytes
ceph_disk_occupation
ceph_health_status
ceph_mon_election_call
ceph_mon_election_lose
ceph_mon_election_win
ceph_mon_metadata
ceph_mon_num_elections
ceph_mon_num_sessions
ceph_mon_quorum_status

Table 48: Ceph metrics (continued)

Metrics
ceph_mon_session_add
ceph_mon_session_rm
ceph_mon_session_trim
ceph_num_objects_degraded
ceph_num_objects_misplaced
ceph_num_objects_unfound
ceph_objecter_0x558cc7ed1d60_op_active
ceph_objecter_0x558cc7ed1d60_op_r
ceph_objecter_0x558cc7ed1d60_op_rmw
ceph_objecter_0x558cc7ed1d60_op_w
ceph_objecter_op_active
ceph_objecter_op_r
ceph_objecter_op_rmw
ceph_objecter_op_w
ceph_osd_apply_latency_ms
ceph_osd_commit_latency_ms
ceph_osd_flag_nobackfill
ceph_osd_flag_nodeep_scrub
ceph_osd_flag_nodown

Table 48: Ceph metrics (continued)

Metrics
ceph_osd_flag_noin
ceph_osd_flag_noout
ceph_osd_flag_norebalance
ceph_osd_flag_norecover
ceph_osd_flag_noscrub
ceph_osd_flag_noup
ceph_osd_in
ceph_osd_metadata
ceph_osd_numpg
ceph_osd_numpg_removing
ceph_osd_op
ceph_osd_op_in_bytes
ceph_osd_op_latency_count
ceph_osd_op_latency_sum
ceph_osd_op_out_bytes
ceph_osd_op_prepare_latency_count
ceph_osd_op_prepare_latency_sum
ceph_osd_op_process_latency_count
ceph_osd_op_process_latency_sum

Table 48: Ceph metrics (continued)

Metrics
ceph_osd_op_r
ceph_osd_op_r_latency_count
ceph_osd_op_r_latency_sum
ceph_osd_op_r_out_bytes
ceph_osd_op_r_prepare_latency_count
ceph_osd_op_r_prepare_latency_sum
ceph_osd_op_r_process_latency_count
ceph_osd_op_r_process_latency_sum
ceph_osd_op_rw
ceph_osd_op_rw_in_bytes
ceph_osd_op_rw_latency_count
ceph_osd_op_rw_latency_sum
ceph_osd_op_rw_out_bytes
ceph_osd_op_rw_prepare_latency_count
ceph_osd_op_rw_prepare_latency_sum
ceph_osd_op_rw_process_latency_count
ceph_osd_op_rw_process_latency_sum
ceph_osd_op_w
ceph_osd_op_w_in_bytes

Table 48: Ceph metrics (continued)

Metrics
ceph_osd_op_w_latency_count
ceph_osd_op_w_latency_sum
ceph_osd_op_w_prepare_latency_count
ceph_osd_op_w_prepare_latency_sum
ceph_osd_op_w_process_latency_count
ceph_osd_op_w_process_latency_sum
ceph_osd_op_wip
ceph_osd_recovery_ops
ceph_osd_stat_bytes
ceph_osd_stat_bytes_used
ceph_osd_up
ceph_osd_weight
ceph_paxos_accept_timeout
ceph_paxos_begin
ceph_paxos_begin_bytes_count
ceph_paxos_begin_bytes_sum
ceph_paxos_begin_keys_count
ceph_paxos_begin_keys_sum
ceph_paxos_begin_latency_count

Table 48: Ceph metrics (continued)

Metrics
ceph_paxos_begin_latency_sum
ceph_paxos_collect
ceph_paxos_collect_bytes_count
ceph_paxos_collect_bytes_sum
ceph_paxos_collect_keys_count
ceph_paxos_collect_keys_sum
ceph_paxos_collect_latency_count
ceph_paxos_collect_latency_sum
ceph_paxos_collect_timeout
ceph_paxos_collect_uncommitted
ceph_paxos_commit
ceph_paxos_commit_bytes_count
ceph_paxos_commit_bytes_sum
ceph_paxos_commit_keys_count
ceph_paxos_commit_keys_sum
ceph_paxos_commit_latency_count
ceph_paxos_commit_latency_sum
ceph_paxos_lease_ack_timeout
ceph_paxos_lease_timeout

Table 48: Ceph metrics (continued)

Metrics
ceph_paxos_new_pn
ceph_paxos_new_pn_latency_count
ceph_paxos_new_pn_latency_sum
ceph_paxos_refresh
ceph_paxos_refresh_latency_count
ceph_paxos_refresh_latency_sum
ceph_paxos_restart
ceph_paxos_share_state
ceph_paxos_share_state_bytes_count
ceph_paxos_share_state_bytes_sum
ceph_paxos_share_state_keys_count
ceph_paxos_share_state_keys_sum
ceph_paxos_start_leader
ceph_paxos_start_peon
ceph_paxos_store_state
ceph_paxos_store_state_bytes_count
ceph_paxos_store_state_bytes_sum
ceph_paxos_store_state_keys_count
ceph_paxos_store_state_keys_sum

Table 48: Ceph metrics (continued)

Metrics
ceph_paxos_store_state_latency_count
ceph_paxos_store_state_latency_sum
ceph_pg_activating
ceph_pg_active
ceph_pg_backfill_toofull
ceph_pg_backfill_unfound
ceph_pg_backfill_wait
ceph_pg_backfilling
ceph_pg_clean
ceph_pg_creating
ceph_pg_deep
ceph_pg_degraded
ceph_pg_down
ceph_pg_forced_backfill
ceph_pg_forced_recovery
ceph_pg_incomplete
ceph_pg_inconsistent
ceph_pg_peered
ceph_pg_peering

Table 48: Ceph metrics (continued)

Metrics
ceph_pg_recovering
ceph_pg_recovery_toofull
ceph_pg_recovery_unfound
ceph_pg_recovery_wait
ceph_pg_remapped
ceph_pg_repair
ceph_pg_scrubbing
ceph_pg_snaptrim
ceph_pg_snaptrim_error
ceph_pg_snaptrim_wait
ceph_pg_stale
ceph_pg_total
ceph_pg_undersized
ceph_pg_unknown
ceph_pool_bytes_used
ceph_pool_dirty
ceph_pool_max_avail
ceph_pool_metadata
ceph_pool_objects

Table 48: Ceph metrics (continued)

Metrics
ceph_pool_quota_bytes
ceph_pool_quota_objects
ceph_pool_raw_bytes_used
ceph_pool_rd
ceph_pool_rd_bytes
ceph_pool_wr
ceph_pool_wr_bytes
ceph_rgw_cache_hit
ceph_rgw_cache_miss
ceph_rgw_failed_req
ceph_rgw_get
ceph_rgw_get_b
ceph_rgw_get_initial_lat_count
ceph_rgw_get_initial_lat_sum
ceph_rgw_keystone_token_cache_hit
ceph_rgw_keystone_token_cache_miss
ceph_rgw_metadata
ceph_rgw_put
ceph_rgw_put_b

Table 48: Ceph metrics (continued)

Metrics
ceph_rgw_put_initial_lat_count
ceph_rgw_put_initial_lat_sum
ceph_rgw_qactive
ceph_rgw_qlen
ceph_rgw_req
ceph_rocksdb_compact
ceph_rocksdb_compact_queue_len
ceph_rocksdb_compact_queue_merge
ceph_rocksdb_compact_range
ceph_rocksdb_get
ceph_rocksdb_get_latency_count
ceph_rocksdb_get_latency_sum
ceph_rocksdb_rocksdb_write_delay_time_count
ceph_rocksdb_rocksdb_write_delay_time_sum
ceph_rocksdb_rocksdb_write_memtable_time_count
ceph_rocksdb_rocksdb_write_memtable_time_sum
ceph_rocksdb_rocksdb_write_pre_and_post_time_count
ceph_rocksdb_rocksdb_write_pre_and_post_time_sum
ceph_rocksdb_rocksdb_write_wal_time_count

Table 48: Ceph metrics (continued)

Metrics
ceph_rocksdb_rocksdb_write_wal_time_sum
ceph_rocksdb_submit_latency_count
ceph_rocksdb_submit_latency_sum
ceph_rocksdb_submit_sync_latency_count
ceph_rocksdb_submit_sync_latency_sum
ceph_rocksdb_submit_transaction
ceph_rocksdb_submit_transaction_sync

18.7.2.1.1.8 CoreDNS metrics

Table 49: CoreDNS metrics

Metrics
coredns_build_info
coredns_cache_hits_total
coredns_cache_misses_total
coredns_cache_size
coredns_dns_request_count_total
coredns_dns_request_duration_seconds_bucket
coredns_dns_request_duration_seconds_count
coredns_dns_request_duration_seconds_sum

Table 49: CoreDNS metrics (continued)

Metrics
coredns_dns_request_size_bytes_bucket
coredns_dns_request_size_bytes_count
coredns_dns_request_size_bytes_sum
coredns_dns_request_type_count_total
coredns_dns_response_rcode_count_total
coredns_dns_response_size_bytes_bucket
coredns_dns_response_size_bytes_count
coredns_dns_response_size_bytes_sum
coredns_health_request_duration_seconds_bucket
coredns_health_request_duration_seconds_count
coredns_health_request_duration_seconds_sum
coredns_panic_count_total

18.7.2.1.1.9 Golang metrics

Table 50: Golang metrics

Metrics
go_gc_duration_seconds
go_gc_duration_seconds_count
go_gc_duration_seconds_sum

Table 50: Golang metrics (continued)

Metrics
go_goroutines
go_info
go_memstats_alloc_bytes
go_memstats_alloc_bytes_total
go_memstats_buck_hash_sys_bytes
go_memstats_frees_total
go_memstats_gc_cpu_fraction
go_memstats_gc_sys_bytes
go_memstats_heap_alloc_bytes
go_memstats_heap_idle_bytes
go_memstats_heap_inuse_bytes
go_memstats_heap_objects
go_memstats_heap_released_bytes
go_memstats_heap_released_bytes_total
go_memstats_heap_sys_bytes
go_memstats_last_gc_time_seconds
go_memstats_lookups_total
go_memstats_mallocs_total
go_memstats_mcache_inuse_bytes

Table 50: Golang metrics (continued)

Metrics
go_memstats_mcache_sys_bytes
go_memstats_msparse_inuse_bytes
go_memstats_msparse_sys_bytes
go_memstats_next_gc_bytes
go_memstats_other_sys_bytes
go_memstats_stack_inuse_bytes
go_memstats_stack_sys_bytes
go_memstats_sys_bytes
go_threads
net_conntrack_dialer_conn_attempted_total
net_conntrack_dialer_conn_closed_total
net_conntrack_dialer_conn_established_total
net_conntrack_dialer_conn_failed_total
net_conntrack_listener_conn_accepted_total
net_conntrack_listener_conn_closed_total
machine_cpu_cores
machine_memory_bytes

18.7.2.1.1.10 Calico metrics

Table 51: Calico metrics

Metrics
felix_active_local_endpoints
felix_active_local_policies
felix_active_local_selectors
felix_active_local_tags
felix_calc_graph_output_events
felix_calc_graph_update_time_seconds
felix_calc_graph_update_time_seconds_count
felix_calc_graph_update_time_seconds_sum
felix_calc_graph_updates_processed
felix_cluster_num_host_endpoints
felix_cluster_num_hosts
felix_cluster_num_policies
felix_cluster_num_profiles
felix_cluster_num_workload_endpoints
felix_exec_time_micros
felix_exec_time_micros_count
felix_exec_time_micros_sum
felix_host

Table 51: Calico metrics (continued)

Metrics
felix_int_dataplane_addr_msg_batch_size
felix_int_dataplane_addr_msg_batch_size_count
felix_int_dataplane_addr_msg_batch_size_sum
felix_int_dataplane_apply_time_seconds
felix_int_dataplane_apply_time_seconds_count
felix_int_dataplane_apply_time_seconds_sum
felix_int_dataplane_failures
felix_int_dataplane_iface_msg_batch_size
felix_int_dataplane_iface_msg_batch_size_count
felix_int_dataplane_iface_msg_batch_size_sum
felix_int_dataplane_messages
felix_int_dataplane_msg_batch_size
felix_int_dataplane_msg_batch_size_count
felix_int_dataplane_msg_batch_size_sum
felix_ipset_calls
felix_ipset_errors
felix_ipset_lines_executed
felix_ipsets_calico
felix_ipsets_total

Table 51: Calico metrics (continued)

Metrics
felix_iptables_chains
felix_iptables_lines_executed
felix_iptables_lock_acquire_secs
felix_iptables_lock_acquire_secs_count
felix_iptables_lock_acquire_secs_sum
felix_iptables_lock_retries
felix_iptables_restore_calls
felix_iptables_restore_errors
felix_iptables_rules
felix_iptables_save_calls
felix_iptables_save_errors
felix_log_errors
felix_logs_dropped
felix_resync_state
felix_resyncs_started
felix_route_table_list_seconds
felix_route_table_list_seconds_count
felix_route_table_list_seconds_sum
felix_route_table_per_iface_sync_seconds

Table 51: Calico metrics (continued)

Metrics
felix_route_table_per_iface_sync_seconds_count
felix_route_table_per_iface_sync_seconds_sum

18.7.2.1.11 Fluentd metrics

Table 52: Fluentd metrics

Metrics
fluentd_output_status_buffer_queue_length
fluentd_output_status_buffer_total_bytes
fluentd_output_status_emit_count
fluentd_output_status_emit_records
fluentd_output_status_num_errors
fluentd_output_status_retry_count
fluentd_output_status_retry_wait
fluentd_output_status_rollback_count
fluentd_output_status_write_count

18.7.2.1.1.12 Prometheus metrics

Table 53: Prometheus metrics

Metrics
prometheus_build_info
prometheus_config_last_reload_success_timestamp_seconds
prometheus_config_last_reload_successful
prometheus_engine_queries
prometheus_engine_queries_concurrent_max
prometheus_engine_query_duration_seconds
prometheus_engine_query_duration_seconds_count
prometheus_engine_query_duration_seconds_sum
prometheus_http_request_duration_seconds_bucket
prometheus_http_request_duration_seconds_count
prometheus_http_request_duration_seconds_sum
prometheus_http_response_size_bytes_bucket
prometheus_http_response_size_bytes_count
prometheus_http_response_size_bytes_sum
prometheus_notifications_alertmanagers_discovered
prometheus_notifications_dropped_total
prometheus_notifications_errors_total
prometheus_notifications_latency_seconds

Table 53: Prometheus metrics (continued)

Metrics
prometheus_notifications_latency_seconds_count
prometheus_notifications_latency_seconds_sum
prometheus_notifications_queue_capacity
prometheus_notifications_queue_length
prometheus_notifications_sent_total
prometheus_rule_evaluation_duration_seconds
prometheus_rule_evaluation_duration_seconds_count
prometheus_rule_evaluation_duration_seconds_sum
prometheus_rule_evaluation_failures_total
prometheus_rule_group_duration_seconds
prometheus_rule_group_duration_seconds_count
prometheus_rule_group_duration_seconds_sum
prometheus_rule_group_interval_seconds
prometheus_rule_group_iterations_missed_total
prometheus_rule_group_iterations_total
prometheus_rule_group_last_duration_seconds
prometheus_sd_azure_refresh_duration_seconds
prometheus_sd_azure_refresh_duration_seconds_count
prometheus_sd_azure_refresh_duration_seconds_sum

Table 53: Prometheus metrics (continued)

Metrics
prometheus_sd_azure_refresh_failures_total
prometheus_sd_consul_rpc_duration_seconds
prometheus_sd_consul_rpc_duration_seconds_count
prometheus_sd_consul_rpc_duration_seconds_sum
prometheus_sd_consul_rpc_failures_total
prometheus_sd_dns_lookup_failures_total
prometheus_sd_dns_lookups_total
prometheus_sd_ec2_refresh_duration_seconds
prometheus_sd_ec2_refresh_duration_seconds_count
prometheus_sd_ec2_refresh_duration_seconds_sum
prometheus_sd_ec2_refresh_failures_total
prometheus_sd_file_mtime_seconds
prometheus_sd_file_read_errors_total
prometheus_sd_file_scan_duration_seconds
prometheus_sd_file_scan_duration_seconds_count
prometheus_sd_file_scan_duration_seconds_sum
prometheus_sd_gce_refresh_duration
prometheus_sd_gce_refresh_duration_count
prometheus_sd_gce_refresh_duration_sum

Table 53: Prometheus metrics (continued)

Metrics
prometheus_sd_gce_refresh_failures_total
prometheus_sd_kubernetes_events_total
prometheus_sd_marathon_refresh_duration_seconds
prometheus_sd_marathon_refresh_duration_seconds_count
prometheus_sd_marathon_refresh_duration_seconds_sum
prometheus_sd_marathon_refresh_failures_total
prometheus_sd_openstack_refresh_duration_seconds
prometheus_sd_openstack_refresh_duration_seconds_count
prometheus_sd_openstack_refresh_duration_seconds_sum
prometheus_sd_openstack_refresh_failures_total
prometheus_sd_triton_refresh_duration_seconds
prometheus_sd_triton_refresh_duration_seconds_count
prometheus_sd_triton_refresh_duration_seconds_sum
prometheus_sd_triton_refresh_failures_total
prometheus_target_interval_length_seconds
prometheus_target_interval_length_seconds_count
prometheus_target_interval_length_seconds_sum
prometheus_target_reload_length_seconds
prometheus_target_reload_length_seconds_count

Table 53: Prometheus metrics (continued)

Metrics
prometheus_target_reload_length_seconds_sum
prometheus_target_scrape_pool_sync_total
prometheus_target_scrapes_exceeded_sample_limit_total
prometheus_target_scrapes_sample_duplicate_timestamp_total
prometheus_target_scrapes_sample_out_of_bounds_total
prometheus_target_scrapes_sample_out_of_order_total
prometheus_target_sync_length_seconds
prometheus_target_sync_length_seconds_count
prometheus_target_sync_length_seconds_sum
prometheus_treecache_watcher_goroutines
prometheus_treecache_zookeeper_failures_total
prometheus_tsdb_blocks_loaded
prometheus_tsdb_compaction_chunk_range_bucket
prometheus_tsdb_compaction_chunk_range_count
prometheus_tsdb_compaction_chunk_range_sum
prometheus_tsdb_compaction_chunk_samples_bucket
prometheus_tsdb_compaction_chunk_samples_count
prometheus_tsdb_compaction_chunk_samples_sum
prometheus_tsdb_compaction_chunk_size_bucket

Table 53: Prometheus metrics (continued)

Metrics
prometheus_tsdb_compaction_chunk_size_count
prometheus_tsdb_compaction_chunk_size_sum
prometheus_tsdb_compaction_duration_seconds_bucket
prometheus_tsdb_compaction_duration_seconds_count
prometheus_tsdb_compaction_duration_seconds_sum
prometheus_tsdb_compactions_failed_total
prometheus_tsdb_compactions_total
prometheus_tsdb_compactions_triggered_total
prometheus_tsdb_head_active_appenders
prometheus_tsdb_head_chunks
prometheus_tsdb_head_chunks_created_total
prometheus_tsdb_head_chunks_removed_total
prometheus_tsdb_head_gc_duration_seconds
prometheus_tsdb_head_gc_duration_seconds_count
prometheus_tsdb_head_gc_duration_seconds_sum
prometheus_tsdb_head_max_time
prometheus_tsdb_head_min_time
prometheus_tsdb_head_samples_appended_total
prometheus_tsdb_head_series

Table 53: Prometheus metrics (continued)

Metrics
prometheus_tsdb_head_series_created_total
prometheus_tsdb_head_series_not_found
prometheus_tsdb_head_series_removed_total
prometheus_tsdb_reloads_failures_total
prometheus_tsdb_reloads_total
prometheus_tsdb_retention_cutoffs_failures_total
prometheus_tsdb_retention_cutoffs_total
prometheus_tsdb_tombstone_cleanup_seconds_bucket
prometheus_tsdb_tombstone_cleanup_seconds_count
prometheus_tsdb_tombstone_cleanup_seconds_sum
prometheus_tsdb_wal_corruptions_total
prometheus_tsdb_wal_fsync_duration_seconds
prometheus_tsdb_wal_fsync_duration_seconds_count
prometheus_tsdb_wal_fsync_duration_seconds_sum
prometheus_tsdb_wal_truncate_duration_seconds
prometheus_tsdb_wal_truncate_duration_seconds_count
prometheus_tsdb_wal_truncate_duration_seconds_sum
promhttp_metric_handler_requests_in_flight
promhttp_metric_handler_requests_total

Table 53: Prometheus metrics (continued)

Metrics
pushgateway_build_info
scrape_duration_seconds
scrape_samples_post_metric_relabeling
scrape_samples_scraped
heapster_exporter_duration_milliseconds
heapster_exporter_duration_milliseconds_count
heapster_exporter_duration_milliseconds_sum
heapster_exporter_last_time_seconds
heapster_kubelet_summary_request_duration_milliseconds
heapster_kubelet_summary_request_duration_milliseconds_count
heapster_kubelet_summary_request_duration_milliseconds_sum
heapster_processor_duration_milliseconds
heapster_processor_duration_milliseconds_count
heapster_processor_duration_milliseconds_sum
heapster_scraper_duration_milliseconds
heapster_scraper_duration_milliseconds_count
heapster_scraper_duration_milliseconds_sum
heapster_scraper_last_time_seconds

18.7.2.1.1.13 CPRO Alertmanager metrics

Table 54: CPRO Alertmanager metrics

Metrics
alertmanager_alerts
alertmanager_alerts_invalid_total
alertmanager_alerts_received_total
alertmanager_build_info
alertmanager_cluster_failed_peers
alertmanager_cluster_health_score
alertmanager_cluster_members
alertmanager_cluster_messages_pruned_total
alertmanager_cluster_messages_queued
alertmanager_cluster_messages_received_size_total
alertmanager_cluster_messages_received_total
alertmanager_cluster_messages_sent_size_total
alertmanager_cluster_messages_sent_total
alertmanager_cluster_peers_joined_total
alertmanager_cluster_peers_left_total
alertmanager_cluster_peers_update_total
alertmanager_cluster_reconnections_failed_total
alertmanager_cluster_reconnections_total

Table 54: CPRO Alertmanager metrics (continued)

Metrics
alertmanager_config_hash
alertmanager_config_last_reload_success_timestamp_seconds
alertmanager_config_last_reload_successful
alertmanager_http_request_duration_seconds_bucket
alertmanager_http_request_duration_seconds_count
alertmanager_http_request_duration_seconds_sum
alertmanager_http_response_size_bytes_bucket
alertmanager_http_response_size_bytes_count
alertmanager_http_response_size_bytes_sum
alertmanager_nflog_gc_duration_seconds
alertmanager_nflog_gc_duration_seconds_count
alertmanager_nflog_gc_duration_seconds_sum
alertmanager_nflog_gossip_messages_propagated_total
alertmanager_nflog_queries_total
alertmanager_nflog_query_duration_seconds_bucket
alertmanager_nflog_query_duration_seconds_count
alertmanager_nflog_query_duration_seconds_sum
alertmanager_nflog_query_errors_total
alertmanager_nflog_snapshot_duration_seconds

Table 54: CPRO Alertmanager metrics (continued)

Metrics
alertmanager_nflog_snapshot_duration_seconds_count
alertmanager_nflog_snapshot_duration_seconds_sum
alertmanager_nflog_snapshot_size_bytes
alertmanager_notification_latency_seconds_bucket
alertmanager_notification_latency_seconds_count
alertmanager_notification_latency_seconds_sum
alertmanager_notifications_failed_total
alertmanager_notifications_total
alertmanager_oversize_gossip_message_duration_seconds_bucket
alertmanager_oversize_gossip_message_duration_seconds_count
alertmanager_oversize_gossip_message_duration_seconds_sum
alertmanager_oversized_gossip_message_dropped_total
alertmanager_oversized_gossip_message_failure_total
alertmanager_oversized_gossip_message_sent_total
alertmanager_peer_position
alertmanager_silences
alertmanager_silences_gc_duration_seconds
alertmanager_silences_gc_duration_seconds_count
alertmanager_silences_gc_duration_seconds_sum

Table 54: CPRO Alertmanager metrics (continued)

Metrics
alertmanager_silences_gossip_messages_propagated_total
alertmanager_silences_queries_total
alertmanager_silences_query_duration_seconds_bucket
alertmanager_silences_query_duration_seconds_count
alertmanager_silences_query_duration_seconds_sum
alertmanager_silences_query_errors_total
alertmanager_silences_snapshot_duration_seconds
alertmanager_silences_snapshot_duration_seconds_count
alertmanager_silences_snapshot_duration_seconds_sum
alertmanager_silences_snapshot_size_bytes

18.7.2.1.14 General metrics

Table 55: General metrics

Metrics
up
process_cpu_seconds_total
process_max_fds
process_open_fds
process_resident_memory_bytes

Table 55: General metrics (continued)

Metrics
process_start_time_seconds
process_virtual_memory_bytes

18.7.2.1.2 Prometheus alert rules

18.7.2.1.2.1 Kube-apiserver alert rules

Table 56: Kube-apiserver alert rules

Alert	Severity	Summary
K8SApiserverDown	critical	API server unreachable
K8SApiServerLatency	warning	Kubernetes apiserver latency is high

18.7.2.1.2.2 ETCD alert rules

Table 57: ETCD alert rules

Alert	Severity	Summary
InsufficientMembers	critical	Etcd cluster insufficient members
NoLeader	critical	Etcd member has no leader
HighNumberOfLeaderChanges	warning	A high number of leader changes within the etcd cluster are happening
HighNumberOfFailed HTTPRequests	warning	Some of HTTP requests are failing

Table 57: ETCD alert rules (continued)

Alert	Severity	Summary
HighNumberOfFailedHTTPRequests	critical	A high number of HTTP requests are failing
HTTPRequestsSlow	warning	Slow HTTP requests
EtcdMemberCommunicationSlow	warning	Etcd member communication is slow
HighNumberOfFailedProposals	warning	A high number of proposals within the etcd cluster are failing
HighFsyncDurations	warning	High fsync durations
HighCommitDurations	warning	High commit durations
EtcdDown	critical	Etcd is down

18.7.2.1.2.3 Kube-state-metrics alert rules

Table 58: Kube-state-metrics alert rules

Alert	Severity	Summary
DeploymentGenerationMismatch	warning	Deployment is outdated
DeploymentReplicasNotUpdated	warning	Deployment replicas are outdated
DaemonSetRolloutStuck	warning	DaemonSet is missing pods
K8SDaemonSetsNotScheduled	warning	Daemonsets are not scheduled
DaemonSetsMissScheduled	warning	Daemonsets are not scheduled correctly
PodFrequentlyRestarting	warning	Pod is restarting frequently

Table 58: Kube-state-metrics alert rules (continued)

Alert	Severity	Summary
KubeNodeNotReady	warning	Kubernetes nodes are Not Ready for a hour
K8SManyNodesNotReady	critical	Many Kubernetes nodes are Not Ready

18.7.2.1.2.4 Kubelet alert rules

Table 59: Kubelet alert rules

Alert	Severity	Summary
K8SKubeletDown	warning	Some Kubelets cannot be scraped
K8SManyKubeletDown	critical	Many Kubelets cannot be scraped
K8SKubeletTooManyPods	warning	Kubelet is close to pod limit

18.7.2.1.2.5 Node alert rules

Table 60: Node alert rules

Alert	Severity	Summary
NodeExporterDown	warning	node-exporter cannot be scraped
K8SNodeOutOfDisk	critical	Node ran out of disk space
K8SNodeMemoryPressure	warning	Node is under memory pressure
K8SNodeDiskPressure	warning	Node is under disk pressure

Table 60: Node alert rules (continued)

Alert	Severity	Summary
NodeCPUUsage	warning	High CPU usage detected
NodeMemoryUsage	warning	High memory usage detected

18.7.2.1.2.6 Fluentd alert rules

Table 61: Fluentd alert rules

Alert	Severity	Summary
FluentdNotRunning	warning	Fluentd is down
FluentdBufferFlushFailed	warning	Fluentd buffer flush failed

18.7.2.1.2.7 Calico alert rules

Table 62: Calico alert rules

Alert	Severity	Summary
CalicoNotRunning	warning	Calico is down
CalicoDataplaneFailuresHigh	warning	A high number of dataplane failures within Felix are happening
CalicoDataplaneAddressMsgBatchSizeHigh	warning	Felix address message batch size is higher
CalicoDatapanelfaceMsgBatchSizeHigh	warning	Felix interface message batch size is higher
CalicoplsetErrorsHigh	warning	A high number of ipset errors within Felix are happening

Table 62: Calico alert rules (continued)

Alert	Severity	Summary
CalicolptableSaveErrorsHigh	warning	A high number of iptable save errors within Felix are happening
CalicolptableRestoreErrorsHigh	warning	A high number of iptable restore errors within Felix are happening

18.7.2.1.2.8 Service alert rules

Table 63: Service alert rules

Alert	Severity	Summary
KubednsNotRunning	warning	Kube-DNS is down
CoreDNSNotRunning	warning	CoreDNS is down
PushgatewayNotRunning	warning	Pushgateway is down
HeapsterNotRunning	warning	Heapster is down
KubeDashboardNotRunning	warning	Kube-dashboard is down
LocalVolumeError	warning	Local_volume is not attached
RookError	warning	Rook is not attached

18.7.2.1.2.9 Prometheus alert rules

Table 64: Prometheus alert rules

Alert	Severity	Summary
PrometheusReloadFailed	warning	Prometheus configuration reload has failed

18.7.2.1.2.10 Grafana alert rules

Table 65: Grafana alert rules

Alert	Severity	Summary
GrafanaDown	warning	No Grafana instance is running, no dashboards available

18.7.2.1.2.11 CPRO Alertmanager alert rules

Table 66: CPRO Alertmanager alert rules

Alert	Severity	Summary
AlertmanagerReloadFailed	warning	Alertmanager configuration reload has failed
AlertmanagerDown	warning	Alertmanager is down

18.7.2.1.2.12 General alert rules

Table 67: General alert rules

Alert	Severity	Summary
TargetDown	warning	Targets are down
DeadMansSwitch	none	Alerting DeadMansSwitch
TooManyOpenFileDescriptors	critical	Too many open file descriptors
FdExhaustionClose	warning	File descriptors will exhausted
FdExhaustionClose	critical	File descriptors soon exhausted

18.7.2.1.3 Prometheus adapter configuration

Prerequisites

Prometheus must be installed before executing this procedure.

1. Login to one of the Control nodes

2. Create the following directory:

```
mkdir /opt/bcmt/app-2.0
```

3. Use scp to copy the Prometheus adapter chart to the Control nodes, into the /opt/bcmt/app-2.0 folder.

4. Change directories into the new folder.

```
cd /opt/bcmt/app-2.0
```

5. Fetch the prometheus-adapter file using the following command:

```
helm fetch stable/prometheus-adapter --version 1.3.0
```

6. Retrieve the Prometheus namespace and service name using the following command. These will be the values of <prometheus namespace> and <prometheus service name>.

```
kubectl get svc
```

Example output:

```
default starfish-cpro-server ClusterIP 10.254.241.192 <none> 80/TCP 15m
```

7. Retrieve the value of **dns_domain** specified by the bcmt_config.json file. This will be used as the value of <dns domain>.

8. Execute the following command to generate the padapter.yaml file:

```
helm inspect values prometheus-adapter-1.3.0.tgz > padapter.yaml
```

9. Edit the padapter.yaml file.

10. Under **image**: set the repository to reference the bcmt-registry.

```
image:  
repository: bcmt-registry:5000/docker.io/prometheus-adapter-amd64  
tag: v0.5.0
```

11. Modify the url: value to access Prometheus.

```
prometheus:
  url: http://<prometheus service name>.<prometheus namespace>.svc.<dns domain>
  port: 80
```

12. Specify custom rules as desired.

```
rules:
  default: true
  custom: [...]
```

13. Install the Prometheus adapter, specifying a value for <adapter-name>.

```
helm install -f padapter.yaml prometheus-adapter-1.3.0.tgz -n <adapter-name>
```

18.7.2.1.4 Dimensioning Prometheus

18.7.2.1.4.1 RAM for Prometheus

Prometheus Memory usage can be divided into the below 2 parts:

1. Cardinality Memory
2. Ingestion memory

First, it is mandatory to determine the values of input variables from your system:

Input Variables	Result(Example Values)	Comment
Number of time series(max_over_time(prometheus_tsdb_head_series[1d]))or Get the max value of numSeries from the Prometheus meta.json files	32961	Please refer the below notes for calculating the number of time series.
Average Labels per time Series	15	This depends on the system,Don't forget to include target labels and the metric name
Number of Unique Label Pairs across all time series	1786	Calculated using prometheus-2.13.0.linux-amd64.tar.gz. Please refer this link for further details.
Average Bytes per label pair(including =,"" and ,)	50	Max byte size in SPS. Eg: calculate the total number of bytes required by the longest label pair including =," and ,

Scrape interval in sec	60	
bytesPerSample	3	rate(prometheus_tsdb_compaction_chunk_size_bytes_sum[1d])/ rate(prometheus_tsdb_compaction_chunk_samples_sum[1d])
Retention time in sec	1296000	Default 15 days

Cardinality memory

This can be calculated with the below formula :

$$((\text{Number of time series} * (732 + \text{Average Labels per time Series} * 32 + \text{Average Labels per time Series} * \text{Average Bytes per label pair} * 2) + 120 * \text{Number of Unique Label Pairs}) * 2 \backslash 1024 \backslash 1024)$$

$$= 171 \text{ MB}$$

Here, 732B per series, 32B per label pair, 120B per unique label value are just worst-case assumptions. These values are taken from the reference link.

Ingestion Memory

This can be calculated with the below formula:

$$(\text{Number of time series} * (\text{Bytes per Sample}/\text{Scrape Interval}) * 3600 * 3 * 2) / 1024 / 1024$$

$$= 34 \text{ MB}$$

Here, the formula is derived from taking a scrape interval, the number of time series, 50% overhead, typical bytes per sample and the doubling from GC. Given how head compaction works, we need to allow for up to 3 hours worth of data

Total Memory:

$$(\text{Cardinality Memory} + \text{Ingestion Memory}) * 2$$

$$= 410 \text{ MB [RAM is doubled for GUI queries impact]}$$



Note:

The following is the procedure for calculating the number of time series for a target endpoint:

- Find the total number of metrics that are being exposed to Prometheus by a target.
- For each metric, find the number of labels and different values that label takes. Eg. Let the metric name be M1, and this metric has n labels(l1 to ln), and each label has different values(V1 to Vn). Then number of time series for this metric would be sum of (V1 to Vn) + 1(metric name).
- Prometheus adds default labels like job, instance to the metric. So add num values for these labels say(Jn, Im).
 - Im- If the metric M1 is exposed from 5 hosts, then Im would be 5.

- b. J_n - if the metric is getting scrapped in multiple jobs, then that many time series will be added.
- 4. So total time series for this metric is sum (v_1 to v_n) + 1 + J_n + I_m
- 5. compute steps 2, 3, 4 for each of the metrics in that target endpoint. Say their values would be N_1 to N_n
- 6. Prometheus adds extra 5 metrics to the endpoints. Say their time series total be E_m .
 - a. $E_m = 5 * (\text{Number of instances} + \text{the number of jobs})$??
- 7. Now total time series for the endpoint would be : Sum of (N_1 to N_n) + E_m
- 8. Add some buffer, as Prometheus may add new labels. Let it be B . Currently it is 2% of step 7
- 9. Total = Sum of (N_1 to N_n) + E_m + B
 - a. The above formula is for fixed time series calculation. If there is a churn in the time series(e.g. Adding/removing a label or adding new value for a label or a new metric), then those needs to be added to the formula.
 - b. The exact number of time series depends on the possible combinations of all labels in a given metric.
 - c. For cases where it is likely that most combinations are assumed to be possible, its good to multiply $V_1 * V_2 * V_3 * \dots * V_n$. This will be the worst-case or the theoretical maximum.
 - d. Since all combinations may really never occur based on use cases, some discretion has to be applied to decide which all combinations will be multiplied and which can be added.

Related information

<https://www.robustperception.io/how-much-ram-does-prometheus-2-x-need-for-cardinality-and-ingestion>

18.7.2.1.4.1.1 Memory limitations with Prometheus

The more Pods used leads to more RAM resource being consumed.

By default, the configuration is set to 4Gb which is too low and will always end up failing on OOkiller due to not enough resources.

To remedy:

To set the limit on a Prometheus server to a higher RAM which should be enough to handle the number of pods deployed in that specific time.

 **Note:** The metrics are only fully downloaded after 2 hours, (and only then will have the final resource claim).

 **Note:** If we are getting an OOkiller message, this might be due to the RAM allocation to the service. By default, this is set with a low value and this will lead cause the RAM value to always end

up with OOkiller. This is because not many PODs with a metric are required to end up eventually consuming a large amount of service RAM.

18.7.2.1.4.2 Disk space for Prometheus

Firstly, the below values need to be obtained from your system:

Value	#	Description
bytes per sample	3	rate(prometheus_tsdb_compaction_chunk_size_bytes_sum[1d])/rate(prometheus_tsdb_compaction_chunk_samples_sum[1d])
Retention time in sec	1296000	15d by default
Number of samples/sec	550	Number of Time series/ Scrape interval(s)

Estimated wal size in MB= ((Number of samples/sec) * size_of_wal_sample_on_disk * num_seconds_to_hold_samples) / (1024*1024))

As per the blog post: <https://www.robustperception.io/how-much-space-does-the-wal-take-up>

- size_of_wal_sample_on_disk = 13
- num_seconds_to_hold_samples = 6 * 3600 (6hrs)

Disk Size can be calculated using the below formula:

((bytes per sample*Retention time in sec*Number of samples/sec)/(1024*1024)) + Estimated WAL size MB
= 2188 MB

18.7.2.1.4.3 CPU for Prometheus

Use 1 vCPU.

18.7.2.2 BTEL Telemetry

18.7.2.2.1 Connection to NetAct

18.7.2.2.1.1 Communication Matrix

SNMP (v3 and v2c) port is configurable by the user, the default value is 31161. PM data FTP port is also configurable, its default value is 30022. To change either of the two ports, note that the new port should be chosen between 30000 to 32767. If it is changed in a live NCS cluster, verify that the port is not used by other service.

Table 68: Port for communication with NetAct

Source System	Source Port	Destination System	Destination Port	Protocols	Port Usage
NCS Edge Node	ephemeral	LB JBI virtual IP	162	SNMP	SNMP notification from NCS
Socks	ephemeral	NCS Edge Node	443	HTTPS	NCS Web App integration
SBI Common Mediation	ephemeral	NCS Edge Node	31161	SNMP	FM operations from NetAct
SBI Common Mediation	ephemeral	NCS Edge Node	30022	SFTP	Fetching PM data from NCS

 **Note:** For cloud environment, the mentioned ports should be allowed in the firewall or rules of security group.

18.7.2.3 Monitoring for Bare-metal

18.7.2.3.1 Zabbix

Zabbix is an enterprise-class open source monitoring solution for network and application monitoring of millions of metrics.

Please check **CZBX - Zabbix Guide** to install Zabbix in NCS.

For a detailed Zabbix template, see section *Zabbix Monitoring templates*.

18.7.2.3.1.1 Zabbix Operations

Zabbix 5.0 is installed and used for hardware monitoring. The user manual can be found online. For more information on Zabbix, go to:

<https://www.zabbix.com/documentation/5.0/> <https://www.zabbix.com/documentation/5.0/manual/api/reference>

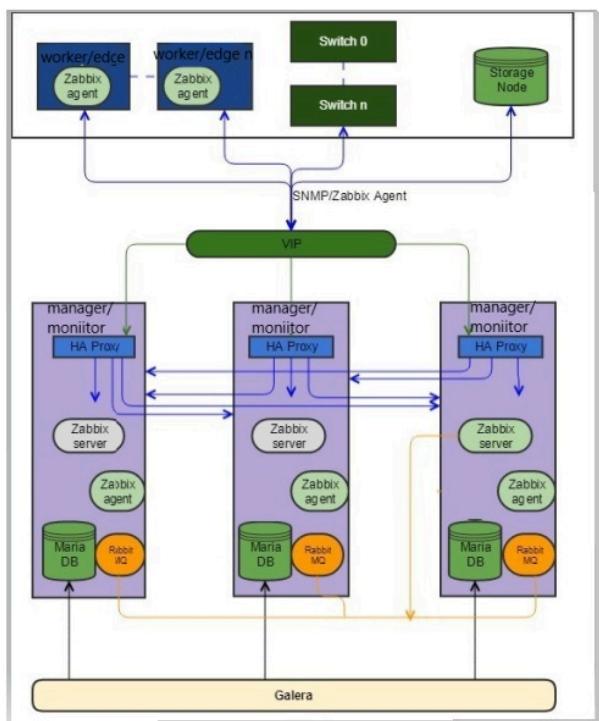
Zabbix is configured during the post-install phase. Different Zabbix templates are configured by default for each, depending on the functionality of the specific physical host. For instructions on viewing, adding/removing and changing what is monitored, see the Zabbix manual.

Modifying existing Zabbix alarms:

- For more information on modifying existing Zabbix alarms, go to: <https://www.zabbix.com/documentation/5.0/manual/config/triggers/trigger> <https://www.zabbix.com/documentation/5.0/manual/config/triggers/expression>

18.7.2.3.1.2 Zabbix HA Monitoring Design

The diagram below shows the HA design and monitoring flow for Zabbix in NCS.



18.7.2.3.1.3 All Zabbix NCS Templates

In NCS, Zabbix is pre-shipped with many templates to monitor the environment. They can be found in the following path:

```
/usr/share/ncs/overcloud/postdeploy/templates/zabbix/templates/
```

Below are listed the supported hardware and services along with their template mappings for host template linkage.

The Zabbix Templates include:

Infrastructure including Node heartbeat checks

- Ingress health check
- Network checks
- Storage checks
- Docker checks
- Repository checks (Helm and Docker registry)
- Other critical process or container checks

Common Zabbix Metrics:

- Percentage of time that the CPU was idle

18.7.2.3.1.3.1 NCS Node Templates

• Manager Templates

- Template NCS General Tests Baremetal
- Smart Disk Monitoring Baremetal
- Template Manage Certificates Validation
- Template NCS Manage Tests

• Master Templates

- Smart Disk Monitoring Baremetal
- Template NCS General Tests Baremetal
- Template NCS Manage Tests
- Template NCS Master Tests
- Template Disk Performance
- Template Manage Certificates Validation
- Template Master Certificates Validation
- Template NCS

• Storage Templates

- Smart Disk Monitoring Baremetal
- Template App OpenStack Ceph Baremetal
- Template NCS General Tests Baremetal
- Template NCS Storage Tests
- Template Disk Performance

- Template HW RAID
- Template SW RAID

- **Worker Templates**

- Smart Disk Monitoring Baremetal
- Template NCS General Tests Baremetal
- Template NCS Worker Tests
- Template Disk Performance
- Template SW RAID

- **Edge Templates**

- Smart Disk Monitoring Baremetal
- Template NCS General Tests Baremetal
- Template Disk Performance

18.7.2.3.1.3.2 Hardware Node Templates

Hardware monitoring is outscoped from NCS and provided by monitoring tools provided by hardware vendors:

- **NADCM** for AirFrame
- **OneView** for HPE
- **OpenManage** for Dell

18.7.2.3.1.3.3 Active Manager/Monitor Monitoring

A new host has been added to Zabbix called the ‘active-Manager/Monitor’. This active manager/Monitor is the node on which Zabbix and virtual IP run.

This host is added to collect data that is common between the three controllers such as Ceph metrics and status of pacemaker resources. Three templates are linked to this host:

- Template App OpenStack Ceph Cluster
- Template Keepalived
- Template App Zabbix Server

18.7.2.3.1.3.4 Importing a Template

About this task

To import a template, do the following:

1. Go to **Configuration > Templates**.

2. Click **Import**.

18.7.2.3.1.3.5 Linking a Template

About this task

To link a template to the host, do the following:

1. Go to **Configuration > Hosts**.

2. Click on the required host and switch to the **Templates** tab.

3. Select one or several templates in the popup window.

4. Click **Add** next to **Link new templates**.

5. Click **Update** in the host attributes form.

18.7.2.3.1.3.6 Switch Monitoring

NCS comes with Zabbix templates for switch monitoring. These templates need to be enabled for each switch, using the following procedure.

Step 1

Configure switch to send SNMP traps to Zabbix



Note:

- There are different configurations for Dell and HP Switches
- The SNMP target IP address refers to the Zabbix public VIP address

Dell Switches (relevant for Dell or AirFrame RM and OR)

Configuring the Dell switches:

```
snmp-server community (default is public) ro.
```

Community can be set as per specific requirement.

```
snmp-server enable traps
snmp-server host {target IP} traps version 2c {public} udp-port {target port}
exit
```

Save the configuration.

Configuring HP switches

```
snmp-agent
snmp-agent trap enable
snmp-agent target-host trap {address} udp-domain {target-address} udp-port 162
  params {securityname} {switchname} v2c
```

Save the configuration.

Step 2

Defining Zabbix host for Switch

1. Login to Zabbix
2. Go to **CONFIGURATION → HOSTS**
3. From the top-right, click **CREATE HOST**
4. Complete:
 - a. The name of the switch.
 - b. A new group (for example: switches) in the "new group" field.
 - c. For **SNMP INTERFACES** click **Add** and insert the IP of the switch (accessible from the controllers).
Leave **CONNECT TO** and **PORT** as the default.
5. For SNMP trap community configuration for the switch in Zabbix:
 - a. Navigate to Configuration.
 - b. Select Hosts.
 - c. Select the switch.
 - d. Navigate to the macro.
 - e. Configure the SNMP trap community as per the example below.
 - f. Click **Update**.
6. Click **Add** on the bottom of the page to complete this action.

Step 3

Associate template to host

1. Go to the host just defined (**CONFIGURATION → Hosts → select relevant host**) and click **TEMPLATES** (top).
2. Select the template **Template Switch Tests** to add, click **Add** and **Update**.

18.7.2.3.1.4 Zabbix Metrics Exporter

18.7.2.3.1.4.1 Introduction to Zabbix Metrics Exporter

The Zabbix Metrics Exporter is a feature for exporting metrics from a Zabbix server into a 3GPP XML file periodically.

In an NCS installation, the Zabbix metrics XML files are located in the primary monitoring servers.

18.7.2.3.1.4.2 Zabbix Metrics Exporter Components

Zabbix Metrics Explorer components include the following:

1. **Python script** - `zbx_metrics_exporter_baremetal.py`, connects to the Zabbix server and collects all the metrics and outputs them into the XML file "`A%Y%m%d.%H%M%z-%H%M%z_zbx_metrics.xml`". It also deletes the corresponding old XML file, for example from exactly two (2) days ago. This is used as a simple XML-rotation mechanism.'
2. **Cron** - Used to schedule the script periodically.
3. **Zabbix Template** – The **Metrics Exporter** template monitors the **Zabbix server** host and is used to monitor the script's action and Cron's activity

18.7.2.3.1.4.3 Zabbix Metrics Exporter Flow

- A Cron job runs the python script every X minutes (currently set to 15 min).
- The script runs and connects to the Zabbix server and exports all metrics into:

`output_dir/A%Y%m%d.%H%M%z-%H%M%z_zbx_metrics.xml` and optionally deletes the corresponding older XML.



Note: Default is 1 day back. (`output_dir/A%Y%m%d.%H%M%z-%H%M%z_zbx_metrics.xml_inprogress` is created during execution).

Example

```
/var/log/zabbix/metrics/A20200913.1315+0000-1330+0000_zbx_metrics.xml
```

- The script outputs any errors into the `last_run.status` file meaning that if it is empty, then the metrics logging has succeeded.
- Zabbix template monitors that:
 - The Cron service is active.
 - The file `last_run.status` exists.
 - `last_run.status` is empty.
 - note



Note: If any error occurred during the above execution, a Zabbix trigger will be triggered and the corresponding Zabbix item's value will show the error text.

18.7.2.3.1.4.4 XML Format

The following table lists the format of every line inside the log:

'measInfold', 'duration', 'endTime', 'measType', 'measObjLdn', 'r', 'error', 'units'

Table 69: XML Field Descriptions

ind	Field	Description	Example
1	measInfold	Measurement type	CPU_CONTROLLER
2	duration	Duration of collected data in seconds	PT900S
3	endTime	Timestamp	2020-09-21T23:15:59+0300
4	measType	Item key	PROCESSOR_LOAD_5avg
5	measObjLdn	Host name	EDGEBM=rca-big-cluster1-edgebm-0
6	r	Value of Counter (Either Item Minimum (over last m minutes), Item Maximum (over last m minutes) or Item Average (over last m minutes))	0.22658
7	error	Item Error	
8	units	Item units	



Note: The Hostname will be the same as defined in Zabbix, unless the item refers to a VFS (virtual file system) resource or an net IF (network interface) in which the Hostname field will be displayed as 'ZabbixHostname [VFS\IF Name]'.

Example

```

- <measInfo measInfoId="CPU_EDGEBM">
  <granPeriod endTime="2020-09-13T13:30:02+0000" duration="PT900S"/>
  <repPeriod duration="PT900S"/>
  <measType p="1">INTERRUPTS_PER_SECOND</measType>
  <measType p="2">PROCESSOR_LOAD_15avg</measType>
  <measType p="3">PROCESSOR_LOAD_1avg</measType>
  <measType p="4">PROCESSOR_LOAD_5avg</measType>
  <measType p="5">CONTEXT_SWITCHES</measType>
  <measType p="6">CPU_IDLE_TIME</measType>
  <measType p="7">CPU_INTERRUPT_TIME</measType>
  <measType p="8">CPU_IOWAIT_TIME</measType>
  <measType p="9">CPU_NICE_TIME</measType>
  <measType p="10">CPU_SOFTIRQ_TIME</measType>
  <measType p="11">CPU_STEAL_TIME</measType>
  <measType p="12">CPU_SYSTEM_TIME</measType>
  <measType p="13">CPU_USER_TIME</measType>
  <measValue measObjId="EDGEbm|rca-big-cluster1-edgebm-0">
    <r p="1">49280.7333333</r>
    <r p="2">0.0109083333333</r>
    <r p="3">0.00770666666667</r>
    <r p="4">0.00888571428571</r>
    <r p="5">5400.0</r>
    <r p="6">99.0058533333</r>
    <r p="7">0.378326666667</r>
    <r p="8">0.0078</r>
    <r p="9">0.00015</r>
    <r p="10">0.175086666667</r>
    <r p="11">0.0</r>
    <r p="12">0.22658</r>
    <r p="13">0.206226666667</r>
  </measValue>
</measInfo>

```

18.7.2.3.1.4.5 Metrics Exporter KPIs

The list of items captured in the metrics exporter, depending on the applied templates/hardware is included in an excel file:

- NCS Zabbix Metrics KPIs.xlsx (see [Zabbix Metric Exporter KPI Template Content](#) on page 553 and [Zabbix Metric Exporter KPI Sheet](#) on page 574).
- Unprotection Password: NCS



Note: Tables and charts should also include Kubernetes metrics.

18.7.2.3.1.4.5.1 Zabbix Metric Exporter KPI Template Content

Template App Zabbix Server	
Trigger Name (23)	Item Name (27)
Zabbix alerter processes more than 75% busy	Zabbix busy alerter processes, in %
Zabbix configuration syncer processes more than 75% busy	Zabbix busy configuration syncer processes, in %
Zabbix db watchdog processes more than 75% busy	Zabbix busy db watchdog processes, in %

Zabbix discoverer processes more than 75% busy	Zabbix busy discoverer processes, in %
Zabbix escalator processes more than 75% busy	Zabbix busy escalator processes, in %
Zabbix history syncer processes more than 75% busy	Zabbix busy history syncer processes, in %
Zabbix housekeeper processes more than 75% busy	Zabbix busy housekeeper processes, in %
Zabbix http poller processes more than 75% busy	Zabbix busy http poller processes, in %
Zabbix icmp pinger processes more than 75% busy	Zabbix busy icmp pinger processes, in %
Zabbix poller processes more than 75% busy	Zabbix busy poller processes, in %
Zabbix proxy poller processes more than 75% busy	Zabbix busy proxy poller processes, in %
Zabbix self-monitoring processes more than 75% busy	Zabbix busy self-monitoring processes, in %
Zabbix snmp trapper processes more than 75% busy	Zabbix busy snmp trapper processes, in %
Zabbix timer processes more than 75% busy	Zabbix busy timer processes, in %
Zabbix trapper processes more than 75% busy	Zabbix busy trapper processes, in %
Zabbix unreachable poller processes more than 75% busy	Zabbix busy unreachable poller processes, in %
More than 100 items having missing data for more than 10 minutes	Zabbix queue over 10m
Less than 25% free in the configuration cache	Zabbix queue
Less than 5% free in the value cache	Zabbix configuration cache, % free

Zabbix value cache working in low memory mode	Zabbix value cache, % free
Less than 25% free in the history cache	Zabbix value cache hits
Less than 25% free in the history index cache	Zabbix value cache misses
Less than 25% free in the trends cache	Zabbix value cache operating mode
	Zabbix history write cache, % free
	Zabbix history index cache, % free
	Zabbix trend write cache, % free
	Values processed by Zabbix server per second

Template App OpenStack Ceph Cluster

Trigger description (6)	Item name (45)
Ceph health is in error state on {HOST.NAME}	Ceph health
Ceph health is in warn state on {HOST.NAME}	Ceph monitor_count
Ceph free capacity is low (< 10%)	Ceph objects_count
Cannot retrieve Ceph metrics from {HOST.NAME} node	Probe ceph osd performance
	Ceph pg_bytes_free
	Ceph pg_bytes_total
	Ceph pg_bytes_used
	Ceph pg_count

	Ceph pg_data_bytes
	Ceph pg_state_count_active
	Ceph pg_state_count_backfill
	Ceph pg_state_count_backfilltoofull
	Ceph pg_state_count_clean
	Ceph pg_state_count_creating
	Ceph pg_state_count_deep
	Ceph pg_state_count_degraded
	Ceph pg_state_count_down
	Ceph pg_state_count_incomplete
	Ceph pg_state_count_inconsistent
	Ceph pg_state_count_peered
	Ceph pg_state_count_peering
	Ceph pg_state_count_recovering
	Ceph pg_state_count_remapped
	Ceph pg_state_count_repair
	Ceph pg_state_count_replay
	Ceph pg_state_count_scrubbing
	Ceph pg_state_count_splitting

	Ceph pg_state_count_stale
	Ceph pg_state_count_undersized
	Ceph pg_state_count_waitbackfill
	Ceph pool_count
	Ceph pool_total_bytes_free
	Ceph pool_total_bytes_total
	Ceph pool_total_bytes_used
	Ceph pool_total_percent_free
	Ceph pool_total_percent_used
	Probe ceph metrics
	Ceph quorum_count
full osd	Ceph full osd
nearfull osd	Ceph nearfull osd
	Ceph read_bytes_sec
	Ceph read_op_per_sec
	Ceph write_bytes_sec
	Ceph write_op_per_sec
	probe ceph racks balance data
Discovery rules (2)	

Ceph Monitor Discovery	
Trigger description (1)	Item name (1)
Ceph monitor {#MONNAME} is down	Ceph monitor {#MONNAME} status
Ceph root nodes discovery	
Trigger description (2)	Item name (2)
Host numbers are not balanced in {#ROOT_NAME} crush	racks hosts balance check per root
Osds numbers are not balanced in {#ROOT_NAME} crush	racks osds balance check per root
Template NCS Master Tests	
Trigger Name (4)	Item Name (5)
DNS nameservers unavailable for {HOST.NAME}	DNS Lookup - nameserver 0
No DNS nameserver redundancy for {HOST.NAME}	DNS Lookup - nameserver 1
Problem with Docker processes on {HOST.NAME}	docker processes
HAProxy not running on {HOST.NAME}	haproxy systemd
	haproxy
Template NCS Manage Tests	
Trigger Name (2)	Item Name (2)
	Rabbitmq status check

Rabbitmq node is not joined to the cluster.	Rabbitmq status
Rabbitmq is in split-brain mode	
Template NCS General Tests Baremetal	
Trigger Name (24)	Item Name (53)
Host name of zabbix_agentd was changed on {HOST.NAME}	Host name of zabbix_agentd running
Zabbix agent on {HOST.NAME} is unreachable	Agent ping
Version of zabbix_agent(d) was changed on {HOST.NAME}	Version of zabbix_agent(d) running
{HOST.NAME} is unavailable by ICMP	ICMP ping
Ping loss is too high on {HOST.NAME}	ICMP loss
Response time is too high on {HOST.NAME}	ICMP response time
Configured max number of opened files is too low on {HOST.NAME}	Maximum number of opened files
Configured max number of processes is too low on {HOST.NAME}	Maximum number of processes
SSH service is down on {HOST.NAME}	SSH Server is listening on port
Too many processes running on {HOST.NAME}	Number of running processes
	zombie process id
SSH Server process is not running on {HOST.NAME}	SSH Server process is running

Processor load is too high on {HOST.NAME}	Number of processes
Disk I/O is overloaded on {HOST.NAME}	/proc/vmstat compact_stall
Hostname was changed on {HOST.NAME}	Host boot time
	Interrupts per second
Host information was changed on {HOST.NAME}	Processor load (15 min average per core)
{HOST.NAME} has just been restarted	Processor load (1 min average per core)
/etc/passwd has been changed on {HOST.NAME}	Processor load (5 min average per core)
Lack of available memory on server {HOST.NAME}	Context switches per second
high usage of network connections	CPU idle time
Too many processes on {HOST.NAME}	CPU interrupt time
OS partition (sda disk) does not exist on {HOST.NAME}	CPU iowait time
	CPU nice time
	CPU softirq time
	CPU steal time
	CPU system time
	CPU user time
	Host name
	Host local time

	System information
	System uptime
	Number of logged in users
	Checksum of /etc/passwd
	Available memory
	Total memory
	CPU guest time
	max allowed network connections
	used network connections
	Free swap memory
	Total swap memory
	Cache memory
	Free memory
	Used memory
	sda disk state
{HOST.NAME} is in maintenance mode	maintenance status
	memory metrics
	ceph installation status check
{ITEM.VALUE}	OpenStack Passwords Expiring Soon

{ITEM.VALUE}	passwd status
	shared memory
Discovery rules (9)	
Isolated CPU discovery	
Trigger description (1)	Item name (1)
CPU Idle time for core {#CPU.NUMBER} on {HOST.NAME1} is low	CPU Idle time for core {#CPU.NUMBER}
Physical network interface discovery	
Trigger description (3)	Item name (11)
Interface {#INTNAME} down on {HOST.NAME}	Outgoing network traffic on {#INTNAME}
Incoming packets rate has been exceed on {#INTNAME}	Interface {#INTNAME} Operational Status
Outgoing packets rate has been exceed on {#INTNAME}	Incoming network packets rate on {#INTNAME}
	Outgoing network packets rate on {#INTNAME}
	Outgoing network bytes on {#INTNAME}
	Outgoing network errors on {#INTNAME}
	Incoming network errors on {#INTNAME}
	Incoming network bytes on {#INTNAME}
	Incoming network traffic on {#INTNAME}

	Outgoing network dropped packets on {#INTNAME}
	Incoming network dropped packets on {#INTNAME}
Physical DPDK network interface discovery	
Trigger description (3)	Item name (11)
Incoming packets rate has been exceed on {#INTNAME}	Outgoing network dropped packets on {#INTNAME}
Outgoing packets rate has been exceed on {#INTNAME}	Outgoing network errors on {#INTNAME}
Interface {#INTNAME} down on {HOST.NAME}	Incoming network packets rate on {#INTNAME}
	Outgoing network packets rate on {#INTNAME}
	Incoming network errors on {#INTNAME}
	Incoming network dropped packets on {#INTNAME}
	Incoming network bytes on {#INTNAME}
	Outgoing network traffic on {#INTNAME}
	Outgoing network bytes on {#INTNAME}
	Interface {#INTNAME} Operational Status
	Incoming network traffic on {#INTNAME}
Physical SRIOV network interface discovery	
Trigger description (3)	Item name (11)

Incoming packets rate has been exceed on {#INTNAME}	Outgoing network dropped packets on {#INTNAME}
Outgoing packets rate has been exceed on {#INTNAME}	Outgoing network errors on {#INTNAME}
Interface {#INTNAME} down on {HOST.NAME}	Incoming network packets rate on {#INTNAME}
	Outgoing network packets rate on {#INTNAME}
	Incoming network errors on {#INTNAME}
	Incoming network dropped packets on {#INTNAME}
	Incoming network bytes on {#INTNAME}
	Outgoing network traffic on {#INTNAME}
	Outgoing network bytes on {#INTNAME}
	Interface {#INTNAME} Operational Status
	Incoming network traffic on {#INTNAME}
Block devices discovery	
Trigger description (0)	Item name (2)
	Disk {#DEVNAME} writes avg1
	Disk {#DEVNAME} reads avg1
Mounted filesystem discovery	
Trigger description (2)	Item name (5)
Free inodes is less than 20% on volume {#FSNAME}	Used disk space on {#FSNAME}

Free disk space is less than 20% on volume {#FSNAME}	Total disk space on {#FSNAME}
	Free inodes on {#FSNAME} (percentage)
	Free disk space on {#FSNAME} (percentage)
	Free disk space on {#FSNAME}
MD devices discovery	
Trigger description (1)	Item name (1)
/dev/{#MDEVICE} - status DEGRADED	/dev/{#MDEVICE} device status
Network interface discovery	
Trigger description (0)	Item name (6)
	Incoming network dropped packets on {#IFNAME}
	Incoming network errors on {#IFNAME}
	Incoming network traffic on {#IFNAME}
	Outgoing network dropped packets on {#IFNAME}
	Outgoing network errors on {#IFNAME}
	Outgoing network traffic on {#IFNAME}
Zombie process ids	
Trigger description (1)	Item name (1)
Process with {#PID} on {HOST.NAME} is zombie	Process with {#PID} is zombie

Template SW RAID	
Trigger description (2)	Item name (2)
RAID installation failed	SW RAID config status
RAID disk failure	RAID disk status
Template NCS Storage Tests	
Trigger description (1)	Item name (1)
Storage node {HOST.NAME} has been detected with reset of megaraid_sas driver	Raid controller flapping status
Template App OpenStack Ceph Baremetal	
Trigger description (4)	Item name (8)
Cannot retrieve Ceph Osd metrics from {HOST.NAME} node	Ceph osd_count_down
	Ceph osd_count_in
	Ceph osd_count_out
	Ceph osd_count_up
	Probe ceph osd metrics
Osd disk labels were changed	check osd disk label
Ceph is not installed on host {HOST.NAME}	check_ceph_installation
There are no OSDs for host {HOST.NAME}	Check ceph osds

Discovery rules (2)	
Ceph Disk Discovery	
Trigger description (1)	Item name (1)
Disk {#OSDDISK} mapped to {#OSDINDEX} is unhealthy	Status for disk {#OSDDISK} mapped to {#OSDINDEX}
Ceph Osd Discovery	
Trigger description (4)	item name (3)
Ceph osd {#OSDINDEX} performance is beyond the configured value on {HOST.NAME}	Ceph osd.{#OSDINDEX} delay
Ceph osd {#OSDINDEX} is down	Ceph osd.{#OSDINDEX} state
Ceph osd {#OSDINDEX} is flipping for more than 3 minutes on {HOST.NAME}	Ceph cluster membership for osd.{#OSDINDEX}
Ceph osd {#OSDINDEX} is out of ceph cluster	
Template Disk Performance	
Trigger description (0)	Item name (0)
Discovery rules (1)	
Disk discovery	
Trigger description (0)	Item name (11)
	Disk:{#DEVICENAME}:Write:Bytes/sec

	Disk:{#DEVICENAME}:Write:Ops per second
	Disk:{#DEVICENAME}:Write:ms
	Disk:{#DEVICENAME}:Read:ms
	Disk:{#DEVICENAME}:Read:Merged
	Disk:{#DEVICENAME}:IO:ms
	Disk:{#DEVICENAME}:Read:Ops per second
	Disk:{#DEVICENAME}:Read:Bytes/sec
	Disk:{#DEVICENAME}:Write:Merged
	Disk:{#DEVICENAME}:IO:Weight:ms
	Disk:{#DEVICENAME}:IO:Currently Active

Template HW RAID

Trigger description (1)	Item name (1)
HW RAID is not optimal on {HOST.NAME}	HW Raid Status

Smart Disk Monitoring Baremetal

Trigger description (0)	Item name (2)
	probe HDDs data
	probe SSDs data

Discovery rules (2)	
HDDs discovery	
Trigger description (3)	Item name (2)
{#DEVICE} overall Health is not OK	{#DEVICE} overall Health
{#DEVICE} Temp is greater than 45C	{#DEVICE} Temperature
{#DEVICE} Temp is greater than 50C	
SSDs discovery	
Trigger description (13)	Item name (12)
current Pending Sector on {#DEVICE} is greater than zero	No Error Logged on {#DEVICE}
ECC Error Rate on {#DEVICE} is greater than zero	Runtime Bad Block on {#DEVICE}
erase Fail Count Total on {#DEVICE} is greater than zero	{#DEVICE} current Pending Sector
Errors Logged on {#DEVICE}	{#DEVICE} Device Temp
program Fail Count Total on {#DEVICE} is greater than zero	{#DEVICE} ECC Error Rate
Runtime Bad Block on {#DEVICE}	{#DEVICE} End to End Error
Temperature is higher than 55C on {#DEVICE}	{#DEVICE} erase fail count total
Temperature is higher than 65C on {#DEVICE}	{#DEVICE} overall Health
there is end to end error on {#DEVICE}	{#DEVICE} program fail count total

uncorrectable error count on {#DEVICE} is greater than zero	{#DEVICE} Serial Number
Wear Leveling Count is less than 10% on {#DEVICE}	{#DEVICE} Uncorrectable Error Count
Wear Leveling Count is less than 15% on {#DEVICE}	{#DEVICE} Wear Leveling Count
{#DEVICE} Health is not OK	
<hr/>	
Template Manage Certificates Validation	
Trigger description (4)	Item name (2)
Certificate is not valid or not found on the {HOST.NAME}.	Manage certificates check
more than one Certificate cant be found or not valid on the {HOST.NAME}.	
Certificate will be expired in less than 30 days on the {HOST.NAME}	Manage certificates check
more than one Certificate will expire in less than 30 days on the {HOST.NAME}.	
<hr/>	
Template Master Certificates Validation	
Trigger description (4)	item name (2)
Certificate is not valid or not found on the {HOST.NAME}.	Master certificates check
more than one Certificate cant be found or not valid on the {HOST.NAME}.	

Certificate will be expired in less than 30 days on the {HOST.NAME}	Master certificates check
more than one Certificate will expire in less than 30 days on the {HOST.NAME}.	
Template NCS	
Trigger description (9)	Item name (44)
	bcmt registry is running
	bcmt yum repo is running
	Calico is running
Calico service is not active on {HOST.NAME}	Calico Service Status
	Chrony Frequency
	Chrony Last Offset
	Chrony Leap Status
	Chrony Reference ID
	Chrony Residual Freq
	Chrony RMS Offset
	Chrony Root Delay
	Chrony Root Dispersion
	Chrony Skew

	Chrony Stratum
	Chrony System Time
	Chrony Update Interval
	CoreDNS is running
	CoreDNS Service Status
	CoreDNS Version
	Etcd API Version
	Etcd is running
Etcd service is not active on {HOST.NAME}	Etcd Service Status
	Heartbeat Service Status
	helm is running
	Kubernetes apiserver is running
kubelet service is not active on {HOST.NAME}	kubelet Service Status
	Kubelet Version
Nginx service is not active on {HOST.NAME}	Nginx Service Status
	Number of auditd process
uwsgi is down on {HOST.NAME}	Number of bcmt Nginx Processes
	Number of CoreDNS processes
	Number of etcd processes

	etcd is down on {HOST.NAME}
	Number of kube-apiserver process
	Number of kube controller manager process
kubelet is down on {HOST.NAME}	Number of kubelet process
	Number of kube proxy process
	Number of kube scheduler process
	Number of metrics server process
Redis-server is down on {HOST.NAME}	Number of redis-server Processes
	Number of running processes
	Probe chrony data
	Registry Proxy Service Status
	Registry Service Status
calico is down on {HOST.NAME}	system.run[systemctl status calico grep Active]

Template Keepalived

Trigger description (1)	Item name (1)
alarm manager is not running on {ITEM.LASTVALUE}	alarm manager status
zabbix server is not running on {ITEM.LASTVALUE}	zabbix server status

Template Worker Tests	
Trigger description (0)	Item name 4)
	Dedicated CPUs
	Number of Cores
	Number of CPUs
	Number of Virtual CPUs (vCPU)

18.7.2.3.1.4.5.2 Zabbix Metric Exporter KPI Sheet

Template App OpenStack Ceph Cluster
Item Name (38)
Ceph read_bytes_sec
Ceph read_op_per_sec
Ceph write_bytes_sec
Ceph write_op_per_sec
Ceph monitor_count
Ceph objects_count
Ceph pg_bytes_free
Ceph pg_bytes_total
Ceph pg_bytes_used

Ceph pg_count
Ceph pg_data_bytes
Ceph pg_state_count_active
Ceph pg_state_count_backfill
Ceph pg_state_count_backfilltoofull
Ceph pg_state_count_clean
Ceph pg_state_count_creating
Ceph pg_state_count_degraded
Ceph pg_state_count_down
Ceph pg_state_count_incomplete
Ceph pg_state_count_inconsistent
Ceph pg_state_count_peered
Ceph pg_state_count_peering
Ceph pg_state_count_recovering
Ceph pg_state_count_remapped
Ceph pg_state_count_repair
Ceph pg_state_count_replay
Ceph pg_state_count_scrubbing
Ceph pg_state_count_splitting

Ceph pg_state_count_stale
Ceph pg_state_count_undersized
Ceph pg_state_count_waitbackfill
Ceph pool_count
Ceph pool_total_bytes_free
Ceph pool_total_bytes_total
Ceph pool_total_bytes_used
Ceph pool_total_percent_free
Ceph pool_total_percent_used
Ceph quorum_count
Template NCS General Tests Baremetal
Item Name (33)
ICMP ping
ICMP loss
ICMP response time
Maximum number of opened files
Maximum number of processes
Number of logged in users

Number of running processes
Number of processes
Host boot time
Interrupts per second
Processor load (15 min average per core)
Processor load (1 min average per core)
Processor load (5 min average per core)
Context switches per second
CPU idle time
CPU interrupt time
CPU iowait time
CPU nice time
CPU softirq time
CPU steal time
CPU system time
CPU user time
System uptime
Cache memory
shared memory

Free memory
Total memory
Free swap memory
Total swap memory
Used memory
Available memory
max allowed network connections
used network connections
Discovery Rules (3)
Mounted filesystem discovery
Item Prototype (5)
Free disk space on {#FSNAME}
Free disk space on {#FSNAME} (percentage)
Free inodes on {#FSNAME} (percentage)
Total disk space on {#FSNAME}
Used disk space on {#FSNAME}
Physical network interface discovery
Item Prototype (10)
Incoming network bytes on {#INTNAME}

Incoming network dropped packets on {#INTNAME}
Incoming network errors on {#INTNAME}
Incoming network packets rate on {#INTNAME}
Incoming network traffic on {#INTNAME}
Outgoing network bytes on {#INTNAME}
Outgoing network dropped packets on {#INTNAME}
Outgoing network errors on {#INTNAME}
Outgoing network packets rate on {#INTNAME}
Outgoing network traffic on {#INTNAME}
Physical SRIOV network interface discover
Item Prototype (10)
Incoming network bytes on {#INTNAME}
Incoming network dropped packets on {#INTNAME}
Incoming network errors on {#INTNAME}
Incoming network packets rate on {#INTNAME}
Incoming network traffic on {#INTNAME}
Outgoing network bytes on {#INTNAME}
Outgoing network dropped packets on {#INTNAME}
Outgoing network errors on {#INTNAME}

Outgoing network packets rate on {#INTNAME}
Outgoing network traffic on {#INTNAME}
Template App OpenStack Ceph Baremetal
Item Name (4)
Ceph osd_count_down
Ceph osd_count_in
Ceph osd_count_out
Ceph osd_count_up
Discovery Rules (1)
Ceph Osd Discovery
Item Prototype (2)
Ceph cluster membership for osd.{#OSDINDEX}
Ceph osd.{#OSDINDEX} state
Template Disk Performance
Discovery Rules (1)
Disk discovery
Item Prototype (3)

Disk:\$1:IO:Currently Active
Disk:\$1:IO:ms
Disk:\$1:IO:Weight:ms
Template NCS Master Tests
Item Name (3)
docker processes
haproxy
haproxy systemd

18.7.2.3.1.5 Replacing the Zabbix Username and Password



Note: Admin is used by default and cannot be changed.

1. From the manager/monitor, (logged in as a root user), run the following:

```
# mysql -u root zabbixdb
update users set passwd=md5('newPassword') where alias='old_zabbix_
username';
update users set alias='new zabbix username' where
alias='old_zabbix username';
```

2. Update the `zabbix_username` and `zabbix_password` in the following file:

```
vim
/usr/share/cbis/data/hieradata/
cbis_openstack_deployment.yaml

cbis::openstack_deployment::zabbix_username:
cbis::openstack_deployment::zabbix_password:
```

-  **Note:** Opening the file and replacing the username default value needs to be updated on all the nodes for any future maintenance or recovery operations, so that the new usernames are available.

18.7.2.3.1.6 Zabbix Housekeeping

You need to set the Override items history period to 15 days.

1. Login to the Zabbix UI.
2. Navigate to the HISTORY section of the Housekeeping page: **ADMINISTRATION → GENERAL → HOUSEKEEPING**
3. Mark the OVERRIDE ITEMS HISTORY PERIOD checkbox.
4. In the DATA STORAGE PERIOD field, enter the value: 15.
5. Click **Update**.

18.7.2.3.1.7 Zabbix Proxy

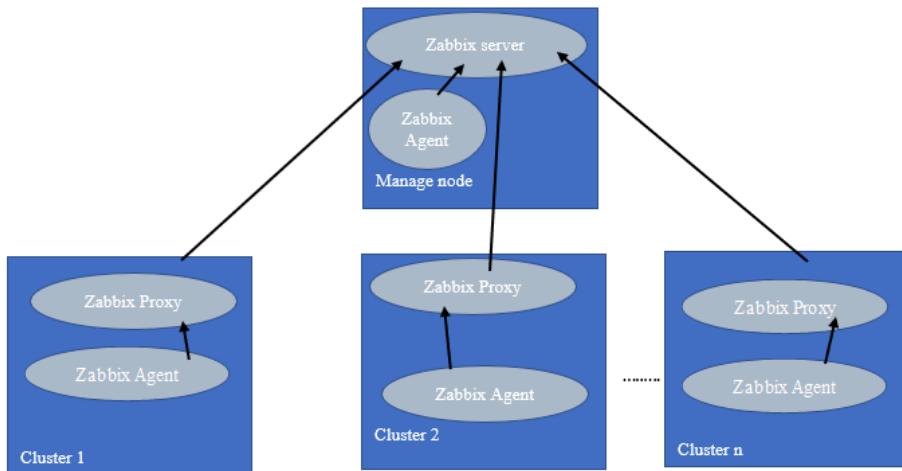
Zabbix proxy 5.0 is installed and used on behalf of Zabbix server on collecting performance and availability data. Zabbix proxy also provides caching mechanism in case of connection lost. For more information on Zabbix proxy, visit: https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies

18.7.2.3.1.7.1 Cache monitoring data and logs on edge site

Zabbix proxy is installed on master nodes, it cashes all data from Zabbix agents from all nodes in the cluster and then send them to Zabbix server. Zabbix proxy also does data preprocessing and simplify maintenance of distributed monitoring.

Certificates are used to secure the connection between Zabbix server and Zabbix proxy. For more information about Zabbix certification, visit: https://www.zabbix.com/documentation/current/manual/encryption/using_certificates

Figure 22: Zabbix Proxy



To see Zabbix proxy on Zabbix UI:

1. Login to Zabbix UI
2. Go to Administration then Proxies

All Proxies will appear with all the information needed such as Zabbix Proxy name, mode, encryption type and hosts monitored by the proxy. See the following figure:

Figure 23: Proxies

Proxies									Create proxy
<input type="text" value="Name"/> <input type="button" value="Mode"/> Any Active Passive									Filter
<input type="checkbox"/>	Name	Mode	Encryption	Compression	Last seen (age)	Host count	Item count	Required performance (vps)	Hosts
<input type="checkbox"/>	rca-test-cluster1	Active	CERT	ON	1s	1	219	2.38	rca-test-cluster1-singlenodecontroller-0

Displaying 1 of 1 four

18.7.2.3.1.8 Database management

This is done to handle the performance of the housekeeper, and the impact of history items. Database Partitions are created every day, and the deletion will be dropping the created partitions.

Python script:

/usr/bin/zabbix_db_partitions_manager.py

Note: The script takes the following arguments.

- **History:** the number of days to keep history tables - the default is set to 3 days, and it is better to not increase it to more than 6 days, otherwise the housekeeper will be running for some items.
- **Trend:** the number of days to keep historical data logging- the default is set to 30 days.
 - **Cron job:** schedule every hour to delete/create partitions if needed.

```
1 * * * * sudo python /usr/bin/zabbix_db_partitions_manager.py --
history=3 -- trend=30
```

Log file: /var/log/zabbix/zabbix_db_partitions_maintenance.log



Note: The file should exist and should be empty.

18.7.2.3.2 Alarms

18.7.2.3.2.1 Alarm Manager

The Alarm Manager is the hub of fault management. Its core manages the alarm life cycle. The life of an alarm starts when the associated fault is first detected and ends when the alarm has been cleared, (when the associated fault has been repaired).

The Alarm Manager communicates with its environment via adapter plug-ins. It has two interfaces, a southbound interface which receives all the traps and a northbound interface for the outgoing traps. The Alarm Manager receives traps containing alarm information from Zabbix. On the other hand, the northbound interface supports both SNMPv2 and SNMPv3 utility operations.



Note: The Alarm Manager is active only on one controller at a time. If one of the active controllers fails, the Alarm Manager resource will be moved to another controller.



Note: Only active alarms are saved in the database.



Note: In case of a Central type deployment with a monitoring cluster, users should perform the following steps:

1. Create a backup of /opt/openstack-ansible/inventory/inventory.sh to a separate folder (**IMPORTANT: do not save the backup under /opt/openstack-ansible/inventory folder!**)

```
cp -rp /opt/openstack-ansible/inventory/inventory.sh /tmp/
inventory.sh.orig
```

2. Open /opt/openstack-ansible/inventory/inventory.sh.

3. Replace the content with:

```
{#!/bin/bash
cat /opt/install/data/cbis-clusters/
```

```
{MONITORING_CLUSTER_NAME} /postconfig-inv.json} } *
```

```
{MONITORING_CLUSTER_NAME}
```

4. Once the alma update has been completed using the Monitoring cluster inventory, `inventory.sh` should be updated again to reflect the updated NCS Baremetal cluster inventory.

18.7.2.3.2.1.1 Monitoring Alarms

Alarms can be monitored using `snmpget` or `snmpwalk` utilities to get the alarms. The Alarm Manager listens for these requests on the 1161 port. The following commands retrieve alarms from the Alarm Manager:

- SNMPv2
- SNMPv3

1. Using SNMPv2

```
snmpwalk -v2c -c <community string> <IP address>:1161 1.3.6.1.4.1.94
```

2. Using SNMPv3

```
snmpwalk -v3 -u <user> -l noAuthNoPriv <IP address>:1161 1.3.6.1.4.1.94
```



Note: 1.3.6.1.4.1.94 is the predefined sysObjectID.

18.7.2.3.2.1.2 Alarm Storage in Logs

The Alarm Manager also stores alarms in logs. Alarms which finish their lifecycle are stored in the following log file: `/var/log/alm/history.log`

In addition, if the Database is full and there is a new alarm created, the new alarm will be stored in the following file: `/var/log/alm/alarms.log`

Finally, all the Alarm Manager Error logs are stored in the file: `/var/log/alm/alma.log`

18.7.2.3.2.1.3 Alarm Notifications

In addition, Alarm Manager also supports auto notification. Alarm Manager can be configured to send SNMPv2 or SNMPv3 Traps to notification targets. The Alarm Manager supports up to 20 notification targets. A notification target can be configured in the following directory in the cluster manager:

`/root/zabbix_configuration/alarm_manager_notification_targets.yaml`



Note: If the NCS Manager was not used to define notification targets, this file will not exist on the cluster manager. In this case, create a new file in the same directory and name it as shown above.

Add a new notification target as follows:

```
#For SNMPv2
- action: initial
  authPassword: ''
  authProtocol: MD5
  community_string: public
  name: MyPCV2C
  passPhrase: ''
  port: 162
  privPassword: ''
  privProtocol: DES
  security_level: noAuthNoPriv
  send_to: <your dest IPadd>
  snmp_version: SNMPv2
  username: MyuserV2C

#For SNMPv3 with authentication and with privacy:
- action: initial
  authPassword: authentication123
  authProtocol: MD5
  community_string: private
  name: MyPCV3sec
  passPhrase: changeyourpassword
  port: 162
  privPassword: privacy123
  privProtocol: DES
  security_level: authPriv
  send_to: <your dest IPadd>
  snmp_version: SNMPv3
  username: MyuserV3_secured

#For SNMPv3 without authentication and without privacy:
- action: initial
  authPassword: ''
  authProtocol: MD5
  community_string: public
  name: MyPCV3notSecured
  passPhrase: ''
  port: 162
  privPassword: ''
  privProtocol: DES
  security_level: noAuthNoPriv
  send_to: <your dest IPadd>
  snmp_version: SNMPv3
  username: MyuserV3_noAuth_noPriv
```

Example

For SNMPv3, there are three possible security levels:

- *noAuthNoPriv*
- *authPriv*
- *authNoPriv*

If *noAuthNoPriv* is selected, fill the authentication and privacy params with empty strings ("").

authProtocol can be either MD5 or SHA. *privProtocol* can be either DES or AES.

The following table helps to explain the previously listed parameters:

Configuration Parameter	SNMP Version	Notes
username	SNMPv3	SNMPv3 username.
name	SNMPv2 and SNMPv3	Alarm manager target notification name.
send_to	SNMPv2 and SNMPv3	IP address of target notification.
community_string	SNMPv2	Authentication String.
port	SNMPv2 and SNMPv3	Notification Target Port.
snmp_version	-	Choose between v3 and v2.
security_level	SNMPv2 and SNMPv3	Select one of the three security levels which are NoAuthNoPriv, AuthNoPriv and AuthPriv.
authProtocol	SNMPv3	Should be either MD5 or SHA.
authPassword	SNMPv3	At least 8 chars in plain text and it must be encrypted.
privProtocol	SNMPv3	Should be either DES or AES.
privPassword	SNMPv3	At least 8 chars in plain text and it must be encrypted.

Configuration Parameter	SNMP Version	Notes
passPhrase	SNMPv3	Used for the encryption of the Privacy and Authentication passwords.

After modifying this file, run the following command from the cluster manager:

```
/usr/local/bin/openstack-ansible --timeout=60 -b -u cbis-admin --private-key /home/cbis-admin/.ssh/id_rsa
/usr/share/cbis/cbis-ansible/post-install/update-alma-notification-targets-bare.yml --extra-vars "cluster_name=CLUSTER-NAME private_key=/home/cbis-admin/.ssh/id_rsa"
```



Note: **CLUSTER-NAME** shall be replaced with the cluster name.



Note: If the monitoring cluster is deployed, make sure that **/opt/openstack-ansible/inventory/inventoy.sh** contains the postconfig inventory file of the monitoring cluster.



Note:

In case of Central type deployment with monitoring cluster, the user should:

- 1- Open `/opt/openstack-ansible/inventory/inventory.sh`
- 2- Replace the content with:

```
#!/bin/bash
cat /opt/install/data/cbis-clusters/
{MONITORING_CLUSTER}/postconfig-inv.json
```

- {MONITORING_CLUSTER} should be replaced with a monitoring cluster name.
- Once alma update is done using Monitoring cluster inventory, `inventory.sh` should be updated again to reflect the NCS Baremetal cluster inventory.

18.7.2.3.2.1.4 Alarm Descriptions for Zabbix

All Zabbix active alarms are sent to the Alarm Manager which stores them and then forwards to the notification targets defined during installation (in the NCS Manager) or by using the method explained in the section above. Alarm Severity mapping from Zabbix to the Alarm Manager is performed when the Alarm notification is sent from Zabbix. The following diagram shows the Alarm flows.

The alarms are identified by the Alarm Manager using both `noiAlarmSpecificProblem` and `noiAlarmSystemDN`. Hence, a combination of the host name, system and alarm id are used to differentiate the alarm from other alarms. Alarm Ids are taken from Zabbix mapping files:

- `/usr/share/cbis/overcloud/postdeploy/templates/zabbix/snmp/zabbix_oid_mapping.yaml`

 **Note:** Zabbix sends the Zabbix Trigger ID as part of the noiAlarmSystemDN to differentiate between alarms that are created according to discovery rules.

 **Note:** For Hardware alarms, the general ID used is '55555'.

 **Note:** For New Zabbix alarms added, the general ID used is '60000'.

18.7.2.3.2.1.5 NCS BareMetal Alarms

Refer to *NCS BareMetal Alarms* and *NCS BareMetal Counters* spreadsheets for NCS BareMetal Alarms.

18.7.2.3.3 ELK

ELK supports the following modes of operation:

Local mode: the Elasticsearch database will be installed locally, and ELK data will be forwarded to that database.

Remote mode: Filebeat will be installed on all nodes, collect the logs and send them to the remote rsyslog server using 514/UDP port. (The remote rsyslog server's IP is the rsyslog servers IPs defined by the user in the NCS-Manager under ELK Optional Configuration section).

18.7.2.3.3.1 Logs in ELK

The following logs are collected from Filebeat:

```
/var/log/audit/audit.log
/var/log/boot.log
/var/log/ceph/ceph.log
/var/log/ceph/ceph*mon*.log
/var/log/ceph/ceph*osd*.log
/var/log/ceph/ceph*mgr*.log
/var/log/cron
/var/log/dmesg
/var/log/maillog
/var/log/mariadb/mariadb.log
/var/log/messages
/var/log/secure
/var/log/spooler
/var/log/yum.log
/var/log/zabbix/*.log
```

The following logs are collected from Logstash and stored in Elasticsearch database:

/var/log/ceph/ceph.log
/var/log/ceph/ceph*mon*.log
/var/log/ceph/ceph*osd*.log
/var/log/ceph/ceph*rgw*.log
/var/log/mariadb/mariadb.log
/var/log/messages
/var/log/secure
/var/log/audit/audit.log

18.7.2.3.3.2 Kibana Usage and Operation

18.7.2.3.3.2.1 Accessing Kibana

After a successful NCS deployment, a link to Kibana will be available in the NCS Manager under the '**External Tools**' tab.

No options on this screen should be changed. Kibana Dashboards are fully configured and accessible from the Dashboards link.

For consecutive visits, Kibana will remember the last visited location and will navigate directly to that page.

18.7.2.3.3.2.2 Kibana Dashboards

All Kibana dashboards can be edited by the operator. We recommend backing up the initial dashboard and visualization configuration. This can be achieved in the **Management -> Saved Objects** screen by clicking '**Export everything**'.

Dashboards can be restored in the same screen using **Import** and uploading the json file that was saved by '**Export everything**'.

Current ELK implementation in NCS offers the following dashboards:

- [Metricbeat HAProxy] Backend
- [Metricbeat HAProxy] Frontend
- [Metricbeat HAProxy] HTTP backend
- [Metricbeat HAProxy] HTTP frontend
- [Metricbeat HAProxy] HTTP server
- [Metricbeat HAProxy] Overview
- [Metricbeat System] Host overview
- [Metricbeat System] Overview

- CephOverview
- Cloud-ErrorsDashboard
- ipmitool for cloud

18.7.2.3.3.2.3 Ceph Overview Dashboard

Ceph dashboards gives an overview of the Ceph cluster status, health and usage.

18.7.2.3.3.2.4 IPMI Tool Dashboard

The IPMI tool gathers important events from the physical host IPMI interfaces and will show any events that occur. Events can be reboots, hardware warnings, etc ...

18.7.2.3.3.2.5 Cloud Errors Dashboard

The Cloud Errors dashboard has data for detected issues in the cloud. In deployments with no issues, this dashboard will not show any data.

18.7.2.3.3.2.6 Metricbeat HAProxy Dashboards

This set of dashboards provides a deep insight into HAProxy traffic. It measures the amount of connections, both TCP and HTTP(S) between the components and the load balancer. It will measure and note downtime in traffic, and can easily show any traffic anomalies on the load balancer.

18.7.2.4 Splunk

Splunk is software used for searching, monitoring, and analyzing machine-generated big data via a Web-style interface.

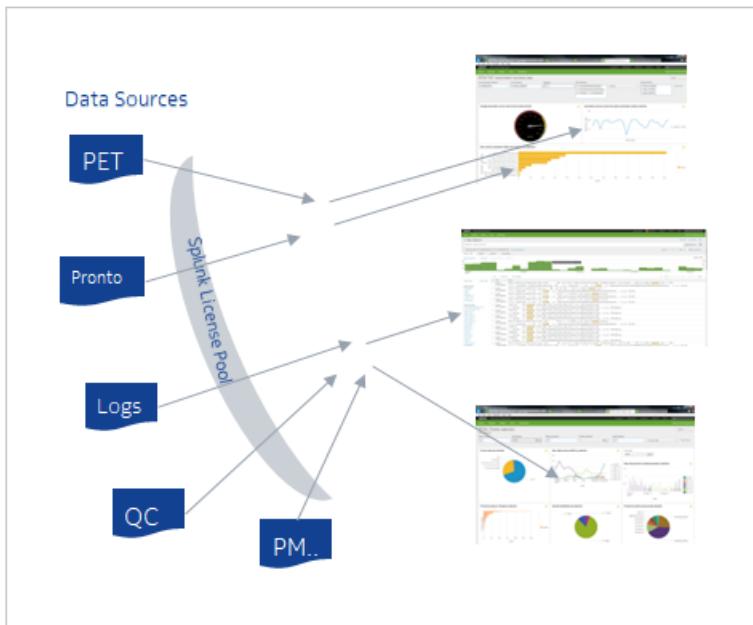
Splunk captures, indexes and correlates real-time data in a searchable repository from which it can generate graphs, reports, alerts, dashboards and visualizations.

Splunk makes machine data accessible across an organization by identifying data patterns, providing metrics, diagnosing problems and providing intelligence for business operations.

Splunk is fully customizable, but also has hundreds of apps that can be used for various use cases in areas like: IT Operations, IT Security, Business Analytics and Data Monitoring. Most of the apps are freely downloadable (like DB connectivity and machine learning packs), but some require a separate license, like the Splunk for Hadoop plugin.

Splunk licensing is based on indexed data volume/day.

NCS supports sending logs to Splunk.



18.7.2.4.1 Using Splunk

NCS versions 20FP1 and up support fluentd integration to Splunk for NCS on VMs.

Starting Splunk

Command Usage

```
ncs service fluentd start [command options] [arguments...]
```

Options

```
--fluentd_es_host <elasticsearch host> specifies the elasticsearch host (optional).
--fluentd_es_port <elasticsearch port> specifies the elasticsearch port (optional).
--fluentd_amqp_host <amqp host> specifies the amqp host (optional).
--fluentd_amqp_port <amqp port> specifies the amqp port (optional).
--fluentd_amqp_user <amqp user> specifies the amqp username (optional).
--fluentd_amqp_pass <amqp password> specifies the amqp password (optional).
--fluentd_kafka_brokers <kafka_brokers> specifies the kafka brokers (optional).
--fluentd_es_user <elasticsearch user> specifies the es username. Only used in SSL mode (optional).
--fluentd_es_pass <elasticsearch password> specifies the es password. Only used in SSL mode (optional).
--fluentd_splunk_host <splunk host> specifies the splunk HEC host (optional).
--fluentd_splunk_port <splunk port> specifies the splunk HEC port (optional).
```

```
--fluentd_splunk_token <splunk token> specifies the splunk HEC token  
(optional).
```

Updating Splunk

This command is used to update fluentd destinations settings.

Command Usage

```
ncs service fluentd update [command options] [arguments...]
```

Options

--fluentd_es_host	<elasticsearch host>	specifies the elasticsearch host (optional).
--fluentd_es_port	<elasticsearch port>	specifies the elasticsearch port (optional).
--fluentd_amqp_host	<amqp host>	specifies the amqp host (optional).
--fluentd_amqp_port	<amqp port>	specifies the amqp port (optional).
--fluentd_amqp_user	<amqp user>	specifies the amqp username (optional).
--fluentd_amqp_pass	<amqp password>	specifies the amqp password (optional).
--fluentd_kafka_brokers	<kafka_brokers>	specifies the kafka brokers (optional).
--fluentd_es_user	<elasticsearch user>	specifies the es username. Only used in SSL mode (optional).
--fluentd_es_pass	<elasticsearch password>	specifies the es password. Only used in SSL mode (optional).
--fluentd_splunk_host	<splunk host>	specifies the splunk HEC host (optional).
--fluentd_splunk_port	<splunk port>	specifies the splunk HEC port (optional).
--fluentd_splunk_token	<splunk token>	specifies the splunk HEC token (optional).

Stopping Splunk

This command is used to stop fluentd.

```
ncs service fluentd stop
```

18.7.2.5 NCS Virtualized Deployment Alarms

Refer to *NCS Virtualized Deployment Alarms* and *NCS Virtualized Deployment Counters* spreadsheets for NCS Virtualized Deployment Alarms.

18.8 Alarm Manager on BareMetal

18.8.1 MTU Size

NetAct expects the MTU size of Ethernet interface not to exceed beyond 1500, which is the standard MTU size of Ethernet interface. To match this, changes may be needed to the external network interface of Edge node(s).

To check the MTU status:

```
# ip addr
```

Or,

```
# netstat -i
```

To apply the setting temporarily:

```
# ifconfig <eth_x> mtu 1500  
<eth_x> is the network interface name used for connection of NetAct
```

To change it permanently:

```
# vim /etc/sysconfig/network-scripts/ifcfg-<eth_x>
```

Add the following content: MTU=1500 Then restart the network:

```
# systemctl restart network
```

19 Cluster Graceful Shutdown and Startup for Bare-metal implementation

In case you need to turn off the live operation of your servers for the sake of occurring maintenance or emergency measures you would execute a graceful shutdown.

Following the execution of the off-duty activities you would execute the startup procedure as described in this chapter.

19.1 Prerequisites for executing Graceful Shutdown

 **Important:** Before you start the procedure, make sure to collect the IPMI addresses for the nodes by executing the following command on each node:

```
sudo ipmitool lan print | grep "IP Address"
```

Prerequisites checks

To make sure the cluster is in a healthy state execute the following steps before starting the shutdown procedure.

1. Check whether Ceph is in `HEALTH OK` state (on any storage or HCI node):

```
sudo ceph -s
```

2. Check mariadb cluster on manager node (expected to be in `synced` state):

```
clustercheck
```

Expected output (indicating the correct synchronized state):

```
[root@cluster-manager-0 ~]# clustercheck
HTTP/1.1 200 OK
Content-Type: text/plain
Connection: close
Content-Length: 40

Percona XtraDB Cluster Node is synced.
```

3. Check NCS cluster health:

```
sudo ncs cluster health-check
```

4. Copy system clock to h/w clock(on all nodes):

```
hwclock --systoh
```

5. Check ntp synchronization(on all nodes):

```
ntpstat | grep synchronised
```

19.2 Cluster Shutdown

19.2.1 NCS Cluster shutdown

Run all the commands in this section from any master nodes

1. Check node names and node status using command "kubectl get nodes":

```
sudo kubectl get nodes
```

Expected output:

NAME	STATUS	ROLES	AGE	VERSION
rca-big-cluster1-edgebm-0	Ready	<none>	10h	v1.18.8
rca-big-cluster1-masterbm-0	Ready	<none>	10h	v1.18.8
rca-big-cluster1-masterbm-1	Ready	<none>	10h	v1.18.8
rca-big-cluster1-masterbm-2	Ready	<none>	10h	v1.18.8
rca-big-cluster1-workerbm-0	Ready	<none>	10h	v1.18.8
rca-big-cluster1-workerbm-1	Ready	<none>	10h	v1.18.8
rca-big-cluster1-workerbm-2	Ready	<none>	10h	v1.18.8
rca-big-cluster1-workerbm-3	Ready	<none>	10h	v1.18.8

2. Use command "kubectl drain" to drain traffic for every node one by one except the storage and manager nodes. See the following example command:

```
sudo kubectl drain rca-big-cluster1-edgebm-0 --ignore-daemonsets --force --delete-local-data
```

Expected output:

```
node "rca-big-cluster1-edgebm-0" cordoned
WARNING: Ignoring DaemonSet-managed pods: kube-proxy-k8ccv
node "rca-big-cluster1-edgebm-0" drained
```



Note: You can check the node status using command `kubectl get nodes`.

The node status should turn to **Ready, SchedulingDisabled**.

Example output:

```
kubectl get nodes
rca-big-cluster1-edgebm-0     Ready,SchedulingDisabled   <none>    10h
                             v1.18.8
rca-big-cluster1-masterbm-0   Ready                      <none>    10h
                             v1.18.8
rca-big-cluster1-masterbm-1   Ready                      <none>    10h
                             v1.18.8
```

rca-big-cluster1-masterbm-2 v1.18.8	Ready	<none>	10h
rca-big-cluster1-workerbm-0 v1.18.8	Ready	<none>	10h
rca-big-cluster1-workerbm-1 v1.18.8	Ready	<none>	10h
rca-big-cluster1-workerbm-2 v1.18.8	Ready	<none>	10h
rca-big-cluster1-workerbm-3 v1.18.8	Ready	<none>	10h



Note: After shutdown there is a known issue in NCS20FP2, which prevents ssh to the node that was shutdown. In case there is no ssh to the node, run the following commands :

```
setenforce 0
systemctl restart network
```

19.2.2 Ceph and services shutdown

To shut down Ceph and services ssh to each node in cluster and execute the following steps:

1. To shut down ceph run the following command on ONLY ONE storage or HCI node:

```
ceph osd set noout
```



Note: The above is only for storage or HCI nodes.

2. Run the following on each storage or HCI node:

```
systemctl stop ceph.target
for i in $(docker ps --filter name=ceph-osd --quiet ) ;do docker stop $i ;
done
timeout -s 9 60 docker stop $(docker ps -q) || true
```

3. Run the following on each Master and storage (or HCI) node:

```
systemctl stop mount_cephfs_share
```

4. To shut down haproxy, execute on control/master nodes the following :

```
systemctl stop haproxy.service && systemctl disable haproxy.service
```

5. Execute the shutdown on all the NCS nodes:

```
shutdown -h now
```

19.2.3 Redhat "NotReady" Status issue

If using Redhat, there may be nodes in "NotReady" status after cluster start up. The following log can be seen in the journal.

```
systemd-logind[1025]: Failed to enable subscription: Failed to activate
service 'org.freedesktop.systemd1': timed out
    systemd-logind[1025]: Failed to fully start up daemon: Connection timed
out
    systemd[1]: systemd-logind.service: main process exited, code=exited,
status=1/FAILURE
    systemd[1]: Failed to start Login Service.
    systemd[1]: Unit systemd-logind.service entered failed state.
    systemd-logind.service failed.
    systemd[1]: systemd-logind.service has no holdoff time, scheduling
restart.
    systemd[1]: Stopped Login Service.
```

This is a known issue of Redhat <https://access.redhat.com/solutions/3900301>, the workaround is to reboot the "NotReady" nodes.

```
# ncs node restart --node_names <notready node_names>
```

19.3 Cluster Startup

For Cluster Startup repeat the following steps for each node in the cluster:

1. Power on node:

```
usr/bin/ipmitool -l lanplus -H -U user -P password power on
```

Wait for 10-15 minutes until the node is running.

2. Check Mariadb cluster is in sync (only for manager nodes):

```
clustercheck
```

3. If step#2 returns a sync error, perform the following:

Run the ansible:

```
/usr/local/bin/openstack-ansible --timeout=60 -b -u cbis-admin /usr/share/
cbis/cbis-ansible/pre-deploy/galera-check.yml --private-key=/home/cbis-
admin/.ssh/id_rsa
```

4. Bring back Ceph services (only for storage or HCI nodes):



Attention: This should only be executed on ONLY ONE storage or hci node.

```
ceph osd unset noout
```

5. Check Ceph Health (only for storage or HCI nodes):

```
[root@rca-big-cluster1-masterbm-0 ~]# ceph health
```

Console response states:

```
HEALTH_OK
```

6. Uncordon NCS Node (skip for Storage nodes and dedicated manager), see an example command as below:

```
kubectl uncordon rca-big-cluster1-edgebm-0
```

7. Check if node is in ready state (skip for Storage nodes):

```
kubectl get nodes
```

8. Wait for 15-20 minutes for all the pods to be running and use NCS health check to sync then run NCS cluster health-check:

```
sudo ncs cluster health-check
```

20 Backup and Restore Operations for Bare-metal Implementation

Nokia Container Services Backup/Restore is used to backup/restore Kubernetes volumes, resources and cluster data. It leverages the design idea of existing CBUR, and provides support to local backup/restore operations.

It will also perform regular backups of the CM database (infrastructure configurations) to restore the manager node in case of failure.

All applications that wish to use Backup/Restore functionality for application data (volumes) need to define custom `BRPolicy` file as described in the guide above

Backup can either run in Backup-now mode, or scheduled backup mode to take regular backups.



Note:

- All backup files are stored under `/root/backup/CLUSTERS/{CLUSTER_NAME}` on the manager node , the customer has to configure nfs mount point on this path to make sure the files are backed up to a remote server or copy them manually.
- User has to specify which application's data to backup from the list in the Backup page.
- For all deployments, the ncs user 'ncm-admin' needs to have the same password before backup and before running the restore operation.

20.1 Storage dimensioning for Backup and Restore

Before installing the NCS cluster, storage dimensioning-related topics must be taken into account.

The following two bullets are for NCS Manager Physical backup and restore (relevant only for BareMetal deployments):

- The size of each backup is varied and depends on the contents.
- The average size for a small cluster is about 4GB. The size to consider is $(\text{rotation} + 1) * (\text{size of backup})$. It is recommended to use an external NFS server to store the backups.

The following bullets are relevant for Kubernetes-related Backup and Restore:

- For both BareMetal and Virtualized Deployments:

In any of the backend endpoint choices available, the underlying application that handles the backup and restore (Containerized Backup and Restore (CBUR)) uses 3 Kubernetes volumes. In BareMetal deployments, by default they are facilitated by CephFS storage class. For Virtualized Deployment, the default storage class for these volumes is GlusterFS. They are used to store at least temporary backups of the NCS Kubernetes cluster and of the applications. Their size is configurable by the user after deployment at the cluster operation *NCS Backup and Restore* and the default assigned for each of the 3 is 50 GB, 150 GB in total.

- /CLUSTER: used for cluster level backup.

Sizing recommendations:

- If the endpoint is set to *local*: $(\text{rotation} + 2) * (\text{the size of local registry} + \text{the size of local Helm repo} + \text{the size of Helm home} + \text{the size of tiller configmaps} / \text{os tuning related crs} / \text{ncm users} / \text{hooks} / \text{Helm releases information})$.
- If the endpoint is not set to *local*: $2 * (\text{the size of 1 cluster backup} = \text{local registry} + \text{the size of local Helm repo} + \text{the size of Helm home} + \text{the size of tiller configmaps} / \text{os tuning related crs} / \text{ncm users} / \text{hooks} / \text{Helm releases information})$.

- /BACKUP: used for application backup and restore when the backend is not set to *local*.

Sizing recommendation: at least double the data size (also consider the situation when backing up multiple applications at the same time).

- /CBUR_REPO: used for application backup and restore when backend is set to *local*. Log files are saved in this volume.

Sizing recommendation: sum of all application storage where backend is set to *local*. Each application requires: $(\text{maxCopy} + 2) * \text{backup data size}$.

- For BareMetal:

- If local endpoint is planned to be used, the directory /opt/management/backup is used to store cluster and app backups on the local FS. In case of non-central (cluster) type deployments this directory is also a shared directory for the Master node (mountpoint for CephFS).
- It is required to configure the directory's allocated size pre-installation, in the installation menu under **Resources → Ceph optional configuration → CephFS Share Point Size**. The default size is 100GB on cluster-type setups.

This affects also the size of the directory /opt/bcmt/storage, so it occupies double the size given on the CephFS cluster.



Note: For MNC clusters, it is not required and this directory is not shared among the master nodes.

When planning to use local backups, the size of this directory must accommodate the size: $(\text{size of cluster backup}) * (\text{rotation} + 1) + \text{sum of } ((\text{each app backup size}) * (\text{apps rotation} + 1))$.

- For Virtualized Deployments:

- Depending on the storage class used in installation - Default is glusterFS - which in default holds the volumes under the directory /data0. Sizing of this directory must be done pre-installation and assessing the amount must be according to as detailed in the following paragraph.



Note: Backup and Restore does not support Veritas Netbackup, AWS, or any other public cloud as the backend for NCS on Virtualized Deployment and NCS on BareMetal Deployment.

20.2 Virtualized Deployments

Backup and restore operations in Virtualized Deployments are done via NCS CLI APIs. The underline application that runs backup and restore operations is called Containerized Backup and Restore (CBUR), configuring it requires editing relevant Helm charts and performing Helm upgrade. For instructions refer to *Containerized Application Backup and Recovery (the Containerized Backup and Restore tool)* on page 646.

You can back up and restore the following:

- Kubernetes cluster infrastructure
- Specific applications
- Everything together

20.2.1 Configuring CBUR before running any backup or restore

Containerized Backup and Restore(CBUR) is running as a pod/container under the 'ncms' namespace. CBUR can be configured both pre and post installation. Containerized Backup and Restore(CBUR) can run backups of the cluster and applications after installation.

Post installation, configuring CBUR is done by editing a values.yaml file and Helm upgrading CBUR. In the following section important fields to consider are shown, and how to configure them:

- To run backups of clusters you must enable k8swatcher.
- Enable cbur-celery pod, in order to run on-demand backups and to increase performance, especially of parallel backups (can happen when having many scheduled backups). This should be done by enabling "direct_redis".
- Enable any desired external backend, meaning a target location to send the backups to. That is since the default is supporting only local backups. Follow the CBUR procedures for instructions on how to set and config each type of backend (for example, with Avamar there are multiple variables and conditions to meet).
 - *Backup or Restore with Backend Mode SFTP* on page 615
 - *Copying the Public Key for the Backup and Restore Tool to the SFTP Server* on page 617
 - *Backup or Restore with Avamar Server* on page 618
- "storageClass" is one field that cannot be changed after the installation. This field refers to the storage class used by CBUR volumes. To change to storageClass with RWX for example (it is required for avamar) for CBUR you will need to re-install, helm delete and helm install CBUR chart.
- Changing the volume sizes assigned for backups. There are 3 volumes named CLUSTER, CBUR_REPO and BACKUP. The first obviously relevant for cluster backups, the other 2 are for apps, when CBUR_REPO is relevant only for local backend and BACKUP is for all the other backends.

The default sizes for each of 3 is 50Gi. You should take into account the current disk space status of your storage class used with CBUR (that is also configurable).

- There are other options for CBUR, given to consideration of the customer, like authentication, user management (connecting to keycloak), using Istio.

The procedure to configure CBUR:

1. Save current configuration then update it:

```
sudo helm -nncms get values -a cbur-master > /home/cloud-user/  
cbur_values.yaml  
sudo vim /home/cloud-user/cbur_values.yaml
```

Example (important variables to check if correct/need change are with comments):



Note: The following repository/registry URLs are provided as examples, you should add your own.

```
COMPUTED VALUES:  
AWSS3: //backend option  
access_key: ""  
access_logging: false  
access_token: ""  
bucket_prefix: ""  
http_proxy: ""  
predefined_bucket: ""  
region: ""  
secret_key: ""  
uid: ""  
CEPHS3: //backend option  
access_key: ""  
bucket_prefix: ""  
ca_crt: ""  
endpoint: ""  
secret_key: ""  
uid: ""  
GLOBAL_BACKEND: ""  
SSH: //SFTP. Below is an example of values.  
host: 10.75.239.162  
hostKey: ""  
mode: sftp  
path: /newupload  
port: 22  
strictHostKeyChecking: autoadd  
username: seenisftp  
accessAllResources: true  
adminRole: true  
affinity: {}  
auditLogStdout:
```

```
enabled: false
auth:
  enabled: false
  useNodePort: false
avamar:                                //backend option - Avamar
  clientName: ""
  dnsPolicy: ClusterFirstWithHostNet
  domain: ARC
  enabled: false
  postScriptTimeoutSeconds: 10800
  preScriptTimeoutSeconds: 10800
resources:
  limits:
    cpu: 1
    memory: 500Mi
  requests:
    cpu: 300m
    memory: 70Mi
server: 10.75.53.234
useHelmPlugin: false
backup_self:                                //Consider using if using local backend
  backend_mode: AWSS3
  cronSpec: 0 0 * * *
  enabled: false
  maxiCopy: 5
  volumes:
    - cbur-repo
cburScope: Cluster
cburaVolume:
  enabled: false
  storage: 500Mi
celery:
  workerConcurrency: "10 --max-tasks-per-child=2"
resources:
  limits:
    cpu: 10
    memory: 8Gi
  requests:
    cpu: 1
    memory: 2Gi
securityContext:
  readOnlyRootFilesystem: false
storageMonitor:
  enables: false
snapshot:
```

```
enabled: true
clusterDomain: ""
clusterId: ""
clusterName: ""
component: master
control_node_ip_list: "" //for avamar
custom:
  job:
    annotations: {}
  pod:
    annotations: {}
direct_redis:
  enabled: true //to enable celery
  redis_image:
    name: crdb/redisio
    pullPolicy: IfNotPresent
    registry: registry
    tag: 3.0-2.2002.el7
  redis_port: 6379
resources:
  limits:
    cpu: 500m
    memory: 500Mi
  requests:
    cpu: 50m
    memory: 50Mi
directPvc:
  backupClusterPvcName: ""
  backupPvcName: ""
  enabled: false
  repoPvcName: ""
enableBrSchemaValidation: true
fullnameOverride: null
global:
  annotations: {}
  containerNamePrefix: ""
  podNamePrefix: ""
  registry: bcmt-registry:5000
  registry1: bcmt-registry:5000
  registry2: csf-docker-inprogress.repo.lab.pl.alcatel-lucent.com
helm2Legacy: false
hostNetwork: //for avamar
  avamar: true
hostPaths:
  dirs:
```

```
bcmt-path: /opt/bcmt/storage/
files:
  ncm-cli: /usr/local/bin/ncm
https:
  certsPath: /opt/bcmt/config/cbur
  enabled: false
  port: 443
  tls:
    ca: ""
    client_cert: ""
    client_key: ""
    enabled: false
    server_cert: ""
    server_key: ""
image:
agent:
  name: cbur/cbura
  pullPolicy: IfNotPresent
  registry: registry
  tag: 1.0.3-3233
avamar:
  name: cbur/cbur-avamar
  pullPolicy: IfNotPresent
  registry: registry
  tag: 1.0.1-3334
croncli:
  name: cbur/cbur-cl
  pullPolicy: IfNotPresent
  registry: registry
  tag: 1.0.3-3233
master:
  name: cbur/cburm
  pullPolicy: IfNotPresent
  registry: registry
  tag: 1.9-06-3335
ingress:
  enabled: false
  path: cbur
integrityCheck:
  local_enable: false
  s3_enable: false
isPvRwx: false          //For avamar this needs to be true and
to have supporting RWX (read write many) storage class
istio:
  cni:
```

```
    enabled: true
  createDrForClient: false
  enabled: false
  gateway:
    annotations: {}
    enabled: false
    ingressPodSelector:
      istio: ingressgateway
    labels: {}
    port: 80
    protocol: HTTP
    tls:
      mode: PASSTHROUGH
  mtls:
    enabled: true
  permissive: true
  sharedHttpGateway:
    name: null
    namespace: null
  version: 1.5
  virtualservice:
    hosts:
      - cbur-master-cbur.cbur
k8swatcher:
  clusterBrEnabled: true
  enabled: true //HAS TO BE ENABLED FOR CLUSTER
  resources:
    limits:
      cpu: 500m
      memory: 500Mi
    requests:
      cpu: 200m
      memory: 80Mi
  keepPvc: false
  logging:
    console_logging_level: INFO
    file_logging_level: DEBUG
    unified_logging: false
  nameOverride: null
  nodeSelector:
    is_control: "true"
  nodeSelectorOverride: {}
  onlyRestore: false
  partOf: cbur
  priorityClassName: system-cluster-critical
```

```
rbac:
  enabled: true
  serviceAccountName: ""
replicaCount: 1
resources:
  limits:
    cpu: 3
    memory: 50Gi
  requests:
    cpu: 300m
    memory: 350Mi
securityContext:
  enabled: true
  fsGroup: auto
  runAsUser: auto
serverPort: 80
storageBackup: 2Gi //BACKUP dir volume size (for apps
on backend != local)
storageBackupCluster: 20Gi //CLUSTER dir volume size (for
cluster backups)
storageClass: "" //will be used to create above cbur-master volumes when
both volumeType.glusterfs and directPvc.enabled are false. When default
"", the default storageclass will be used. By NCS default, glusterfs will
be used
storageMonitor:
  backup_high_threshold: 85%
  backup_low_threshold: 75%
  cluster_backup_high_threshold: 85%
  cluster_backup_low_threshold: 75%
  enabled: true
  repo_high_threshold: 85%
  repo_low_threshold: 75%
storageRepo: 8Gi //CBUR_REPO dir volume size (for
apps on backend == local)
swiftS3: //backend option
  accessKeyId: ""
  bucketPrefix: ""
  caCrt: ""
  endpointUrl: ""
  region: ""
  secretAccessKey: ""
  uid: ""
timezone:
  mountHostLocaltime: true
tolerations:
```

```

- effect: NoExecute
  key: is_control
  operator: Equal
  value: "true"
- effect: NoExecute
  key: is_edge
  operator: Equal
  value: "true"
- effect: NoExecute
  key: is_storage
  operator: Equal
  value: "true"
- effect: NoSchedule
  operator: Exists
volumeType:
  backupClusterEndpoints: glusterfs-cluster
  backupClusterPath: cbur-glusterfs-backup-cluster
  backupEndpoints: glusterfs-cluster
  backupPath: cbur-glusterfs-backup
  glusterfs: true                                //Toggle glusterfs use
  repoEndpoints: glusterfs-cluster
  repoPath: cbur-glusterfs-repo

```

2. Helm upgrade CBUR:

```
sudo helm -n ncms upgrade cbur-master stable/cbur -f /home/cloud-user/cbur_values.yaml --recreate-pods
```

Or if the above fails:

```

sudo bash

helm -n ncms upgrade cbur-master stable/cbur -f /home/cloud-user/cbur_values.yaml --recreate-pods

```

20.2.2 Procedure to Back up the NCS Cluster

1. Log in as an administrator.

```
ncs config set --endpoint 'https://<control_ip>:8082/ncm/api/v1/' && ncs
user login --username 'ncm-admin' --password <password>
```

2. Save somewhere external to the cluster in a safe place, the user and password you used.

3. Back up the current NCS configuration manually, and save it somewhere external in a safe place.

```
ncs config export > ncs_config_$(date +%d-%m-%y_%H:%M:%S).json
```

4. Back up the cluster.

Example:

```
ncs cluster backup --endpoint sftp --schedule none --rotation 5 --override true
```

Explanation of the flags:

--list	<boolean>	Lists the backup archives of an endpoint (optional, if value is true, lists the archives of an endpoint)
--disable	<disable>	Disables the scheduling defined in brpolicy
--schedule	<schedule>	Specifies the schedule for the backup - for example - 30 10 * * * backs up at 10:30 everyday; none for immediate back up, disable to disable backup
--endpoint	<endpoint>	Specifies the endpoint for the backup; value should be: local, sftp, avamar, AWSS3, CEPHS3, SWIFTS3. (optional, default is local)
--rotation	<rotation>	Specifies the rotation of the backup; value [1-10] is valid (optional, default is 1)
--override	<override>	Specifies whether to override or not; only true or false is valid (optional, default is false)
--help, -h		Shows help (default: false)

5. View the progress as follows.

Generally, at any time you can check the Containerized Backup and Restore(CBUR) tool logs by running the following command (if not using celery it would be the cbur-master pod):

```
kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur | less OR
kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur -f
OR# To write into file to review later kubectl -nncms logs -l app=cbur-
master-cbur-mycelery -c cbur > somelogfile.log
```

- If it is an on-demand backup, the output of the previous command should give a task ID. Then you should check the progress of the backup with that task ID, like this:

```
[root@server-name]# ncs cluster backup --endpoint sftp --schedule none
--rotation 5 --override true
{"result":"Very Important: Please manually save cluster config by
command <ncm config export>.
This cluster config will be used to create a new cluster if disaster
recovery.
backupCluster task = 7911fca8-81e7-4e4b-a0a9-f025c961921c is on!"}
[root@server-name]#
[root@server-name]# helm backup -i 7911fca8-81e7-4e4b-a0a9-
f025c961921c
{
  "apiVersion": "v1",
  "code": 200,
  "details": {},
  "kind": "Status",
  "message": {
    "task_info": {
      "7911fca8-81e7-4e4b-a0a9-f025c961921c": {
        "backend": {
          "mode": "sftp"
        },
        "request": "backupCluster",
        "rotation": 5
      }
    }
  },
  "metadata": {},
  "reason": "",
  "status": "InProgress"
}
```

- If it is a scheduled backup, you can verify if the request itself (not the backup) was successful, and then check the logs of CBUR once the backup finished:

```
[root@server-name]# kubectl -nncms get cronjob -o wide
NAME                      SCHEDULE      SUSPEND   ACTIVE   LAST
SCHEDULE     AGE     CONTAINERS
                  SELECTOR
backup-cluster-by-cbur-ncms   0 21 * * *   False      0      10h
                                32h   backup-cluster-by-cbur-ncms
                                cbur/cbur-cli:1.0.3-3233   <none>
```

```
[root@server-name]# kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur | less
```

6. To view existing backups:

Backup should take ~45 min.

```
ncs cluster backup --list true --endpoint <endpoint>
```

#OR - to see the actual file that exists in the Containerized Backup and Restore tool pod:

```
[root@server-name]# kubectl -nncms get po -l app=cbur-master-cbur-mycelery
NAME READY STATUS RESTARTS AGE
cbur-master-cbur-mycelery-556c664f4-b65nv 2/2 Running 0 29d# copy the name
of the pod for the below command:
[root@server-name]# kubectl -nncms exec -ti cbur-master-cbur-
mycelery-556c664f4-b65nv -c cbur bashbash-4.2# ls -l /CLUSTER/local/
total 24348578
-r-----. 1 root root 15113095056 Feb 2 11:05
20220202103010_e03_123456123422_cburm.tgz
```



Note: Rotation impacts the number of backups you see. For example, if latest backup was run with rotation=1, you see only one backup at any given time.

20.2.3 Restore Cluster



Note: Before running a restore operation with local endpoint, it is recommended to check and disable any backups that are supposed to occur in relevant namespaces. Check the currently scheduled backups by running:

```
sudo kubectl get cronjobs -A
```

Check if there is a currently any running backup by checking CBUR logs:

```
sudo kubectl -nncms logs $(sudo kubectl -nncms get po -l app=cbur-
master-cbur-mycelery -o custom-columns=:metadata.name" --no-headers) -
c cbur | less
```

If there are scheduled backups for the planned duration of the restore flow (the time duration depends on the amount and size of apps), for any app that needs it, save the original schedule and disable it:

```
sudo helm backup -n <app namespace> -t <app release name> -a disable
```

Some applications with backup schedules cannot be disabled. In that case, continue normally with the cluster restore operation.

Then run cluster restore.

Afterwards re-enable all of the app backups by running the application backups flow through the NCS Manager (or one by one via CLI with the same command as above with "-a enable" at the end instead of the disable).



Note: You must record in a safe place the passwords for **ncs-admin** and **ncm-admin** before running the `ncs cluster restore` command. The **ncs/m-admin** user must have the same password as before.

1. Use your manually backed up `ncs_config.json` to redeploy NCS cluster as it was.
2. After a successful installation, log in to an admin user **with the same password you did before the relevant backup was done.**

```
ncs config set --endpoint 'https://<control_ip>:8084/ncm/api/v1/' && ncs
user login --username 'ncm-admin' --password <password>
```

3. Verify that the backup you want to restore from exists at the relevant endpoint and that the Containerized Backup and Restore(CBUR) tool knows of it (same as step 5 from backup):

```
ncs cluster backup --list true --endpoint <endpoint>
```

Example:

```
[root@server-name]# ncs cluster backup --list true --endpoint sftp
{
    "sftp": [
        "20220118104005",
        "20220118124137",
        "20220118172213"
    ]
}
```

4. Run cluster restore. If you want to choose a specific backup ID, like this:

```
sudo ncs cluster restore --endpoint="sftp" --all=true --
app_restore_timeout=600 --id 20220118172213
```

--all = means that besides restoring all Kubernetes resources related to operation of the cluster infrastructure, also app data will be restored.

5. Like in on-demand backups, you can check the progress of the restore both in CBUR logs and the status itself.

```
helm restore -i <task id received from restore command>
```

If you do not have access to the output of the restore command you can check the task ID by reviewing the restore - brpolicy:

```
kubectl -nncms get brpolices.cbur.csf.nokia.com -o yaml | less
```

Here you should look for the relevant status by knowing the timestamp (or it should be the last restoreCluster in the list of requests). There should be a *details* key that contains the task ID. Generally, at any time you can check the Containerized Backup and Restore(CBUR) tool logs by running the following command (if not using celery it would be the cbur-master pod):

```
request: restoreCluster
kind: Status
message: restoreCluster task = 785a8c51-3b93-4b65-88c4-596359801e9d is on!
metadata: {}
reason: ""
status: Success
```

```
kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur | less
```

Or:

```
kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur -f
```

Or to write into file to review later:

```
kubectl -nncms logs -l app=cbur-master-cbur-mycelery -c cbur >
somelogfile.log
```

Another option is the following:

```
kubectl -nncms exec cbur-master-cbur-mycelery-7cc45b4d5d-4cmg9 -c cbur
cat /CBUR_REPO/log/BR.log > cburlogs.log
```

 **Note:** cbur-master-cbur-mycelery-7cc45b4d5d-4cmg9 is the full name of the cbur-mycelery pod.

6. For configmaps and secrets related to a certain helm release (application), if they are not mentioned in its BrPolicy, they will not be restored automatically restored by NCS cluster restore flow. However, after the restore flow was done successfully, they can be restored via the following procedure:

- a. Create 'dummy', default valued such resources with the same names as they had when they were backed up.
- b. For each one manually restore the resource.

For example:

```
ncs cluster restore --configmap <cm name> --secret <secret name> --  
namespace <ns> --endpoint <sftp / avamar/ local>
```

20.2.4 App backups

Generally, the backup policy of an app (like scheduling, rotation and backend) is defined within each app's helm charts, in a special resource called BrPolicy.

Basic backup:

```
helm -n <relevant namespace> -t <app helm release> -a <enable=enable  
scheduling defined in brpolicy / none = on demand backup>
```

20.2.5 Additional procedures

The Containerized Backup and Restore tool procedures:

- [BCMT Cluster Backup and Restore](#) on page 651
- [Copying the Public Key for the Backup and Restore Tool to the SFTP Server](#) on page 617
- [Backup or Restore with Avamar Server](#) on page 618
- [#unique_487](#)
- [Restore between clusters](#) on page 705
- [To configure Avamar if using AVAMAR as the backup or restore repository](#) on page 706
- [#unique_490](#)
- [To set ncm endpoint and login ncm](#) on page 706
- [To disable a scheduled cluster backup](#) on page 709
- [To begin the immediate cluster backup](#) on page 709

20.2.6 Backup or Restore with Backend Mode SFTP

Backup/Restore with backend mode "SFTP"

Configure BrPolicy

Change backend mode to "SFTP". It is case insensitive.

Configure the access to SFTP server

You need to configure the following sftp settings via the Containerized Backup and Restore tool Helm chart or Kubernetes secret "cburm-ssh-config". The secret has the higher priority, if there is secret "cburm-ssh-config" in application's namespace, the Containerized Backup and Restore tool will use the settings in it instead of the ones in helm chart to access sftp server.

- mode: only support "sftp" currently.
- host: the IP address or FQDN of the remote server
- port: the port to be used for sftp on the remote server
- path: the target directory on the remote server (i.e. the directory containing the backup data)
- username: the username to connect to the remote server
- strictHostKeyChecking: Whether to do host key checking for sftp server. It only supports "yes" / "no" / "autoadd".

1. If the argument is set to "yes", SSH will use the value of "hostKey" to check the host key of sftp server.
 2. If the argument is set to "no", SSH will skip the host key checking even if "hostKey" is set.
 3. If the argument is set to "autoadd", SSH will automatically add new host key to the user known hosts file when connecting for the first time and ssh will refuse to connect to hosts whose host key has changed. If "hostKey" is also set, the behavior will be like "yes", the Containerized Backup and Restore tool will use the value of "hostKey" to do host key checking and will not add host key to known hosts file.
 4. For "yes" and "autoadd", when the host key has changed, user needs to update the new value to the secret "cburm-ssh-config" or use "helm upgrade" to update the 'hostKey' field. But please be cautious that the "helm upgrade" will cause pods to restart, for detailed steps, refer to section "Upgrade procedure" in the Containerized Backup and Restore tool user guide.
- hostKey: the Containerized Backup and Restore tool will use this host key to do host key checking. This field only takes effect when SSH.strictHostKeyChecking is set to "yes" or "autoadd".



Note: For application backup and cbur-mster backup, the backup data will be saved in /BACKUP temporarily before transferring to sftp server, you need to allocate enough space for /BACKUP volume.

1) Example of sftp settings in helm chart

```
# sftp setting in helm chart  
  
SSH:
```

```

mode: "sftp"
username: "user1"
host: "10.75.145.6"
port: 22
path: "/sftp-root/bcmtnname"
strictHostKeyChecking: !!str "yes"    # only supports "yes", "no", "autoadd"
hostKey: "ecdsa-sha2-nistp256
AAAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAbmlzdHAyNTYAAABBD5oFeLow1QLEkhoTHM8 /
apxDOTCDlFbkbhHPmqrBHoysmsSifj807J/bTRI8wBAaGWtzgHZDBMRU1YJcAP5gWw="
```

2) Example of secret "cburm-ssh-config"

```

# sftp setting in secret

apiVersion: v1
kind: Secret
metadata:
  name: cburm-ssh-config
  namespace: default
type: Opaque
stringData:
  port: "22"
  host: 10.75.145.11
  mode: sftp
  username: user1
  path: /sftp-root/bcmtnname
  strictHostKeyChecking: "no"
  hostKey: ""
```

User Key Authentication

User needs to get the public key from secret “cburm-ssh-public-key” in cbur-master’s namespace and add it to the “`~/.ssh/authorized_keys`” of sftp user in sftp server.

```
kubectl get secret -n ncms cburm-ssh-public-key -o
jsonpath={.data.ssh_public_key} | base64 -d
```

backup/restore with SFTP

Run helm backup and restore command to do backup/restore.

1. helm backup -t app -a none/enable/disable
2. helm restore -t app -b backup_id

20.2.7 Copying the Public Key for the Backup and Restore Tool to the SFTP Server

About this task

This is a post procedure for backup and restore to work with SFTP.

1. On the cluster CLI (on one of the controller/master nodes), run the command: `kubectl get secret -n ncms cburm-ssh-public-key -o jsonpath={.data.ssh_public_key} | base64 -d && echo`
2. Copy the output and paste it in the SFTP server, in the last line of the file, `/<SFTP user home directory>/ .ssh/authorized_keys`

20.2.8 Backup or Restore with Avamar Server

Install or Upgrade

If no CBUR chart is installed, execute following command:

```
helm install csf-helm-releases/cbur --version 1.3.5 -n cbur-master --namespace ncms --set avamar.enabled=true,avamar.server=10.75.53.220,avamar.clientName=<user_defined_name>,avamar.domain=ARC,control_node_ip_list="10.x.x.x 10.x.x.x 10.x.x.x"
```

If CBUR installed with an old version, execute following command:

```
helm upgrade cbur-master csf-helm-releases/cbur --version 1.3.5 --namespace ncms --set avamar.enabled=true,avamar.server=10.75.53.220,avamar.clientName=<user_defined_name>,avamar.domain=ARC,control_node_ip_list="10.x.x.x 10.x.x.x 10.x.x.x"
```

 **Important:** The following values must be set correctly, or Avamar backup/restore may fail

avamar.enabled: must be set to "true"

avamar.server: IP address of Avamar server, AKA, MSC IP address. For example, 10.40.32.23 which is owned/maintained by NetAct

avamar.clientName: user defined client name, or use the BCMT control node name prefix by default, optional parameter.

avamar.domain: Avamar server domain name where Avamar client will register to

control_node_ip_list: External OAM IP list of bcmt control nodes which can be directly reached from avamar server. Although It can be empty, if it is empty and Kubernetes reallocates the avamar pod to another Control node, avmar client would not be able to re-register to Avamar server again and following backup/restore will fail.

You must use a RWX volume:

- If using direct GlusterFS volume, `volumeType.glusterfs` must be set to "true".
- If using dynamic PVC, RWX PVC must be used and `isPvRwx` must be set to "true", `volumeType.glusterfs` must be set to "false".

hostNetwork.avamar: Must set to "false" when "istio.enabled=true". Otherwise, istio-proxy sidecar cannot be injected to Avamar pod.

There are the following side effects when "hostNetwork.avamar=false":

- In Avamar GUI, you can see the backup/restore result, you cannot see the detailed logs for a specific backup/restore. You can check the logs in Backup and Recovery Avamar pod (dir: /var/avamar/clientlogs).
- The backup/restore work may start a little late. In Avamar lab-6 and lab-7, it is less than 60 seconds.

The following command could change the Avamar client setting, then the Avamar client pod will restart and re-register to Avamar server.

```
kubectl edit deployment cbur-master-cbur-avamar -nncms
```

 **Note:** You should use the ARC-FP6 related Nokia Archive Cloud lab for arccli because of BRCLOUD-3266.

 **Important:** You should check the MTU of the network interface by which avamar client accesses the server. Because if the MTU is inconsistent between Avamar server and client, some unexpected errors may happen. You should consult Avamar server administrator for the right MTU size.

Example:

The following example uses MTU 1500, the following steps should be done in all controller nodes.

1. To find related network interface (give ethx as example) whose IP is in `control_node_ip_list` used to access Avamar server
2. To check the MTU status of this network interface ethx:

```
$ ip addr or $ netstat -i
```

3. To apply the setting temporarily:

```
$ ifconfig mtu 1500
```

4. To change it permanently:

```
$ vim /etc/sysconfig/network-scripts/ifcfg-ethx
```

Add the following content:

```
MTU=1500
```

5. Then restart the network:

```
$ systemctl restart network
```

Check Avamar Client Registration

- Log in to the ARC Management Node via SSH on port 7722 as the brc user. Execute:

```
[user@somenode~]# ssh brc@<MANAGEMENT_NODE_IP> -p 7722
```

- Provide a password for brc user..
- Listing all Clients

After install/upgrade successfully, you should be able to find the registered client from ARC management node with user defined client name or use the BCMT control node name prefix by default. For example, BCMT control node name is 'mybcmtn-g01-c-01/02/03', then the client registered on Aamvar is "mybcmtn-g01-c". So it is important that you have unique name for different cluster.

Here the client name is tw_test which is activated and enabled.

```
[brc@arc-mgmt-cli ~]$ arccli client list
2020-03-12 03:40:46,600 INFO      Preparing clients list
+-----+-----+-----+
| Host           | Activated | Enabled |
+-----+-----+-----+
| 10.58.164.194 | Yes       | Yes     |
| 10.58.164.196 | Yes       | Yes     |
| 10.58.164.197 | Yes       | Yes     |
| 10.58.164.198 | Yes       | Yes     |
| amokfutas117-infra | Yes       | Yes     |
| kejittas62      | Yes       | Yes     |
| kejittas62-infra | Yes       | Yes     |
| twcburvm        | Yes       | Yes     |
| tw_test          | Yes       | Yes     |
+-----+-----+-----+
```

For listing all clients, the following headers are visible:

- * Host is a hostname of the Client
- * Activated is the status of Client activation: Yes or No
- * Enabled is the status of Client to back up files: Yes or No

Backup configuration

To add a new backup configuration, a file with Client's backup configuration must be prepared. The file must be in JSON format and contain details about:

- name
- paths to backup
- backup schedule

- data retention time

The following example shows a backup configuration file content:

```
{
  "backupConfigurationIdentifier" : {
    "clientType" : "NE",
    "clientVersion" : "18.2",
    "configurationName" : "CCBUR"
  },
  "backupDefinitions" : [ {
    "name" : "ALL",
    "description" : "Configuration files",
    "dataset" : {
      "paths" : [ "/BACKUP/default/DEPLOYMENT_cbura-cbur/" ],
      "preBackupScript" : "pre-backup.sh -t cbura-cbur",
      "postBackupScript" : "clean.sh -t cbura-cbur",
      "type" : "FILESYSTEM"
    },
    "schedule" : {
      "times" : [ "11:30" ],
      "maxRunHours" : 3,
      "type" : "DAILY"
    },
    "retention" : {
      "duration" : 7,
      "unit" : "DAYS"
    }
  }
]
}
```

According to this example:

- Backup Definition is called "ALL".
- This Backup Definition will have the cbura-cbur backed up every day at 11:30 hour, and data stored for 7 days.

1. Backup Configuration properties

The following table presents the possible values for the backupConfigurationIdentifier category:

Path	Type	Description	Constraints
backupConfigurationIdentifier	Object	Identifies given Backup Configuration	Required
backupDefinitions	Array	The list of Backup Definitions	Required

2. Backup Configuration Identifier properties

Backup Configuration Identifier identifies given Backup Configuration. The following table presents the Backup Configuration Identifier properties:

Path	Type	Description	Regular expression	Required
clientType	String	Type of the client given backup configuration is targeted for.	^[a-zA-Z0-9.-]+ \$	Yes
clientVersion	String	Version of the client given backup configuration is targeted for.	^[a-zA-Z0-9.-]+ \$	Yes
configurationName	String	Name uniquely identifying the backup configuration for targeted client type and version.	^[a-zA-Z0-9.-]+ \$	Yes



Note: Cumulative length of clientType, clientVersion, configurationName and name of Backup Definition cannot exceed 45 characters.

3. Backup Definition properties

Backup Definition contains details such as data for backup, when backup runs and how long backed up data is stored. It is possible to have multiple Backup Definitions in each Backup Configuration.

The following table presents the backup definition properties:

Path	Type	Description	Constraints
name	String	Name of the backup definition. The name must be unique among all backup definitions for given backup configuration.	Required. Must match the regular expression: ^[a-zA-Z0-9.-]+\$
description	String	A description for the Backup Definition.	Optional
dataset	Object	Defines backup data, preand post-backup scripts.	Required
schedule	Object	Defines when backup runs.	Required
retention	Object	Defines how long backup should be kept.	Required

- **Dataset settings** Dataset defines what needs to be backed up. the Containerized Backup and Restore tool use Filesystem Dataset. The possible attributes are presented in the following table:

Path	Type	Description	Constraints
paths	Array	1. For backup cburm self volume, then the paths in dataset should be "/CBUR_REPO/data and /CBUR_REPO/record" . If the volume is only cbur-repo, then it is not necessary to provision pre-backup and post restore script.	Required

Path	Type	Description	Constraints
		<p>To backup volume that attach to cbur-m, then needs to run pre-backup.sh and post-restore.sh script for cbur-master.</p> <p>2. For application backup/restore, input paths in format of <code>"/BACKUP/[NAMESPACE]/[DEPLOYMENT STATEFULSET DAEMONSET K8SOBJECTS PVC]_[brpolicy_name]/"</code>.</p> <ul style="list-style-type: none"> – To backup volumes, the name of daemonset/deployment/statefulset should be same with the Br Policy. <p>Here, Kubernetes deployment name is cbura-cbur, which is in default namespace, so the path is <code>"/BACKUP/default/DEPLOYMENT_cbura-cbur/"</code>.</p> <p>If you want to backup your statefulset sdc-sdc in namespace: ncms, the path would be <code>"/BACKUP/ncms/STATEFULSET_sdc-sdc/"</code>.</p> <ul style="list-style-type: none"> – If the application to backup is none of Deployment / StatefulSet / DaemonSet: <ul style="list-style-type: none"> a. If 'k8sobjects' is defined in BrPolicy, the path is <code>"/BACKUP/[NAMESPACE]/K8SOBJECTS_[brpolicy_name]/"</code>; b. If only 'pvc' is defined in BrPolicy, the path is <code>"/BACKUP/[NAMESPACE]/PVC_[brpolicy_name]/"</code> <p>If you use release scripts to backup/restore the whole helm release, input paths for all the Br Policies in the helm release according to the above format.</p> <p>For example:</p> <pre>"paths" : ["/BACKUP/foo/STATEFULSET_foo-jk-crdbredisio-server/", "/BACKUP/foo/STATEFULSET_foo-jk-foo/"]**</pre>	
pre Backup Script	String	<p>Name of the script to run before backup.</p> <p>1. To backup one BrPolicy pre-backup.sh -t Here is: pre-backup.sh -t cbura-cbur</p> <p>2. To backup one helm release release-pre-backup.sh -t</p>	Required

Path	Type	Description	Constraints
post Backup Script	String	<p>Name of the script to run after backup.</p> <p>1. To backup one BrPolicy clean.sh -t cbura</p> <p>2. To Backup one helm release release-clean.sh -t</p>	Required
type	String	<p>Type of backed up data.</p> <p>Must be: FILESYSTEM</p>	Required

preBackupScript for backup one BrPolicy

This command backs up a BrPolicy.

Usage:

```
pre-backup.sh -t BrPolicy [flags]
```

Flags:

- c timer counter, wait [counter]*5s for backup to complete, default is 120, which means 10 minutes. It only makes sense when celery is enable on cbur master
- n namespace, use default namespace by default
- H cburm service name/IP. If not set, use default cbur service name. usually, pls don't set it, unless you have another cburm service available rather than default.
- U specify which login username to be verified when auth enabled (option)
- P specify which login password to be verified when auth enabled (option)
- m, --max-attempts The max attempts to run backup in case of failure (Default: 1, i.e. will not retry)
- i, --retry-interval Time in seconds between each retry (default 60)

postBackupScript for backup one BrPolicy

```
clean.sh -t BrPolicy [-n namespace]
```

preBackupScript for backup one Helm release

This command backs up a helm release.

Usage:

```
release-pre-backup.sh -t HELM_RELEASE [flags]
```

Flags:

```

-t HELM_RELEASE      Specify the helm release name to backup.
-n NAMESPACE         Specify the namespace of this helm release. (Only
                     support helm version 3)
-l, --tiller-namespace TILLER_NAMESPACE
                     Specify the tiller_namespace of this helm release.
Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION      Only support 2 (helm version 2) and 3 (helm verison 3).
                     When "-v" is not specified,
                     - if "--tiller-namespace" is specified, the
Containerized Backup and Restore tool will process it as helm version 2;
                     - otherwise if "-n" is specified, the Containerized
Backup and Restore tool will process it as helm version 3;
                     - otherwise as helm version 2.
-U CBUR_USERNAME     Specify the username of the Containerized Backup and
Restore tool when auth is enabled
-P CBUR_PASSWORD     Specify the password of the Containerized Backup and
Restore tool when auth is enabled
-m, --max-attempts   The max attempts to run backup in case of failure
(Default: 1, i.e. will not retry)
-i, --retry-interval Time in seconds between each retry (default 60)
-T, --timeout DURATION When celery on, how long to wait for the backup
task to complete (default: 1h0m0s).
                     Please note, it is for one backup attempt, the timer
will be resumed for each retry.
                     The format can be:
                     1) <d+>, time in seconds, e.g. 600 (600 seconds)
                     2) <d+>h<d+>m<d+>s, e.g. 2h (2 hours), 2h30m (2 hours
30 mins), 30s (30 seconds)
-h/--help             Display this help text and exit.

```

Example:

```

release-pre-backup.sh -t csdc -n ns1 # helm version 3
release-pre-backup.sh -t csdc -l app_tiller_ns # helm version 2
release-pre-backup.sh -t csdc --tiller-namespace app_tiller_ns # helm
version 2
release-pre-backup.sh -t csdc -v 2 # helm version 2
# If backup fails, only retry once after 120 seconds. And will wait at most
2 hours for each try to complete.
release-pre-backup.sh -t csdc -m 2 -i 120 -T 2h
release-pre-backup.sh -t csdc --max-attempts 2 --retry-interval 120 --
timeout 2h
release-pre-backup.sh -h

```

postBackupScript for backup one helm release

This command cleans up the directory in cbur which is to save the backup data for the helm release.

Usage:

```
release-clean.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to clean up.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified,
	- if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2;
	- otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3;
	- otherwise as helm version 2.
-h/---help	Display this help text and exit.

Example:

```
release-clean.sh -t csdc -n ns1 # helm version 3
release-clean.sh -t csdc --tiller-namespace app_tiller_ns # helm version 2
release-clean.sh -h
```

- **Schedule settings**

The schedule defines when backup runs. There are several schedule types supported:

- Daily Schedule
- Weekly Schedule
- Monthly Schedule:
 - Monthly Schedule for specific day of month
 - Monthly Schedule for specific day of specific week of month

Daily Schedule

The daily schedule properties are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: DAILY
times	Array	Times of day in HH:MM format when backups shall start. Interval between backup start times must be greater or equal to maximum backup execution time limit specified by max RunHours property.	Required
maxRunHours	Integer	Maximum backup execution time in hours. This parameter is taken into account after a first successful backup. The first execution can be longer.	Required Integer between 1 and 23 inclusive

The following example shows the usage of the daily schedules properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
    }
  ]
}
```

```

    "schedule": {
        "type": "DAILY",
        "times": [
            "01:00",
            "16:00"
        ],
        "maxRunHours": 1
    },
    "retention": {
        "duration": 10,
        "unit": "WEEKS"
    }
}
]
}

```

Weekly Schedule

The possible properties of weekly schedule are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: WEEKLY
daysOfWeek	Array	Days of the week when backups runs.	Required. Size must be between 1 and 7 inclusive. The allowed values are: <ul style="list-style-type: none"> • SUNDAY • MONDAY • TUESDAY • WEDNESDAY • THURSDAY • FRIDAY • SATURDAY
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish. This parameter is taken into account after a first successful backup. The first execution can be longer.	Required The format is HH:MM

The following example shows the usage of the weekly schedules properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
      "schedule": {
        "type": "WEEKLY",
        "daysOfWeek": [
          "SATURDAY",
          "SUNDAY"
        ],
        "runAfter": "00:00",
        "runBefore": "02:30"
      },
      "retention": {
        "duration": 10,
        "unit": "WEEKS"
      }
    }
  ]
}
```

Monthly Schedule

Monthly Schedule for specific day of month:

This type of schedule allows configuring backups to be run on specific day of each month. The possible properties of monthly schedule for specific day of month are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: MONTHLY
dayOfMonth	String	Specifies the day of month when backup is executed.	Required Range between values 1 and 28 inclusive or LAST
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish.	Required. The format is HH:MM

The following example shows the usage of monthly schedule for specific day of month properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
      "schedule": {
        "type": "MONTHLY",
        "dayOfMonth": "LAST",
        "runAfter": "00:00",
        "runBefore": "02:30"
      },
      "retention": {
        "duration": 7,
        "count": 1
      }
    }
  ]
}
```

```

        "unit": "DAYS"
    }
}
]
}
```

Monthly Schedule for specific day of specific week of month:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: MONTHLY_ON_SPECIFIC_WEEKDAY
dayOfMonth	String	The week of the month when backup runs.	Required The allowed values are: • FIRST • SECOND • THIRD • LAST
dayOfWeek	String	The day of the specified week of month when backup runs.	Required The allowed values are: • SUNDAY • MONDAY • TUESDAY • WEDNESDAY • THURSDAY • FRIDAY • SATURDAY
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish.	Required The format is HH:MM

The following example shows the usage of the monthly schedules for specific day of specific week of month properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "unit": "DAYS"
    }
  ]
}
```

```

    "name": "ALL",
    "description": "Configuration files",
    "dataset": {
        "paths": [
            "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
    },
    "schedule": {
        "type": "MONTHLY_ON_SPECIFIC_WEEKDAY",
        "weekOfMonth": "FIRST",
        "dayOfWeek": "WEDNESDAY",
        "runAfter": "00:00",
        "runBefore": "02:30"
    },
    "retention": {
        "duration": 7,
        "unit": "DAYS"
    }
}
]
}

```

Retention settings

The retention defines how long a backup must be kept. The possible properties are presented in the following table:

Path	Type	Description	Constraints
duration	Number	The duration backup should be kept.	Required Integer, minimum value is 1
unit	String	Time unit retention duration is expressed with.	Required The allowed values are: • DAYS • WEEKS • MONTHS • YEARS

The following example shows the usage of the retention schedules properties.

```
{
}
```

```

"backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
},
"backupDefinitions": [
{
    "name": "ALL",
    "description": "Configuration files",
    "dataset": {
        "paths": [
            "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
    },
    "schedule": {
        "type": "MONTHLY_ON_SPECIFIC_WEEKDAY",
        "weekOfMonth": "FIRST",
        "dayOfWeek": "WEDNESDAY",
        "runAfter": "00:00",
        "runBefore": "02:30"
    },
    "retention": {
        "duration": 10,
        "unit": "WEEKS"
    }
}
]
}

```

Client management

- 1. Adding Backup Configuration** To add Backup Configuration, execute the arccli backup-configuration create command:

```
[brc@arc-mgmt-cli~]$ arccli backup-configuration create --file
<PATH_TO_BACKUP_CONF_FILE>
```

where: is a path to file with prepared Backup Configuration. Example:

```
[brc@arc-mgmt-cli ~]$ arccli backup-configuration create -f ./twcbur.json
2020-02-27 10:10:36,241 INFO      Adding backup definition:
NE_18.2_CCBUR_ALL
```

```

2020-02-27 10:10:36,242 INFO Adding schedule with name:
NE_18.2_CCBUR_ALL
2020-02-27 10:10:47,432 INFO Adding retention with name:
NE_18.2_CCBUR_ALL
2020-02-27 10:10:54,793 INFO Adding dataset: NE_18.2_CCBUR_ALL
2020-02-27 10:11:04,632 INFO Adding dataset: NE_18.2_CCBUR_ALL targets
2020-02-27 10:11:10,360 INFO Adding dataset: NE_18.2_CCBUR_ALL options
2020-02-27 10:11:27,226 INFO Adding group: NE_18.2_CCBUR_ALL
2020-02-27 10:11:34,337 INFO Backup definition: NE_18.2_CCBUR_ALL has
been added
2020-02-27 10:11:34,338 INFO Backup configuration: NE_18.2_CCBUR has
been added

```

2. Adding Client to Backup Configuration

Prerequisites:

- Client is integrated with Backup Engine Node.
- Backup Configuration exists.
- Client is not integrated with any Backup Configuration. To add Client to existing Backup Configuration, execute command, where: is a hostname of the Client to be added is a Backup Configuration Identifier.

Example:

```

[brc@arc-mgmt-cli ~]$ arccli backup-configuration add-client -c tw_test
NE_18.2_CCBUR
2020-02-27 10:13:28,688 INFO Getting tw_test client details
2020-02-27 10:13:36,230 INFO Attaching client: tw_test to backup
configuration: NE_18.2_CCBUR
2020-02-27 10:13:39,802 INFO Adding client: tw_test to backup
definition: NE_18.2_CCBUR_ALL
2020-02-27 10:13:46,050 INFO Client: tw_test has been added to backup
configuration: NE_18.2_CCBUR

```

3. Showing Client details

To show details about single Client integrated to Backup Engine Node, execute the arccli client show command specifying client hostname, as shown below:

```
[brc@arc-mgmt-cli ~]$ arccli client show <CLIENT_HOSTNAME>
```

Example:

```
[brc@arc-mgmt-cli ~]$ arccli client show tw_test
2020-03-12 04:00:24,696 INFO Displaying client: tw_test details
```

Host	Client Type	Client Version	Configuration Name	Backup Definitions	Activated	Enabled
tw_test	NE	18.2	NE_18.2_CCBUR	NE_18.2_CCBUR_ALL	Yes	Yes

For listing single Client details, the following headers are visible:

- Host is client name
- Client Type is the type of network element
- Client Version specifies network element software release
- Configuration Name is a Backup Configuration name
- Backup Definitions are Backup Definition IDs
- State is the status of Client: Activated or Not activated

Perform Backup with ARCLI

1. Configure APP/chart brpolicy

Set **backend mode to "Avamar"** in brpolicy file of target app in helm chart. cburn will base on this customized setting performing different backup/restore procedure for different target app/charts existing in the same cluster.

2. Triggering on demand backup

Backup starts automatically at time determined in the Backup Definition Schedule. It is possible to trigger backup on demand by using the arccli command. On demand backup can be triggered for all Backup Definitions of Client's Backup Configuration at once or for a selected one.

where: is name of the integrated Client.

The backup on demand starts and the following log information is visible in the console:

```
[brc@arc-mgmt-cli ~]$ arccli backup run -c tw_test
2020-03-09 10:05:32,412 INFO      Starting backup on demand for client:
tw_test
2020-03-09 10:05:52,068 INFO      Client tw_test backup scheduled for
backup definition: NE_18.2_CCBUR_ALL
```

3. Listing backups

Listing backups allows listing available backups for specified Client in a table format. On the list, both scheduled and on demand backups are visible.

To list backups, execute the arccli backup list command by specifying Client hostname in the following way:

```
[brc@managementnode-cli ~]$ arccli backup list -c <CLIENT_HOSTNAME>
```

where: is a hostname of the Client for which backups will be listed.

Example:

The table with details of available backups is visible, as follows:

```
[brc@arc-mgmt-cli ~]$ arccli backup list -c tw_test
2020-03-06 09:52:22,306 INFO      Displaying client tw_test backups
+-----+-----+-----+
+-----+-----+-----+
| Backup Definition | Label Number | Created           | Expires
| Size             | Location       |                   |
+-----+-----+-----+
+-----+-----+-----+
| NE_18.2_CCBUR_ALL | 2            | 2020-03-06 09:35 AM | 2020-03-13
09:31 AM | 28.5 KB     | DD - 10.91.56.108 |
| NE_18.2_CCBUR_ALL | 1            | 2020-03-06 08:46 AM | 2020-03-13
08:38 AM | 205.3 MB    | DD - 10.91.56.108 |
+-----+-----+-----+
+-----+-----+-----+
```

For listing backups, the following headers are visible:

- a. Backup Definition**
- b. Backup Definitions ID.**
- c. Label Number:** number assigned to each backup
- d. Created:** date and time when backup is created
- e. Expires:** date and time when backup expires
- f. Size:** size of backed up data
- g. Location:** location where backup is saved

4. Activity management

To list all activities, execute:

```
[brc@managementnode-cli ~]$ arccli activity list [options]
```

Available options:

- a. active to show active activities
- b. completed to show completed activities
- c. queued to show queued activities
- d. client (or -c) to show activities for specified client

Example: The table with activities is visible:

```
[brc@arc-mgmt-cli ~]$ arccli activity list -c tw_test
2020-03-09 03:59:59,441 INFO      Displaying activities for client: tw_test
+-----+-----+-----+
+-----+-----+-----+
| ID          | Backup Definition    | Status        | Start
Time           | End Time             | Client       |
+-----+-----+-----+
+-----+-----+-----+
| 9158348349561009 | NE_18.2_CCBUR_ALL   | Completed    |
| 2020-03-06 09:33 CET | 2020-03-06 09:35 CET | tw_test      |
| 9158348028230309 | NE_18.2_CCBUR_ALL   | Completed    |
| 2020-03-06 08:40 CET | 2020-03-06 08:46 CET | tw_test      |
+-----+-----+-----+
+-----+-----+-----+
```

Visible headers:

- ID is an activity ID
- Backup Definition is Backup Definition ID.
- Status is a status of the activity.
- Start Time is the date and time when activity began.
- End Time is the date and time when activity ended.
- Client is an ARC Client name.

5. Showing activity details

To show activity details, execute:

```
[brc@managementnode-cli ~]$ arccli activity show <ACTIVITY_ID>
```

Example: The table with activity details is visible.

```
[brc@arc-mgmt-cli ~]$ arccli activity show 9158348349561009
2020-03-04 09:51:32,054 INFO      Displaying activity: 9158348349561009
details
+-----+-----+
| Attribute      | Value        |
+-----+-----+
```

ID	9158348349561009
Status	Completed
Start Time	2020-03-06 09:33 CET
Elapsed	00h:02m:01s
End Time	2020-03-06 09:35 CET
Type	Scheduled Backup
Progress Bytes	288.5 MB
New Bytes	0%
Client	tw_test
Client Release	18.2.100-134
Scheduled Start Time	2020-03-06 09:33 CET
Scheduled End Time	2020-03-06 09:35 CET
Backup Definition	NE_18.2_CCBUR_ALL
Plug-In	Linux File System
Retention	D

Visible attributes:

- Id is an activity ID
- Status is a status of the activity.
- Start Time is date and time that this activity began
- Elapsed is an elapsed time for this activity
- End Time is date and time that this activity completed
- Type is the type of activity.
- Progress Bytes is a total number of bytes examined during this activity
- New Bytes is a percentage of new bytes backed up
- Client is an ARC Client hostname
- Client Release is the Avamar client software version
- Scheduled Start Time is date and time that this activity was scheduled to begin
- Scheduled End Time is date and time that this activity was scheduled to end
- Backup Definition is Backup Definition ID.
- Plug-In is the plugin that is used for this activity
- Retention are retention types that are assigned to this backup

Perform Restore with Avamar GUI

Click Backup & Restore from top level for GUI. → Click ARC/[Your own Domain] → click the client name → find the backup file.

Click Action → Restore Now → On new pop out window, click More Option → on another new pop out window, click show Advanced Options to input pre and post script → click OK → click OK

To restore 1 BrPolicy:

Pre-Script: clean.sh -t

Post-Script: post-restore.sh -t

To restore 1 Helm release:

Pre-Script: release-clean.sh -t

Post-Script: release-post-restore.sh -t

```
#Pre-Script to restore one BrPolicy

clean.sh -t BrPolicy -n [namespace]
#Post-Script to restore one BrPolicy
```

This command restores a BrPolicy.

Usage:

```
post-restore.sh -t BrPolicy [flags]
```

Flags:

```
-c timer counter, wait [counter]*5s for restore to complete, default is
120, which means 10 minutes. It only makes sense when celery is enable on
cbur master

-n namespace, use default namespace by default

-H cburm service name/IP. If not set, use default cbur service name.
usually, pls don't set it, unless you have another cburm service available
rather that default.

-U specify which login username to be verified when auth enabled

-P specify which login password to be verified when auth enabled

-v specify whether to restore volume data or not, supported values is
"true/false", default is false

-p specify which k8sobject type to be restore, it can be a list or a
string, e.g. "all"/"secret"/"secret configmaps"

-m specify which k8sobject name will be restored for specified
object_type. It only supports string format

if both -v and -p args not provided, it means restore all that in BrPolicy.

--max-attempts The max attempts to run restore in case of failure
(Default: 1, i.e. will not retry)

--retry-interval Time in seconds between each retry (default 60)

#Pre-Script to restore one helm release
```

This command cleans up the direcotry in cbur which is to save the backup data for the helm release.

Usage:

```
release-clean.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to clean up.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified, - if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2; - otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3; - otherwise as helm version 2.
-h/--help	Display this help text and exit.

Example:

```
release-clean.sh -t csdc -n ns1 # helm version 3
release-clean.sh -t csdc --tiller-namespace app_tiller_ns # helm version 2
release-clean.sh -h
#Post-Script to restore one helm release
```

This command restores a helm release.

Usage:

```
release-post-restore.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to restore.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified, - if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2; - otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3; - otherwise as helm version 2.

```

-U CBUR_USERNAME      Specify the username of the Containerized Backup and
Restore tool when auth is enabled
-P CBUR_PASSWORD      Specify the password of the Containerized Backup and
Restore tool when auth is enabled
--brpolicies          Restore multiple BrPolicies separated by commas in the
specified order.
--exclude-brpolicies To exclude the given BrPolicies separated by commas.
Note:
If there are multiple BrPolicies in "--brpolicies" or
"--exclude-brpolicies",
please add single quotes ('') or double quotes ("")
around the value when using this script in Avamar GUI.
--volume true|false   Specify whether to restore volume data or not. Default
is false.
--object_type OBJECT_TYPE
Specify the k8sobject type to restore, it can be a list
or a string, e.g. "all"/"secrets"/"secrets configmaps"
or repeat multiple times to specify multiple object
types
--object_name OBJECT_NAME
Specify the k8sobject name to restore for specified
object_type. It only supports string format
and will be applied to all object_type.
--max-attempts        The max attempts to run restore in case of failure
(Default: 1, i.e. will not retry)
--retry-interval      Time in seconds between each retry (default 60)
--timeout DURATION    When celery on, how long to wait for the restore task
to complete (default: 1h0m0s).
Please note, it is for one restoration attempt, the
timer will be resumed for each retry.
The format can be:
1) <d+>, time in seconds, e.g. 600 (600 seconds)
2) <d+>h<d+>m<d+>s, e.g. 2h (2 hours), 2h30m (2 hours
30 mins), 30s (30 seconds)
-h/--help              Display this help text and exit.

```

The Scope of Data to Restore:

- * The pvcs defined in BrPolicy are always restored.
- * If '--volume', '--object_type' and '--object_name' are all not provided, then all items defined in BrPolicy are restored.
- * Otherwise, if one of them are provided:
 - if '--volume true' is not provided, then no volumes are restored (even if they are defined in BrPolicy).
 - if '--object_type' is not provided, then no k8s objects are restored (even if they are defined in BrPolicy).

Example:

```
release-post-restore.sh -t csdc -v 2 # helm version 2
release-post-restore.sh -t csdc --brpolicies 'br1,br2,br3' --object_type all
--tiller-namespace app_tiller_ns    # helm version 2
release-post-restore.sh -t csdc --exclude-brpolicies "br1,br2" --volume true
-n ns1   # helm version 3
release-post-restore.sh -t csdc --object_type "secrets" --object_name
"applname" -n ns1   # helm version 3
# If the restoration fails, at most retry two times, each after 120 seconds.
And will wait at most 2 hours for each try to complete.
release-post-restore.sh -t csdc --max-attempts 3 --retry-interval 120 --
timeout 2h
release-post-restore.sh -h
```

You should be able to see a new restore job from the Activity window.

Once it completes, you should be able to see "Completed" without any error code.

Perform Restore from Avamar Pod

The restore procedure can be performed on Client by using the Avamar Client CLI avtar command. Syntax for restoring data by using the avtar command should look as follows:

```
avtar --extract [options]
```

The avtar command options are presented in the following table:

Option	Description
--path=/ARC/	<ul style="list-style-type: none"> • Avamar server account path • Mandatory parameter
--id=@/ARC	<ul style="list-style-type: none"> • The authentication Avamar user ID • The predefined restore user can be used • Mandatory parameter
--password=	<ul style="list-style-type: none"> • The authentication password for the Avamar user • Optional parameter (when not provided, command will prompt for the password) Note: Now

Option	Description
	we get the password from Nokia Archive Cloud (password=testtest)
--labelnumber=	<ul style="list-style-type: none"> Specifies backup to be restored • Backup label numbers can be determined by Listing backups • Mandatory parameter
--run-at-start=	<ul style="list-style-type: none"> Name of the script to run before restoring starts The script must be located on Client in the /etc/avamar/scripts folder • Mandatory parameter1) To restore one BrPolicy:clean.sh -t 2) To restore one helm releaserelease-clean.sh -t
--run-at-end=	<ul style="list-style-type: none"> Name of the script to run after restoring starts The script must be located on Client in the /etc/avamar/scripts folder • Mandatory parameter1) To restore one BrPolicypost-restore.sh -t 2) To restore one helm releaserelease-post-restore.sh -t

1. Get Avamar pod on BCMT cluster node

```
# Get Avamar pod

[root@server-name]# kubectl get pods -n ncms | grep avamar
cbur-master-cbur-avamar-5f867cf749-97sn4    1/1      Running     0          4d3h
```

2. Log into Avamar pod to perform restore

```
[root@server-name]# kubectl exec -it -nncms cbur-master-cbur-
avamar-5f867cf749-97sn4 bash
````-4.2# avtar --extract --path=/ARC/tw_test --id=restore@/ARC --
password=testtest --labelnumber=1 --run-at-start="clean.sh -t cbura-cbur" --
run-at-end="post-restore.sh -t cbura-cbur"
```

Expected outcome:

```
Restore Logs

````-4.2# avtar --extract --path=/ARC/tw_test --id=restore@/ARC --
password=testtest --labelnumber=1 --run-at-start="clean.sh -t cbura-cbur" --
run-at-end="post-restore.sh -t cbura-cbur"
```

```
avtar Info <5551>: Command Line: /usr/local/avamar/bin/avtar.bin --  
vardir=/usr/local/avamar/var --bindir=/usr/local/avamar/bin --sysdir=/usr/  
local/avamar/etc --extract --account=/ARC/tw_test --id=restore@/ARC --  
password=***** --sequencenumber=1 --run-at-start="clean.sh -t  
cbura-cbur" --run-at-end="post-restore.sh -t cbura-cbur"  
avtar Info <7977>: Starting at 2020-03-09 11:03:18 CST [avtar Dec 5 2018  
07:44:00 18.2.100-134 Linux-x86_64]  
avtar Info <5916>: Executing run-at-start '/usr/local/avamar/etc/scripts/  
clean.sh -t cbura-cbur'  
avtar Info <5917>: Back from run-at-start, exit code 0  
avtar Info <6555>: Initializing connection  
avtar Info <5552>: Connecting to Avamar Server (10.75.53.220)  
avtar Info <5554>: Connecting to one node in each datacenter  
avtar Info <5583>: Login User: "restore", Domain: "default", Account: "/ARC/  
tw_test"  
avtar Info <5580>: Logging in on connection 0 (server 0)  
avtar Info <5582>: Avamar Server login successful  
avtar Info <10632>: Using Client-ID='0208bfaa1797945b978cb498330db88559c2460e'  
avtar Info <5550>: Successfully logged into Avamar Server [18.2.0-134]  
avtar Info <5295>: Starting restore at 2020-03-09 11:03:23 CST as "root" on  
"tw-control-01" (4 CPUs) [18.2.100-134]  
avtar Info <10534>: Initializing session on Data Domain: 1, mode:RESTORE,  
boost:ENABLED, compressed-restore:disabled, max-streams:1, pool-  
size:67108864, queue-size:6, read-size:131072  
avtar Info <40174>: Multi-stream restore via cloning is enabled.  
avtar Info <10632>: Using Client-ID='0208bfaa1797945b978cb498330db88559c2460e'  
avtar Info <40062>: GSAN connected via: IPv4, retrieved Data Domain IPv4  
hostname = 10.91.56.108  
avtar Info <10539>: Connecting to Data Domain Server "10.91.56.108"(1) (LSU:  
avamar-1580813747, User: "*****")  
avtar Info <40206>: Setting default storage unit to 'avamar-1580813747' for  
handle 1  
avtar Info <41440>: Data Domain handle:1 capabilities:0x0820021B  
avtar Info <10543>: Data Domain login to 10.91.56.108 successful  
avtar Info <18845>: Restore mode - No Data Domain read compression.  
avtar Info <41359>: Backup #1 had been created by avtar version 18.2.100-134  
as plugin 1001-Filesystem  
avtar Info <5949>: Backup file system character encoding is UTF-8.  
avtar Info <8745>: Backup from Linux host "/ARC/tw_test" (tw-control-01) with  
plugin 1001 - Linux Filesystem  
avtar Info <5538>: Backup #1 label "NE_18.2_CCBUR_ALL" timestamp 2020-03-09  
10:56:42 CST, 1 files, 144 bytes  
avtar Warning <19107>: Unable to determine if there is enough disk space for  
the specified restore.  
avtar Info <5259>: Restoring backup to directory ""
```

```
avtar Info <18076>: Opening DD stored file system container 1 in /  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4 for restore  
operation, using pre-fetched ddr_files.xml.  
avtar Info <15265>: Opening file system container /  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4//  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4/container.1.cdsf  
for restore.  
avtar Info <5262>: Restore completed  
avtar Info <7925>: Restored 144 bytes from selection(s) with 144 bytes in 1  
files, 5 directories  
avtar Info <6090>: Restored 144 bytes in 0.33 minutes: 25.73 KB/hour (183  
files/hour)  
avtar Info <7883>: Finished at 2020-03-09 11:03:38 CST, Elapsed time:  
0000h:00m:19s  
avtar Info <5916>: Executing run-at-end '/usr/local/avamar/etc/scripts/post-  
restore.sh -t cbura-cbur'  
avtar Info <6031>: Begin STDOUT from run-at-end:  
Restore succeeded  
avtar Info <6032>: End of STDOUT  
avtar Info <5917>: Back from run-at-end, exit code 0  
avtar Info <6645>: Not sending wrapup anywhere.  
avtar Info <5314>: Command completed (1 warning, exit code 0: success)
```

20.3 Containerized Application Backup and Recovery (the Containerized Backup and Restore tool)

Backup and restore commands for Virtualized Deployments are done through NCS CLIs and Helm plugins.

20.3.1 Introduction to Backup and Recovery

Containerized Backup and Recovery is an asset which can back up and restore Kubernetes volumes, resources and cluster data. Currently, Backup and Recovery can work in both NCS, Minikube and OpenShift environment. It leverages the design idea of existing Backup and Recovery, and provides support to the following backup/restore modes:

- Local backup and restore (backend: LOCAL)
- Third party EMC Avamar backup/restore (backend: AVAMAR, only support in NCS and Minikube Environment)
- Amazon S3 backup/restore (backend: AWSS3)
- Ceph cluster S3 API backup and restore (backend: CEPHS3)
- Swift cluster S3 API backup and restore (backend: SWIFTS3)

- Backup/restore with target server (backend: SFTP)

Cluster Backup / Restore is mainly for Disaster Recovery and is only supported in NCS Environment.

 **Note:** Because of framework changes in Backup and Recovery 19.06, pay attention to following case, if Backup and restore is upgraded from old versions to 19.06, all backup data will not be recognized by backup and restore 19.06, backup your application as soon as possible after upgrading backup and restore to 19.06.

Namespace backup/restore is supported from 20.10 FP4.

20.3.2 Prerequisites

Before deploying Backup and Restore, storage space must be planned in advance, which includes:

- volumes in `cbur-master`
- `/tmp` volume in `cbura-siddercar`

How to calculate wanted cluster volume size:

- In case of local backup: $(\text{rotation}+1) * (\text{size of cluster backup}) + 5\text{GB}$.
- In case of SFTP/Avamar backup: $2 * (\text{size of cluster backup}) + 5\text{GB}$

`cluster_backup_size = size of (/opt/bcmt/storage/docker-registry/ + /opt/bcmt/storage/charts/ + /opt/bcmt/storage/user-db4keycloak/ + /opt/bcmt/storage/hooks/ + /opt/bcmt/storage/helm_home/)`

Application volume sizes are described as follows.

Volumes in `cbur-master`

There are three volumes in `cbur-master`:

1. **/BACKUP:** used for application level backup and restore when the backend is not set to “local”.
Sizing recommendation: at least double the data size (also consider the situation when backing up multiple applications at the same time).
2. **/CBUR_REPO:** used for application level backup and restore when backend is set to “local”. Log files are saved in this volume.
Sizing recommendation: sum of all application storage where backend is set to “local”. Each application requires: $(\text{maxCopy} + 2) * \text{backup data size}$
3. **/CLUSTER:** used for cluster level backup (*only required for NCS environments*).

Sizing recommendations:

- If the endpoint is set to “local”: $(\text{rotation} + 2) * (\text{the size of local registry} + \text{the size of local Helm repo} + \text{the size of Helm home} + \text{the size of tiller configmaps / os tuning related crs / ncm users / hooks / Helm releases information})$

- If the endpoint is not set to "local": 2 * (the size of local registry + the size of local Helm repo + the size of Helm home + the size of tiller configmaps / os tuning related crs / ncm users / hooks / Helm releases information)

When designing the volumes size, consider the thresholds in `StorageMonitor`.

If they are enabled:

- when the storage usage exceeds high thresholds: the backup/restore will fail and trigger a critical alarm.
- when the storage usage exceeds low thresholds: the backup/restore will proceed and trigger a warning alarm.

The volumes can be defined using a direct GlusterFS volume (default option) or a dynamic PVC. The dynamic PVC can be GlusterFS volumes created by `storageClass`.

- In Virtualized environments, when using a direct GlusterFS volume, prepare the endpoints and volumes before proceeding with the installation. For example: `backupEndpoints`, `backupPath`, `repoEndpoints`, `repoPath`, `backupClusterEndpoints`, `backupClusterPath`.

In Virtualized environments, the default GlusterFS volumes are prepared in `/data0`. When designing the size of `/data0`, consider the following:

1. Base `/data0` size requirements.
2. The size of the three `cbur-master` volumes.
3. The space of `/tmp` in `cbura-sidecar` if using `/data0`.

- When using a dynamic PVC, use the fields `storageBackup`, `storageRepo`, `storageBackupCluster` and `storageClass` to define the volumes.

To perform backup and restore using an Avamar server, the direct GlusterFS volume or a **RWX** PVC must be used.

Volumes in `cbura-sidecar`

The `/tmp` volume in `cbura-sidecar` will contain both the tarball and uncompressed files, so the storage space should be larger than the sum of data volumes to backup. Ensure enough space has been allocated to this volume.

By default, `/tmp` in `cbura-sidecar` will use the node storage (`/data0` in BCMT environments).

The following workflows are included for a better understanding on `/tmp` usage.

- Backup workflow for application volume(s):
 1. For backup volume(s) specified in `brpolicy`, create a tarball in `/tmp` of `cbura-sidecar` of application pod.
 2. Copy the tarball to `/BACKUP` (if backend is not set to "local") or `/CBUR_REPO` (if backend is set to "local") in the `cbur` pod (`cbur-master` if celery "disabled", otherwise `cbur-master-cbur-mycelery`).

3. If the backend is not set to "local", copy the tarball from /BACKUP to the third party storage.

- Restore workflow for application volume(s):

1. If the backend is not set to "local", copy the tarball from the third party storage to the /BACKUP volume in the `cbur` pod.
2. In the `cbura-sidecar` of application pod, decompress into /tmp, then copy the files from /tmp to the backup volume(s).

Unique application Pod labels

When running application backup and restore, the `brpolicy` name is used to find the Deployment/StatefulSet/DaemonSet.

The `matchlabels` defined in Deployment/StatefulSet/DaemonSet is used to find pods, therefore the application Helm chart must satisfy the following requirements:

1. The `matchlabels` of the Deployment/StatefulSet/DaemonSet must be the same as the labels of pod(s).
2. The labels of the pod(s) must be unique, to ensure that only the correct pods are identified as belonging to this Deployment/StatefulSet/DaemonSet.

20.3.3 NCS installation

NCS pre-installs the backup and restore service using `cburScope`: "Cluster" in the "ncms" namespace. It can be upgraded with user defined settings.

1. The Containerized Backup and Restore tool volumes

The default setting is to use the GlusterFS volumes provided by NCS. These volumes reside within /data0, which is also used by containers. When /data0 is full, pods will get evicted.

To avoid using the predefined NCS GlusterFS volumes, use 1 of the following options:

- For GlusterFS volumes, specify the following parameters:

Example values:

```
volumeType:  
  glusterfs: true  
  backupEndpoints: "glusterfs-ep"  
  backupPath: "cbur-backup"  
  repoEndpoints: "glusterfs-ep"  
  repoPath: "cbur-repo"  
  backupClusterEndpoints: "glusterfs-ep"  
  backupClusterPath: "cbur-cluster"
```

- For PVCs, specify the following parameters:

Example values:

```

volumeType:
  glusterfs: false
  storageBackup: "2Gi"
  storageRepo: "8Gi"
  storageBackupCluster: "20Gi"
  # Set to the storageClass you want to use "kubectl get sc"
  # Or use the default storageClass by leave it empty
  storageClass: "xxx"

```

2. The Containerized backup and restore tool supports PSP creation**3. Namespace scoped the Containerized backup and restore tool**

Install backup and restore with the following settings:

```

cburScope: "Namespaced"
volumeType:
  glusterfs: false
  # Set to the storageClass you want to use "kubectl get sc"
  # Or use the default storageClass by leave it empty
  storageClass: ""
  storageBackup: "2Gi"
  storageRepo: "8Gi"
  storageBackupCluster: "20Gi"
  securityContext:
    enabled: true
    # Change it to the uid you want to use
    runAsUser: 2000
    # Change it to the gid you want to use
    fsGroup: 2000
  k8swatcher:
    clusterBrEnabled: false

```

Since NCS installs backup and restore using the "Cluster" scope by default, the BrPolicy CRD and BrHook CRD are added to the cluster.

20.3.4 Backup and Recovery Infrastructure in NCS

Backup and Recovery is a key asset to support backup and restore lifecycle event. It provides REST APIs which can be accessed within a cluster, details of these APIs are documented in Nokia Container Services (NCS) 20FP2SU2 Integration Guide.

In NCS, the Containerized Backup and Restore tool is installed in Control node as the *Cluster* level, and handles requests from the BCMT application API server. Nokia provides a Helm backup and restore plugin, which gives you the flexibility to invoke backup or restore by the Helm CLI.

As shown in the picture, the containerized Backup and Recovery consists of the following parts:

- **Containerized Backup and Restore tool-m:** the Containerized Backup and Restore tool master provides REST API interface and core backup/restore functionality. It can run any worker nodes and control nodes
- **Containerized Backup and Restore tool-a:** the Containerized Backup and Restore tool agent which acts as a sidecar container run in the application POD, it can be inserted/removed on demand
- **3rd party server client:** third party server client that communicates with third party svr. The pod is started when 3rd party backup (AVAMAR) is enabled in the Containerized Backup and Restore tool helm chart. The client must be run on control nodes since it needs to backup to a 3rd party server out of the cluster.
- **cbur-m-celery:** Celery is an asynchronous task queue/job queue. If celery is enabled, when cbur-m receives the API request, it passes the backup/restore task to celery and just returns status 200 with task ID. And celery will take the backup/restore job. The task result can be tracked with a curl command. Celery is set to disable as default value in CNF 18.10.

For application backup/restore:

- For Kubernetes deployments, the data is the same among all the pod replicates, so backup and restore is performed for only 1 of the pods.
- For Kubernetes statefulsets, multiple pods backup and restore are supported. This is defined in the application's BrPolicy.
- For Kubernetes daemonsets, multiple pods restore is supported. This is defined in the application's BrPolicy.

20.3.5 Back Up and Restore Process

20.3.5.1 BCMT Cluster Backup and Restore

1. *Introduction* on page 651
2. *Back up BCMT cluster* on page 652
3. *Recover BCMT Cluster in the event of disaster* on page 659
4. *Recover APPs* on page 660
5. *Recover APP PV Data* on page 664

20.3.5.1.1 Introduction

Containerized Backup and Restore support for NCS cluster backup/restore operations covers the following data:

- infrastructure configs used for creating a cluster You can invoke `<ncm config export>` manually to export the bcmt config and save it by yourself. When disaster happens, you can use this config to create a new cluster with the same configs as old cluster.
- local container registry
- local chart repo
- helm home (glusterfs volume bcmt-helm-home)
- helm release info
- tunables like kernel and docker Backup related custom resources for those CRDs.
- NCM users
- If `/opt/bcmt/storage/hooks/` exists, upon restore, the Containerized Backup and Restore tool restores the directory and invokes the hooks, *pre-install-app* before re-installing helm releases and *post-install-app* after re-installing helm releases.
- All Secrets from all namespaces
- All ConfigMaps from all namespaces

If `multi_tenant` is enabled in `cluster_config`. Backup related custom resources for multi-tenancy.

- Tenant Custom Resource(CR)
- TenantNamespace CR

The NCS cluster backup does not include application backup (it assumes that applications have been independently backed up). The NCS cluster restore executes a helm rollback to reinstall the application on the cluster and then automatically executes an application restore on each application using its latest backup archive.

 **Note:**

- All applications should support event "helm rollback" because Backup and Restore will use "helm rollback" to recover them.
- Per-release BrHooks will not be executed when applications are restored in the context of NCS cluster restore.

20.3.5.1.2 Back up BCMT cluster

Users can use ncm CLI to begin the backup cron job and choose a backup storage server for saving data.

 **Note:** Only to backup data to a remote storage server is meaningful for disaster recovery.
Currently, Avamar, ceph3, awss3, swifts3 and sftp are supported as the storage server.

To do cluster backup, below functions must be enabled:

- [Enable k8swatcher.enabled on page 653](#)
- [Set ncm endpoint and login ncm on page 654](#)

Related information

[Enable k8swatcher and third-party client together on page 653](#)

[Kubernetes cronjob backup on page 655](#)

To begin the backup as Kubernetes cronjob

[Disable a scheduled cluster backup on page 657](#)

To disable a scheduled cluster backup

[Begin the immediate cluster backup on page 657](#)

[Checking cbur-m container std output log to see if backup succeeds on page 658](#)

By checking cbur-m container std output log to see if backup succeeds

[Back up cbur-master self volume on page 658](#)

To backup cbur-master self volume

[Manual back up if using local backend on page 658](#)

Manual backup if using local backend (Optional)

20.3.5.1.2.1 Enable k8swatcher.enabled

k8swatcher.enabled must be set to true

You can check if this function is enabled by following methods:

- Use "helm get values -a cbur-master" to check all values settings when installing Backup and Restore. If k8swatcher.enabled is true, then this functionality is enabled.

```
# helm get values -a cbur-master
```

- Use "helm status cbur-master | grep k8swatcher" to check the status of cbur-master-cbur-k8swatcher pod. If this pod is running, then this functionality is enabled.

```
# helm status cbur-master | grep k8swatcher
```

20.3.5.1.2.2 Enable k8swatcher and third-party client together

If not enabled, do the following according to your requirements to enable k8swatcher.

- Only enable k8swatcher

```
helm upgrade cbur-master csf-helm-releases/cbur --version 1.3.4 --namespace ncms --reuse-values --set k8swatcher.enabled=true
```

- Then use "helm status cbur-master" to check if Backup and Recovery is ready (For example, following is the status of Avamar and k8swatcher).

```
# helm status cbur-master
```

LAST DEPLOYED: Thu Jan 3 01:58:17 2019

NAMESPACE: ncms

STATUS: DEPLOYED

RESOURCES:

==> v1/ClusterRoleBinding

NAME	AGE
cbur-master-cbur	11h

==> v1/Service

cbur-master-cbur	11h
------------------	-----

==> v1/Deployment

cbur-master-cbur	11h
cbur-master-cbur-k8swatcher	16m
cbur-master-cbur-avamar	16m

==> v1/Pod(related)

NAME	READY	STATUS	RESTARTS
AGE			
cbur-master-cbur-7d9f7ff47b-2nj6z	1/1	Running	0
cbur-master-cbur-k8swatcher-647644cbf4-b7969	1/1	Running	0
16m			
cbur-master-cbur-avamar-79df7f97db-dxwfz	1/1	Running	0
			2m

==> v1/Secret

NAME	AGE
cbur-ncm-secret	16m

==> v1/ServiceAccount

cbur-master-cbur	11h
------------------	-----

NOTES:

An API service which backup/restore sensitive data in volumes of k8s deployment or statefulset

Sevice Impact: May cause target pods restart twice if sidecar is not predefined.

20.3.5.1.2.3 Set ncm endpoint and login ncm

```
# ncm config set --endpoint https://<any one of control nodes' IP>:8082/ncm/
api/v1
# ncm user login --username="ncm-admin" --password="<your password>"
```

If you have not initialized ncm user and password, log in and reset password first.

20.3.5.1.2.4 Kubernetes cronjob backup

To begin the backup as Kubernetes cronjob

1. By ncm cli to set cron job schedule, backup storage method, max keeping copies and support multiple tillers.

```
NAME:
  ncm tool cluster backup - backup cluster

USAGE:
  ncm tool cluster backup [command options] [arguments...]

OPTIONS:
  --list      <boolean>      list the backup archives of an
  endpoint(optional,if value is true, will list the archives of an
  endpoint)
  --schedule   <schedule>     specifies the schedule for the backup; none
  for immediate backup, disable for disable backup
  --endpoint   <endpoint>     specifies the endpoint for the backup;value
  should be:local,avamar,cephs3,awss3,swifts3,sftp;(optional,default is
  local)
  --rotation    <rotation>     specifies the rotation of the backup;
  value[1-10] is valid(optional, default is 1)
  --override    <override>     specifies whether to override or not; only
  true or false is valid(optional,default is false)
```

2. Give an example:

```
# ncm tool cluster backup --schedule "0 23 * * *" --endpoint "sftp" --
rotation 1
"Very Important: Please manually save cluster config by command <ncm
config export>. This cluster config will be used to create a new cluster
if disaster recovery.
Finally, a k8s cronjob backup-cluster-by-cbur is created for scheduled
cluster backup.
```

```
And you can check the result in requestHistory in cluster BrPolicy by
<kubectl get brpolices.cbur.csf.nokia.com backup-cluster-by-cbur -n ncms
-o yaml>"
```

If NCS cluster backup is successful, the command will return a note about cluster deploying configs and created Kubernetes cronjob name.

3. Save cluster deploying configs by yourself in case cluster happens disaster and everything is lost.

```
root@server-name]# ncs config export
```

4. Check the "requestHistory" in brpolicy backup-cluster-by-cbur to see if Kubernetes cron job is created successfully.



Note: To list the brpolices, user should try both `kubectl -nncms get br` and `kubectl -nncms get brpolices.cbur.csf.nokia.com`

```
# kubectl get br -n ncms
NAME          AGE
backup-cluster-by-cbur   2m

# # kubectl get br backup-cluster-by-cbur -n ncms -o yaml
apiVersion: cbur.csf.nokia.com/v1
kind: BrPolicy
metadata:
  creationTimestamp: "2019-07-03T12:53:39Z"
  generation: 8
  name: backup-cluster-by-cbur
  namespace: ncms
  resourceVersion: "3409727"
  selfLink: /apis/cbur.csf.nokia.com/v1/namespaces/ncms;brpolices/backup-
cluster-by-cbur
  uid: 9477089d-9d91-11e9-a7c8-fa163e956783
spec:
  backend:
    mode: sftp
    cronSpec: '*/2 * * * *'
  cronjob_name: backup-cluster-by-cbur
  ignore: "true"
  k8sType: backupall
  maxiCopy: 1
  override: "true"
status:
  requestHistory:
    addCronjob:
    - "20190703125340":
```

```

apiVersion: v1
code: 200
details:
  name: backup-cluster-by-cbur
  namespace: ncms
kind: Status
message: Successfully to create cronjob backup-cluster-by-cbur
metadata: {}
reason: ""
status: Success

```

5. Check the cronjob details by using kubectl commands.

```

# kubectl get cronjob -n ncms
NAME                  SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE
AGE
backup-cluster-by-cbur 0 23 * * *  False       0          <none>
37s

# kubectl describe cronjob backup-cluster-by-cbur -n ncms

```

20.3.5.1.2.5 Disable a scheduled cluster backup

To disable a scheduled cluster backup

You can disable a scheduled cluster backup with the following command, the cronjob will be deleted.

```
ncs tool cluster backup --schedule 'disable'
```

20.3.5.1.2.6 Begin the immediate cluster backup

An immediate cluster backup can be triggered by the following command:

```
# ncm tool cluster backup --schedule 'none' --endpoint <ENDPOINT>
```

The output should be as follows, or a timeout when Celery is off.

```

# ncm tool cluster backup --schedule 'none' --endpoint local
{"result":"Very Important: Please manually save cluster config by command <ncm config export>.
This cluster config will be used to create a new cluster if disaster recovery.
backupCluster task = f6edb4f8-ae07-4301-a18c-6c1741b1871e is on!
And you can check the result in requestHistory in cluster BrPolicy by
<kubectl get brpolices.cbur.csf.nokia.com backup-cluster-by-cbur -n ncms -o yaml>"}

```

The immediate backup will not create BrPolicy if not exist, so request history will not be recorded in BrPolicy. Check the request result from the cbur-m pod standard output log.

20.3.5.1.2.7 Checking cbur-m container std output log to see if backup succeeds

By checking cbur-m container std output log to see if backup succeeds

The cluster backup request result will be put in BrPolicy backup-cluster-by-cbur and you can check the "requestHistory" to see the result of backupCluster.

 **Note:** There are at most 20 result records for each request in BrPolicy "requestHistory". If exceeding 20 records, CBUR will remove the oldest one from requestHistory. Backup cluster will take time, wait for the request result if celery is disabled.

After scheduled time, you can monitor the logs of cbur-master to see if backup succeeds.

```
# kubectl logs -f cbur-master-cbur-7d9f7ff47b-2nj6z -n ncms
```

 **Note:** For Avamar, use the Avamar server GUI to backup data from client before next scheduled job happens. Otherwise, the data will be deleted in next scheduled job. During the restoration process, use Avamar server GUI to restore the data first, then use ncm CLI to restore app data.

20.3.5.1.2.8 Back up cbur-master self volume

To backup cbur-master self volume

If you are using Local as backend storage, backup the CBUR after the cluster backup is complete.

For Local backend, edit CBUR's BrPolicy to set a remote storage backend and add "cbur-repo" to volumes since everything is lost in a disaster.

Refer to section *Backup or Restore for cbur-master self volume* on page 694 for details.

20.3.5.1.2.9 Manual back up if using local backend

Manual backup if using local backend (Optional)

For Local backend, the data is saved locally in Backup and Restore volumes. To support disaster recovery, you can choose to manually backup local cluster data and CBUR's volume data to your own storage server.

For cluster data, when you set cluster backup endpoint to be "local", manually copy all files under following directory to your own storage server.

- In controller node: /data0/glusterfs-backup-cluster
- In cbur-master pod: /CLUSTER

For CBUR's volume data, manually copy all files under following directory to your own storage server.

- In controller node: /data0/glusterfs-cbur-repo/repo/
- In cbur-master pod: /CBUR_REPO



Note: All files under the directory means "not contain the directory itself".

20.3.5.1.3 Recover BCMT Cluster in the event of disaster

1. Re-create BCMT cluster: After disaster happens, you can create the BCMT cluster again by your saved deployment configs.



Note: If you have saved configs for example, CLCM's user_input.yml, BCMT's bcmt_config.json by yourself directly re-install BCMT cluster by BCMT Installation Guides and ignore this section.



Note: If `embedded_clcm = false`, then no clcm related config and you must prepare CLCM user_input.yml again to re-create the cluster infrastructure.

- a. If `embedded_clcm = false`, then the saved configs can be used for BCMT Installation Guides to deploy BCMT.
- b. If `embedded_clcm = true`, then the saved configs includes all cluster deployment related configs.
To create a new BCMT cluster can directly use "`ncm cluster install --config=`"
 - i. Copy bcmt_config.json to your deploy server's directory /opt/bcmt as usual
 - ii. Login deploy server
 - iii. Login BCMT admin container
 - iv. Directly install BCMT:

```
ncm cluster install --config=/opt/bcmt/bcmt_config.json
```



Note: If you want to keep the IPs of cluster nodes, create your BCMT cluster nodes with static IPs.

2. Recover Backup and Restore manually:

- a. Enable Backup and Restore DR function again
- b. If Backup and Restore has been upgraded to a newer version before disaster, upgrade again, including related helm plugins
- c. If Backup and Restore was customized by Helm set values before disaster, re-set it. If you edit cbur's own brpolicy manually, edit it again
- d. If using ceph3/awss3/swift3, all the secret key/access key info needed to access S3 must be recovered manually
- e. If using "Local" backend before disaster, manually restore cluster data and cbur's volume data to your own storage server.
 - For cluster data, when you set cluster backup endpoint to be "local" before disaster, manually copy backed up files from your own storage server to following directory.

- In cbur-master pod: /CLUSTER.
- Or manually mount glusterfs volume cbur-glusterfs-backup-cluster to a directory in controller node and after copying, unmount it.
- For cbur's volume data, manually copy all files under following directory to your own storage server.
- It is /CBUR_REPO in cbur-master pod (or cbur-master-cbur-mycelery pod if enabling celery).
- Or manually mount glusterfs volume cbur-glusterfs-repo to a directory in controller node and after copying, unmount it.

Example:

```
# mkdir /home/test
# gluster volume info cbur-glusterfs-repo
Volume Name: cbur-glusterfs-repo
Type: Replicate
Volume ID: f27f1317-a181-4514-8913-986158398df3
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: 192.168.168.17:/data0/glusterfs-cbur-repo
Brick2: 192.168.168.12:/data0/glusterfs-cbur-repo
Brick3: 192.168.168.15:/data0/glusterfs-cbur-repo
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off

# mount -t glusterfs 192.168.168.15:cbur-glusterfs-repo /home/test

##After restoring files
# umount /home/test
# rm -rf /home/test
```

20.3.5.1.4 Recover APPs

Cluster Restore restores the cluster backup data, the restore sequence is as follows:

- NCM Users ETCD data
- Cluster local data, include:

```
1. local container registry(/opt/bcmt/storage/docker-registry/)  
2. Helm repo(/opt/bcmt/storage/charts/)  
3. Helm home(/root/.helm/)  
4. NCM hooks(/opt/bcmt/storage/hooks/)  
5. NCM Users keycloak data(/opt/bcmt/storage/keycloak/)  
  
- tunables like kernel and docker.  
  Related custom resources for those crds will be re-apply if exist in backup  
  data.  
  
- Applications  
  1. helm release info  
  2. application data (if backup data exist and recorded in Backup History in  
  BrPolicy.)
```



Notice: For Multi-Tillers case, ensure other Tillers except kube-system are available before restore application data for these Tillers.

- From 19.06, APPs' BrPolicy can define the Kubernetes objects to backup including Secrets and ConfigMaps.
- From 19.06.1, when backup cluster, Backup and Restore backups all Secrets and ConfigMaps from all namespaces in BCMT cluster.

There are following kinds of Secrets / ConfigMaps.

- App's own Secrets / ConfigMaps defined in its own helm chart.
- App's Secrets / ConfigMaps but not defined in its own helm chart.
- Cluster level or namespace level Secrets / ConfigMaps like service account related token secrets.
Multiple apps may use them and may not be aware of their existence.

Backup and Restore recommends that for first 2 kinds of Secrets / ConfigMaps, all APPs should backup them by using their **BrPolicy** not by cluster backup.

The function of cluster level backup of Secrets and ConfigMaps is mainly for the third one - cluster level or namespace level Secrets / ConfigMaps which can be called **Standalone Secrets / ConfigMaps**.

From 19.06.1, the cluster backup will backup all Secrets and ConfigMaps from all namespaces in BCMT cluster including all of above 3 kinds, there may be some conflicts when restoring the Secrets / ConfigMaps from both APP level data and cluster level data during disaster recovery.

If you are not aware of these details, the following is the recommended procedure for disaster recovery.

Apps' Secrets / ConfigMaps mean that an app can backup its Secrets / ConfigMaps in its BrPolicy. And the standalone Secrets / ConfigMaps mean the third kind of Secrets / ConfigMaps.

Pre-requisites: An APP's BrPolicy defines the Secrets / ConfigMaps to backup and also it is using the Containerized Backup and Restore tool 19.06.1, following are the restoring scenarios.

**Note:**

- To restore, apps will automatically restore the latest app-level backup data including apps' Secrets / ConfigMaps. But if an app already exists, cluster level restore will not help restore it. Because cluster level Secrets / ConfigMaps restore will only patch existing ones, app should first be re-installed with its objects re-created before restoring its backup data.
- If no Standalone Secrets / ConfigMaps, ignore them.

1. NCM CLI for cluster restore

```
# ncm tool cluster restore --help
NAME:
  ncm tool cluster restore - restore cluster

USAGE:
  ncm tool cluster restore [command options] [arguments...]

OPTIONS:
  --endpoint          <endpoint>      specifies
  the endpoint for the restore; value should
  be:local,avamar,cephs3,awss3,swifts3,sftp(optional, default is local)
  --app               <app>           specifies the apps for
  restore, valid values(all, apps split by comma),(optional, default is
  None)
  --configmap         <configmap>     specifies the configmap for
  restore, valid values(all, configmaps split by comma),(optional, default
  is None)
  --secret            <secret>         specifies the secret for
  restore, valid values(all, configmaps split by comma),(optional, default
  is None)
  --force             <force>          specifies whether or not force to
  restore, only true or false is valid(optional, default is false)
  --namespace          <namespace>       specifies namespace for the
  restore(optional, default is None)
  --tiller_namespace   <namespace>       specifies tiller namespace
  for the restpre, if there are multiple tiller_namespaces, split with
  comma.Example:namespacel,namespace2;(optional, default is kube-system)
  --all                <all>            specifies whether to restore all
  or not; only true or false is valid(optional, default is false)
  --id                <id>             specifies the restore id; restore
  id can be listed by 'ncm tool cluster'
```

a. Restore Cluster with the latest data

Trigger Cluster Restore by below command, which will restore the latest backup. The "ENDPOINT" is the endpoint of your backup data, it should be local, avamar, cephss3, awss3, swifts3, sftp.

```
# ncm tool cluster restore --endpoint <ENDPOINT>
```

b. Restore cluster with a specific backup id

- i. Get the backup ID list for a specific backup endpoint by following command

```
ncm tool cluster backup --list true --endpoint <ENDPOINT>
```

- ii. Restore cluster with the ID you want

```
ncm tool cluster restore --id <BACKUP ID> --endpoint <ENDPOINT>
```

2. Check the restore-cluster-by-cbur if restore cluster successfully.

```
# kubectl get brpolices.cbur.csf.nokia.com restore-cluster-by-cbur -n ncms
-o yaml
apiVersion: cbur.csf.nokia.com/v1
kind: BrPolicy
metadata:
  creationTimestamp: "2019-07-03T03:07:14Z"
  generation: 2
  name: restore-cluster-by-cbur
  namespace: ncms
  resourceVersion: "39997"
  selfLink: /apis/cbur.csf.nokia.com/v1/namespaces/ncms;brpolices/restore-
cluster-by-cbur
  uid: a88289d5-9d3f-11e9-ba29-fa163e70f964
spec:
  backend:
    mode: cephss3
    ignore: "true"
    k8sType: restoreCluster
status:
  requestHistory:
    restoreCluster:
    - "20190703031730":
      apiVersion: v1
      code: 200
      details:
        name: restore-cluster-by-cbur
        namespace: ncms
        kind: Status
      message: Successfully to restore current cluster.
```

```

    metadata: {}
    reason: ""
    status: Success

```

3. Restart the Containerized Backup and Restore tool master Pod and k8swatcher Pod.

Due to the authentication cache in the Containerized Backup and Restore tool master and k8swatcher pod, restart them to clear the cache when authentication enabled, for example:

```

kubectl delete pod/cbur-master-cbur-748d766b5-67t2x -nncms
kubectl delete pod/cbur-master-cbur-k8swatcher-7565c75884-kfgn2 -nncms

```

Then wait about 1 minute to ensure the Containerized Backup and Restore tool master and k8swatcher pod restart automatically.

20.3.5.1.5 Recover APP PV Data

If you want to recover your APP's pv data, Backup and Restore should backup itself to a remote server. Refer to section *Backup or Restore for cbur-master self volume* on page 694 to recover cbur-master. Use Helm restore to recover each APP's PV data.

20.3.5.2 Backup or Restore with Avamar Server

Install or Upgrade

If no CBUR chart is installed, execute following command:

```

helm install csf-helm-releases/cbur --version 1.3.5 -n cbur-master --namespace
ncms --set avamar.enabled=true,avamar.server=10.75.53.220,
avamar.clientName=<user_defined_name>,avamar.domain=ARC,
control_node_ip_list="10.x.x.x 10.x.x.x 10.x.x.x"

```

If CBUR installed with an old version, execute following command:

```

helm upgrade cbur-master csf-helm-releases/cbur --version 1.3.5 --namespace
ncms --set avamar.enabled=true,avamar.server=10.75.53.220,
avamar.clientName=<user_defined_name>,avamar.domain=ARC,
control_node_ip_list="10.x.x.x 10.x.x.x 10.x.x.x"

```

 **Important:** The following values must be set correctly, or Avamar backup/restore may fail

avamar.enabled: must be set to "true"

avamar.server: IP address of Avamar server, AKA, MSC IP address. For example, 10.40.32.23 which is owned/maintained by NetAct

avamar.clientName: user defined client name, or use the BCMT control node name prefix by default, optional parameter.

avamar.domain: Avamar server domain name where Avamar client will register to

control_node_ip_list: External OAM IP list of bcmt control nodes which can be directly reached from avamar server. Although It can be empty, if it is empty and Kubernetes reallocates the avamar pod to another Control node, avmar client would not be able to re-register to Avamar server again and following backup/restore will fail.

You must use a RWX volume:

- If using direct GlusterFS volume, **volumeType.glusterfs** must be set to "true".
- If using dynamic PVC, RWX PVC must be used and **isPvRwx** must be set to "true", **volumeType.glusterfs** must be set to "false".

hostNetwork.avamar: Must set to "false" when "istio.enabled=true". Otherwise, istio-proxy sidecar cannot be injected to Avamar pod.

There are the following side effects when "hostNetwork.avamar=false":

- In Avamar GUI, you can see the backup/restore result, you cannot see the detailed logs for a specific backup/restore. You can check the logs in Backup and Recovery Avamar pod (dir: /var/avamar/clientlogs).
- The backup/restore work may start a little late. In Avamar lab-6 and lab-7, it is less than 60 seconds.

The following command could change the Avamar client setting, then the Avamar client pod will restart and re-register to Avamar server.

```
kubectl edit deployment cbur-master-cbur-avamar -nncms
```

 **Note:** You should use the ARC-FP6 related Nokia Archive Cloud lab for arccli because of BRCLOUD-3266.

 **Important:** You should check the MTU of the network interface by which avamar client accesses the server. Because if the MTU is inconsistent between Avamar server and client, some unexpected errors may happen. You should consult Avamar server administrator for the right MTU size.

Example:

The following example uses MTU 1500, the following steps should be done in all controller nodes.

1. To find related network interface (give ethx as example) whose IP is in **control_node_ip_list** used to access Avamar server
2. To check the MTU status of this network interface ethx:

```
$ ip addr or $ netstat -i
```

3. To apply the setting temporarily:

```
$ ifconfig mtu 1500
```

4. To change it permanently:

```
$ vim /etc/sysconfig/network-scripts/ifcfg-ethx
```

Add the following content:

MTU=1500

5. Then restart the network:

```
$ systemctl restart network
```

Check Avamar Client Registration

- Log in to the ARC Management Node via SSH on port 7722 as the brc user. Execute:

```
[user@somenode~]# ssh brc@<MANAGEMENT_NODE_IP> -p 7722
```

- Provide a password for brc user..
- Listing all Clients

After install/upgrade successfully, you should be able to find the registered client from ARC management node with user defined client name or use the BCMT control node name prefix by default. For example, BCMT control node name is 'mybcmtnode-g01-c-01/02/03', then the client registered on Aamvar is "mybcmtnode-g01-c". So it is important that you have unique name for different cluster.

Here the client name is tw_test which is activated and enabled.

```
[brc@arc-mgmt-cl ~]$ arccli client list
2020-03-12 03:40:46,600 INFO      Preparing clients list
+-----+-----+-----+
| Host          | Activated | Enabled |
+-----+-----+-----+
| 10.58.164.194 | Yes       | Yes     |
| 10.58.164.196 | Yes       | Yes     |
| 10.58.164.197 | Yes       | Yes     |
| 10.58.164.198 | Yes       | Yes     |
| amokfutas117-infra | Yes       | Yes     |
| kejittas62      | Yes       | Yes     |
| kejittas62-infra | Yes       | Yes     |
| twcburvm        | Yes       | Yes     |
| tw_test          | Yes       | Yes     |
+-----+-----+-----+
```

For listing all clients, the following headers are visible:

- * Host is a hostname of the Client
- * Activated is the status of Client activation: Yes or No
- * Enabled is the status of Client to back up files: Yes or No

Backup configuration

To add a new backup configuration, a file with Client's backup configuration must be prepared. The file must be in JSON format and contain details about:

- name
- paths to backup
- backup schedule
- data retention time

The following example shows a backup configuration file content:

```
{
  "backupConfigurationIdentifier" : {
    "clientType" : "NE",
    "clientVersion" : "18.2",
    "configurationName" : "CCBUR"
  },
  "backupDefinitions" : [ {
    "name" : "ALL",
    "description" : "Configuration files",
    "dataset" : {
      "paths" : [ "/BACKUP/default/DEPLOYMENT_cbura-cbur/" ],
      "preBackupScript" : "pre-backup.sh -t cbura-cbur",
      "postBackupScript" : "clean.sh -t cbura-cbur",
      "type" : "FILESYSTEM"
    },
    "schedule" : {
      "times" : [ "11:30" ],
      "maxRunHours" : 3,
      "type" : "DAILY"
    },
    "retention" : {
      "duration" : 7,
      "unit" : "DAYS"
    }
  }
]
}
```

According to this example:

- Backup Definition is called "ALL".
- This Backup Definition will have the cbura-cbur backed up every day at 11:30 hour, and data stored for 7 days.

1. Backup Configuration properties

The following table presents the possible values for the backupConfigurationIdentifier category:

Path	Type	Description	Constraints
backupConfigurationIdentifier	Object	Identifies given Backup Configuration	Required
backupDefinitions	Array	The list of Backup Definitions	Required

2. Backup Configuration Identifier properties

Backup Configuration Identifier identifies given Backup Configuration. The following table presents the Backup Configuration Identifier properties:

Path	Type	Description	Regular expression	Required
clientType	String	Type of the client given backup configuration is targeted for.	^[a-zA-Z0-9.-]+\$	Yes
clientVersion	String	Version of the client given backup configuration is targeted for.	^[a-zA-Z0-9.-]+\$	Yes
configurationName	String	Name uniquely identifying the backup configuration for targeted client type and version.	^[a-zA-Z0-9.-]+\$	Yes



Note: Cumulative length of clientType, clientVersion, configurationName and name of Backup Definition cannot exceed 45 characters.

3. Backup Definition properties

Backup Definition contains details such as data for backup, when backup runs and how long backed up data is stored. It is possible to have multiple Backup Definitions in each Backup Configuration.

The following table presents the backup definition properties:

Path	Type	Description	Constraints
name	String	Name of the backup definition. The name must be unique among all backup definitions for given backup configuration.	Required. Must match the regular expression: ^[a-zA-Z0-9.-]+\$
description	String	A description for the Backup Definition.	Optional
dataset	Object	Defines backup data, preand post-backup scripts.	Required
schedule	Object	Defines when backup runs.	Required
retention	Object	Defines how long backup should be kept.	Required

- **Dataset settings** Dataset defines what needs to be backed up. the Containerized Backup and Restore tool use Filesystem Dataset. The possible attributes are presented in the following table:

Path	Type	Description	Constraints
paths	Array	1. For backup cburm self volume, then the paths in dataset should be "/CBUR_REPO/data and /CBUR_REPO/record"	Required

Path	Type	Description	Constraints
		<p>. If the volume is only cbur-repo, then it is not necessary to provision pre-backup and post restore script.</p> <p>To backup volume that attach to cbur-m, then needs to run pre-backup.sh and post-restore.sh script for cbur-master.</p> <p>2. For application backup/restore, input paths in format of "/BACKUP/[NAMESPACE]/[DEPLOYMENT STATEFULSET DAEMONSET K8SOBJECTS PVC]_[brpolicy_name]/".</p> <ul style="list-style-type: none"> – To backup volumes, the name of daemonset/deployment/statefulset should be same with the Br Policy. <p>Here, Kubernetes deployment name is cbura-cbur, which is in default namespace, so the path is "/BACKUP/default/DEPLOYMENT_cbur-cbur/".</p> <p>If you want to backup your statefulset sdc-sdc in namespace: ncms, the path would be "/BACKUP/ncms/STATEFULSET_sdc-sdc/".</p> <ul style="list-style-type: none"> – If the application to backup is none of Deployment / StatefulSet / DaemonSet: <ol style="list-style-type: none"> a. If 'k8sobjects' is defined in BrPolicy, the path is "/BACKUP/[NAMESPACE]/K8SOBJECTS_[brpolicy_name]/"; b. If only 'pvc' is defined in BrPolicy, the path is "/BACKUP/[NAMESPACE]/PVC_[brpolicy_name]/" <p>If you use release scripts to backup/restore the whole helm release, input paths for all the Br Policies in the helm release according to the above format.</p> <p>For example:</p> <pre>"paths": ["/BACKUP/foo/STATEFULSET_foo-jk-crdbredisio-server/", "/BACKUP/foo/STATEFULSET_foo-jk-foo/"]**</pre>	
pre Backup Script	String	<p>Name of the script to run before backup.</p> <p>1. To backup one BrPolicy pre-backup.sh -t</p> <p>Here is: pre-backup.sh -t cbura-cbur</p>	Required

Path	Type	Description	Constraints
		2. To backup one helm release release-pre-backup.sh -t	
post Backup Script	String	<p>Name of the script to run after backup.</p> <p>1. To backup one BrPolicyclean.sh -t Here is:clean.sh -t cbura</p> <p>2. To Backup one helm release release-clean.sh -t</p>	Required
type	String	<p>Type of backed up data.</p> <p>Must be: FILESYSTEM</p>	Required

preBackupScript for backup one BrPolicy

This command backs up a BrPolicy.

Usage:

```
pre-backup.sh -t BrPolicy [flags]
```

Flags:

- c timer counter, wait [counter]*5s for backup to complete, default is 120, which means 10 minutes. It only makes sense when celery is enable on cbur master
- n namespace, use default namespace by default
- H cburm service name/IP. If not set, use default cbur service name. usually, pls don't set it, unless you have another cburm service available rather than default.
- U specify which login username to be verified when auth enabled (option)
- P specify which login password to be verified when auth enabled (option)
- m, --max-attempts The max attempts to run backup in case of failure (Default: 1, i.e. will not retry)
- i, --retry-interval Time in seconds between each retry (default 60)

postBackupScript for backup one BrPolicy

```
clean.sh -t BrPolicy [-n namespace]
```

preBackupScript for backup one Helm release

This command backs up a helm release.

Usage:

```
release-pre-backup.sh -t HELM_RELEASE [flags]
```

Flags:

```
-t HELM_RELEASE      Specify the helm release name to backup.  
-n NAMESPACE        Specify the namespace of this helm release. (Only  
support helm version 3)  
-l, --tiller-namespace TILLER_NAMESPACE  
                      Specify the tiller_namespace of this helm release.  
Default value is "kube-system". (Only support helm version 2)  
-v HELM_VERSION     Only support 2 (helm version 2) and 3 (helm verison 3).  
                      When "-v" is not specified,  
                      - if "--tiller-namespace" is specified, the  
Containerized Backup and Restore tool will process it as helm version 2;  
                      - otherwise if "-n" is specified, the Containerized  
Backup and Restore tool will process it as helm version 3;  
                      - otherwise as helm version 2.  
-U CBUR_USERNAME    Specify the username of the Containerized Backup and  
Restore tool when auth is enabled  
-P CBUR_PASSWORD    Specify the password of the Containerized Backup and  
Restore tool when auth is enabled  
-m, --max-attempts  The max attempts to run backup in case of failure  
(Default: 1, i.e. will not retry)  
-i, --retry-interval Time in seconds between each retry (default 60)  
-T, --timeout DURATION When celery on, how long to wait for the backup  
task to complete (default: 1h0m0s).  
                      Please note, it is for one backup attempt, the timer  
will be resumed for each retry.  
                      The format can be:  
                      1) <d+>, time in seconds, e.g. 600 (600 seconds)  
                      2) <d+>h<d+>m<d+>s, e.g. 2h (2 hours), 2h30m (2 hours  
30 mins), 30s (30 seconds)  
-h/--help            Display this help text and exit.
```

Example:

```
release-pre-backup.sh -t csdc -n ns1 # helm version 3  
release-pre-backup.sh -t csdc -l app_tiller_ns # helm version 2  
release-pre-backup.sh -t csdc --tiller-namespace app_tiller_ns # helm  
version 2  
release-pre-backup.sh -t csdc -v 2 # helm version 2  
# If backup fails, only retry once after 120 seconds. And will wait at most  
2 hours for each try to complete.  
release-pre-backup.sh -t csdc -m 2 -i 120 -T 2h  
release-pre-backup.sh -t csdc --max-attempts 2 --retry-interval 120 --  
timeout 2h  
release-pre-backup.sh -h
```

postBackupScript for backup one helm release

This command cleans up the directory in cbur which is to save the backup data for the helm release.

Usage:

```
release-clean.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to clean up.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified,
	- if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2;
	- otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3;
	- otherwise as helm version 2.
-h/-help	Display this help text and exit.

Example:

```
release-clean.sh -t csdc -n ns1 # helm version 3
release-clean.sh -t csdc --tiller-namespace app_tiller_ns # helm version 2
release-clean.sh -h
```

- **Schedule settings**

The schedule defines when backup runs. There are several schedule types supported:

- Daily Schedule
- Weekly Schedule
- Monthly Schedule:
 - Monthly Schedule for specific day of month
 - Monthly Schedule for specific day of specific week of month

Daily Schedule

The daily schedule properties are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: DAILY
times	Array	Times of day in HH:MM format when backups shall start. Interval between backup start times must be greater or equal to maximum backup execution time limit specified by max RunHours property.	Required
maxRunHours	Integer	Maximum backup execution time in hours. This parameter is taken into account after a first successful backup. The first execution can be longer.	Required Integer between 1 and 23 inclusive

The following example shows the usage of the daily schedules properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
    }
  ]
}
```

```

    "schedule": {
        "type": "DAILY",
        "times": [
            "01:00",
            "16:00"
        ],
        "maxRunHours": 1
    },
    "retention": {
        "duration": 10,
        "unit": "WEEKS"
    }
}
]
}

```

Weekly Schedule

The possible properties of weekly schedule are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: WEEKLY
daysOfWeek	Array	Days of the week when backups runs.	Required. Size must be between 1 and 7 inclusive. The allowed values are: <ul style="list-style-type: none"> • SUNDAY • MONDAY • TUESDAY • WEDNESDAY • THURSDAY • FRIDAY • SATURDAY
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish. This parameter is taken into account after a first successful backup. The first execution can be longer.	Required The format is HH:MM

The following example shows the usage of the weekly schedules properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
      "schedule": {
        "type": "WEEKLY",
        "daysOfWeek": [
          "SATURDAY",
          "SUNDAY"
        ],
        "runAfter": "00:00",
        "runBefore": "02:30"
      },
      "retention": {
        "duration": 10,
        "unit": "WEEKS"
      }
    }
  ]
}
```

Monthly Schedule

Monthly Schedule for specific day of month:

This type of schedule allows configuring backups to be run on specific day of each month. The possible properties of monthly schedule for specific day of month are presented in following table:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: MONTHLY
dayOfMonth	String	Specifies the day of month when backup is executed.	Required Range between values 1 and 28 inclusive or LAST
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish.	Required. The format is HH:MM

The following example shows the usage of monthly schedule for specific day of month properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "name": "ALL",
      "description": "Configuration files",
      "dataset": {
        "paths": [
          "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
      },
      "schedule": {
        "type": "MONTHLY",
        "dayOfMonth": "LAST",
        "runAfter": "00:00",
        "runBefore": "02:30"
      },
      "retention": {
        "duration": 7,
        "count": 1
      }
    }
  ]
}
```

```

        "unit": "DAYS"
    }
}
]
}
```

Monthly Schedule for specific day of specific week of month:

Path	Type	Description	Constraints
type	String	Type of the schedule.	Must be: MONTHLY_ON_SPECIFIC_WEEKDAY
dayOfMonth	String	The week of the month when backup runs.	Required The allowed values are: • FIRST • SECOND • THIRD • LAST
dayOfWeek	String	The day of the specified week of month when backup runs.	Required The allowed values are: • SUNDAY • MONDAY • TUESDAY • WEDNESDAY • THURSDAY • FRIDAY • SATURDAY
runAfter	String	Time after which backup starts.	Required The format is HH:MM
runBefore	String	Time before which backup should finish.	Required The format is HH:MM

The following example shows the usage of the monthly schedules for specific day of specific week of month properties:

```
{
  "backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
  },
  "backupDefinitions": [
    {
      "unit": "DAYS"
    }
  ]
}
```

```

    "name": "ALL",
    "description": "Configuration files",
    "dataset": {
        "paths": [
            "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
    },
    "schedule": {
        "type": "MONTHLY_ON_SPECIFIC_WEEKDAY",
        "weekOfMonth": "FIRST",
        "dayOfWeek": "WEDNESDAY",
        "runAfter": "00:00",
        "runBefore": "02:30"
    },
    "retention": {
        "duration": 7,
        "unit": "DAYS"
    }
}
]
}

```

Retention settings

The retention defines how long a backup must be kept. The possible properties are presented in the following table:

Path	Type	Description	Constraints
duration	Number	The duration backup should be kept.	Required Integer, minimum value is 1
unit	String	Time unit retention duration is expressed with.	Required The allowed values are: • DAYS • WEEKS • MONTHS • YEARS

The following example shows the usage of the retention schedules properties.

```
{
}
```

```

"backupConfigurationIdentifier": {
    "clientType": "NE",
    "clientVersion": "18.2",
    "configurationName": "CCBUR"
},
"backupDefinitions": [
{
    "name": "ALL",
    "description": "Configuration files",
    "dataset": {
        "paths": [
            "/BACKUP/default/DEPLOYMENT_cbura-cbur/"
        ],
        "preBackupScript": "pre-backup.sh -t cbura-cbur",
        "postBackupScript": "clean.sh -t cbura-cbur",
        "type": "FILESYSTEM"
    },
    "schedule": {
        "type": "MONTHLY_ON_SPECIFIC_WEEKDAY",
        "weekOfMonth": "FIRST",
        "dayOfWeek": "WEDNESDAY",
        "runAfter": "00:00",
        "runBefore": "02:30"
    },
    "retention": {
        "duration": 10,
        "unit": "WEEKS"
    }
}
]
}

```

Client management

- 1. Adding Backup Configuration** To add Backup Configuration, execute the arccli backup-configuration create command:

```
[brc@arc-mgmt-cli~]$ arccli backup-configuration create --file
<PATH_TO_BACKUP_CONF_FILE>
```

where: is a path to file with prepared Backup Configuration. Example:

```
[brc@arc-mgmt-cli ~]$ arccli backup-configuration create -f ./twcbur.json
2020-02-27 10:10:36,241 INFO      Adding backup definition:
NE_18.2_CCBUR_ALL
```

```

2020-02-27 10:10:36,242 INFO Adding schedule with name:
NE_18.2_CCBUR_ALL
2020-02-27 10:10:47,432 INFO Adding retention with name:
NE_18.2_CCBUR_ALL
2020-02-27 10:10:54,793 INFO Adding dataset: NE_18.2_CCBUR_ALL
2020-02-27 10:11:04,632 INFO Adding dataset: NE_18.2_CCBUR_ALL targets
2020-02-27 10:11:10,360 INFO Adding dataset: NE_18.2_CCBUR_ALL options
2020-02-27 10:11:27,226 INFO Adding group: NE_18.2_CCBUR_ALL
2020-02-27 10:11:34,337 INFO Backup definition: NE_18.2_CCBUR_ALL has
been added
2020-02-27 10:11:34,338 INFO Backup configuration: NE_18.2_CCBUR has
been added

```

2. Adding Client to Backup Configuration

Prerequisites:

- Client is integrated with Backup Engine Node.
- Backup Configuration exists.
- Client is not integrated with any Backup Configuration. To add Client to existing Backup Configuration, execute command, where: is a hostname of the Client to be added is a Backup Configuration Identifier.

Example:

```

[brc@arc-mgmt-cli ~]$ arccli backup-configuration add-client -c tw_test
NE_18.2_CCBUR
2020-02-27 10:13:28,688 INFO Getting tw_test client details
2020-02-27 10:13:36,230 INFO Attaching client: tw_test to backup
configuration: NE_18.2_CCBUR
2020-02-27 10:13:39,802 INFO Adding client: tw_test to backup
definition: NE_18.2_CCBUR_ALL
2020-02-27 10:13:46,050 INFO Client: tw_test has been added to backup
configuration: NE_18.2_CCBUR

```

3. Showing Client details

To show details about single Client integrated to Backup Engine Node, execute the arccli client show command specifying client hostname, as shown below:

```
[brc@arc-mgmt-cli ~]$ arccli client show <CLIENT_HOSTNAME>
```

Example:

```
[brc@arc-mgmt-cli ~]$ arccli client show tw_test
2020-03-12 04:00:24,696 INFO Displaying client: tw_test details
```

Host	Client Type	Client Version	Configuration Name	Backup Definitions	Activated	Enabled
tw_test	NE	18.2	NE_18.2_CCBUR	NE_18.2_CCBUR_ALL	Yes	Yes

For listing single Client details, the following headers are visible:

- Host is client name
- Client Type is the type of network element
- Client Version specifies network element software release
- Configuration Name is a Backup Configuration name
- Backup Definitions are Backup Definition IDs
- State is the status of Client: Activated or Not activated

Perform Backup with ARCLI

1. Configure APP/chart brpolicy

Set **backend mode to "Avamar"** in brpolicy file of target app in helm chart. cburn will base on this customized setting performing different backup/restore procedure for different target app/charts existing in the same cluster.

2. Triggering on demand backup

Backup starts automatically at time determined in the Backup Definition Schedule. It is possible to trigger backup on demand by using the arccli command. On demand backup can be triggered for all Backup Definitions of Client's Backup Configuration at once or for a selected one.

where: is name of the integrated Client.

The backup on demand starts and the following log information is visible in the console:

```
[brc@arc-mgmt-cli ~]$ arccli backup run -c tw_test
2020-03-09 10:05:32,412 INFO      Starting backup on demand for client:
tw_test
2020-03-09 10:05:52,068 INFO      Client tw_test backup scheduled for
backup definition: NE_18.2_CCBUR_ALL
```

3. Listing backups

Listing backups allows listing available backups for specified Client in a table format. On the list, both scheduled and on demand backups are visible.

To list backups, execute the arccli backup list command by specifying Client hostname in the following way:

```
[brc@managementnode-cli ~]$ arccli backup list -c <CLIENT_HOSTNAME>
```

where: is a hostname of the Client for which backups will be listed.

Example:

The table with details of available backups is visible, as follows:

```
[brc@arc-mgmt-cli ~]$ arccli backup list -c tw_test
2020-03-06 09:52:22,306 INFO      Displaying client tw_test backups
+-----+-----+-----+
+-----+-----+-----+
| Backup Definition | Label Number | Created           | Expires
| Size             | Location       |                   |
+-----+-----+-----+
+-----+-----+-----+
| NE_18.2_CCBUR_ALL | 2            | 2020-03-06 09:35 AM | 2020-03-13
09:31 AM | 28.5 KB     | DD - 10.91.56.108 |
| NE_18.2_CCBUR_ALL | 1            | 2020-03-06 08:46 AM | 2020-03-13
08:38 AM | 205.3 MB    | DD - 10.91.56.108 |
+-----+-----+-----+
+-----+-----+-----+
```

For listing backups, the following headers are visible:

- a. Backup Definition**
- b. Backup Definitions ID.**
- c. Label Number:** number assigned to each backup
- d. Created:** date and time when backup is created
- e. Expires:** date and time when backup expires
- f. Size:** size of backed up data
- g. Location:** location where backup is saved

4. Activity management

To list all activities, execute:

```
[brc@managementnode-cli ~]$ arccli activity list [options]
```

Available options:

- a. active to show active activities
- b. completed to show completed activities
- c. queued to show queued activities
- d. client (or -c) to show activities for specified client

Example: The table with activities is visible:

```
[brc@arc-mgmt-cli ~]$ arccli activity list -c tw_test
2020-03-09 03:59:59,441 INFO      Displaying activities for client: tw_test
+-----+-----+-----+
+-----+-----+-----+
| ID           | Backup Definition   | Status          | Start
Time          | End Time            | Client          |
+-----+-----+-----+
+-----+-----+-----+
| 9158348349561009 | NE_18.2_CCBUR_ALL    | Completed       |
| 2020-03-06 09:33 CET | 2020-03-06 09:35 CET | tw_test         |
| 9158348028230309 | NE_18.2_CCBUR_ALL    | Completed       |
| 2020-03-06 08:40 CET | 2020-03-06 08:46 CET | tw_test         |
+-----+-----+-----+
+-----+-----+-----+
```

Visible headers:

- ID is an activity ID
- Backup Definition is Backup Definition ID.
- Status is a status of the activity.
- Start Time is the date and time when activity began.
- End Time is the date and time when activity ended.
- Client is an ARC Client name.

5. Showing activity details

To show activity details, execute:

```
[brc@managementnode-cli ~]$ arccli activity show <ACTIVITY_ID>
```

Example: The table with activity details is visible.

```
[brc@arc-mgmt-cli ~]$ arccli activity show 9158348349561009
2020-03-04 09:51:32,054 INFO      Displaying activity: 9158348349561009
details
+-----+-----+
| Attribute        | Value          |
+-----+-----+
```

ID	9158348349561009
Status	Completed
Start Time	2020-03-06 09:33 CET
Elapsed	00h:02m:01s
End Time	2020-03-06 09:35 CET
Type	Scheduled Backup
Progress Bytes	288.5 MB
New Bytes	0%
Client	tw_test
Client Release	18.2.100-134
Scheduled Start Time	2020-03-06 09:33 CET
Scheduled End Time	2020-03-06 09:35 CET
Backup Definition	NE_18.2_CCBUR_ALL
Plug-In	Linux File System
Retention	D

Visible attributes:

- Id is an activity ID
- Status is a status of the activity.
- Start Time is date and time that this activity began
- Elapsed is an elapsed time for this activity
- End Time is date and time that this activity completed
- Type is the type of activity.
- Progress Bytes is a total number of bytes examined during this activity
- New Bytes is a percentage of new bytes backed up
- Client is an ARC Client hostname
- Client Release is the Avamar client software version
- Scheduled Start Time is date and time that this activity was scheduled to begin
- Scheduled End Time is date and time that this activity was scheduled to end
- Backup Definition is Backup Definition ID.
- Plug-In is the plugin that is used for this activity
- Retention are retention types that are assigned to this backup

Perform Restore with Avamar GUI

Click Backup & Restore from top level for GUI. → Click ARC/[Your own Domain] → click the client name → find the backup file.

Click Action → Restore Now → On new pop out window, click More Option → on another new pop out window, click show Advanced Options to input pre and post script → click OK → click OK

To restore 1 BrPolicy:

Pre-Script: clean.sh -t

Post-Script: post-restore.sh -t

To restore 1 Helm release:

Pre-Script: release-clean.sh -t

Post-Script: release-post-restore.sh -t

```
#Pre-Script to restore one BrPolicy

clean.sh -t BrPolicy -n [namespace]
#Post-Script to restore one BrPolicy
```

This command restores a BrPolicy.

Usage:

```
post-restore.sh -t BrPolicy [flags]
```

Flags:

```
-c timer counter, wait [counter]*5s for restore to complete, default is
120, which means 10 minutes. It only makes sense when celery is enable on
cbur master
-n namespace, use default namespace by default
-H cburm service name/IP. If not set, use default cbur service name.
usually, pls don't set it, unless you have another cburm service available
rather that default.
-U specify which login username to be verified when auth enabled
-P specify which login password to be verified when auth enabled
-v specify whether to restore volume data or not, supported values is
"true/false", default is false
-p specify which k8sobject type to be restore, it can be a list or a
string, e.g. "all"/"secret"/"secret configmaps"
-m specify which k8sobject name will be restored for specified
object_type. It only supports string format
if both -v and -p args not provided, it means restore all that in BrPolicy.
--max-attempts The max attempts to run restore in case of failure
(Default: 1, i.e. will not retry)
--retry-interval Time in seconds between each retry (default 60)
#Pre-Script to restore one helm release
```

This command cleans up the direcotry in cbur which is to save the backup data for the helm release.

Usage:

```
release-clean.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to clean up.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified, - if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2; - otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3; - otherwise as helm version 2.
-h/--help	Display this help text and exit.

Example:

```
release-clean.sh -t csdc -n ns1 # helm version 3
release-clean.sh -t csdc --tiller-namespace app_tiller_ns # helm version 2
release-clean.sh -h
#Post-Script to restore one helm release
```

This command restores a helm release.

Usage:

```
release-post-restore.sh -t HELM_RELEASE [flags]
```

Flags:

-t HELM_RELEASE	Specify the helm release name to restore.
-n NAMESPACE	Specify the namespace of this helm release. (Only support helm version 3)
--tiller-namespace TILLER_NAMESPACE	Specify the tiller_namespace of this helm release. Default value is "kube-system". (Only support helm version 2)
-v HELM_VERSION	Only support 2 (helm version 2) and 3 (helm verison 3). When "-v" is not specified, - if "--tiller-namespace" is specified, the Containerized Backup and Restore tool will process it as helm version 2; - otherwise if "-n" is specified, the Containerized Backup and Restore tool will process it as helm version 3; - otherwise as helm version 2.

```

-U CBUR_USERNAME      Specify the username of the Containerized Backup and
Restore tool when auth is enabled
-P CBUR_PASSWORD      Specify the password of the Containerized Backup and
Restore tool when auth is enabled
--brpolicies          Restore multiple BrPolicies separated by commas in the
specified order.
--exclude-brpolicies To exclude the given BrPolicies separated by commas.
Note:
If there are multiple BrPolicies in "--brpolicies" or
"--exclude-brpolicies",
please add single quotes ('') or double quotes ("")
around the value when using this script in Avamar GUI.
--volume true|false   Specify whether to restore volume data or not. Default
is false.
--object_type OBJECT_TYPE
Specify the k8sobject type to restore, it can be a list
or a string, e.g. "all"/"secrets"/"secrets configmaps"
or repeat multiple times to specify multiple object
types
--object_name OBJECT_NAME
Specify the k8sobject name to restore for specified
object_type. It only supports string format
and will be applied to all object_type.
--max-attempts        The max attempts to run restore in case of failure
(Default: 1, i.e. will not retry)
--retry-interval      Time in seconds between each retry (default 60)
--timeout DURATION    When celery on, how long to wait for the restore task
to complete (default: 1h0m0s).
Please note, it is for one restoration attempt, the
timer will be resumed for each retry.
The format can be:
1) <d+>, time in seconds, e.g. 600 (600 seconds)
2) <d+>h<d+>m<d+>s, e.g. 2h (2 hours), 2h30m (2 hours
30 mins), 30s (30 seconds)
-h/--help              Display this help text and exit.

```

The Scope of Data to Restore:

- * The pvcs defined in BrPolicy are always restored.
- * If '--volume', '--object_type' and '--object_name' are all not provided, then all items defined in BrPolicy are restored.
- * Otherwise, if one of them are provided:
 - if '--volume true' is not provided, then no volumes are restored (even if they are defined in BrPolicy).
 - if '--object_type' is not provided, then no k8s objects are restored (even if they are defined in BrPolicy).

Example:

```
release-post-restore.sh -t csdc -v 2 # helm version 2
release-post-restore.sh -t csdc --brpolicies 'br1,br2,br3' --object_type all
--tiller-namespace app_tiller_ns # helm version 2
release-post-restore.sh -t csdc --exclude-brpolicies "br1,br2" --volume true
-n ns1 # helm version 3
release-post-restore.sh -t csdc --object_type "secrets" --object_name
"applname" -n ns1 # helm version 3
# If the restoration fails, at most retry two times, each after 120 seconds.
And will wait at most 2 hours for each try to complete.
release-post-restore.sh -t csdc --max-attempts 3 --retry-interval 120 --
timeout 2h
release-post-restore.sh -h
```

You should be able to see a new restore job from the Activity window.

Once it completes, you should be able to see "Completed" without any error code.

Perform Restore from Avamar Pod

The restore procedure can be performed on Client by using the Avamar Client CLI avtar command. Syntax for restoring data by using the avtar command should look as follows:

```
avtar --extract [options]
```

The avtar command options are presented in the following table:

Option	Description
--path=/ARC/	<ul style="list-style-type: none"> • Avamar server account path • Mandatory parameter
--id=@/ARC	<ul style="list-style-type: none"> • The authentication Avamar user ID • The predefined restore user can be used • Mandatory parameter
--password=	<ul style="list-style-type: none"> • The authentication password for the Avamar user • Optional parameter (when not provided, command will prompt for the password) Note: Now

Option	Description
	we get the password from Nokia Archive Cloud (password=testtest)
--labelnumber=	<ul style="list-style-type: none"> Specifies backup to be restored • Backup label numbers can be determined by Listing backups • Mandatory parameter
--run-at-start=	<ul style="list-style-type: none"> Name of the script to run before restoring starts The script must be located on Client in the /etc/avamar/scripts folder • Mandatory parameter1) To restore one BrPolicy:clean.sh -t 2) To restore one helm releaserelease-clean.sh -t
--run-at-end=	<ul style="list-style-type: none"> Name of the script to run after restoring starts The script must be located on Client in the /etc/avamar/scripts folder • Mandatory parameter1) To restore one BrPolicypost-restore.sh -t 2) To restore one helm releaserelease-post-restore.sh -t

1. Get Avamar pod on BCMT cluster node

```
# Get Avamar pod

[root@server-name]# kubectl get pods -n ncms | grep avamar
cbur-master-cbur-avamar-5f867cf749-97sn4    1/1      Running     0          4d3h
```

2. Log into Avamar pod to perform restore

```
[root@server-name]# kubectl exec -it -nncms cbur-master-cbur-
avamar-5f867cf749-97sn4 bash
````-4.2# avtar --extract --path=/ARC/tw_test --id=restore@/ARC --
password=testtest --labelnumber=1 --run-at-start="clean.sh -t cbura-cbur" --
run-at-end="post-restore.sh -t cbura-cbur"
```

Expected outcome:

```
Restore Logs

````-4.2# avtar --extract --path=/ARC/tw_test --id=restore@/ARC --
password=testtest --labelnumber=1 --run-at-start="clean.sh -t cbura-cbur" --
run-at-end="post-restore.sh -t cbura-cbur"
```

```
avtar Info <5551>: Command Line: /usr/local/avamar/bin/avtar.bin --  
vardir=/usr/local/avamar/var --bindir=/usr/local/avamar/bin --sysdir=/usr/  
local/avamar/etc --extract --account=/ARC/tw_test --id=restore@/ARC --  
password=***** --sequencenumber=1 --run-at-start="clean.sh -t  
cbura-cbur" --run-at-end="post-restore.sh -t cbura-cbur"  
avtar Info <7977>: Starting at 2020-03-09 11:03:18 CST [avtar Dec 5 2018  
07:44:00 18.2.100-134 Linux-x86_64]  
avtar Info <5916>: Executing run-at-start '/usr/local/avamar/etc/scripts/  
clean.sh -t cbura-cbur'  
avtar Info <5917>: Back from run-at-start, exit code 0  
avtar Info <6555>: Initializing connection  
avtar Info <5552>: Connecting to Avamar Server (10.75.53.220)  
avtar Info <5554>: Connecting to one node in each datacenter  
avtar Info <5583>: Login User: "restore", Domain: "default", Account: "/ARC/  
tw_test"  
avtar Info <5580>: Logging in on connection 0 (server 0)  
avtar Info <5582>: Avamar Server login successful  
avtar Info <10632>: Using Client-ID='0208bfaa1797945b978cb498330db88559c2460e'  
avtar Info <5550>: Successfully logged into Avamar Server [18.2.0-134]  
avtar Info <5295>: Starting restore at 2020-03-09 11:03:23 CST as "root" on  
"tw-control-01" (4 CPUs) [18.2.100-134]  
avtar Info <10534>: Initializing session on Data Domain: 1, mode:RESTORE,  
boost:ENABLED, compressed-restore:disabled, max-streams:1, pool-  
size:67108864, queue-size:6, read-size:131072  
avtar Info <40174>: Multi-stream restore via cloning is enabled.  
avtar Info <10632>: Using Client-ID='0208bfaa1797945b978cb498330db88559c2460e'  
avtar Info <40062>: GSAN connected via: IPv4, retrieved Data Domain IPv4  
hostname = 10.91.56.108  
avtar Info <10539>: Connecting to Data Domain Server "10.91.56.108"(1) (LSU:  
avamar-1580813747, User: "*****")  
avtar Info <40206>: Setting default storage unit to 'avamar-1580813747' for  
handle 1  
avtar Info <41440>: Data Domain handle:1 capabilities:0x0820021B  
avtar Info <10543>: Data Domain login to 10.91.56.108 successful  
avtar Info <18845>: Restore mode - No Data Domain read compression.  
avtar Info <41359>: Backup #1 had been created by avtar version 18.2.100-134  
as plugin 1001-Filesystem  
avtar Info <5949>: Backup file system character encoding is UTF-8.  
avtar Info <8745>: Backup from Linux host "/ARC/tw_test" (tw-control-01) with  
plugin 1001 - Linux Filesystem  
avtar Info <5538>: Backup #1 label "NE_18.2_CCBUR_ALL" timestamp 2020-03-09  
10:56:42 CST, 1 files, 144 bytes  
avtar Warning <19107>: Unable to determine if there is enough disk space for  
the specified restore.  
avtar Info <5259>: Restoring backup to directory ""
```

```
avtar Info <18076>: Opening DD stored file system container 1 in /  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4 for restore  
operation, using pre-fetched ddr_files.xml.  
avtar Info <15265>: Opening file system container /  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4//  
cur/0208bfaa1797945b978cb498330db88559c2460e/1D5F5BE5CEB8BB4/container.1.cdsf  
for restore.  
avtar Info <5262>: Restore completed  
avtar Info <7925>: Restored 144 bytes from selection(s) with 144 bytes in 1  
files, 5 directories  
avtar Info <6090>: Restored 144 bytes in 0.33 minutes: 25.73 KB/hour (183  
files/hour)  
avtar Info <7883>: Finished at 2020-03-09 11:03:38 CST, Elapsed time:  
0000h:00m:19s  
avtar Info <5916>: Executing run-at-end '/usr/local/avamar/etc/scripts/post-  
restore.sh -t cbura-cbur'  
avtar Info <6031>: Begin STDOUT from run-at-end:  
Restore succeeded  
avtar Info <6032>: End of STDOUT  
avtar Info <5917>: Back from run-at-end, exit code 0  
avtar Info <6645>: Not sending wrapup anywhere.  
avtar Info <5314>: Command completed (1 warning, exit code 0: success)
```

20.3.5.3 Backup or Restore with Backend Mode SFTP

Backup/Restore with backend mode "SFTP"

Configure BrPolicy

Change backend mode to "SFTP". It is case insensitive.

Configure the access to SFTP server

You need to configure the following sftp settings via the Containerized Backup and Restore tool Helm chart or Kubernetes secret "cburm-ssh-config". The secret has the higher priority, if there is secret "cburm-ssh-config" in application's namespace, the Containerized Backup and Restore tool will use the settings in it instead of the ones in helm chart to access sftp server.

- mode: only support "sftp" currently.
- host: the IP address or FQDN of the remote server
- port: the port to be used for sftp on the remote server
- path: the target directory on the remote server (i.e. the directory containing the backup data)
- username: the username to connect to the remote server

- strictHostKeyChecking: Whether to do host key checking for sftp server. It only supports "yes" / "no" / "autoadd".
 1. If the argument is set to "yes", SSH will use the value of "hostKey" to check the host key of sftp server.
 2. If the argument is set to "no", SSH will skip the host key checking even if "hostKey" is set.
 3. If the argument is set to "autoadd", SSH will automatically add new host key to the user known hosts file when connecting for the first time and ssh will refuse to connect to hosts whose host key has changed. If "hostKey" is also set, the behavior will be like "yes", the Containerized Backup and Restore tool will use the value of "hostKey" to do host key checking and will not add host key to known hosts file.
 4. For "yes" and "autoadd", when the host key has changed, user needs to update the new value to the secret "cburm-ssh-config" or use "helm upgrade" to update the 'hostKey' field. But please be cautious that the "helm upgrade" will cause pods to restart, for detailed steps, refer to section "Upgrade procedure" in the Containerized Backup and Restore tool user guide.
- hostKey: the Containerized Backup and Restore tool will use this host key to do host key checking. This field only takes effect when SSH.strictHostKeyChecking is set to "yes" or "autoadd".

 **Note:** For application backup and cbur-master backup, the backup data will be saved in /BACKUP temporarily before transferring to sftp server, you need to allocate enough space for /BACKUP volume.

1) Example of sftp settings in helm chart

```
# sftp setting in helm chart

SSH:
mode: "sftp"
username: "user1"
host: "10.75.145.6"
port: 22
path: "/sftp-root/bcmtnname"
strictHostKeyChecking: !!str "yes"    # only supports "yes", "no", "autoadd"
hostKey: "ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBD5oFeLow1QLEkhoTHM8 /
apxDotCDlFbkbhHPmqrBHoysmsSifj807J/bTRI8wBAaGWTzgHZDBMRUYJcAP5gWw="
```

2) Example of secret "cburm-ssh-config"

```
# sftp setting in secret

apiVersion: v1
kind: Secret
metadata:
  name: cburm-ssh-config
```

```
namespace: default
type: Opaque
stringData:
  port: "22"
  host: 10.75.145.11
  mode: sftp
  username: user1
  path: /sftp-root/bcmtnname
  strictHostKeyChecking: "no"
  hostKey: ""
```

User Key Authentication

User needs to get the public key from secret “cburm-ssh-public-key” in cbur-master’s namespace and add it to the “~/.ssh/authorized_keys” of sftp user in sftp server.

```
kubectl get secret -n ncms cburm-ssh-public-key -o
jsonpath={ .data.ssh_public_key } | base64 -d
```

backup/restore with SFTP

Run helm backup and restore command to do backup/restore.

1. helm backup -t app -a none/enable/disable
2. helm restore -t app -b backup_id

20.3.5.4 Backup or Restore for cbur-master self volume

cbur-master volume

Backup and Recovery can support backup/restore self volume data since CNF 18.10.

Backup and Recovery can support backup following types of self volume:

1. cbur-repo volume, which is used to store backup data for other application. The backend can only be set to third party server, such as AVAMAR
2. Other share volume that can be accessed by cbur-master. This kind of volume is used to save other application data. The backend can be 'LOCAL' or 'AVAMAR'.

For share volume, follow the following rules:

- The share volume can be a list
- The mount path must be the same name with the volume name. For example, the volume name is csdc-data, then the mount path must be /csdc-data.

- The option backup_self only can be enabled by upgrading Backup and Recovery after installation. Because the CRD BrPolicy is created during runtime and the backup_self option will use the kind BrPolicy, all CRs of BrPolicy must be created after CRD creation.

Enable BrPolicy file

To backup/restore for cbur-master self volume, cbur-master must enable its own BrPolicy file.

The BrPolicy file is the same with other APP policy file but with one difference. A new parameter is_cburm is added for backend. This parameter is only needed for backup cbur-master volumes.

You can set backup_self.backend_mode, backup_self.cronSpec, backup_self.maxiCopy when set backup_self.enabled=true

```
# BrPolicy file for cbur-master

helm upgrade cbur-master csf-helm-stable/cbur --version 1.3.4 --reuse-values
--set backup_self.enabled=true
```

Backup or Restore



Note: Before the backup, you can edit Backup and Recovery's BrPolicy to change its parameters. If all apps' data are not saved in LOCAL, you can remove cbur-repo from volumes.

```
kubectl -nncms edit br cbur-master-cbur
```

```
helm backup -t cbur-master -a none
helm restore -t cbur-master -b "backup_id"
```

Use CRD BrHook to run hook jobs

Backup and Recovery supports a CRD-based hook implementation which is used to run hook jobs pre/post backup/restore. User can specify their hooks by including BrHook objects in their Helm charts. When Backup and Restore API receives backup/restore requests, it reads all relevant BrHook objects and executes them at appropriate times.

Define weights in values.yaml

For the weights in BrPolicy and BrHook, suggest to define them in the values.yaml of helm chart, so they can be easily modified when the chart is used as a subchart.

per-release BrHook

Define per-release BrHook (targetType: release) if the chart may be used as a subchart. If a BrHook is defined as per-release, when the chart is included as a subchart, the BrHook will be run around the whole release, not this subchart.

The feature does not change the behavior of backup cron job, which only supports per-BrPolicy cron backup. The cron backup will not run per-release BrHook.

The "jobhookenable" in Helm chart only governs behavior of the Helm plugin hook jobs, not the jobs defined in BrHook. If you begin using BrHook while leaving Helm plugin hook jobs for backwards compatibility, you can set "jobhookenable" as "false", then the hook jobs in "ncms/backup & ncms/restore" will not be executed. And user can consider to wrap {{- if not jobhookenable -}} around BrHook definitions, so for an older BCMT without BrHooks, you can set "jobhookenable" to "true" and the BrHooks will not be installed.

Define CRD 'BrHook'

 **Note:** The following repository/registry URLs are provided as examples, the user should add their own.

```
apiVersion: cbur.csf.nokia.com/v1
kind: BrHook
metadata:
  name: {{ template "etcd.fullname" . }}-br-postbackup-a # will be used as
  k8s job name
  namespace: {{ .Release.Namespace }} # will be used as k8s job namespace
  labels: # will be used as k8s job labels
spec:
  properties:
    targetType: [ brpolicy | release ] # required
    targetName: {{ template "etcd.fullname" . }} # required
    hookPoint: postbackup
    weight: {{ .Values.brhookWeight.app1BrHookpostBackupA }} # integer,
  default: 0
  enable: [ true | false ] # boolean, default: true
  timeout: 900 # integer, default: 600 (seconds)
  deletePolicy: hook-succeeded,before-hook-creation # before-hook-creation:
  default and always ON
  template: # Contains Job spec. "required"
  ##### The following is an example #####
  spec:
    template:
      spec:
        containers:
          - name: foo-test
            image: csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
cbur-cli:1.0.3-1673
            command: ["/bin/sh", "-c", "for i in {1..1}; do date && echo hello;
done && sleep 1"]
            imagePullPolicy: IfNotPresent
            resources: {}
            restartPolicy: Never
            securityContext:
              runAsUser: 999
```

```
terminationGracePeriodSeconds: 30
```

- **targetType**: [brpolicy | release] "brpolicy" means the hook/job is run pre/post the backup/restore of one BrPolicy; "release" means the hook/job is run pre/post the backup/restore of all BrPolicies in the Helm release; "required"
- **targetName**: release name or brpolicy name depending on targetType. It only supports one brpolicy name or release name; "required"
- **hookPoint**: one or more from "prebackup, postbackup, prerestore, postrestore" separated by comma ", "; "required"
- **weight**: integer, default: 0, Backup and Restore will run the hook with the lowest weight first (negative to positive). If some BrHooks have the same weight, they will be executed randomly.
- **enable**: boolean, default: true, disable or enable this hook job
- **timeout**: integer, default: 600 (seconds), Backup and Recovery needs to wait for the job completion and user should give the maximum wait time
- **deletePolicy**: one or more from "before-hook-creation, hook-succeeded, hook-failed" separated by comma ", "; before-hook-creation: Delete the previous resource before this hook job is launched again (default) (**always ON**); hook-succeeded: Delete the resource after the hook is successfully executed; hook-failed: Delete the resource if the hook failed during execution.

Add weight in BrPolicy

Multiple BrPolicies will be run in weighted order with the lowest weight first (negative to positive). If several BrPolicies have the same weight, they will be run randomly.

```
spec:
  weight: {{ .Values.brpolicyWeight.app1BrPolicyA }} # integer, default: 0
```

Backup or Restore Sequence

For multiple BrHooks / BrPolicies, it will not start until the previous one runs to completion, they are sequential. The backup/restore will fail immediately and throw exception when any BrHook or any BrPolicy fails. The remaining BrHooks / BrPolicies will not be run. If user wants to check the logs or status of hook job specifically, do not include "hook-failed" in deletePolicy.

Backup sequence

Restore sequence

Job hook events for backup/restore plugin

For new users who need to run hook job pre/post backup/restore, use the CRD 'BrHook' functionality, which is introduced in the Containerized Backup and Restore tool 20.03 and meant to replace the Helm hook method in this section. The Helm hook method has some shortcomings, for example:

- All hook jobs are executed before/after all brpolicies, the execution order cannot be controlled.
- When celery is on, the post hook jobs may be run before 'backup'/'restore' completes.

To the users using this old way (job hook events for backup/restore plugin):

This old way has many drawbacks. And in **OpenShift**, Backup and Restore only supports Rest APIs for users to use. Change your hook job to use BrHook as soon as possible. Backup and Restore will not maintain this way and will obsolete it in the near future.

To use backup/restore hookjobs, application should have pre/post templates in their Charts. Backup/Restore plugin executes the pre/post backup/restore hooks present in template/ncms/backup or template/ncms/restore.



Note: At present, postbackup/postrestore job hook is not supported if celery is enabled.

The following parameters should be declared in your chart values.yaml.



Note: To define them in the global section in case your chart will be in a umbrella chart or as a sub-chart.

```
# values.yaml

global:
  jobhookenable: true
  prebackup: 0
  postbackup: 0
  prerestore: 0
  postrestore: 0
  jobtimeout: 25 #set by default for the hookjobs .. if your job takes longer
then 25s you can add change the variable in your chart values.yaml.

          #It's a smart timeout, will wait untill the job is completed.
```

Chart Package structure

```
# tree structure of chart for job hook

app-hook
├── Chart.yaml
├── templates
│   └── deployment.yaml
```

```
|   └── _helpers.tpl
|   └── ncms
|       ├── backup
|       |   ├── postbackup.yaml
|       |   └── prebackup.yaml
|       └── restore
|           ├── postrestore.yaml
|           └── prerestore.yaml
|   └── NOTES.txt
|   └── policy.yaml
|   └── pv-claim.yaml
|   └── service.yaml
└── values-model.yaml
└── values-template.yaml.j2
└── values.yaml
```

prebackup.yaml

```
# prebackup.yaml

{{ if gt (int .Values.global.prebackup) 0 }}

# Templated kubernetes jobs

{{- end }}
```

postbackup.yaml

```
# postbackup.yaml

{{ if gt (int .Values.global.postbackup) 0 }}

# Templated kubernetes jobs

{{- end }}
```

rerestore.yaml

```
# prerestore.yaml

{{ if gt (int .Values.global.prerestore) 0 }}
```

```
# Templated kubernetes jobs

{{{- end }}}}
```

postrestore.yaml

```
# postrestore.yaml

{{ if gt (int .Values.global.postrestore) 0 }}

# Templated kubernetes jobs

{{{- end }}}}
```

20.3.5.5 Namespace Backup and Restore

Introduction

The Containerized Backup and Restore tool supports the ability to backup/restore all k8s resources, or partial k8s resources based on Kind, label_selector in one defined namespace. Namespace Backup and Restore is supported for environments in which the Containerized Backup and Restore tool itself is cluster-scoped and namespace-scoped. All backend modes are supported, including local, S3, Avamar, and SFTP.



Note: The namespace backup refers to the Kubernetes resources and does not include the volume data in 1 APP. To backup the volume data, use the "helm backup -t <app-name>" to backup.

Namespace Backup

You can use the Containerized Backup and Restore tool CLI to begin the backup.

```
helm backup -N NAMESPACE [flags]

Flags:
  -a ACTION           the schedule to take. Allowed values: none, enable,
  disable (default "none")
    - none: one-time backup
    - enable: enable cron backup for specified
  namespace.
    - disable: disable cron backup for the specified
  namespace.
```

```

        - "*/5 * * * *": the schedule of cronjobs in the
namespace specified to conduct namespace backup
-h/---help           help for backup
-i TASK_ID          the celery task id to query the task status.
-N NAMESPACE        specify the namespace of namespace backup and
restoration.
--backend string    specify the backend mode of namespace backup and
restoration.
-r MAXIMUM_COPY     specified the maximum copy of namespace backup
--labels string      (Optional) specify the labels selector to backup
selective resources. Use comma to separate the list.
--types string       (Optional) specify the types selector to backup
selective resources, which is case sensitive and the same as K8s KIND.
                           Use comma to separate the list.
--override true|false Specified override the brpolicy values. (default
false)

```

- 1. Namespace backup.** The following example means to backup all the resources in default namespace, with the backend is local and rotation is 2.

```
# helm backup -N default --backend local -r 2
```

- 2. Create cronjob to backup namespace k8s resources.** Below example means to backup all the Kubernetes resources in default namespace every 5 seconds.

```
# helm backup -N default -a "*/5 * * * *"
```

- 3. Selective backup with specified label_selector in specific namespace.** The following example means to backup Kubernetes resources containing the label_selector app=true or release=app in the default namespace.

```
# helm backup -N default --labels app=true,release=app
```

- 4. Selective backup with specified types in specific namespace.** The following example means to backup Secret and ConfigMap in the default namespace.

```
# helm backup -N default --types Secret,ConfigMap
```

Namespace Restore

You can use the Containerized Backup and Restore tool CLI to begin the restore.

```
helm restore -N NAMESPACE [flags]
```

Flags:

```

-b BACKUP_ID           specify the timestamp or backup_id(s) to restore. It
can be:
  - timestamp 'YYYYMMDDHHMMSS': restore the helm
release with the data of a specific timestamp;
    If the backup id with this timestamp doesn't exist
for one of the BrPolicies, restore will exit with failure.
  - if "-b" is not specified: Restore the latest data
for each BrPolicy in the namespace.

The BrPolicies will use the data of different
timestamps if the latest timestamps are different;
  - backup_ids separated by whitespace, which belongs
to the same release.

The backup_id can be found
in .status.backupHistory of BrPolicy;
-h/--help               help for restore
-i TASK_ID              the celery task id to query the task status.
-N NAMESPACE            specify the namespace of namespace backup and
restoration.
--backend string        specify the backend mode of namespace backup and
restoration.
--labels string          (Optional) specify the labels selector to restore
selective resources. Use comma to separate the list.
--types string           (Optional) specify the types selector to restore
selective resources, which is case sensitive and the same as K8s KIND.
                           Use comma to separate the list.
--appdata true|false     restore volume contents of all releases in the
namespace. the default value is false.
--force true|false       whether force to recreate the resources in
namespace. the default value is false.

```

1. Namespace restore. The following example means to restore all the Kubernetes resources in default namespace, the backend and rotation are default values

```
# helm restore -N default
```

2. Selective restore with specified label_selector in specific namespace. The following example means to restore Kubernetes resources containing the label_selector app=true or release=app in the default namespace.

```
# helm restore -N default --labels app=true,release=app
```

3. Selective restore with specified types in specific namespace. The following example means to restore Secret and ConfigMap in the default namespace.

```
# helm restore -N default --types Secret,ConfigMap
```

4. Restore volume contents of all releases in the namespace.

Compared with namespace backup CLI, the namespace restore CLI adds one new option "appdata", it means to restore volume contents of all releases in the namespace if set appdata as true. To use it, ensure to use "helm backup -t" to backup the release volume contents before restore.

For example, there are 2 APPs app1 and app2 in default namespace, to restore the default namespace and including the volume data in app1 and app2, need to run "helm backup -t app1" and "helm backup -t app2" to do the volume content backup, then run "helm backup -N default" to backup the Kubernetes resources in default namespace.

```
[root@server-name]# helm ls
NAME          REVISION        UPDATED         STATUS        CHART
             APP VERSION      NAMESPACE
app1          1              Tue May 25 07:18:35 2021  DEPLOYED    app-1
               1.1            default
app2          1              Tue May 25 07:18:35 2021  DEPLOYED    app-2
               1.1            default
[root@server ~]# helm backup -t app1
[root@server ~]# helm backup -t app2
[root@server ~]# helm backup -N default
[root@server ~]# helm restore -N default --appdata true
```

The following example means to restore the Kubernetes resources in the default namespace, and the Kubernetes resources include the volume contents of all releases in default namespace.

```
# helm restore -N default --appdata true
```

Namespace Backup/Restore via REST API

Namespace Backup API:

```
curl -skL --post30 -X POST http://<cbur-master svc name>.<cbur-master
namespace>.svc:80/v2/backupns/<namespace> -H 'accept: application/json' -H
'Content-Type:application/x-www-form-urlencoded' -d "<parameter list>"
```

Namespace Backup API example:

```
curl -skL --post30 -X POST http://cbur-master-cbur.ncms.svc:80/v2/backupns/
default -H 'accept: application/json' -H 'Content-Type:application/x-www-form-
urlencoded' -d "backend=local"
```

Namespace Restore API:

```
curl -skL --post30 -X POST http://<cbur-master svc name>.<cbur-master namespace>.svc:80/v2/restorens/<namespace> -H 'accept: application/json' -H 'Content-Type:application/x-www-form-urlencoded' -d "<parameter list>"
```

Namespace Restore API example:

```
curl -skL --post30 -X POST http://cbur-master-cbur.ncms.svc:80/v2/restorens/default -H 'accept: application/json' -H 'Content-Type:application/x-www-form-urlencoded' -d "force=true"
```

The limitation for Namespace Backup/Restore

1. You should explicitly backup the Containerized Backup and Restore tool itself before doing a Namespace Backup, to ensure that in the event of the loss of a namespace, the Containerized Backup and Restore tool PVC data (i.e. application backup archives) is preserved.
2. You will need to explicitly install the Containerized Backup and Restore tool in the namespace before restoring it.
3. For Helm 2, since namespaced the Containerized Backup and Restore tool does not have the permission to backup tiller namespace metadata, users need to manually backup the metadata in tiller namespace. This is not an issue for Helm 3.

The Namespace Backup/Restore procedure in multitenancy OpenShift env

Refer to Namespace scoped the Containerized Backup and Restore tool for multitenancy OpenShift env to install the Containerized Backup and Restore tool in multi-tenancy OpenShift env.

The following is a namespace backup/restore example used in OpenShift env provided by QD Application team:

Install the Containerized Backup and Restore tool in multi-tenancy OpenShift env with the Containerized Backup and Restore tool itself is namespace-scope

```
```bash
helm install cbur-master cbur --set
 volumeType.glusterfs=false,k8swatcher.clusterBrEnabled=false,timezone.
 mountHostLocaltime=false,clusterId="cbur",clusterName="cbur-
 master",cburScope="Namespaced",accessAllResources=
 false,nodeSelectorOverride."nodetype"=worker
```

```

Namespace Backup

```
```bash
```

```

curl -X POST http://cbur-master-cbur-test.apps.okd.nokia.com:80/v2/backups/<namespace> -H 'accept: application/json' -H 'Content-Type:application/x-www-form-urlencoded' -d "<parameter list>"

```
or  

```bash
helm backup -N <namespace>
```

```

One limitation to use helm plugin to do namespace backup in multitenancy OpenShift env: if you do not have permission to get the namespace, such as, "kubectl get namespace" reports permission denied, the helm plugin for namespace backup/restore cannot be used and will report following error, it will be enhanced in future releases

```

```bash
[root@server-name]# helm backup -N default
Error from server (Forbidden): namespaces is forbidden: User "xxx" cannot list resource "namespaces" in API group "" at the cluster scope
[KO] Can't find namespace name default. Please enter the valid namespace name or check the usage for syntax
Error: plugin "backup" exited with error
```

```

Namespace Restore

```

curl -X POST http://cbur-master-cbur-test.apps.okd.nokia.com:80/v2/restorens/<namespace> -H 'accept: application/json' -H 'Content-Type:application/x-www-form-urlencoded' -d "<parameter list>"
```

or

```
helm restore -N <namespace>
```

Note:

1. In multi-tenancy OpenShift env with the Containerized Backup and Restore tool itself is namespace-scope, the Containerized Backup and Restore tool combine the admin cluster-role be default, so the Containerized Backup and Restore tool will only backup/restore the resources the admin cluster role can access. WARNING logs will be reported in the cbur pod logs if the resource is forbidden to get.
2. Users can bind their specific service account name when install the Containerized Backup and Restore tool to define their rules when the default rules do not meet request. `rbac: enabled: false serviceAccountName: "sa"`

20.3.5.6 Restore between clusters

Restore application with backup data on another cluster

the Containerized Backup and Restore tool supports application restore from a backup archive created on another cluster. Take below scenario as example.

1. On cluster A, application did backup
2. Then on cluster B, Users want to restore application data that used the backup archive created on cluster A

the Containerized Backup and Restore tool can support this kind of scenario but with below prerequisite:

- The namespace and release name must be the same in the two cluster
- The backend must be the same in the two cluster
- On cluster B, if there is no backupHistory of the BrPolicy in cluster B, then users can directly run helm restore to restore the data with backup data created with cluster A.
- On cluster B, if users already do backup before restore, that is there is already backupHistory of the BrPolicy in cluster B, then users must run a image sync first before running restore.

The sync command is(replace the namespace and brpolicy name in the API)

```
curl -kL --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v1/images/<namespace>/<brpolicyname>?force=true'
```

20.3.5.7 To configure Avamar if using AVAMAR as the backup or restore repository

Note: Skip this step if the endpoint is not set to "AVAMAR" when invoking < ncm tool cluster backup --endpoint avamar --schedule "0 0 * * *" --rotation 1>

It is similar to the above section except following configs:

- All related name settings can be your BCMT cluster name.
- In **Select Files and Folder**, input folder /CLUSTER
- No need to configure **pre-script and post-script**

20.3.5.8 To set ncm endpoint and login ncm

```
# ncm config set --endpoint https://<any one of control nodes' IP>:8082/ncm/api/v1
# ncm user login --username="ncm-admin" --password="<your password>"
```

If you have not initialized the ncm user and password, log in and reset the password.

20.3.5.9 To begin the back up as Kubernetes cronjob

By ncm CLI to set cron job schedule, backup storage method, max keeping copies and support multiple tillers.

```
NAME:
  ncm tool cluster backup - backup cluster

USAGE:
  ncm tool cluster backup [command options] [arguments...]

OPTIONS:
  --list      <boolean>      list the backup archives of an
  endpoint(optional,if value is true, will list the archives of an endpoint)
  --schedule   <schedule>     specifies the schedule for the backup; none for
  immediate backup, disable for disable backup
  --endpoint   <endpoint>     specifies the endpoint for the backup; value
  should be:local,avamar,cephs3,awss3,swifts3,sftp;(optional,default is local)
  --rotation    <rotation>     specifies the rotation of the backup;
  value[1-10] is valid(optional, default is 1)
  --override    <override>     specifies whether to overide or not; only true
  or false is valid(optional,default is false)
```

Example:

```
# ncm tool cluster backup --schedule "0 23 * * *" --endpoint "netbackup" --
rotation 1
"Very Important: Manually save cluster config by command <ncm config export>.
This cluster config will be used to create a new cluster if disaster
recovery.
Finally, a k8s cronjob backup-cluster-by-cbur is created for scheduled cluster
backup.
And you can check the result in requestHistory in cluster BrPolicy by <kubectl
get brpolices.cbur.csf.nokia.com backup-cluster-by-cbur -n ncms -o yaml>"
```

If ncm tool cluster backup is successful, the command will return a note about cluster deploying configs and created k8s cronjob name.

You need to save cluster deploying configs by yourself in case cluster happens disaster and everything is lost.

```
[root@server-name]# ncm config export
```

Can check the "requestHistory" in brpolicy backup-cluster-by-cbur to see if Kubernetes cron job is created successfully.

```
# kubectl get br -n ncms
```

```

NAME                AGE
backup-cluster-by-cbur   2m

# # kubectl get br backup-cluster-by-cbur -n ncms -o yaml
apiVersion: cbur.csf.nokia.com/v1
kind: BrPolicy
metadata:
  creationTimestamp: "2019-07-03T12:53:39Z"
  generation: 8
  name: backup-cluster-by-cbur
  namespace: ncms
  resourceVersion: "3409727"
  selfLink: /apis/cbur.csf.nokia.com/v1/namespaces/ncms;brpolices/backup-
cluster-by-cbur
  uid: 9477089d-9d91-11e9-a7c8-fa163e956783
spec:
  backend:
    mode: netbackup
    cronSpec: '*/2 * * * *'
    cronjob_name: backup-cluster-by-cbur
    ignore: "true"
    k8sType: backupall
    maxiCopy: 1
    override: "true"
status:
  requestHistory:
    addCronjob:
      - "20190703125340":
          apiVersion: v1
          code: 200
          details:
            name: backup-cluster-by-cbur
            namespace: ncms
            kind: Status
            message: Successfully to create cronjob backup-cluster-by-cbur
            metadata: {}
            reason: ""
            status: Success

```

And you can check the cronjob details by using kubectl commands.

```

# kubectl get cronjob -n ncms
NAME           SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
backup-cluster-by-cbur  0 23 * * *  False      0        <none>   37s

```

```
# kubectl describe cronjob backup-cluster-by-cbur -n ncms
```

20.3.5.10 To disable a scheduled cluster backup

You can disable a scheduled cluster backup by command as below, the cronjob will be deleted.

```
ncm tool cluster backup --schedule 'disable'
```

20.3.5.11 To begin the immediate cluster backup

An immediate cluster backup can be triggered by following command:

```
# ncm tool cluster backup --schedule 'none' --endpoint <ENDPOINT>
```

The output should be as follows, or else a timeout when Celery is off.

```
# ncm tool cluster backup --schedule 'none' --endpoint local
{"result":"Very Important: Please manually save cluster config by command <ncm config export>.
This cluster config will be used to create a new cluster if disaster recovery.
backupCluster task = f6edb4f8-ae07-4301-a18c-6c1741b1871e is on!
And you can check the result in requestHistory in cluster BrPolicy by
<kubectl get brpolices.cbur.csf.nokia.com backup-cluster-by-cbur -n ncms -o yaml>"}
```

The immediate backup will not create BrPolicy if not exist, so request history will not be recorded in BrPolicy. Check the request result from the cbur-m pod standard output log.

20.3.6 Deployment

If you are using BCMT, users can choose how to install Backup and Restore in NCS.

- **Automatic installation during NCS deployment** NCS deployment installs Backup and Recovery by default.

> **Note**: In default, the container images of Avamar are not in BCMT's container registry. So you need to download them by yourself and push them to BCMT's container registry if you want to use Avamar.

- **Manual installation**

You can choose not to install Backup and Recovery during NCS deployment. After deploying NCS successfully, to install Backup and Recovery like other common apps.

If it is not in the NCS, you can install it manually, using the Helm install command. Refer to the *Nokia Container Services (NCS) Manual Installation Guide* for a detailed manual installation procedure.

20.3.7 Back Up and Restore via NCM CLI

The following method is only supported in NCS.

```
$ ncs appli backup --id <app helm release name> --action none
/
$ ncs appli restore --id <app helm release name>
```

20.3.8 Back Up and Restore via REST API

This way is supported both in NCS and non-NCS. Backup and Restore currently supports backup and restore on:

- helm release level
- per BrPolicy level
- NCS cluster level

For NCS Cluster level, refer to [3.11 BCMT Cluster Backup and Restore for details](#).

For back up:

```
$ curl -s --header 'Accept: application/json' -X POST http://<cbur-master svc name>.<cbur-master namespace>.svc:80/v2/helm/release/backup/<app namespace>/<app release name>?helm_version=<helm version>
```

For restore:

```
$ curl -kL --header 'Accept: application/json' -X POST --header 'Content-Type: application/x-www-form-urlencoded' -d 'backup_id=<backup_id>&volume=&object_type=&object_name=' http://<cbur-master svc name>.<cbur-master namespace>.svc:80/v2/helm/release/restore/<app namespace>/<app release name>?helm_version=<helm version>
```

Restore command example:

BCMT:

```
curl -s -kL --post301 --cert /opt/bcmt/config/cbur/client.pem --key /opt/bcmt/config/cbur/client-key.pem --header 'Accept: application/json' -X POST --header 'Content-Type: application/x-www-form-urlencoded' -d 'backup_id=20210508032354_e03_LOCAL_csdcc_jk-csdcc&volume=&object_type=&object_name=' http://cbur-master-cbur.ncms.svc:80/v2/helm/release/restore/csdcc_jk?helm_version=3
```



Note:

- <helm version> can support helm2 and helm3, i.e. helm_version=2
- If https enabled for cbur-master, the --key and --cert option must be included in the API
- If the backup_id can not be referred in brpolicy history or the backup_id is not analyzed successfully, the restored volume would be the latest one rather than the specific version, and the curl command would print "backup_id": "null"

20.3.9 Install or Upgrade Helm Plugins

There are following ways to install or upgrade backup and restore Helm plugins.

- Install the backup and restore plugin separately.

```
### For helm version 2
# Remove them first if they have been installed.
$ helm plugin remove backup
$ helm plugin remove restore

# Install the latest version
$ helm plugin install https://example/artifactory/csf-generic-delivered-local/
helm-plugins/backup-3.1.4.tgz
$ helm plugin install https://example/artifactory/csf-generic-delivered-local/
helm-plugins/restore-3.1.4.tgz

### For helm version 3
# Uninstall them first if they have been installed.
$ helm3 plugin uninstall backup
$ helm3 plugin uninstall restore

# Install the latest version
$ helm3 plugin install https://example/artifactory/csf-generic-delivered-
local/helm-plugins/backup-3.1.4.tgz
$ helm3 plugin install https://example/artifactory/csf-generic-delivered-
local/helm-plugins/restore-3.1.4.tgz
```

 **Note:** For helm backup/restore plugins, use the version in the same Backup and Restore release as Backup and Restore chart. The version info can be found in "SW packages" of Backup and Restore release notes.

20.3.10 Helm Version 2 Back Up and Restore Plugins Usage

```
# helm backup -h
```

Back up a Helm release or enable / disable cron backup for a Helm release.

Usage:

```
backup -t RELEASE [flags]
```

Flags:**-a ACTION**

the action to take. Allowed values: none, enable, disable (default "none")

- none: one-time backup
- enable: enable cron backup for all BrPolicies in the release, the release BrHooks (if any) will not be run.
- disable: disable cron backup for all BrPolicies in the release

-h/--help

help for backup

-i TASK_ID

the celery task id to query the task status.

-n CBURM_TILLER_NAMESPACE

the tiller namespace of cbur-master (default "kube-system")

-p LOCAL_CHART

the path to the local chart if the chart is not in repo. It is only used for helm hook job(s).

--ca-file string

Verify certificates of HTTPS-enabled servers using this CA bundle. It is for secure chart repo when using helm hook job(s).

--cert-file string

Identify HTTPS client using this SSL certificate file. It is for secure chart repo when using helm hook job(s).

--key-file string

Identify HTTPS client using this SSL key file. It is for secure chart repo when using helm hook job(s).

--password string

Chart repository password where to locate the requested chart. It is for secure chart repo when using helm hook job(s).

-t RELEASE

the release name to backup

-x CBURM_SERVICE

specify the cbur-master service in format 'dns_name:port' or 'ip:port'

Global Flags:**--tiller-namespace TILLER_NAMESPACE**

namespace of Tiller. (default "kube-system")

Example:

```
# one-time backup release 'app1'
helm backup -t app1
# one-time backup release 'app1', cbur-master in tiller-namespace
'cburm_tiller_ns'
helm backup -t app1 -a none -n cburm_tiller_ns
# cbur-master at '10.197.19.86:8080'
helm backup -t app1 -a none -x '10.197.19.86:8080'
```

```
# Enable cron backup for all BrPolicies of 'app1'  
helm backup -t app1 -a enable  
# Query the status of backup task  
helm backup -i 5ed9a664-3dcb-42b6-aaf5-04f937817417
```

```
# helm restore -h
```

Restore a helm release with the latest or specified backup id(s).

Usage:

```
restore -t RELEASE [flags]
```

Flags:

-b BACKUP_ID

specify the timestamp or backup_id(s) to restore. It can be:

- timestamp 'YYYYMMDDHHMMSS': restore the helm release with the data of a specific timestamp. If the backup id with this timestamp does not exist for one of the BrPolicies, restore will exit with failure.
- if "-b" is not specified: Restore the latest data for each BrPolicy in the release. The BrPolicies will use the data of different timestamps if the latest timestamps are different;
- backup_ids separated by whitespace, which belongs to the same release. The backup_id can be found in .status.backupHistory of BrPolicy;

-h/--help

help for restore

-i TASK_ID

the celery task id to query the task status.

-n CBURM_TILLER_NAMESPACE

the tiller namespace of cbur-master (default "kube-system")

-p LOCAL_CHART

the path to the local chart if the chart is not in repo. It is only used for helm hook job(s).

--ca-file string

Verify certificates of HTTPS-enabled servers using this CA bundle. It is for secure chart repo when using helm hook job(s).

--cert-file string

Identify HTTPS client using this SSL certificate file. It is for secure chart repo when using helm hook job(s).

--key-file string

Identify HTTPS client using this SSL key file. It is for secure chart repo when using helm hook job(s).

--password string

Chart repository password where to locate the requested chart. It is for secure chart repo when using helm hook job(s).

-t RELEASE

the release name to backup

-x CBURM_SERVICE

specify the cbur-master service in format 'dns_name:port' or 'ip:port'

--volume true/false

whether to restore volume(s) (default "false")

--object_type TYPE

the k8s object type to restore. It can be "all", one or a list of object types separated by whitespace, e.g. "all"/"secrets"/"secrets configmaps" or repeat multiple times to specify multiple object types. The object_type must be same with what is configured in BrPolicy. Otherwise, for example, it's 'secret' in BrPolicy, but --object_type as 'secrets', the resource will not be restored.

--object_name NAME

the k8s object's full name or prefix to restore for the specified object_type(s). It only supports string format and will be applied to all specified object_type(s).

Global Flags:

--tiller-namespace TILLER_NAMESPACE

namespace of Tiller. (default "kube-system")

The Scope of Data to Restore:

- The pvcs defined in BrPolicy are always restored.
- If both '--volume' and '--object_type' are not provided, then all items defined in the BrPolicy are restored.
- If '--volume' is provided and '--object_type' is not, then no k8s objects are restored (even if they are defined in BrPolicy).
- If '--object_type' is provided but '--volume' is not, then no volumes are restored (even if they are defined in BrPolicy).

Example:

```
# Restore release 'appl' with the latest data of each BrPolicy
helm restore -t appl

# Restore release 'appl' with the data of timestamp "20200225090021", cbur-
master in tiller-namespace "cburm_tiller_ns"
helm restore -t appl -b "20200225090021" -n cburm_tiller_ns
# Only restore volumes, cbur-master at '10.197.19.86:8080'
helm restore -t appl --volume true -x 10.197.19.86:8080

# Restore 'csdc-at' with backup-ids '20200228112009_e02_LOCAL_default_csd-
at-csdc' (BrPolicy 'csdc-at-csdc') and '20200228112010_e02_LOCAL_default_csd-
at-csdc-1' (BrPolicy 'csdc-at-csdc-1')
# If 'csdc-at' has other BrPolicies, they will not be restored.
helm restore -t csdc-at -b "20200228112009_e02_LOCAL_default_csd-
at-csdc-1" "20200228112010_e02_LOCAL_default_csd-
at-csdc-1"

# Query the status of restore task
helm restore -i fc4fd25d-6fc1-4d26-9c42-0cbd479e0501

# Restore the secrets and configmaps which names start with 'applname'
helm restore -t appl --object_type "secrets configmaps" --object_name
"applname"
```

20.3.11 Helm Version 3 Back Up and Restore Plugins Usage

```
# helm3 backup -h
```

Back up a Helm release or enable / disable cron backup for a Helm release.

Usage:

```
backup -t RELEASE [flags]
```

Flags:

-a ACTION

the action to take. Allowed values: none, enable, disable (default "none")

- none: one-time backup
- enable: enable cron backup for all BrPolicies in the release, the release BrHooks (if any) will not be run.
- disable: disable cron backup for all BrPolicies in the release

-h/--help

help for backup

-i TASK_ID

the celery task id to query the task status.

-p LOCAL_CHART

the path to the local chart if the chart is not in repo. It is only used for helm hook job(s).

--ca-file string

Verify certificates of HTTPS-enabled servers using this CA bundle. It is for secure chart repo when using helm hook job(s).

--cert-file string

Identify HTTPS client using this SSL certificate file. It is for secure chart repo when using helm hook job(s).

--key-file string

Identify HTTPS client using this SSL key file. It is for secure chart repo when using helm hook job(s).

--password string

Chart repository password where to locate the requested chart. It is for secure chart repo when using helm hook job(s).

-t RELEASE

the release name to backup

-x CBURM_SERVICE

specify the cbur-master service in format 'dns_name:port' or 'ip:port'

-x HOST/PORT

For multitenancy OpenShift env, example: "-x cbur-master-cbur-jk.apps.okd.nokia.com".

Global Flags:

-n NAMESPACE

the namespace of this Helm release

Example:

```
# one-time backup release 'app1' in default namespace
helm3 backup -t app1
# one-time backup release 'app1' in namespace 'ns1'
helm3 backup -t app1 -a none -n ns1
# cbur-master at '10.197.19.86:8080'
helm3 backup -t app1 -a none -x '10.197.19.86:8080'
# Use the K8s route to find the Containerized Backup and Restore tool
service in multitenancy OpenShift env
helm3 backup -t app1 -x cbur-master-cbur-jk.apps.okd.nokia.com
# Enable cron backup for all BrPolicies of 'app1'
helm3 backup -t app1 -a enable
# Query the status of backup task
helm3 backup -i 5ed9a664-3dcf-42b6-aaf5-04f937817417
```

helm3 restore -h

Restore a helm release with the latest or specified backup id(s).

Usage:

restore -t RELEASE [flags]

Flags:**-b BACKUP_ID**

specify the timestamp or backup_id(s) to restore. It can be:

- timestamp 'YYYYMMDDHHMMSS': restore the helm release with the data of a specific timestamp. If the backup id with this timestamp doesn't exist for one of the BrPolicies, restore will exit with failure.
- if "-b" is not specified: Restore the latest data for each BrPolicy in the release. The BrPolicies will use the data of different timestamps if the latest timestamps are different;
- backup_ids separated by whitespace, which belongs to the same release. The backup_id can be found in .status.backupHistory of BrPolicy;

-h/--help

help for restore

-i TASK_ID

the celery task id to query the task status.

-p LOCAL_CHART

the path to the local chart if the chart is not in repo. It is only used for helm hook job(s).

--ca-file string

Verify certificates of HTTPS-enabled servers using this CA bundle. It is for secure chart repo when using helm hook job(s).

--cert-file string

Identify HTTPS client using this SSL certificate file. It is for secure chart repo when using helm hook job(s).

--key-file string

Identify HTTPS client using this SSL key file. It is for secure chart repo when using helm hook job(s).

--password string

Chart repository password where to locate the requested chart. It is for secure chart repo when using helm hook job(s).

-t RELEASE

the release name to backup

-x CBURM_SERVICE

specify the cbur-master service in format 'dns_name:port' or 'ip:port'

-x HOST/PORT

For multitenancy OpenShift env, e.g "-x cbur-master-cbur-jk.apps.okd.nokia.com"

--volume truemfalse

whether to restore volume(s) (default "false")

--object_type TYPE

the k8s object type to restore. It can be "all", one or a list of object types separated by whitespace, e.g. "all"/"secrets"/"secrets configmaps" or repeat multiple times to specify multiple object types. The object_type must be same with what is configured in BrPolicy. Otherwise, for example, it's 'secret' in BrPolicy, but --object_type as 'secrets', the resource will not be restored.

--object_name NAME

the k8s object's full name or prefix to restore for the specified object_type(s). It only supports string format and will be applied to all specified object_type(s).

Global Flags:**-n NAMESPACE**

the namespace of this Helm release

The Scope of Data to Restore:

- The pvcs defined in BrPolicy are always restored.
- If both '--volume' and '--object_type' are not provided, then all items defined in the BrPolicy are restored.
- If '--volume' is provided and '--object_type' is not, then no k8s objects are restored (even if they are defined in BrPolicy).
- If '--object_type' is provided but '--volume' is not, then no volumes are restored (even if they are defined in BrPolicy).

Example:

```
# Restore release 'app1' in default namespace with the latest data of each
BrPolicy
helm3 restore -t app1
# Restore release 'app1' in namespace 'ns1' with the data of timestamp
"20200225090021"
helm3 restore -t app1 -b "20200225090021" -n ns1
# Only restore volumes, cbur-master at '10.197.19.86:8080'
```

```

helm3 restore -t appl --volume true -x 10.197.19.86:8080
# Use the K8s route to find the Containerized Backup and Restore tool
service in multitenancy OpenShift env
helm3 restore -t appl -x cbur-master-cbur-jk.apps.okd.nokia.com
# Restore 'csdc-at' with backup-ids '20200228112009_e02_LOCAL_default_csdca
t-csdc' (BrPolicy 'csdc-at-csdc') and '20200228112010_e02_LOCAL_default_csdca
t-csdc-1' (BrPolicy 'csdc-at-csdc-1')
# If 'csdc-at' has other BrPolicies, they will not be restored.
helm3 restore -t csdc-at -b "20200228112009_e02_LOCAL_default_csdca
t-csdc 20200228112010_e02_LOCAL_default_csdca
t-csdc-1"
# Query the status of restore task
helm3 restore -i fc4fd25d-6fc1-4d26-9c42-0cbd479e0501
# Restore the secrets and configmaps which names start with 'applname'
helm3 restore -t appl --object_type "secrets configmaps" --object_name
"applname"

```

20.3.12 Scheduled Backup

Scheduled Backup

You can use CronJobs to run backup jobs on a time-based schedule. These automated jobs run like Cron tasks on a Linux or UNIX system. Scheduled backups can be enabled via two ways, automatic and manual, details follow.

cronjob name

For general, the cronjob name is a string with the format brpolicyname-namespace, it may be changed based on following scenarios:

- if the string is less than 53 characters, cronjob name will be no change
- if the string is more than 53:

Then check if cbur_scope is "Namespaced", if brpolicyname < 52 characters, cronjob = brpolicyname

- all other scenarios, CBUR will trunc the cronjobname to 53 characters (53 includes character "-").

CBUR will trunc "brpolicyname-namesapce" to 50 chacters plus "%s%s", %s will be the random combinations in (all accii lower case and string digits)

Automatically Enable or Disable

There are two parameters in BrPolicy that impact cronjob automatic enable or disable.

.spec.autoEnableCron: If BrPolicy contains spec.cronsing that is not empty, autoEnableCron = true indicates that the cron job is immediately scheduled when the BrPolicy is created, while autoEnableCron = false indicates that scheduling of the cron job should be done on a subsequent manual backup request. This option only works when k8swatcher.enabled is true

.spec.autoUpdateCron: This parameter is used to indicate if subsequent update of cronjob will be done via brpolicy update. Its value can be true or false. true means cronjob must be updated via brpolicy update, false means cronjob must be updated via manual helm backup -t app -a enable/disable command.

| autoEnableCron | autoUpdateCron | Result |
|----------------|----------------|--|
| FALSE | FALSE | <p>Cronjobs are not scheduled at Br Policy creation, and subsequent scheduling of cronjobs can only be done via manual commands.</p> <p>Action: Do nothing</p> |
| TRUE | FALSE | <p>Cronjobs are scheduled at Br Policy creation, and subsequent scheduling of cronjobs can only be done via manual commands.</p> <p>Action: If .status.scheduleEnable is not set, then add cronjob, status changed to enable2. if cronspec not exist, throwfail</p> |
| FALSE | TRUE | <p>Cronjobs are not scheduled at Br Policy creation, and subsequent scheduling of cronjobs does not take effect since autoEnableCron is false.</p> <p>Action: If .status.scheduleEnable is enabled, delete cronjob, status changed to disable if .status. scheduleEnable is disable/none, do nothing.</p> |
| TRUE | TRUE | <p>Cronjobs are scheduled at Br Policy creation, and subsequent scheduling of cronjobs can only be done via Helm upgrade w/ modification of the BrPolicy's cronspec.</p> <p>Action: If .status.scheduleEnable is enabled, update cronjoba</p> |

| autoEnableCron | autoUpdateCron | Result |
|-----------------------|-----------------------|---|
| | | <p>if cronspec not exist, delete cronjob, status->disableb) if cronspec exist, update cronjob, status is still enabled.</p> <p>if .status.scheduleEnable is disable/none, add cronjoba) if cronspec not exist, throwfail, status is still noneb) if cronspec exist, add cronjob, status none-> enable</p> |

| autoUpdateCron | Status | autoEnableCron | cronsing | Result |
|-----------------------|---------------|-----------------------|-----------------|--|
| TRUE | none | FALSE | present | Do nothing. |
| TRUE | none | FALSE | not present | Do nothing |
| TRUE | none | TRUE | present | status = enable.
Schedule cronjob. |
| TRUE | none | TRUE | not present | Do nothing |
| TRUE | enable | FALSE | present | status = disable.
Delete cronjob |
| TRUE | enable | FALSE | not present | status = disable.
Delete cronjob |
| TRUE | enable | TRUE | present | status remains enable. Update cronjob. |
| TRUE | enable | TRUE | not present | status = disable.
Delete cronjob |
| TRUE | disable | FALSE | present | Do nothing |

| autoUpdateCron | Status | autoEnableCron | cronspec | Result |
|-----------------------|---------------|-----------------------|-----------------|---|
| TRUE | disable | FALSE | not present | Do nothing |
| TRUE | disable | TRUE | present | status = enable.
Schedule cronjob. |
| TRUE | disable | TRUE | not present | Do nothing. |
| FALSE | none | FALSE | present | Do nothing. |
| FALSE | none | FALSE | not present | Do nothing. |
| FALSE | none | TRUE | present | status = enable.
Schedule cronjob. |
| FALSE | none | TRUE | not present | Do nothing |
| FALSE | enable | FALSE | present | Do nothing |
| FALSE | enable | FALSE | not present | Do nothing |
| FALSE | enable | TRUE | present | Do nothing |
| FALSE | enable | TRUE | not present | Do nothing (only manual command can delete cronjob and change status) |
| FALSE | disable | FALSE | present | Do nothing |
| FALSE | disable | FALSE | not present | Do nothing |
| FALSE | disable | TRUE | present | Do nothing (only manual command can enable cronjob and change status) |

| autoUpdateCron | Status | autoEnableCron | cronspec | Result |
|----------------|---------|----------------|-------------|------------|
| FALSE | disable | TRUE | not present | Do nothing |

 **Note:** There is one case may have compatibility issue, when autoEnableCron is first set to false, then change it to true, the Containerized Backup and Restore tool only adds cronjob when .status.ScheduleEnable is not set in brpolicy.

Manual Enable or Disable

The user can manually enable, disable or update cronjob via set action to enable in the Helm backup command.

```
$ # helm backup -t appl -a enable
```

Scheduled backup is removed when the value of action is set to disable.

```
$ # helm backup -t appl -a disable
```

To see the results of the command:

```
$ # kubectl get jobs -n ncms
NAME          DESIRED   SUCCESSFUL   AGE
app1-cronjob-1536025800  1          1           27m
app1-cronjob-1536114000  1          1           17m
app1-cronjob-1536115500  1          1            7m
```

After creating the cron job, get its status using this command:

```
$ # kubectl get jobs -n ncms
NAME          DESIRED   SUCCESSFUL   AGE
app1-cronjob-1536025800  1          1           27m
app1-cronjob-1536114000  1          1           17m
app1-cronjob-1536115500  1          1            7m
```

After creating the cron job, get its status using this command:

```
$ # kubectl get cronjob app1-cronjob -n ncms
NAME      SCHEDULE      SUSPEND      ACTIVE      LAST SCHEDULE      AGE
app1-cronjob  */10 * * * *  False        0          <none>        3m
```

If you want to know the backup result, you can get the `requestHistory` in the `BrPolicy`.

```
$ kubectl get brpolicy [BrPolicy-name] -n [namespace] -o json | jq  
".status.requestHistory.backup"
```

Manual Enable or Disable via API

The Containerized Backup and Restore tool supports enable or disable schedule backup via API for both single `BrPolicy` and a Helm release.

Single BrPolicy:

- Enable schedule backup for single `brpolicy` for Helm 3

```
$ curl -v --header 'Accept: application/json' -X PUT http://cbur-master-  
cbur.sps-btel-sm.svc.cluster.local:80/v2/cronjob/{namespace}/{policynname}?  
helm_version=3
```

- Enable schedule backup for single `brpolicy` for Helm 2

```
$ curl -v --header 'Accept: application/json' -X PUT http://cbur-master-  
cbur.sps-btel-sm.svc.cluster.local:80/v2/cronjob/{tiller_namesapce}/  
{policynname}?helm_version=2
```

- Disable schedule backup

```
curl -v --header 'Accept: application/json' -X DELETE http://cbur-master-  
cbur.sps-btel-sm.svc.cluster.local:80/v2/cronjob/{tiller_namesapce}/  
{policynname}?helm_version=2
```

A Helm release:

- Enable schedule backup for Helm release for Helm 3

```
$ curl -v --header 'Accept: application/json' -X PUT http://cbur-master-  
cbur.ncms.svc:80/v2/helm/release/cronjob/{namespace}/{releasename}?  
helm_version=2
```

- Enable schedule backup for Helm release for Helm 2

```
$ curl -v --header 'Accept: application/json' -X PUT http://cbur-master-  
cbur.ncms.svc:80/v2/helm/release/cronjob/{tiller_namespace}/{releasename}?  
helm_version=2
```

- Disable schedule backup for Helm release

```
curl -v --header 'Accept: application/json' -X DELETE http://cbur-master-
cbur.ncms.svc:80/v2/helm/release/cronjob/{namespace}/{releasename}?
helm_version=3
```

20.3.13 Retrieve RequestHistory

The API supports retrieve requestHistory for single policy/helm release/cluster BR/namespaced BR

API format

/v2/policy/{namespace}/{policyname}/requesthistory For cluster backup, the policy name is backup-cluster-by-cbur For cluster restore, the policy name is restore-cluster-by-cbur.

Example

```
HTTPS is not enabled for cbur-master
curl -kL --header 'Accept: application/json' -X GET http://
cbur-master-cbur.ncms.svc:80/v2/default/csdc/requesthistory | jq
".message.status.requestHistory"
```

20.3.14 Retrieve backup_id

backup_id format

From 19.06, backup_id is saved in the .status.BackupHistory of BrPolicy and this section is automatically updated by Backup and Recovery.

The backup_id is named by following rule since Backup and Recovery chart version 1.3.0:

```
<timestamp format "%Y%m%d%H%M%S">_e02_<backend mode>_<namespace>_<BrPolicy
name>
```

Before Backup and Recovery chart version 1.3.0, the backup_id is named as:

```
<timestamp format "%Y%m%d%H%M%S">_e01_<backend mode>_<BrPolicy name>
```



Note:

If use old backup_id format to do restore, then restore will fail.

So we suggest, if you installed the new Backup and Restore version(>=1.3.0), do a backup first, then restore. Restore with earlier backup_id may fail.

Get backup_id from BrPolicy

We can directly check it in BrPolicy.

Firstly to get your BrPolicy name which is equal to the target StatefulSet or Deployment or DaemonSet name by using helm status.

```
$ helm status <your helm release name>
RESOURCES:
==> v1/BrPolicy
NAME          AGE
testbr-cbur-testing  28d
```

OR

```
$ kubectl get brpolicy -n <your helm release namespace>
NAME          AGE
testbr-cbur-testing  28d
```

Then get its backupHistory.

```
$ kubectl get brpolicy [BrPolicy-name] -o=jsonpath='{.status.backupHistory}' -n [namespace]
```

 **Note:**

After application delete and re-install:

- If k8swatcher is not enabled, user needs to run the following API explicitly to sync the backupHistory to contain the previous backup ids of the same BrPolicy.
- Otherwise, k8swatcher will sync the backupHistory. Besides these ways, backup and restore operation will also sync the backupHistory.

Get backup_id via API

By default, if the backupHistory contains valid items (the backend in backup id is the same as the current backend), Backup and Restore will think the backupHistory to be accurate, and will not sync with the real backend. If user wants to sync with the real backend, please add "?force=true" at the end of API:

- For "local" backend, Backup and Restore will sync with the data in /CBUR_REPO.
- For "AWSS3" / "CEPHS3" / "SWIFTS3" / "SFTP" backend, Backup and Recovery will sync with the data in AWS S3 / CEPH S3 / SWIFT S3 / SFTP server.

 **Note:** There is no backupHistory for AVAMAR backend.

1. HTTPS is not enabled for cbur-master

```
curl -kL --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v2/images/<your helm release namespace>/<your BrPolicy name?force=true(default:false)>'  

2. HTTPS is enabled for cbur-master  

curl -kL --post301 --cert /opt/bcmt/config/cbur/client.pem --key /opt/bcmt/config/cbur/client-key.pem --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v2/images/<your helm release namespace>/<your BrPolicy name?force=true(default:false)>'
```

To delete specific backup

```
1. HTTPS is not enabled for cbur-master  

curl -X DELETE -kL --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v2/images/<your helm release namespace>/<your BrPolicy name>?backup_id=<backup id>'  

2. HTTPS is enabled for cbur-master  

curl -X DELETE -kL --post301 --cert /opt/bcmt/config/cbur/client.pem --key /opt/bcmt/config/cbur/client-key.pem --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v2/images/<your helm release namespace>/<your BrPolicy name>?backup_id=<backup id>'
```

20.3.15 Backup and Recovery Scope

`cburScope="Namespaced" | "Cluster"`

- "Cluster" means Backup and Recovery's functions are open to all namespaces in a cluster.
- "Namespaced" means Backup and Recovery will be installed per namespace and its functions will be limited in its namespace, such as k8swatcher/role/backup/restore, and so on.

See following matrix for the Backup and Restore function limitation. **It is recommend to set `cburScope` to "Namespaced" in non-NCS openshift env, and to "Cluster" in NCS env.**

Table 70: Matrix

| <code>cburScope</code> | <code>rbac.enabled</code> | <code>securityContext.enabled</code> | <code>Result</code> |
|------------------------|---------------------------|--------------------------------------|---|
| Cluster | TRUE | TRUE | 1. CBUR will combine with cluster-admin role
2. CBUR pod and container run as root |
| Cluster | FALSE | TRUE | 1. CBUR will use the SA defined in security Context.serviceAccountName as the pod |

Table 70: Matrix (continued)

| cburScope | rbac.enabled | securityContext.enabled | Result |
|------------------|---------------------|--------------------------------|---|
| | | | <p>SA. The SA must be binding to specific roles to satisfy your need if "", cburm will failed to come up.</p> <p>2. If securityContext.runAsUser set to "auto" or securityContext.enabled=false:</p> <ul style="list-style-type: none"> • In openshift, it will run as arbitrary user. • In NCS, CBUR pod will come up with default user "root" in image. And may fail to be up if the SA is not privileged permission. <p>3. If securityContext.runAsUser and fsGroup is set to specific value:</p> <ul style="list-style-type: none"> • In openshift, be sure the value is in allowed range of corresponding SCC. • In NCS, be sure the value is in allowed range of corresponding PSP. |
| Namespaced | TRUE | TRUE | <p>1. Backup and Recovery will come up with basic roles that defined in helm chart 2. If securityContext.runAsUser set to "auto":</p> <ul style="list-style-type: none"> • In openshift, it will run as arbitrary user. • In NCS, CBUR pod will come up with default user "root" in image. And will failed to come up since "root" is not permitted for basic roles. <p>2. If securityContext.runAsUser and fsGroup is set to specific value:</p> <ul style="list-style-type: none"> • In openshift, be sure the value is in allowed range of corresponding SCC. • In NCS, be sure the value is in allowed range of corresponding PSP. |

Table 70: Matrix (continued)

| cburScope | rbac.enabled | securityContext.enabled | Result |
|------------|--------------|-------------------------|---|
| Namespaced | FALSE | TRUE | <p>1. Use the SA defined in securityContext.serviceAccountName as the pod SA. The SA must be binding to specific roles to statisfy your need if "", cburm will failed to come up.</p> <p>2. If securityContext.runAsUser set to "auto" or securityContext.enabled=false:</p> <ul style="list-style-type: none"> • In openshift, it will run as arbitrary user. • In NCS, CBUR pod will come up with default user "root" in image and may failed to come up since "root" is not permitted for basic roles. To resolve it, the SA must have previleged roles. <p>3. If securityContext.runAsUser anf fsGroup is set to specific value:</p> <ul style="list-style-type: none"> • In openshift, be sure the value is in allowed range of corresponding SCC. • In NCS, be sure the value is in allowed range of corresponding PSP. |

20.3.16 The Containerized Backup and Restore tool-a sidecar

To collect the volume data within a pod, Backup and Restore needs to insert a sidecar container to the pod and mount the target volume to the sidecar container.

The sidecar container is only used to tar/untar the volumes and copy the tarball to/from CBUR pod. The chart developers can also use their own container images (which includes GNU tar) for the sidecar container.

By default, Backup and Recovery will add the sidecar container on-demand, that is, when a backup or restore request is received and the sidecar container will be removed once the backup/restore task is finished.

There are two kinds of sidecar, user pre-defined sidecar and auto-injected sidecar.

- **auto-injected sidecar:**

- Before 18.12: with BCMT18.05, rbac is enabled in the BCMT cluster. "root" login will not be allowed. Backup and Recovery will add rbac related setting to the APP pod so that sidecar can come up with root. And rbac will setting will be removed after backup/restore finish.
 - From 18.12: Sidecar will be set the same fsGroup and runAsUser as your pod securityContext. If cannot find explicit securityContext definition, Backup and Recovery will try to find the uid / gid from the first container of hooks in BrPolicy.
- **pre-defined sidecar:** See *Pre-defined the Containerized Backup and Restore tool-a sidecar* on page 729 for details.

20.3.16.1 Pre-defined the Containerized Backup and Restore tool-a sidecar

Add Backup and Restore agent as sidecar container is **optional**.

In Kubernetes, the adding/removing of sidecar container will result in the creation of a new pod and then the termination of the old pod, this could result in service impact. Especially for the case when replicas equal to 1, there will be short service stop. If the above service impact is a concern, application chart developer could include the configuration of sidecar container in their chart file directly. Then, the sidecar container will always run in the pod. Following is the example deployment with pre-configured sidecar container.



Note:

- With BCMT18.05, rbac is enabled in the BCMT cluster.
- From 18.12: Pre-defined sidecar cbura should be configured with the same runAsUser and fsGroup as pod explicitly.
- Always use the latest cbura image in pre-defined the Containerized Backup and Restore tool-a sidecar.



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
  labels:
    app: {{ template "demo.fullname" . }}
    chart: demo-0.1.0
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ template "demo.fullname" . }}
  template:
    metadata:
```

```
labels:
  app: {{ template "demo.fullname" . }}
```

spec:

```
  securityContext:
    fsGroup: <your container's supplemental gid>
    runAsUser: <your container's uid>
    runAsGroup: <your container's gid>
```

containers:

```
  - name: demo
    image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/app/
yourapp:1.0-0"
    imagePullPolicy: IfNotPresent
    volumeMounts:
      - mountPath: /BACKUP1
        name: "app-data1"
      - mountPath: /BACKUP2
        name: "app-data2"
      - name: cbura-sidecar #<---predefined sidecar name, hardcoded. If any
another name used, cbur will not treat it as a valid predefined sidecar
        image: "csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
cbura:1.0.3-983"
        imagePullPolicy: IfNotPresent
    securityContext:
      runAsUser: <your container's uid>
      runAsGroup: <your container's gid>
    volumeMounts:
      - mountPath: /app-data1 #<---must have same name as the volume name
        name: "app-data1"
      - mountPath: /app-data2 #<---must have same name as the volume name
        name: "app-data2"
    resources:
      limits:
        cpu: 1
        memory: 256Mi
      requests:
        cpu: 50m
        memory: 128Mi
  volumes:
    - name: "app-data1"
      persistentVolumeClaim:
        claimName: demo-pv-claim
    - name: "app-data2"
      hostPath:
        path: /data
```

If you have multiple containers in the same pod with different uid / gid, please specify securityContext under cbura-sidecar with the same runAsUser / fsGroup as one of your containers that need to backup.

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
  labels:
    app: {{ template "demo.fullname" . }}
    chart: demo-0.1.0
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ template "demo.fullname" . }}
  template:
    metadata:
      labels:
        app: {{ template "demo.fullname" . }}
    spec:
      securityContext:
        fsGroup: <all containers' supplemental gid>
      containers:
        - name: demo1
          image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/app/
yourapp1:1.0-0"
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsUser: <the current container uid>
            runAsGroup: <the current container gid>
          volumeMounts:
            - mountPath: /BACKUP1
              name: "app-data1"
            - mountPath: /BACKUP2
              name: "app-data2"
        - name: demo2
          image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/app/
yourapp2:1.0-0"
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsUser: <the current container uid>
            runAsGroup: <the current container gid>
          volumeMounts:
```

```

    - mountPath: /BACKUP1
      name: "app-data1"
    - mountPath: /BACKUP2
      name: "app-data2"
  - name: cbura-sidecar #<--predefined sidecar name, hardcoded. If any
    another name used, cbur will not treat it as a valid predefined sidecar
    image: "csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
    cbura:1.0.3-983"
    imagePullPolicy: IfNotPresent
    securityContext:
      runAsUser: <One of your containers uid>
    volumeMounts:
      - mountPath: /app-data1 #<--must have same name as the volume name
        name: "app-data1"
      - mountPath: /app-data2 #<--must have same name as the volume name
        name: "app-data2"
    resources:
      limits:
        cpu: 1
        memory: 256Mi
      requests:
        cpu: 50m
        memory: 128Mi
    volumes:
      - name: "app-data1"
        persistentVolumeClaim:
          claimName: demo-pv-claim
      - name: "app-data2"
        hostPath:
          path: /data

```



Note: At present, if multiple containers in the same pod with different uid/gid all want to backup data, cbura only can be given one uid/gid of those containers. So make sure the backed up data can at least let cbura read it but cannot keep the data's original ownership after restoration.

Before 18.12: For pre-defined sidecar, there are two options:

- Using cbur (UID:1000, GID:1000) as the security context in the sidecar. You might not be able to backup/restore file due to linux access control. APP need to open access permission to cbur user. **This option is not suggested because there will be file ownership change issue if file ownership is cared a lot.**



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```
apiVersion: apps/v1beta2
```

```
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
  labels:
    app: {{ template "demo.fullname" . }}
    chart: demo-0.1.0
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ template "demo.fullname" . }}
  template:
    metadata:
      labels:
        app: {{ template "demo.fullname" . }}
    spec:
      containers:
        - name: demo
          image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/
app/yourapp:1.0-0"
            imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: /BACKUP1
              name: "app-data1"
            - mountPath: /BACKUP2
              name: "app-data2"
            - name: cbura-sidecar #<--predefined sidecar name, hardcoded.
If any another name used, cbur will not treat it as a valid predefined
sidecar
            securityContext:
              runAsUser: 1000 #<--Run sidecar with 'cbur'
              image: "csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
cbura:1.0.3-983"
            imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: /app-data1 #<--must have same name as the volume
name
              name: "app-data1"
            - mountPath: /app-data2 #<--must have same name as the volume
name
              name: "app-data2"
      resources:
        limits:
          cpu: 1
          memory: 256Mi
```

```

    requests:
      cpu: 50m
      memory: 128Mi
    volumes:
      - name: "app-data1"
        persistentVolumeClaim:
          claimName: demo-pv-claim
      - name: "app-data2"
        hostPath:
          path: /data

```

- using root(UID:0, GID:0) as the security context in the sidecar. APP must have there own serviceAccount and RolingBinding defined in CHART.



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```

apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
  labels:
    app: {{ template "demo.fullname" . }}
    chart: demo-0.1.0
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ template "demo.fullname" . }}
  template:
    metadata:
      labels:
        app: {{ template "demo.fullname" . }}
    spec:
      serviceAccountName: {{ template "demo.fullname" . }} #<--Add
      serviceAccountName with the name when define serviceAccount resource
      containers:
        - name: demo
          image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/
            app/yourapp:1.0-0"
            imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: /BACKUP1
              name: "app-data1"
            - mountPath: /BACKUP2
              name: "app-data2"

```

```

        - name: cbura-sidecar #<--predefined sidecar name, hardcoded.
If any another name used, cbur will not treat it as a valid predefined
sidecar
        securityContext:
            runAsUser: 0          #<--Run sidecar with root
            image: "csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
cbura:1.0.3-983"
            imagePullPolicy: IfNotPresent
            volumeMounts:
                - mountPath: /app-data1 #<--must have same name as the volume
name
                    name: "app-data1"
                - mountPath: /app-data2 #<--must have same name as the volume
name
                    name: "app-data2"
            resources:
                limits:
                    cpu: 1
                    memory: 256Mi
                requests:
                    cpu: 50m
                    memory: 128Mi
            volumes:
                - name: "app-data1"
                    persistentVolumeClaim:
                        claimName: demo-pv-claim
                - name: "app-data2"
                    hostPath:
                        path: /data

```

```

apiVersion: v1
kind: ServiceAccount
metadata:
    name: {{ template "demo.fullname" . }}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
    name: {{ template "demo.fullname" . }}
roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: psp:privileged
subjects:
    - kind: ServiceAccount

```

```
name: {{ template "demo.fullname" . }}
```

```
namespace: {{ .Release.Namespace }}
```

20.3.16.2 Attach volume for sidecar

Containerized Backup and Restore(CBUR) sidecar is used to collect volume data and the data is saved in /tmp folder which will use host storage. To not flood the host disk, attach specific volume is supported now.

For auto-injected sidecar, it is configurable in BrPolicy via --set
cburaVolume.storageClass=**, cburaVolume.storage=Gi.

For pre-defined sidecar, users can also attach a volume for sidecar. Backup and Restrore uses directory /tmp/ in sidecar for building tarball, so you can mount a volume to /tmp for pre-defined sidecar.

 **Note:** The folder /tmp in cbura sidecar consumes the host disk. If the target backup data is so large that it will take up lots of host disk space, you need to attach a separate volume to /tmp folder in cbura sidecar.

 **Note:** The following repository/registry URLs are provided as examples, the user should add their own.

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: {{ template "demo.fullname" . }}
  labels:
    app: {{ template "demo.fullname" . }}
    chart: demo-0.1.0
spec:
  replicas: 1
  selector:
    matchLabels:
      app: {{ template "demo.fullname" . }}
  template:
    metadata:
      labels:
        app: {{ template "demo.fullname" . }}
    spec:
      securityContext:
        fsGroup: <your container's supplemental gid>
        runAsUser: <your container's uid>
        runAsGroup: <your container's gid>
      containers:
        - name: demo
          image: "csf-docker-candidates.repo.lab.pl.alcatel-lucent.com/app/
yourapp:1.0-0"
          imagePullPolicy: IfNotPresent
```

```
volumeMounts:
  - mountPath: /BACKUP1
    name: "app-data1"
  - mountPath: /BACKUP2
    name: "app-data2"
  - name: cbura-sidecar #<---predefined sidecar name, hardcoded. If any
another name used, cbur will not treat it as a valid predefined sidecar
    image: "csf-docker-delivered.repo.lab.pl.alcatel-lucent.com/cbur/
cbura:1.0.3-983"
    imagePullPolicy: IfNotPresent
  securityContext:
    runAsUser: <your container's uid>
    runAsGroup: <your container's gid>
  volumeMounts:
    - mountPath: /app-data1 #<---must have same name as the volume name
      name: "app-data1"
    - mountPath: /app-data2 #<---must have same name as the volume name
      name: "app-data2"
    - mountPath: /tmp #<--- Attach a PVC to the folder /tmp in cbura
sidecar
      name: "cbura-tmp-pvc"
  resources:
    limits:
      cpu: 1
      memory: 256Mi
    requests:
      cpu: 50m
      memory: 128Mi
  volumes:
    - name: "app-data1"
      persistentVolumeClaim:
        claimName: demo-pv-claim
    - name: "cbura-tmp-pvc"
      persistentVolumeClaim:
        claimName: demo-cbura-tmp-pv-claim
    - name: "app-data2"
      hostPath:
        path: /data
```

20.3.16.3 Support of Special Characters for Login and Password

APIs

- Backup and Recovery with CKEY authentication

Username cannot be modified currently. Password works with all the special characters: !@#\$%^&*()?:;`{|}+=_-.,;

- Backup and Recovery with Basic authentication

Currently, Backup and Recovery with Basic authentication does not support to modify username and password.

Portal

- Backup and Restore with CKEY authentication

Username cannot be modified currently. Password works with all the special characters: !@#\$%^&*()?:;`{|}+=_-.,;

- Backup and Recovery with CKEY authentication

Backup and Recovery with Basic authentication does not support to modify username and password.

20.3.17 BrPolicy

For application backup and restore, write a Backup and Recovery policy file is **mandatory**. Backup and Recovery defines a Custom Resource, the kind is: BrPolicy.

From 20.03, Backup and Recovery can be configured to enable BrPolicy schema validation. In Backup and Recovery chart, enableBrSchemaValidation is false by default.

Starting from Backup and Recovery 20.10 FP1 (that is, cbur-1.4.5), enableBrSchemaValidation is set to true, which means BrPolicies are checked for schema validation by default. Make sure your BrPolicies definition strictly follow the specification below, or it will fail to install. It is not required that all the parameters specified below included in your BrPolicies, but for the parameters that included in your BrPolicy, it must follow the specification below.

The preBackupCmd / postBackupCmd / preRestoreCmd / postRestoreCmd are monitored by the kubelet parameter streamingConnectionIdleTimeout. If the commands have no output longer than the timer, the connection to the pod is disconnected. So make sure to have output periodically.

- For preBackupCmd / preRestoreCmd, Backup and Recovery will report error on streamingConnectionIdleTimeout times out.
- For postBackupCmd / postRestoreCmd, the Containerized Backup and Restore tool cannot detect the timeout of streamingConnectionIdleTimeout, since kubectl exec returns code 0, the Containerized Backup and Restore tool will think the post commands have completed although they are actually running.

In a BCMT environment, streamingConnectionIdleTimeout is 60 mins. But in other environments, the value may be different.

Following is an example of BrPolicy configuration for a demo application:

```

apiVersion: "cbur.csf.nokia.com/v1"      #the apiVersion "cbur.bcmt.local/v1" is DEPRECATED and will be removed in cbur-1.9.0. Please update with "cbur.csf.nokia.com/v1".
kind: BrPolicy
metadata:
  name: {{ template "demo.fullname" . }} #--must have the same name as deployment/statefulset/daemonset which you want to backup
spec:
  autoEnableCron: false
  autoUpdateCron: false
  volumes:
    - app-data1    #---volume name you want to backup. NOTE: not mount path
    - app-data2
  weight: {{ .Values.brpolicyWeight.applBrPolicyA }}    #integer, default: 0, only supported for the Containerized Backup and Restore tool 20.03 and later
  backend:
    mode: "local"          #--Modes supported now:
    "local", "AVAMAR", "CEPHS3", "AWSS3", "SWIFTS3", "SFTP", case insensitive
    cburaVolume: #-- If defined, will use for auto-injected sidecar and pvc backup
    storageClass: glusterfs-storageclass    #--storage class type user want to define
    storageSize: 1Gi    #-- storage size user want to define
    resources:    #-- If defined, will use for auto-injected sidecar and pvc backup
    limits:
      cpu: 1    #-- limit cpu configuration for auto-injected sidecar
      memory: 50Mi  #-- limit memory configuration for auto-injected sidecar
    requests:
      cpu: 100m  #-- request cpu configuration for auto-injected sidecar
      memory: 20Mi  #-- request memory configuration for auto-injected sidecar
  bypassPod:
    matchLabels:
      key: value    #-- bypass the pods for performing backup
  selectPod:
    matchLabels:
      key: value    #-- select the pods for performing backup
  cronSpec: "*/*10 * * * *"  #--cronjob frequency, here means every 10 minutes of every day
  dataEncryption:
    enable: true        #--Whether to encrypt the backed up data or not.
    secret: appl-br

```

```
k8sType: statefulset      #<--deployment/statefulset/daemonset/", case
insensitive. Please remove this field or leave it empty if the application
is not a deployment/daemonset/statefulset. This field is newly added for
CSF18.09.

brOption: [0|1|2]          #<--This value only applies to statefulset (0 or 1
or 2) and daemonset (0 or 1).

maxiCopy: 5               #<--the maximum copy you want to saved.

ignoreFileChanged: false  #<--Whether to ignore the file change(s) or not
when creating tar file in cbura sidecar.

k8sobjects:
  - object-type: [string]
    match-criteria: [name | label]
    label-key: [dict]
    match-string: [string]
  - object-type: [string]
    match-criteria: [name | label]
    label-key: [dict]
    match-string: [string]

pvc:
  - matchCriteria: [fullname | startname | endname | label]
    matchLabels: [dict]
    matchString: [string]
  - matchCriteria: [fullname | startname | endname | label]
    matchLabels: [dict]
    matchString: [string]

hooks:
  - name: demo-app-container-name  #<--container name in the pod where you
want to execute hooks.

  commands:
    preBackupCmd: ["sh", "-c", "echo backup start"]           #<--command will
be executed after sidecar insert
    preBackupCmdOnAll: false                                    #<--Whether to run
preBackupCmd on all pods, default: false
    postBackupCmd: []                                         #<--command will
be executed before removing sidecar and after backup(scp)
    postBackupCmdOnAll: false                                #<--Whether to run
postBackupCmd on all pods, default: false
    preRestoreCmd: []                                       #<--command will
be executed after sidecar insert
    preRestoreCmdOnAll: false                               #<--Whether to run
preRestoreCmd on all pods, default: false
    postRestoreCmd: ["sh", "-c", "echo restore finished"] #<-- command will
be executed before removing sidecar and after retore(scp)
    postRestoreCmdOnAll: false                            #<--Whether to run
postRestoreCmd on all pods, default: false
```

- `metadata.name`: should be same as the deployment/statefulset/daemonset name in the application's chart file.
- `spec.volumes`: <list> a list of the volume name that you want to do backup, that is, Backup and Recovery will backup the volume data to a persistent volume.
- `spec.weight`: <integer> (default: 0) Multiple BrPolicies will be run in weighted order with the lowest weight first (negative to positive). If several BrPolicies have the same weight, they will be run randomly. Only supported for the Containerized Backup and Restore tool 20.03 and later.
- `spec.backend.mode`: <string> The mode can be one of: "local", "AVAMAR", "AWSS3", "CEPHS3", "SWIFTS3" and "SFTP", case insensitive. If not provided, will use "local" as default.



Note: The mode will be overwritten by GLOBAL_BACKEND which is set in cbur chart values.yaml.

- `spec.cronSpec`: <string> The schedule time to run backup. Empty string is allowed if there is no need to schedule backup cronjob. It has 5 fields: 'minute', 'hour', 'day of the month', 'month', 'day of the week'. Each field supports "" or "number" or "/number". For example, * /5 in the minutes field indicates every 5 minutes. Other cron expressions (for example, comma ",", minus sign "-", percent "%", "L", "W", hash "#", "?", "H", macros "@yearly"/"@hourly"/"reboot") are not supported currently.
- `spec.dataEncryption.enable`: <boolean> (default: true). When "false", Backup and Recovery will not encrypt the backed up data. This option does not apply to cluster backup.
- `spec.dataEncryption.secret`: <string> If this field is defined and not empty, Backup and Recovery will get the passphrase from the secret and use it to encrypt/decrypt the tarballs during backup/restore.
- `spec.maxiCopy`: <integer> Limit the number of copies that can be saved. Once it is reached, the oldest one will be deleted. The range for maxiCopy is [1,100].
- `spec.hooks`: <list> specify the container name and it is hook commands. "preBackupCmdOnAll / postBackupCmdOnAll / preRestoreCmdOnAll / postRestoreCmdOnAll" are used to control whether to run the corresponding hook commands on all pods (including the pod to do backup/restore and other pods which will not do backup/restore). If the value is not 'true' or not defined, the hook logic will be as before, that is, the commands will only be run on the pod which is selected to do backup/restore.
- `spec.cburaVolume`: specify the type and size for CBUR sidecar volume
- `spec.cburaVolume.storageClass`: <string>
- `spec.cburaVolume.storageSize`: <string>
- `spec.resources`: specify the limits and requests cpu and memory
- `spec.resources.cpu`: <string or int>, for example, 1 or 100m
- `spec.resources.memory`: <string>
- `spec.bypassPod`: bypass the pods for performing backup

- `spec.bypassPod.matchLabels`: <dict>label must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character . If `spec.bypassPod` exists, `spec.bypassPod.matchLabels` is required.
- `spec.selectPod`: select the pods for performing backup
- `spec.selectPod.matchLabels`: <dict>label must consist of lower case alphanumeric characters or '-', and must start and end with an alphanumeric character. If `spec.selectPod` exists, `spec.selectPod.matchLabels` is required.
 - When choosing which pod(s) to backup, the Containerized Backup and Restore tool-m will first eliminate all pods that match the labels in the `bypassPod` field (if it is present).
 - Then of the remaining pods, the Containerized Backup and Restore tool-m will eliminate any that do not match the labels in the `selectPod` field (if present).
 - If there are remaining pods match `bypassPod` and `selectPod` labels, then do backup, otherwise, raise alarm.
 - If the backup is for a stateful set, Backup and Recovery will apply the rule specified by the `brOption` field to the remaining selected pods. And if `brOption` is 2, and the number of remaining selected pods is less than the number of all pods, then backup will fail and throw fail message.
- `spec.k8sType`: <string> specify the app Kubernetes type. The values should be deployment/statefulset/daemonset, case insensitive. Backup and Recovery also supports Kubernetes objects backup and restore for application which is not Deployment or StatefulSet or DaemonSet. For such application, remove the field "k8sType" or leave it empty.
- `spec.autoEnableCron`: <boolean> If BrPolicy contains `spec.cronspec` that is not empty, `autoEnableCron = true` indicates that the cron job is immediately scheduled when the BrPolicy is created, while `autoEnableCron = false` indicates that scheduling of the cron job should be done on a subsequent backup request. This option only works when `k8swatcher.enabled` is true.
- `spec.autoUpdateCron`: <boolean> (default: false).Indicate if subsequent update of cronjob will be done via brpolicy update. true means cronjob must be updated via brpolicy update, false means cronjob must be updated via manual "helm backup -t app -a enable/disable" command.
- `spec.brOption`: <integer> This value only applies to statefulset and daemonset. For statefulset, the value can be 0,1 or 2. For daemonset, the value can be 0 or 1.
 - "0": backup any one of the PODs and restore to any one of PODs.
 - "1": backup any one of the POD, and restore it to all PODs one by one by its index.
 - "2": backup all PODs and restore them one by one based by its index.
- `spec.k8sobjects`: <list> For details, refer to chapter k8s-objects-backuprestore
- `spec.preBackupCmd`: <list> it must be list format. The same rule for `spec.postBackup`, `spec.preRestoreCmd`, `spec.postRestoreCmd`.
- `spec.ignoreFileChanged`: <boolean> (default: false). When "true", Backup and Recovery will ignore the following issues when creating tar file in cbura sidecar:

- "File changed as we read it"
- "File removed before we read it"

This option doesn't apply to the tar operation in cbur-master (that is, backup cbur-master self volume(s), BCMT cluster backup, and application's Kubernetes objects backup). For details, refer to CSFLCM-3663.

The backupHistory, requestHistory and alarms are recorded in BrPolicy's status section which is updated by Backup and Recovery automatically and not user-configurable.

- .status.backupHistory: a list of backups (images). Avamar backup history will not be updated here.
- .status.requestHistory: the backup or restore request's result will be recorded here. If enabled celery, you can check this section for task ID and then manually check the backup or restore result by this task ID. And if enabled scheduled backup, you can check this section for backup result or task ID (if enabled celery).
- .status.scheduleEnable: to show the latest schedule enable/disable status. It will be added/updated after backup with action set to enable/disable.
- .status.alarms: if backup or restore failed, related alarms are recorded here to let next success clear them. They also are put in the stdout of Backup and Recovery.



Note: There are at most 20 result records for each request in BrPolicy "requestHistory". If exceeding 20 records, CBUR will remove the oldest one from requestHistory. Before 19.06, backupHistory is located in spec and if your cluster is upgraded from old versions to 19.06, then you can still see spec.backupHistory but it will not be used anymore.

The BrPolicy file is designed for backup/restore per statefulset/deployment/daemonset. For umbrella charts scenario when multiple statefulset/deployment/daemonset are included in a helm/chart release, you can create multiple BrPolicy files.

20.3.18 Async

Async Backup and Restore with celery

Install or Upgrade

By default, the async function is disabled. The Containerized Backup and Restore tool's async function uses Celery as the async task queue and Redis as message broker.

Before the Containerized Backup and Restore tool 20.10 FP1 (i.e. cbur-1.4.5), the Containerized Backup and Restore tool supports two ways for redis function:

- CRDB: the Containerized Backup and Restore tool helm chart includes crdb-redisio sub chart which is in charge to create the redis container.
- redis image: the Containerized Backup and Restore tool uses redis image and creates the redis container directly

Backup or Restore

```
$ helm backup -t <app> -a <action>
```

Or:

```
$ helm restore -t <app> -b <backup_id>
```

cbur-m passes the backup and restore job to celery. Status code 200 is returned with the task ID.

```
{
  "apiVersion": "v1",
  "code": 202,
  "details": {
    "3e4a164d-d9b7-49c7-b527-23b46f877ac8": {
      "action": "none",
      "fn": "backup_name",
      "name": "app1",
      "namespace": "default"
    }
  },
  "kind": "Status",
  "message": "backup_name task = 3e4a164d-d9b7-49c7-b527-23b46f877ac8 is on!",
  "metadata": {},
  "reason": "",
  "status": "Success"
}
```

Use the following commands to query the backup or restore result:

```
helm backup -i 3e4a164d-d9b7-49c7-b527-23b46f877ac8
helm restore -i e6e0aaa6-ea98-492e-b03d-a1bdbd0ae83a
```

Or:

```
curl -X GET --header 'Content-Type: multipart/form-data' --header 'Accept: application/json' 'http://cbur-master-cbur.ncms.svc:80/v1/task/3e4a164d-d9b7-49c7-b527-23b46f877ac8'
```



Note:

- **ncms** is the namespace that cbur-m installed in.
- **80** is the server port of cbur-m.

- **3e4a164d-d9b7-49c7-b527-23b46f877ac8** is the task_id got after helm backup/ restore.

20.3.19 Containerized Backup and Restore tool Installation Configuration

The detailed configuration of cbur-master is set in values.yaml of the Containerized Backup and Restore tool helm chart.

General parameter definitions of values.yaml are defined as follows.

 **Note:** The following repository/registry URLs are provided as examples, the user should add their own.

| Parameter | Description | Default |
|-----------------------------|--|--|
| global.registry | Global container image registry | csf-docker-delivered.repo.lab.pl.alcatel-lucent.com |
| global.registry1 | Global container image registry1 | csf-docker-candidates.repo.lab.pl.alcatel-lucent.com |
| global.registry2 | Global container image registry2 | csf-docker-inprogress.repo.lab.pl.alcatel-lucent.com |
| global.podNamePrefix | The name prefix of the Containerized Backup and Restore tool pods | nil |
| global.containerName Prefix | The name prefix of the Containerized Backup and Restore tool containers | nil |
| global.annotations | These annotations will be added to all the Containerized Backup and Restore tool pods (including cronjob pod). | nil |
| custom.pod.annotations | These annotations will be added to the Containerized Backup and Restore tool pods (not including cronjob pod). | nil |
| custom.job.annotations | These annotations will be added to the Containerized Backup and Restore tool cronjob pod. | nil |

| Parameter | Description | Default |
|--------------------------|---|------------------|
| replicaCount | Desired number of cbur replicas. It only supports "1" for now. | 1 |
| image.master.name | The Containerized Backup and Restore tool master image name | cbur/cburm |
| image.master.tag | The Containerized Backup and Restore tool master image tag | {TAG_NAME} |
| image.master.pullPolicy | the Containerized Backup and Restore tool master image pull policy | IfNotPresent |
| image.master.registry | the Containerized Backup and Restore tool master image registry | registry |
| image.agent.name | cbura sidecar image name | cbur/cbura |
| image.agent.tag | cbura sidecar image tag | {TAG_NAME} |
| image.agent.pullPolicy | cbura sidecar image pull policy | IfNotPresent |
| image.agent.registry | cbura sidecar image registry | registry |
| image.croncli.name | The Containerized Backup and Restore tool cronjob image name | cbur/cbur-cli |
| image.croncli.tag | The Containerized Backup and Restore tool cronjob image tag | {TAG_NAME} |
| image.croncli.pullPolicy | The Containerized Backup and Restore tool cronjob image pull policy | IfNotPresent |
| image.croncli.registry | The Containerized Backup and Restore tool cronjob image registry | registry |
| image.avamar.name | The Containerized Backup and Restore tool Avamar client image name | cbur/cbur-avamar |
| image.avamar.tag | The Containerized Backup and Restore tool Avamar client image tag | {TAG_NAME} |

| Parameter | Description | Default |
|-------------------------|--|--------------|
| image.avamar.pullPolicy | The Containerized Backup and Restore tool Avamar client image pull policy | IfNotPresent |
| image.avamar.registry | The Containerized Backup and Restore tool Avamar client image registry | registry |
| component | Component name used as part of the label. | master |
| partof | All Kubernetes objects created by the Containerized Backup and Restore tool will have label: app.kubernetes.io/part-of: {{ .Values.partOf }} | cbur |
| fullnameOverride | Use this to configure the fullname, which will be used as (or as part of) the name of Kubernetes objects of the Containerized Backup and Restore tool. If both nameOverride and fullnameOverride are specified, fullnameOverride will take the precedence. | nil |
| nameOverride | When it is set and not part of the Release Name, the fullname would be "Release Name-nameOverride". | nil |
| cburScope | Only supports "Cluster" and "Namespaced". "Cluster" means the Containerized Backup and Restore tool can serve the whole cluster, e.g. backup / restore the apps in any namespace; "Namespaced" means the Containerized Backup and Restore tool can only backup/ restore the apps in the same namespace as the Containerized Backup and Restore tool. | Cluster |
| accessAllResources | This only takes effect when cburScope is "Namespaced". <ul style="list-style-type: none"> • If "true", the Containerized Backup and Restore tool will add access ("get", "list", "create", "update", "patch") to all resources in the Kubernetes namespace. | true |

| Parameter | Description | Default |
|-----------|--|---------|
| | <ul style="list-style-type: none"> If "false", the Containerized Backup and Restore tool only adds limited access to limited resources in the Kubernetes namespace. <p>For details, refer to "basic_role.yaml" in the Containerized Backup and Restore tool chart or by "kubectl get role -basic-namespaced -o yaml". If user wants permissions in between, the user provides the own service account and adds related roles and installs the Containerized Backup and Restore tool with "--set rbac.enabled=false,rbac.serviceAccountName=xxx". For "Cluster" scoped the Containerized Backup and Restore tool, the Containerized Backup and Restore tool cluster role binds the Containerized Backup and Restore tool's service account to "cluster-admin".</p> | |
| adminRole | <p>This only takes effect when cburScope is "Namespaced".</p> <ul style="list-style-type: none"> If "true" and accessAllResources is "false", the Containerized Backup and Restore tool will bind the admin to serviceaccount. If "false" and accessAllResources is "false", the Containerized Backup and Restore tool only adds limited access to limited resources in the Kubernetes namespace. <p>For details, please refer to "basic_role.yaml" in the Containerized Backup and Restore tool chart or by "kubectl get role -basic-namespaced -o yaml". If user wants permissions in between, the user provides the own service account and adds related roles and installs the Containerized Backup and Restore tool with "--set rbac.enabled=false,rbac.serviceAccountName=xxx". For "Cluster" scoped the Containerized Backup and Restore tool, the Containerized Backup and Restore tool cluster role binds the Containerized Backup and Restore tool's service account to "cluster-admin".</p> | true |

| Parameter | Description | Default |
|---------------------------|---|---------|
| serverPort | The Containerized Backup and Restore tool http port | 80 |
| resources.limits.cpu | Limits CPU computation of 'cbur' container | 3 |
| resources.limits.memory | Limits memory size of 'cbur' container | 1Gi |
| resources.requests.cpu | Requests CPU computation of "cbur" container | 300m |
| resources.requests.memory | Requests memory size of 'cbur' container | 350Mi |
| rbac.enabled | "true" means the Containerized Backup and Restore tool will add ServiceAccount and the related permissions, and the Containerized Backup and Restore tool pods will use this ServiceAccount; "false" means to the Containerized Backup and Restore tool pods will use the "rbac.serviceAccountName" | true |
| rbac.serviceAccountName | The serviceAccountName used by the Containerized Backup and Restore tool pods when "rbac.enabled" is "false" | nil |
| securityContext.enabled | If "true", add securityContext to the Containerized Backup and Restore tool pods and container. | true |
| securityContext.runAsUser | If "auto", add "securityContext: " to "cbur" container; otherwise, use it as the runAsUser of "cbur" container. | auto |
| securityContext.fsGroup | If "auto", add "securityContext: " to cbur pods; otherwise, use it as the fsGroup of cbur pods. | auto |
| https.enabled | Specifies if https is enabled for API request | false |

| Parameter | Description | Default |
|----------------------------|--|-----------------------|
| https.port | The Containerized Backup and Restore tool https port | 443 |
| https.certsPath | The host path that includes https cert files when https.tls.enabled is "false". | /opt/bcmt/config/cbur |
| https.tls.enabled | If "true", get the https certs from the fields in "https.tls", instead of from https.certsPath | false |
| https.tls.client_cert | The client private key | nil |
| https.tls.client_key | The client public cert file | nil |
| https.tls.ca | The private key of the root CA | nil |
| https.tls.server_key | The sever public cert file | nil |
| https.tls.server_cert | The server private key | nil |
| volumeType.glusterfs | If "true", mount the static glusterfs volumes specified in "volumeType" as the Containerized Backup and Restore tool volumes. If false, create and mount PVCs as the Containerized Backup and Restore tool volumes. When volumeType.glusterfs is true, users need to manually create the GlusterFS endpoints and volumes. If you are using NCS, NCS has created these volumes by default. But it's not recommended to use the default ones since they are sharing the "/data0" of NCS. Please refer to https://github.com/kubernetes/examples/tree/master/volumes/glusterfs | true |
| volumeType.backupEndpoints | Glusterfs endpoints for "/BACKUP" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. The "/BACKUP" volume is used as a temporary storage when backup/restore application data to/from third-party storage server. | glusterfs-cluster |

| Parameter | Description | Default |
|-----------------------------------|--|-------------------------------|
| volumeType.backupPath | Glusterfs volume name for "/BACKUP" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. | cbur-glusterfs-backup |
| volumeType.repoEndpoints | Glusterfs endpoints for "/CBUR_REPO" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. The "/CBUR_REPO" volume is used as the local storage. When the spec. backend.mode is "local" in BrPolicy, the data is saved here. | glusterfs-cluster |
| volumeType.repoPath | Glusterfs volume name for "/CBUR_REPO" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. | cbur-glusterfs-repo |
| volumeType.backupClusterEndpoints | Glusterfs endpoints for "/CLUSTER" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. The "/CLUSTER" volume is used when backup/restore NCS cluster. If the backend is not "local", it's used as a temporary storage during the process of cluster backup/restore; If the backend is "local", the cluster backup data is also saved here. For cluster backup/restore, the Containerized Backup and Restore tool only supports NCS cluster. So this is only needed when user needs to backup/restore NCS cluster. | glusterfs-cluster |
| volumeType.backupClusterPath | Glusterfs volume name for "/CLUSTER" volume in the Containerized Backup and Restore tool. It is used only when "volumeType.glusterfs=true". Must exist before used. | cbur-glusterfs-backup-cluster |
| isPvRwx | When both volumeType.glusterfs and directPvc.enabled are false, the Containerized Backup and Restore tool will create and mount the PVCs as specified in the following parameters (storageBackup / storageRepo / storage | false |

| Parameter | Description | Default |
|----------------------|--|---------|
| | <p>BackupCluster).The "isPvRwx" indicates whether the accessModes of these PVCs is ReadWriteMany.</p> <p> Note: Only when isPvRwx is true or volumeType.glusterfs is true, Avamar can be used.</p> <p>.</p> | |
| storageBackup | When both volumeType.glusterfs and directPvc.enabled are false, a pv will be created by PVC with this size using the StorageClass specified in parameter storageClass. the Containerized Backup and Restore tool will mount it as "/BACKUP" volume. The "/BACKUP" volume is used as a temporary storage when backup/restore application data to/from third-party storage server. | 2Gi |
| storageRepo | When both volumeType.glusterfs and directPvc.enabled are false, a pv will be created by PVC with this size using the StorageClass specified in parameter storageClass. the Containerized Backup and Restore tool will mount it as "/CBUR_REPO" volume. The "/CBUR_REPO" volume is used as the local storage. When the spec.backend.mode is "local" in BrPolicy, the data is saved here. | 8Gi |
| storageBackupCluster | When both volumeType.glusterfs and directPvc.enabled are false, a pv will be created by PVC with this size using the StorageClass specified in parameter storageClass. The Containerized Backup and Restore tool will mount it as "/CLUSTER" volume. The "/CLUSTER" volume is used when backup/restore NCS cluster. If the backend is not "local", it is used as a temporary storage during the process of cluster backup/restore; If the backend is "local", the cluster backup data is also saved here. For cluster backup/restore, the Containerized Backup and Restore tool only supports NCS cluster. So this is only needed when user needs to backup/restore NCS cluster. | 8Gi |

| Parameter | Description | Default |
|-------------------------------------|--|---------|
| storageClass | This storageclass will be used to create above cbur-master volumes (storage Backup / storageRepo / storageBackup Cluster) when both volumeType .glusterfs and directPvc.enabled are false. When use default "", the default storageclass will be used. | nil |
| directPvc.enabled | If volumeType.glusterfs is false and directPvc.enabled is true, the Containerized Backup and Restore tool will use the existing PVCs (not created by the Containerized Backup and Restore tool) specified in directPvc as the Containerized Backup and Restore tool volumes. | false |
| directPvc.backupPvcName | PVC for the Containerized Backup and Restore tool "/BACKUP" volume when volumeType.glusterfs is false and directPvc.enabled is true. | nil |
| directPvc.repoPvcName | PVC for the Containerized Backup and Restore tool "/CBUR_REPO" volume when volumeType.glusterfs is false and directPvc.enabled is true. | nil |
| directPvc.backupClusterPvcName | PVC for the Containerized Backup and Restore tool "/CLUSTER" volume when volumeType.glusterfs is false and directPvc.enabled is true. | nil |
| storageMonitor.enabled | Enable volume monitor for the Containerized Backup and Restore tool volumes: /BACKUP, /CBUR_REPO, /CLUSTER | true |
| storageMonitor.backup_low_threshold | Check disk usage before application level backup/restore to third party. Fire warning alarm 'VOLUME_EXCEED_THRESHOLD' if the disk usage of '/BACKUP' reaches this threshold, but does not reach the storageMonitor.backup_high_threshold | 75% |
| storageMonitor.repo_low_threshold | Check disk usage before application level backup/restore to local. Fire warning alarm 'VOLUME_EXCEED_THRESHOLD' if | 75% |

| Parameter | Description | Default |
|--|--|---------|
| | the disk usage of '/CBUR_REPO' reaches this threshold, but does not reach the storageMonitor.repo_high_threshold | |
| storageMonitor.cluster_backup_low_threshold | Check disk usage before cluster level backup/restore. Fire warning alarm 'VOLUME_EXCEED_THRESHOLD' if the disk usage of '/CLUSTER' reaches this threshold, but does not reach the storage Monitor.cluster_backup_high_threshold | 75% |
| storageMonitor.backup_high_threshold | Check disk usage before application level backup/restore to third party. Fail backup/restore and fire critical alarm 'VOLUME_EXCEED_THRESHOLD' if the disk usage of '/BACKUP' reaches this threshold. | 85% |
| storageMonitor.repo_high_threshold | Check disk usage before application level backup/restore to local. Fail backup/restore and fire critical alarm 'VOLUME_EXCEED_THRESHOLD' if the disk usage of '/CBUR_REPO' reaches this threshold. | 85% |
| storageMonitor.cluster_backup_high_threshold | Check disk usage before cluster level backup / restore. Fail backup/restore and fire critical alarm 'VOLUME_EXCEED_THRESHOLD' if the disk usage of '/CLUSTER' reaches this threshold. | 85% |
| cburaVolume.enabled | <p>If true, create a PVC with the default StorageClass and mount it to "/tmp" of cbura auto-injected sidecar. It will be used to save temporary data during application backup/restore. If false, the temporary data will consume the node storage on which the pod runs.</p> <p> Note: The "cburaVolume" setting in BrPolicy has the priority. That is, cburaVolume in BrPolicy > cburaVolume in the Containerized Backup and Restore tool helm values > use node storage</p> | false |

| Parameter | Description | Default |
|----------------------|--|---------|
| cburaVolume.storage | "/tmp" volume size when cburaVolume.enabled is true. | 500Mi |
| GLOBAL_BACKEND | If set, it will override the ".spec.backend.mode" in application's BrPolicy. This setting does not apply to cluster backup/restore. | nil |
| CEPHS3.endpoint | The CEPH S3 storage endpoint. For example, <code>http://10.67.40.80:8080/</code> or <code>https://10.67.40.80:8080/</code> | nil |
| CEPHS3.access_key | The AccessKey of CEPH S3 storage. If user also creates secret "cburm-s3-key" in application's namespace, the key in secret will be used. | nil |
| CEPHS3.secret_key | The SecretKey of CEPH S3 storage. If user also creates secret "cburm-s3-key" in application's namespace, the key in secret will be used. | nil |
| CEPHS3.uid | The "uid" is used as part of CEPH S3 bucket name. It is not used for authentication. It can be any string consisting of lowercase letters, numbers, dots(.) (not recommended), and hyphens ('-'). User may define it as the user ID who created the credentials ("AccessKey" and "SecretKey"), but this is not mandatory. If user also creates secret "cburm-s3-key" in application's namespace, the uid in secret will be used. | nil |
| CEPHS3.bucket_prefix | The bucket prefix for the CEPH S3 bucket name. If not specified, the "clusterId" OR "clusterName" will be used. | nil |
| CEPHS3.ca_crt | It contains the SSL certificates for HTTPS endpoint. If it is not defined, do not verify SSL certificates. Enter the base64 encoded string of SSL CA cert: <code>cat ca.crt base64 -w 0</code> | nil |
| AWSS3.access_key | The access key ID of AWS S3 storage. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore | nil |

| Parameter | Description | Default |
|----------------------|---|---------|
| | tool's namespace, the key in secret will be used. | |
| AWSS3.secret_key | The secret access key of AWS S3 storage. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore tool's namespace, the key in secret will be used. | nil |
| AWSS3.access_token | The session token of AWS S3 storage. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore tool's namespace, the token in secret will be used. | nil |
| AWSS3.region | The region name of AWS S3 storage. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore tool's namespace, the region name in secret will be used. | nil |
| AWSS3.uid | The "uid" is used as part of AWS S3 bucket name. It is not used for authentication. It can be any string consisting of lowercase letters, numbers, dots(.) (not recommended), and hyphens ('-'). User may define it as the AWS account, but this is not mandatory. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore tool's namespace, the uid in secret will be used. | nil |
| AWSS3.http_proxy | http_proxy that will be used to access the AWS s3. If user also creates secret "cburm-s3-key" in the Containerized Backup and Restore tool's namespace, the http_proxy in secret will be used. | nil |
| AWSS3.bucket_prefix | The bucket prefix for the AWS S3 bucket name. If not specified, the "clusterId" OR "clusterName" will be used. | nil |
| AWSS3.access_logging | If true, will enable AWS S3 server access logging. A separate bucket "-logs" will be created to save the access logs. If false, will disable AWS S3 server access logging. | false |

| Parameter | Description | Default |
|-------------------------|--|---------|
| AWSS3.predefined_bucket | The existing predefined bucket for AWSS3 data storage. If provided, cbur will not create new bucket | nil |
| swiftS3.endpointUrl | The Swift S3 storage endpoint. For example, <code>http://10.67.40.80:8080/</code> or <code>https://10.67.40.80:8080/</code> | nil |
| swiftS3.accessKeyId | The access key ID of Swift S3 storage. If user also creates secret "cburm-swifts3-config" in application's namespace, the key in secret will be used. | nil |
| swiftS3.secretAccessKey | The secret access key of Swift S3 storage. If user also creates secret "cburm-swifts3-config" in application's namespace, the key in secret will be used. | nil |
| swiftS3.region | The region name of Swift S3 storage. If user also creates secret "cburm-swifts3-config" in the Containerized Backup and Restore tool's namespace, the region name in secret will be used. | nil |
| swiftS3.uid | The "uid" is used as part of Swift S3 bucket name. It is not used for authentication. It can be any string consisting of lowercase letters, numbers, dots (.) (not recommended), and hyphens ('-'). User may define it as the user id who created the credentials ("accessKeyId" and "secretAccessKey"), but this is not mandatory. If user also creates secret "cburm-swifts3-config" in application's namespace, the uid in secret will be used. | nil |
| swiftS3.bucketPrefix | The bucket prefix for the Swift S3 bucket name. If not specified, the "clusterId" OR "clusterName" will be used. | nil |
| swiftS3.caCrt | It contains the SSL certificates for https endpoint. If it's not defined, do not verify SSL certificates. Enter the base64 encoded string of SSL CA cert: <code>cat ca.crt base64 -w 0</code> If user also creates secret "cburm-swifts3-config" | nil |

| Parameter | Description | Default |
|---------------------------|---|------------|
| | in application's namespace, the caCrt in secret will be used. | |
| SSH.mode | Only support "sftp" currently. | sftp |
| SSH.username | The username used to access sftp server. | nil |
| SSH.host | The IP address or FQDN of sftp server. | nil |
| SSH.port | The port of sftp server. | 22 |
| SSH.path | The path on sftp server to save the backed up data. | nil |
| SSH.strictHostKeyChecking | <p>Whether to do host key checking for sftp server:</p> <ul style="list-style-type: none"> If the argument is set to "yes", SSH will use the value of "hostKey" to check the host key of sftp server. If the argument is set to "no", SSH will skip the host key checking even if "hostKey" is set. If the argument is set to "autoadd" , ssh will automatically add new host key to the user known hosts file when connecting for the first time and ssh will refuse to connect to hosts whose host key has changed. If "hostKey" is also set, the behavior will be like "yes", for example, the Containerized Backup and Restore tool will use the value of "hostKey" to do host key checking and will not add host key to known hosts file. For "yes" and "autoadd", when the host key has changed, user needs to update the new value to the secret "cburm-ssh-config" or use "helm upgrade" to update the 'hostKey' field. But please be cautious that the "helm upgrade" will cause pods to | !!str "no" |

| Parameter | Description | Default |
|----------------------|---|---|
| | restart, for detailed steps, refer to section "Upgrade procedure" in the Containerized Backup and Restore tool user guide. | |
| SSH.hostKey | The Containerized Backup and Restore tool will use this host key to do host key checking. This field only takes effect when SSH.strictHostKeyChecking is set to "yes" or "autoadd". | nil |
| nodeSelectorOverride | There is helm bug before version 3.2, that nodeSelector cannot be unset to null or other value(s) via --set. Use this parameter to override the default value of nodeSelector. | { } |
| nodeSelector | nodeSelector for the Containerized Backup and Restore tool pods. | is_control: 'true' |
| tolerations | tolerations for the Containerized Backup and Restore tool pods. | <pre>- effect: NoExecute
 key: is_control
 operator: Equal
 value: "true"
 -
 effect: NoExecute
 key: is_edge
 operator: Equal
 value: "true"
 -
 effect: NoExecute
 key: is_storage
 operator: Equal
 value: "true"</pre> |
| affinity | affinity for the Containerized Backup and Restore tool pods. | { } |

| Parameter | Description | Default |
|----------------------|---|-------------------------|
| control_node_ip_list | Cluster control nodes IP list. Only used for Avamar backends. | nil |
| hostNetwork.avamar | If true, Avamar client pod will use host Network. Set hostNetwork.avamar=false when enable Istio. Otherwise, istio-proxy sidecar cannot be injected to Avamar pod. There are the following side effects with this setting: <ul style="list-style-type: none"> In Avamar GUI, you can only see the backup/restore result, you cannot see the detailed logs for a specific backup/restore. But you can check the logs in the Containerized Backup and Restore tool avamar pod (dir: /var/avamar/clientlogs). The backup/restore work may start a little late. In Avamar lab-6 and lab-7, it is less than 60 seconds. | true |
| avamar.enabled | Specify if avamar is enabled. It must be set to true when backend is AVAMAR. | false |
| avamar.server | Avamar server IP address. | 10.75.53.234 |
| avamar.domain | Domain name in Avamar server. | ARC |
| avamar.clientName | Avamar client name. | nil |
| avamar.dnsPolicy | If "hostNetwork.avamar" is "true", this parameter should be set to "ClusterFirst WithHostNet". If "hostNetwork.avamar" is "false", but dnsPolicy is "ClusterFirst WithHostNet", it also works since Avamar pod still uses the cluster DNS first per our testing result. | ClusterFirstWithHostNet |
| avamar.useHelmPlugin | Support helm plugin hook jobs in Avamar scripts. BrHook jobs have been introduced. Do not use this one. | false |

| Parameter | Description | Default |
|--------------------------------------|--|--------------|
| avamar.resources.limits.cpu | Limits CPU computation of Avamar client container. | 1 |
| avamar.resources.limits.memory | Limits memory size of Avamar client container. | 500Mi |
| avamar.resources.requests.cpu | Requests CPU computation of Avamar client container. | 300m |
| avamar.resources.requests.memory | Requests memory size of Avamar client container. | 70Mi |
| avamar.preScriptTimeoutSeconds | The Avamar Pre backup/restore script timeout seconds. | 10800 |
| avamar.postScriptTimeoutSeconds | The Avamar Post backup/restore script timeout seconds. | 10800 |
| direct_redis.enabled | - If true, turn on celery asynchronous mode. | false |
| direct_redis.redis_image.name | redis image name | crdb/redisio |
| direct_redis.redis_image.tag | redis image tag | 2.10-2.1526 |
| direct_redis.redis_image.pullPolicy | redis image pull policy | IfNotPresent |
| direct_redis.redis_image.registry | redis image container registry | registry |
| direct_redis.redis_port | redis port | 6379 |
| direct_redis.resources.limits.cpu | Limits CPU computation of redis container. | 500m |
| direct_redis.resources.limits.memory | Limits memory size of redis container. | 500Mi |

| Parameter | Description | Default |
|--|---|---------|
| direct_redis.resources.requests.cpu | Requests CPU computation of redis container. | 50m |
| direct_redis.resources.requests.memory | Requests memory size of redis container. | 50Mi |
| celery.workerConcurrency | The number of worker processes/threads in celery | 10 |
| celery.resources.limits.cpu | Limits CPU computation of celery. | 10 |
| celery.resources.limits.memory | Limits memory size of celery. | 1Gi |
| celery.resources.requests.cpu | Requests CPU computation of celery. | 300m |
| celery.resources.requests.memory | Requests memory size of celery. | 500Mi |
| k8swatcher.enabled | Specify if k8swatcher is enabled. It must be set to true when you want to backup/restore BCMT cluster or auto enable/update scheduled backup or support existing multi-tenancy capabilities, for example, isolation of backup policies. | false |
| k8swatcher.clusterBrEnabled | This option works only when k8swatcher.enabled is true and indicates if to provide NCS cluster backup / restore function. This option can only be set to true when cburScope is "Cluster". For non-NCS env, set this option to false since cluster BR is not supported in non-NCS env like OpenShift. | true |
| k8swatcher.resources.limits.cpu | Limits CPU computation of k8swatcher. | 500m |
| k8swatcher.resources.limits.memory | Limits memory size of k8swatcher. | 500Mi |

| Parameter | Description | Default |
|--------------------------------------|---|-----------------------------|
| k8swatcher.resources.requests.cpu | Requests CPU computation of k8swatcher. | 200m |
| k8swatcher.resources.requests.memory | Requests memory size of k8swatcher. | 80Mi |
| helmHome.glusterfs.enabled | If it is true, use glusterfs volume as Helm home. For NCS env, this parameter cannot be "true" starting from NCS20 FP1. The Containerized Backup and Restore tool only mounts Helm home for one of the following two use cases: <ul style="list-style-type: none"> NCS cluster backup/restore, controlled by k8swatcher.enabled=true, k8swatcher.clusterBrEnabled=true The set of Avamar scripts which supports backup/restore helm hook jobs as a tmp solution and it only supports helm2, controlled by avamar.useHelmPlugin=true | false |
| helmHome.glusterfs.endpoints | The glusterfs endpoints for helm home | glusterfs-cluster |
| helmHome.glusterfs.path | The glusterfs volume name for helm home | bcmt-helm-home |
| helmHome.hostPath | When helmHome.glusterfs.enabled is false, the Containerized Backup and Restore tool will mount this hostPath as Helm home. If this is also not configured: <ul style="list-style-type: none"> If Kubernetes version >= v1.18.8 (NCS20 FP1), the Containerized Backup and Restore tool will mount "/opt/bcmt/storage/helm_home" as Helm home. Otherwise, mount "/root/.helm" as Helm home. The Containerized Backup and Restore tool only mounts Helm home for one of the following two use cases: | /opt/bcmt/storage/helm_home |

| Parameter | Description | Default |
|-------------------------------|--|-----------|
| | <p>1. NCS cluster backup/restore, controlled by k8swatcher . enabled=true ,k8swatcher . clusterBrEnabled=true</p> <p>2. The set of Avamar scripts which supports backup/restore helm hook jobs as a tmp solution and it only supports helm2, controlled by avamar.useHelmPlugin=true</p> | |
| logging.unified_logging | Specify if unified_logging is enabled. If set to true, all backup/restore logs will be printed with unified logging format. | false |
| logging.file_logging_level | Specify the logging level for log file / CBUR_REPO/logs/BR.log. Valid values: CRITICAL , ERROR , WARNING , INFOorDEBUG | DEBUG |
| logging.console_logging_level | Specify the logging level for console log. Valid values: CRITICAL , ERROR , WARNING , INFOorDEBUG | INFO |
| integrityCheck.local_enable | Enable md5sum integrity check for intra-cluster data transmission. | false |
| integrityCheck.s3_enable | Enable md5sum integrity check for S3 data transmission.

 Note: this will only work for file size < 5GB. | false |
| backup_self.enabled | Enable backup the Containerized Backup and Restore tool itself. | false |
| backup_self.backend_mode | The backend of backup the Containerized Backup and Restore tool self. | AWSS3 |
| backup_self.cronSpec | The cronSpec of backup the Containerized Backup and Restore tool self. The scheduled backup will not be enabled automatically. User needs to run "helm backup -t -a enable" or add "autoEnableCron: true" to the Containerized | 0 0 * * * |

| Parameter | Description | Default |
|------------------------|--|---|
| | Backup and Restore tool BrPolicy when k8swatcher is enabled. | |
| backup_self.maxiCopy | The max number of backups for backing up the Containerized Backup and Restore tool. When the copies exceed the number, the oldest one will be deleted. | 5 |
| backup_self.volumes | When the Containerized Backup and Restore tool backups itself, the volumes to backup. These volumes must have been mounted to the Containerized Backup and Restore tool. The Containerized Backup and Restore tool will add them to the BrPolicy of the Containerized Backup and Restore tool. | - cbur-repo |
| hostPaths.dirs | The host dirs used for NCS cluster backup/restore. Do not change this setting. | bcmt-path: "/opt/bcmt/storage/"bcmt-config-path: "/opt/bcmt/config/bcmt-cmdb" |
| hostPaths.files | The host files used for NCS cluster backup/restore. Do not change this setting. | ncm-cli: "/usr/local/bin/ncm" |
| auth.enabled | Enable the Containerized Backup and Restore tool auth. | false |
| auth.useNodePort | Use node port as the Containerized Backup and Restore tool service port. It can be used by the Containerized Backup and Restore tool portal in non-NCS env. | false |
| ingress.enabled | Enable the Containerized Backup and Restore tool ingress. | false |
| ingress.path | Configure the Containerized Backup and Restore tool ingress path. | cbur |
| auditLogStdout.enabled | If true, also print audit log to STDOUT. | false |
| clusterName | The cluster name, any unique string. For NCS cluster, if none of clusterName and | nil |

| Parameter | Description | Default |
|-----------------------------|---|---------|
| | clusterId is defined, the Containerized Backup and Restore tool will query the cluster info from NCS. For non-NCS cluster, one or both of clusterName and clusterId must be specified. | |
| clusterId | The cluster ID, any unique string. For NCS cluster, if none of clusterName and clusterId is defined, the Containerized Backup and Restore tool will query the cluster info from NCS. For non-NCS cluster, one or both of clusterName and clusterId must be specified. | nil |
| clusterDomain | The cluster domain name. Currently it is only used for Istio traffic management. | nil |
| onlyRestore | Set onlyRestore to be true if you only want the Containerized Backup and Restore tool to do app restore or cluster restore. | false |
| enableBrSchemaValidation | Enable BrPolicy schema validation. | true |
| timezone.mountHostLocaltime | If true, mount host local time to the Containerized Backup and Restore tool pods. For OpenShift env, set this parameter to false. | true |
| istio.version | Current Istio version inside cluster | 1.5 |
| istio.enabled | If true, istio will be used for connecting and managing microservices. | false |
| istio.cni.enabled | If true, istio will use CNI (Container Network Interface). | true |
| istio.mtls.enabled | If true, enforce mTLS connection. | true |
| istio.permissive | Allow mutual TLS as well as clear text for deployment. | false |
| istio.createDrForClient | Create destinationRule when application is installed in istio-injection=enabled | false |

| Parameter | Description | Default |
|---------------------------------------|--|-----------------------------|
| | namespace, but is configured with istio.enabled=false. | |
| istio.virtualservice.hosts | The hosts used to reach the Containerized Backup and Restore tool virtualservice. | ["cbur-master-cbur.cbur"] |
| istio.sharedHttpGateway.namespace | The namespace for shared Istio gateway | nil |
| istio.sharedHttpGateway.name | The name of shared Istio gateway | nil |
| istio.prefixReleaseNameForGatewayName | If true, CKEY chart will add Release name to the beginning of every gateway names. | true |
| istio.gateway.enabled | If true, Istio will use gateway for connecting. | false |
| istio.gateway.labels | Custom label for CKEY Istio gateway | { } |
| istio.gateway.annotations | Custom annotations for Istio gateway | { } |
| istio.gateway.ingressPodSelector | Customized selectors for istio gateway | istio: ingressgateway |
| istio.gateway.port | Gateway port for this gateway | 80 |
| istio.gateway.protocol | Protocol associated with this Istio gateway port | HTTP |
| istio.gateway.hosts | List of istio gateway hosts | - ckey.io |
| istio.gateway.tls.redirect | If true, Istio will redirect from HTTP ports to TLS port. | false |
| istio.gateway.tls.mode | Istio gateway TLS mode | PASSTHROUGH |
| istio.gateway.tls.credentialName | Istio gateway secret name that contains certificates for TLS connection | nil |

| Parameter | Description | Default |
|--------------------------|---|---------|
| istio.gateway.tls.custom | Custom TLS configuration for Istio gateway | { } |
| priorityClassName | priorityClassName used for the Containerized Backup and Restore tool pod | " " |
| helm2Legacy | <p>The Containerized Backup and Restore tool uses "heritage: {{ .Release.Service }}" in some objects. For Helm v2, the .Release.Service is "Tiller", For Helm v3, it is "Helm". When the Containerized Backup and Restore tool Helm release converts from V2 to V3, the later V3 helm upgrade will fail due to the matchLabels in deployments are immutable. Set this variable to "true" will keep using "Tiller" in the Containerized Backup and Restore tool deployments so the Helm upgrade will succeed.</p> <p> Note: the "heritage: {{ .Release.Service }}" in other objects are not controlled by this variable.</p> | false |

20.3.20 Alarm Management

The Containerized Backup and Restore tool fires alarms using unified logging provided by CLOG, and outputs alarms to stdout.

 **Note:** The underlying container environment (Docker engine, Kubernetes, etc.) configuration directs the stdout to log files. Fluentd collects, parses/filters and then distributes them to subsequent subsystems (e.g. CALM). The handling from stdout to CALM is out of the scope of the Containerized Backup and Restore tool.

The following alarms are defined currently. And the fired alarm names are recorded in brpolicy['status']['alarms'].

```
ALARM_DEF['BACKUP_FAIL'] = { 'id': 3001600, 'text': 'the Containerized Backup and Restore tool data backup failed.' }
ALARM_DEF['RESTORE_FAIL'] = { 'id': 3001601, 'text': 'the Containerized Backup and Restore tool data restoration failed.' }
ALARM_DEF['CLUSTER_BACKUP_FAIL'] = { 'id': 3001602, 'text': 'the Containerized Backup and Restore tool cluster backup failed.' }
```

```
ALARM_DEF['CLUSTER_RESTORE_FAIL'] = { 'id': 3001603, 'text': 'the  
Containerized Backup and Restore tool cluster restoration failed.' }  
ALARM_DEF['VOLUME_EXCEED_THRESHOLD'] = { 'id': 3001604, 'text': 'The volume  
usage exceeds threshold.' }  
ALARM_DEF['CRON_ADD_FAIL'] = { 'id': 3001605, 'text': 'the Containerized  
Backup and Restore tool add cron job failed.' }  
ALARM_DEF['CRON_DEL_FAIL'] = { 'id': 3001606, 'text': 'the Containerized  
Backup and Restore tool delete cron job failed.' }
```

For alarms (except for 'VOLUME_EXCEED_THRESHOLD'):

- 1) When operation fails, check if the alarm name (e.g. BACKUP_FAIL) is in br policy, if not, fire 'major' alarm and record the alarm name in br policy.
- 2) When operation succeeds, check if the alarm name is in br policy, if yes, fire 'cleared' alarm and remove the alarm name from br policy.

The Containerized Backup and Restore tool does not fire the same alarm repetitively before it is cleared.

For alarm 'VOLUME_EXCEED_THRESHOLD':

The Containerized Backup and Restore tool monitors the following three volumes: /BACKUP, /CBUR_REPO, /CLUSTER. For each of them, there are two thresholds: low threshold (default: 75%) and high threshold (default: 85%). Before backup / restore / backup cluster / restore cluster, check the disk usage.

Check the disk usage at the beginning of backup / restore / backup cluster / restore cluster.

- If the disk usage reaches the high threshold, reports critical alarm "VOLUME_EXCEED_THRESHOLD" and fails the backup/restore. The corresponding alarm ('BACKUP_FAIL', 'RESTORE_FAIL', 'CLUSTER_BACKUP_FAIL', 'CLUSTER_RESTORE_FAIL') will also be reported.
- If the disk usage reaches the low threshold, reports warning alarm "VOLUME_EXCEED_THRESHOLD" and continues with the operation.

20.3.21 The Containerized Backup and Restore Tool API Reference Guide

From 19.09, the Containerized Backup and Restore tool supports restful API interface v2. Here are the APIs for the Containerized Backup and Restore tool component.

Figure 24: List of the Containerized Backup and Restore tool APIs

Response code supported as of R20.09

| Code | Meaning |
|-------------|---|
| 200 | BACKUP/RESTORE Succeed |
| 202 | Request sent to celery, but not completed |
| 401 | HTTP client unauthorized |
| 404 | Failed to query task status/The backup ID is not in backup history/The task id doesn't exist any more. It may be deleted because of expiration of 7 days. |
| 500 | Internal Server Error |
| 560 | TargetPodsUnstableAfterBackup/Restore |

RESTful API Usage

First, get the auth status from the Containerized Backup and Restore tool Master:

```
curl -s -kL --post301 -X GET http://cbur-master-cbur.ncms.svc:80/v2/auth/status | jq ".message"
```

You get a return code and the authentication type mapping is as below:

| Return Code | Authentication Type | Note |
|--------------------|------------------------------|-------------|
| 0 | Authentication Disabled | |
| 1 | CEKY Authentication Enabled | |
| 2 | Basic Authentication Enabled | |

Then we can use the Containerized Backup and Restore tool RESTful API based on the authentication type.

Authentication disabled

For Authentication disabled, we can access RESTful API directly, here are some examples:

1. Backup a BrPolicy

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/backup/
<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -H "accept: application/json"
```

2. Restore a BrPolicy

```
curl -skL --post301 -X PUT http://cbur-master-cbur.ncms.svc:80/
v2/restore/<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -d
'backup_id=&volume=&object_type=&object_name=' -H "accept: application/
json" -H "Content-Type: application/x-www-form-urlencoded"
```

3. Backup a Helm Release

A. Helm version 2 (pass the tiller namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/
helm/release/backup/<TILLER_NAMESPACE>/<RELEASE_NAME>?helm_version=2 -H
"accept: application/json"
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/helm/
release/backup/<RELEASE_NAMESPACE>/<RELEASE_NAME>?helm_version=3 -H
"accept: application/json"
```

4. Restore a Helm Release

A. Helm version 2 (pass the tiller namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/
v2/helm/release/restore/<TILLER_NAMESPACE>/<RELEASE_NAME>?
helm_version=2 -H "accept: application/json" -H
"Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-
cbur.ncms.svc:80/v2/helm/release/restore/<RELEASE_NAMESPACE>/
<RELEASE_NAME>?helm_version=3 -H "accept: application/json"
-H "Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
```

CEKY Authentication Enabled

For CEKY Authentication, we use the token to access the Containerized Backup and Restore tool RESTful API.

First, use below command to get access token (here use **defalut user "br-admin" with password "br@the Containerized Backup and Restore tool"**, about the detail user management, refer to Section "6.3.2 Keycloak authentication")

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/auth/users/login -H "accept: application/json" -H 'Content-Type: application/json' -d '{"username": "br-admin", "password": "br@the Containerized Backup and Restore tool"}' | jq ".message.accessToken"
```

Second, use "accessToken" to access other RESTful APIs. Need to pass argument like this: **-H "Authorization: Bearer <ACCESS_TOKEN>"**

Here are some examples:

1. Backup a BrPolicy

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/backup/<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -H "accept: application/json" -H "Authorization: Bearer <ACCESS_TOKEN>"
```

2. Restore a BrPolicy

```
curl -skL --post301 -X PUT http://cbur-master-cbur.ncms.svc:80/v2/restore/<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -d 'backup_id=&volume=&object_type=&object_name=' -H "accept: application/json" -H "Content-Type: application/x-www-form-urlencoded" -H "Authorization: Bearer <ACCESS_TOKEN>"
```

3. Backup a Helm Release

A. Helm version 2 (pass the tiller namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/helm/release/backup/<TILLER_NAMESPACE>/<RELEASE_NAME>?helm_version=2 -H "accept: application/json" -H "Authorization: Bearer <ACCESS_TOKEN>"
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/helm/release/backup/<RELEASE_NAMESPACE>/<RELEASE_NAME>?helm_version=3 -H "accept: application/json" -H "Authorization: Bearer <ACCESS_TOKEN>"
```

4. Restore a Helm Release

A. Helm version 2 (pass the tiller namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/helm/release/restore/<TILLER_NAMESPACE>/<RELEASE_NAME>?
```

```
helm_version=2 -H "accept: application/json" -H
"Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
-H "Authorization: Bearer <ACCESS_TOKEN>"
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-
cbur.ncms.svc:80/v2/helm/release/restore/<RELEASE_NAMESPACE>/
<RELEASE_NAME>?helm_version=3 -H "accept: application/json"
-H "Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
-H "Authorization: Bearer <ACCESS_TOKEN>"
```

Basic Authentication Enabled

For Basic Authentication, we use the "username:password" pair to access the Containerized Backup and Restore tool RESTful API.

First, use below command to get username and password:

```
username: kubectl get secret -n ncms cbur-basic-auth -
o=jsonpath='`{.data.username}`' | base64 -d

password: kubectl get secret -n ncms cbur-basic-auth -
o=jsonpath='`{.data.password}`' | base64 -d
```

Second, use "username:password" pair to access other RESTful APIs. Need to pass argument like this: `-u "<username>:<password>"`

Here are some examples:

1. Backup a BrPolicy

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/backup/
<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -H "accept: application/json" -u
<USERNAME>:<PASSWORD>
```

2. Restore a BrPolicy

```
curl -skL --post301 -X PUT http://cbur-master-cbur.ncms.svc:80/
v2/restore/<BRPOLICY_NAMESPACE>/<BRPOLICY_NAME> -d
'backup_id=&volume=&object_type=&object_name=' -H "accept: application/
json" -H "Content-Type: application/x-www-form-urlencoded" -u
<USERNAME>:<PASSWORD>
```

3. Backup a Helm Release

A. Helm version 2 (pass the tiller namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/
helm/release/backup/<TILLER_NAMESPACE>/<RELEASE_NAME>?helm_version=2 -H
"accept: application/json" -u <USERNAME>:<PASSWORD>
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/v2/helm/
release/backup/<RELEASE_NAMESPACE>/<RELEASE_NAME>?helm_version=3 -H
"accept: application/json" -u <USERNAME>:<PASSWORD>
```

4. Restore a Helm Release**A. Helm version 2 (pass the tiller namespace):**

```
curl -skL --post301 -X POST http://cbur-master-cbur.ncms.svc:80/
v2/helm/release/restore/<TILLER_NAMESPACE>/<RELEASE_NAME>?
helm_version=2 -H "accept: application/json" -H
"Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
-u <USERNAME>:<PASSWORD>
```

B. Helm version 3 (pass the release namespace):

```
curl -skL --post301 -X POST http://cbur-master-
cbur.ncms.svc:80/v2/helm/release/restore/<RELEASE_NAMESPACE>/
<RELEASE_NAME>?helm_version=3 -H "accept: application/json"
-H "Content-Type: application/x-www-form-urlencoded" -d
"backup_id=&volume=&object_type=&object_name=&brpolicies=&exclude_brpolicies="
-u <USERNAME>:<PASSWORD>
```

20.4 Backup

20.4.1 Nokia Container Services Backup

When entering the backup page, user has to specify CBUR volumes sizes (50GB by default), these volumes are used to run the backup process and produce the final backup tar files(will keep ROTATION number of backups). Follow the following formula to edit the volume sizes according to your cluster size and application data:

- Cluster Backup Volume Size: Used for cluster level backup. Size: (rotation+1) * (the size of local registry + the size of local helm repo + the size of helm home + the size of tiller configmaps / os tuning related crs / ncs users / hooks / helm releases information)

- Application backup Volume Size: Used for application level backup/restore. Size: sum of storage for all applications with backend as “local”. Each application needs: * (maxCopy + 1) * backup data size



Note: If the backup volumes sizes are not sufficient, backup will fail.

20.4.2 CM backup

This is configured in the installation page and will perform regular backups of the CM database(infrastructure configurations). The path, rotation, and schedule for this backup are only configured during installation. Customer also needs to make sure to backup the files to nfs or copy regularly to a remote location.

The NCS Bare-Metal manager nodes is:

- Automatically backed up every day at the configured time interval.

The backup file is:

- Encrypted via **OPENSSL**
- Stored in the configured place as per `backup_nfs_mountpoint` (Backup path in installation page)
- The `mountpoint` is relative to the installer node. If given a network path, the script will copy it there, not in a folder on the installer node.

Ensure that:

- The folder exists.
- A new dated folder will be created for every backup.
- The backup consists of the databases of the management nodes and some configuration files.

To start a manual backup on a management node, issue the following command:

- `/usr/local/bin/openstack-ansible --timeout=1200 -u cbis-admin /opt/openstack-ansible/playbooks/backup.yml --private-key=/home/cbis-admin/.ssh/id_rsa`

The backup process can be followed in the logs: `/var/log/cbis/backup.log`.

Average running time for a complete backup is 50 minutes.



Important: Manual changes to the configuration will not be backed up. If changes are made to the configuration, it is necessary to reconfigure it after a restore operation.

Example:

Backup and Restore configuration

```
backup_password: b4cK1tup!! # password for backup encryption
backup_hour: 2 # backup hour in cron format
backup_minute: 0 # backup minute in cron format
```

```
backup_nfs_mountpoint: /root/backup # default path for backup storage on
installer node
```

20.5 NCS Manager Restore Central Deployment Type

This process restores the management databases from the backup using the backup file from `backup_nfs_mountpoint` folder. It does not look in the **sub folders**, like those created by the backup process.

With the example configuration as mentioned above, the process will look for it in the following location:

- `/root/backup/cbis_backup.enc`
- Manual changes to the configuration will not be backed up.
- If changes are made to the configuration it is necessary to configure it again after a restore operation.

The manager node restore can be started via CLI from any of the manager node via the following command (current working directory does not matter):

```
python /usr/lib/python2.7/site-packages/nokia/backup_restore_bm/restore.py
--manager_replace      --backup_folder <backup_folder_path>      --
backup_password '<backup_password>' 
--cluster '<your_cluster_name>'
```



Note:

- After a restore procedure, a **Replace Master** operation must be performed.
- Managing Backup and Restore is possible through the NCS Manager menu. Refer to the NCS Manager Reference Guide for more details.

20.6 NCS Manager Reinstall/Restore Procedure Cluster Deployment Type

1. Install a new node (as the first node) with the same build and IP and configuration as previously was installed.

An explanation for the first node installation is described in the Bare-metal Installation Guide.

2. Backup the latest `cbis_backup.enc` from the backup folder (configured with the `backup_nfs_mountpoint` variable).

3. Export the configuration of the cluster from the NCS Manager.

4. Shutdown the node and install a new one with the same build and IP and configuration.

5. Copy the `cbis_backup.enc` to the backup folder directly.

6. Import the cluster configuration that was exported earlier from the NCS Manager or enter the same configuration.

7. After that, in the NCS Manager, run the first two deployment steps: generate inventory and bootstrap manager.
8. Copy the latest `cbis_backup.enc` from the backup folder (configured with the `backup_nfs_mountpoint` variable) on the new installed cluster-manager.
9. Start the restore procedure with these added parameters:

```
python /usr/lib/python2.7/site-packages/nokia/backup_restore_bm/restore.py
```

- `--manager_recovery --cluster <cluster_name>`
- `--backup_folder <your_backups_path> --backup_password <your_backup_password>`

 **Note:** As part of any backup and restore procedure, you must apply security hardening after the restore has completed, to align all nodes.

 **Note:** When deploying the first master node in a cluster, run the above procedure to restore the NCS manger on one of the two other masters, then follow Baremetal Master Replace in the NCS Manager Guide to run Replace for the failed master node.

For details on executing Backup and Restore using the NCS GUI, see *Baremetal Clusters Backup and Restore* in the *NCS Manager Guide*.

20.7 NCS on Virtualized Deployments: Backup the Deploy-server using Snapshots

20.7.1 Introduction

Currently, there is no official procedure for backing up the deploy-server of an NCS cluster on top of a virtualization layer.

The deploy-server consists of essential components: bcmt-yum-repo, dm-registry, clcm-admin, bcmt-admin that needs to be up and running in order to have a healthy NCS cluster.

The following sections describe how to backup the deploy-server in an NCS CN\A deployment by taking a snapshot of the deploy-server.

 **Note:** Snapshot takes a backup of the VM for a specific Point In Time (PIT). The snapshot size right after the creation is minor and from that moment it will save the changes that were done on the VM, meaning that the size of the snapshot will get bigger with time. You can "refresh" the snapshot by creating a new snapshot and deleting the previous snapshot. A snapshot is a only a 'Point In Time' backup, therefore, after any LCM operation (for example: scale-in, scale-out) "refresh" the snapshot by creating a new and updated snapshot.

20.7.2 NCS on top of Openstack deployment

This is applicable for NCS on top of CBIS and on top of any other OpenStack deployment.

1. Log in to the OpenStack Horizon UI.
2. Navigate to "Computes" → "Instances"
3. In the instances list, locate the deploy-server and press on the 'Create Snapshot' button on the right side.

Figure 25: Create a snapshot

| Instance Name | Image Name | IP Address | Flavor | Key Pair | Status | Availability Zone | Task | Power State | Time since created | Actions |
|---------------------|---------------------------|-------------------------------|----------|-------------|--------|-------------------|------|-------------|--------------------|----------------------------------|
| my_deployer | bcmt-deployserver-20.12.0 | 10.5.203.200
192.168.34.21 | m1.large | deployemode | Active | zoneovs | None | Running | 3 weeks | <button>Create Snapshot</button> |
| ncs-deploy | bcmt-deployserver-20.12.0 | 10.5.203.163
172.30.254.18 | m1.large | ncs | Active | zoneovs | None | Running | 1 month, 3 weeks | <button>Create Snapshot</button> |
| ncs-01-deployserver | bcmt-deployserver-20.12.0 | 10.5.203.155
192.168.34.3 | m1.large | deployemode | Active | zoneovs | None | Running | 5 months, 2 weeks | <button>Create Snapshot</button> |

4. In the pop-up window, name the snapshot with the instance name and creation date. Then select 'Create Snapshot':

Figure 26: Create Snapshot

Snapshot Name *

Description:

A snapshot is an image which preserves the disk state of a running instance.

Cancel
Create Snapshot

5. You will be automatically redirected to the 'images' list. Wait until the snapshot creation is done successfully.

OpenStack Restore operation:

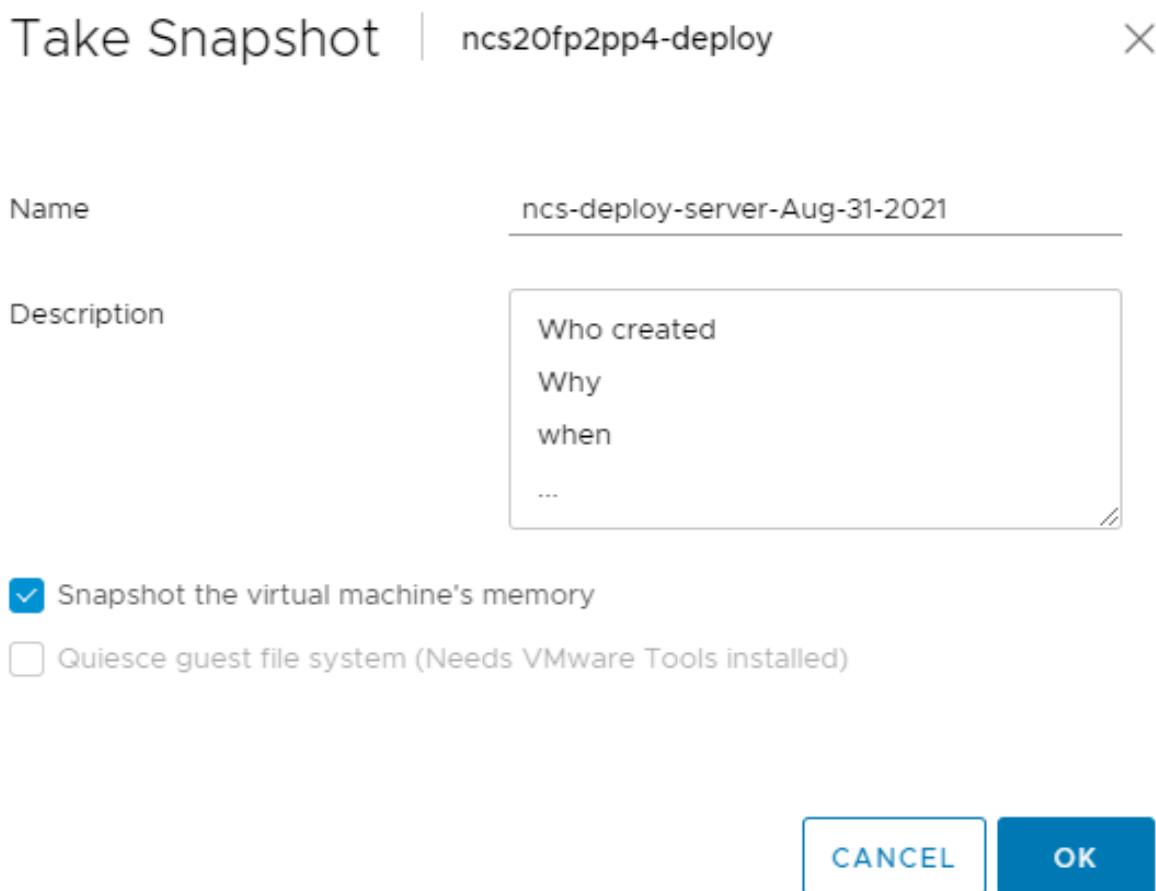
Make sure the 'original' deploy-server is powered off.

To restore the deploy-server, select the 'Launch' option on the right side of the snapshot name and follow the menu that opens. In the pop-up window: Make sure to chose all the correct parameters (Flavor, source, Availability-Zone, etc'). In the 'Network Ports' tab, select the original deploy-server interfaces in order to keep the IP addresses the same as the original machine had.

20.7.3 NCS on top of vCenter deployment

1. Login to vSphere client
2. Navigate to the "VMs and Template view" → the deploy-server VM → ACTIONS → Snapshots → Take Snapshot.
3. In the pop-up window, name the snapshot with the creation date and provide any description you want. Keep the first option selected and select 'OK'.

Figure 27: Name snapshot



4. You can observe the task running in the tasks panel in the bottom part of the UI.
5. Wait until the snapshot creation is done successfully.

Figure 28: Snapshot creation

| Recent Tasks | Alarms | Task Name | Target | Status | Details | Initiator | Queued For | Start Time | Completion Time | Server |
|--------------|--------|---------------------------------|--------------------|-------------|---------|----------------------------|------------|------------------------|------------------------|------------------------------|
| | | Create virtual machine snapshot | ncs20fp2pp4-deploy | ✓ Completed | | VSPHERE LOCAL/Administr... | 9 ms | 08/31/2021, 2:55:32 PM | 08/31/2021, 2:55:50 PM | cto0004vmf.netactnsn-rdma... |

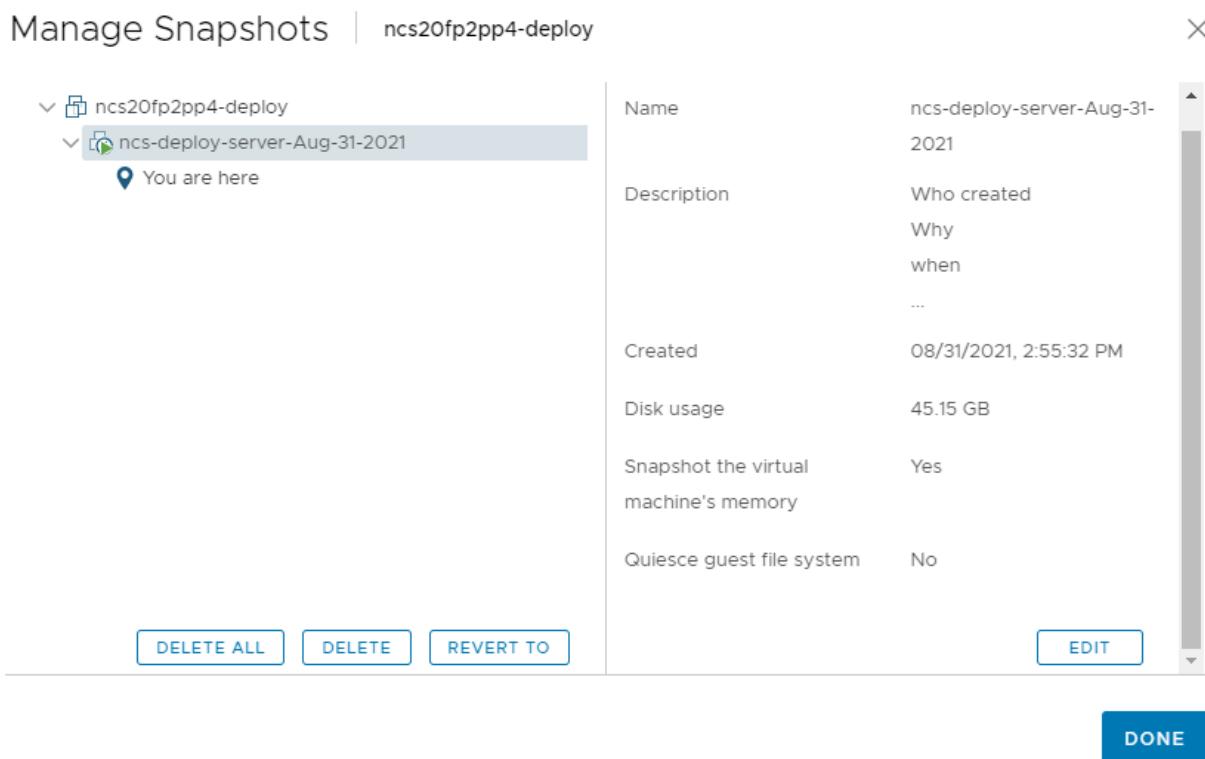
vCenter restore operation:

To restore the deploy-server, select 'Manage Snapshot', chose the desired snapshot and select the 'REVERT TO' button.

 **Note:** It will run-over the previous machine.

vCenter additional resources:

Figure 29: vCenter additional resources



| Name | Value |
|---------------------------------------|-----------------------------------|
| Description | Who created
Why
when
... |
| Created | 08/31/2021, 2:55:32 PM |
| Disk usage | 45.15 GB |
| Snapshot the virtual machine's memory | Yes |
| Quiesce guest file system | No |

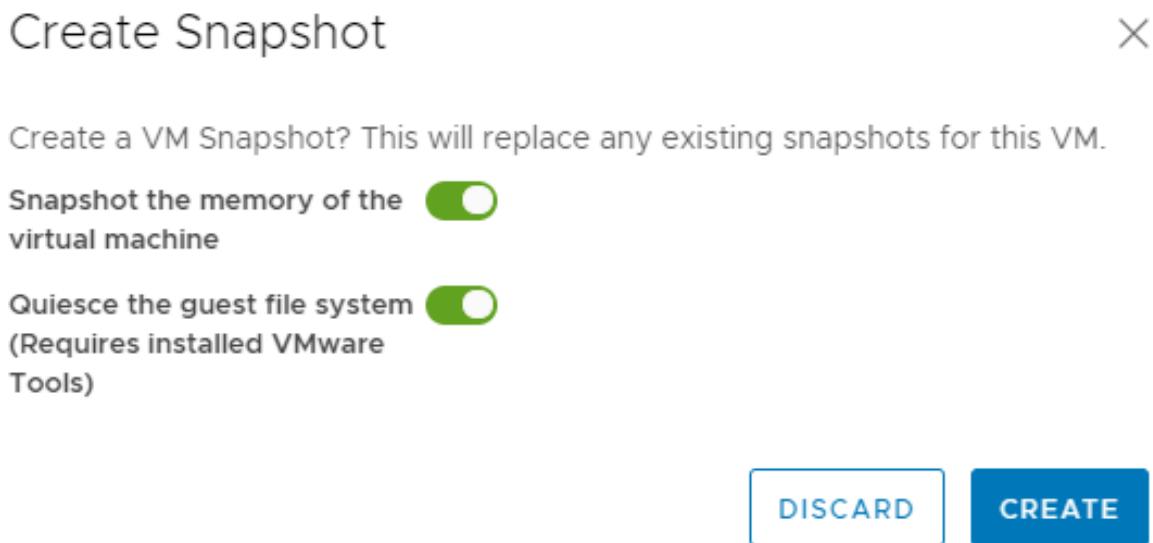
After the restore operation is done, make sure the VM is up and running. If not, turn it on.

20.7.4 NCS on top of vCloud deployment

1. Login to the VMware-Cloud-Director UI.
2. Navigate to "Data Centers" → the desired Data Centers
3. Navigate to "Virtual Machines" → the deploy-server VM → Actions → "Create Snapshot"

4. In the new window, enable both options and create the snapshot

Figure 30: Create a Snapshot



5. Wait until the snapshot creation is done successfully.

vCloud restore operation:

In the "Actions" option of the VM, select the "Revert to Snapshot" option.



Note: It will run-over the previous machine

20.8 Troubleshooting

20.8.1 Administration or Operations Issues

20.8.1.1 glusterfs volume almost full, occupied nearly 85% disk, manual cleaning required

Error message:

```
glusterfs-server:/bcmt-helm-home
    79G   63G   12G  85% /root/.helm
glusterfs-server:/bcmt-glusterfs
    79G   63G   12G  85% /opt/bcmt/storage
```

It is actually /data0/glusterfs-cbur-backup which is almost full, cannot mount this volume via mount -t glusterfs (it hangs), pods for cbur get evicted

because of volume full, deleting whole volume on glusterfs didn't help either.

Analysis:

Cluster glusterfs volume bricks created at /data0 directory, if these volumes occupied more than 85% space, POD evicted will be triggered on the host.

Possible fixes:

1. Manually mount the volume and clean up the data, for example, if need to clean up volume "cbur-glusterfs-backup":

```
$ mkdir /root/tmp-mount; mount -t glusterfs glusterfs-server:/cbur-glusterfs-backup /root/tmp-mount;

cd /root/tmp-mount;

rm example-file1 example-file2

cd /root/

umount /root/tmp-mount
```

2. If the volume was deleted already, need to clean up the bricks directory on /data0 to release the spaces, and recreate the volume.

For example, to clean up bricks for volume "cbur-glusterfs-backup": on each control node, remove and recreate /data0/glusterfs-cbur-backup

```
# rm -fr /data0/glusterfs-cbur-backup; mkdir /data0/glusterfs-cbur-backup
then on one of the control, recreate and start the glusterfs volume:
```

```
gluster volume create cbur-glusterfs-backup replica 3 transport
tcp control1_interneal-service-ip:/data0/glusterfs-cbur-backup
control2_interneal-service-ip:/data0/glusterfs-cbur-backup
control3_interneal-service-ip:/data0/glusterfs-cbur-backup

gluster volume start cbur-glusterfs-backup
```

3. User could install CPRO to monitor the host status to detect this kind of issue as early as possible.

20.8.1.2 glusterfs self heal daemon occupied nearly 50% CPU, Glusterfs volume self heal daemon occupied too CPU, manual cleaning required

Error message:

```
[root@coral-cwe-03 ~]# ps -ef|grep glustershd|wc -l 173
```

```
[root@coral-cw-01 /var/lib/glusterd/glustershd]# systemctl status glusterd

glusterd.service - GlusterFS, a clustered file-system server
Loaded: loaded (/etc/systemd/system/glusterd.service; enabled; vendor preset:
disabled)

Active: active (running) since Fri 2019-07-26 14:52:10 IST; 2 days ago
  Process: 13937 ExecStart=/usr/sbin/glusterd -p /var/run/glusterd.pid --log-
level $LOG_LEVEL $GLUSTERD_OPTIONS (code=exited, status=0/SUCCESS)

Main PID: 13938 (glusterd)

Tasks: 24003

Memory: 62.9G (limit: 62.9G)

CGroup: /system.slice/glusterd.service

    └─ 317 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...

        ├─ 345 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...

        ├─ 1521 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...

        ├─ 2352 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...

        ├─ 2507 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...

        ├─ 3725 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...
```

```
|  └─ 3878 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...
|  └─ 3924 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...
|  └─ 4329 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/
glusterfs/g...
|  └─ 5119 /usr/sbin/glusterfs -s 192.168.1.3 --volfile-id gluster/
glustershd -p /var/run/gluster/glustershd/glustershd.pid -l /var/log/glust...
...
Node response very slow, from top command, CPU is occupied by glusterfs self
heal daemon nearly 50%.
```

Analysis:

Glusterfs self heal process is designed to heal the volume automatically quickly, but it will consume much CPU sometimes and try to heal the deleted volume.

Possible fixes:

Manually restart glusterd service on each such node(for example, control node) to trigger it reload the volume file to avoid it heal delete volume and reduce the self heal daemon process quantity.

```
systemctl restart glusterd
```

20.8.1.3 With restoration, errors are encountered due to previous request history

Analysis:

If the restore operation fails accoring to the NCS UI, check the CBUR logs. If CBUR logs show successful operation the NCS UI can be ignored and in this case further operations can be run.

20.8.2 Upgrade/Rollback failures

Heal glusterfs volume failed during upgrade

Root cause:

Glusterfs self heal daemon process fault.

Workaround:

Execute "gluster volume status" to the volume ID that failed to heal on one of the control node, check the self-heal-daemon status, the "Online" flag of each self-heal Daemon should be "Y", if any one is "N", go to that node, execute "systemctl restart glusterd" and then recheck.

After fixing its status, re-run upgrade.

21 Cluster Scale-in and Scale-out operations for Bare-metal implementation

Infrastructure scaling operations add or remove nodes from a cluster. The idea here is to enable users to build new clusters from the ground up on bare-metal server infrastructures, eliminating the need to deploy hypervisors to abstract the physical hardware. Using dedicated servers, users can be given exclusive access to the entire cluster.

Before you proceed to scale storage nodes, you should consider the following prerequisites:

1. Supported Infrastructures and Platforms

2. Supported Configurations for:

- Dynamically created storage
- Local Storage

 **Attention:** If any of the servers are in failed state or any of the servers become unreachable for any reason within the NCS Baremetal cluster, Scale-in and Scale-out will also fail.

 **Attention:** Before starting the desired Scale-in/Scale-out procedure you need to Scale-in the faulty server and with the network repaired in this way you can then proceed with the Scale-in/Scale-out as originally planned.

 **Note:** Node infrastructure needs to be created or destroyed manually or via Cluster admin if `embedded_clcm=false`.

 **Note:** Only OpenStack, Vcenter, VCD, AWS and Azure platforms are supported to set `embedded_clcm` to true.

 **Note:** NCS does not support scaling in or scaling out of control nodes.

 **Note:** Before performing any scale in/out operation, verify there is no powered-off/unavailable node in the setup.

 **Note:** Scale in the powered-off/unavailable node from the cluster one by one.

 **Note:** The password used for accessing the cluster is different from the password used for accessing the NCS Manager.

21.1 Scale-out and scale-in failed for one worker node

Scale out failed pre NCS node creation part

When the user re-attempts scale out, the same GUI tile is presented with the same options. After fixing the issue, the user should then scale in and only afterwards, scale out again.

Scale out fails during NCS node creation

When the user enters scale out they will have a status written as unfinished and cannot do anything else but just run deploy. Here, the scale out will continue only from the NCS part and will not need to run all the steps again. It is therefore recommended to fix the issue and just run scale out again. (There is no need to scale in).

Scale in possibilities

Once pg number is calculated and set to each pool on installation, or be updated on scale out operation, it can never be decreased even in scale in, because ceph does not allow decreasing pg number of pools.

Ceph define maximum number of allowed pgs per OSD.

As a result, if the user does too many scale in operations, i.e. number of hosts is decreased by about 40% related to the maximum number it has once been, the scale in may fail with error that after scale in, number of pgs per OSD will be more than allowed.

21.2 Baremetal Cluster Scale-In

- Select **NCS Operation** → **Baremetal Cluster Scale in** → **Open**
- Set the values for listed **Scale-in parameters**

Table 71: Baremetal Cluster Scale-In Parameters

| Parameter | Description |
|---|--|
| Ignore Ceph errors (advanced users only!) (Enable/Disable) | If enabled, Ceph errors will be ignored. This could permanently delete data in the cloud. This is useful for an advanced user who wants to remove faulty storage nodes (Ceph status is not HEALTH_OK before beginning of operation).

 Remember: Storage and other constraints are also not checked. |
| Select roles | Select from drop down list: storage, worker, edge |
| Last NCS scale in status | Enter status |
| Select Nodes to remove | Enter Nodes names (Can only scale-in worker, edge, monitoring, or Storage nodes) |

21.2.1 Removing a Node

Users may wish to remove nodes for many reasons including increasing system efficiency.

 **Note:** When a node is deleted, the node object is deleted in Kubernetes, but the pods that exist on the node itself are not deleted. Any bare pods not backed by a replication controller would be inaccessible to the cluster, pods backed by replication controllers would be rescheduled to other available nodes would need to be manually deleted. The process is usually performed in this order.

1. Mark the **node** related to the machine to be shut down as unschedulable;
2. Start the pod(s) that is running in the **node** in other **node(s)**;
3. Gracefully **delete** the pod(s) that is running in the **node**;
4. **Delete the node.**

If the `embedded_clcm` flag is set to true, NCS co-operates with IaaS to handle infrastructure resources automatically.

NCS commands to scale-in one or several cluster nodes are:

```
ncs cluster scale-in --node_names=bcmt-01-worker-01, bcmt-01-worker-02
```

If the `embedded_clcm` flag is set to false, the node info needs to be added before cluster scale out or deleted after cluster scale in.

```
ncs node delete --node_name=bcmt-01-worker-01
```

Related information

[Baremetal Cluster Scale-In](#) on page 788

21.3 Baremetal Cluster Scale-out

Scale out to a new node for additional storage capacity for deployment, before scaling out the old node to be retired.

1. Select **NCS Operations** → **Baremetal Cluster Scale out** → **Open**
2. Enter **Cluster Username** and **Cluster Password**

 **Note:** The password used for accessing the cluster is different from the password used for accessing the NCS Manager.

 **Note:** Using the NCS Manager GUI is the optimal way to perform Bare metal scale-out.

21.3.1 General

21.3.1.1 Hardware

The following parameters can be set in Hardware, refer to [Hardware Parameters](#) on page 790:

Table 72: Hardware Parameters

| Parameters | Descriptions |
|---------------------------|--|
| IPMI Username | Username used to access management interface (IPMI). |
| IPMI Password | Password used to access management interface (IPMI). |
| Last ncs scale out status | Enter status. |

21.3.2 Customize Host Groups

To add new host groups:

1. Select "+" icon.
2. Enter name for **Add Panel** window.
3. Select **Create**

The following items are factory integrated, out-of-box default installation:

- Master BM
- Edge BM
- Worker BM
- Storage BM
- AllinOne
- Monitor Scale out

21.3.2.1 Monitor BM

Table 73: Monitor BM Parameters

| Parameters | Descriptions |
|----------------|-----------------|
| Platform Usage | Enter platform. |

Table 73: Monitor BM Parameters (continued)

| Parameters | Descriptions |
|---|---|
| Ceph Block Storage, Size in megabytes (Root device) | root device size in MB, when set to 0 partition for OSD will not be created, if set, partition_layout_path will be used as reference for root device partitioning while this value will be used for ceph_block |
| Enable software RAID1 | If enabled, a software raid will be initiated in first two devices (sda and sdb) |
| Select RAID1 primary OS root device hint | Select on which device the raid primary root file system will reside. If left empty, the first device identified as sda will be selected for primary root FS. This field has to be in a 'disk name' format, for example: /dev/sda |
| Select RAID1 secondary OS root device hint | Select on which device the secondary root file system will reside. If left empty, the second device identified as sdb will be selected for secondary root FS. This field has to be in a 'disk name' format, for example: /dev/sdb |
| CaaS Roles | Monitor |
| Select OS root device hint | Select on which device root file system will reside. If left empty, the first device identified as sda will be selected for root FS, one exception is hp-c7kg10_sep_controller's Controllers where it will be nvmeOn1. This field has to be in a JSON format following ironic hint format: {'key':'value'}.

The key is the identifier of the disk, for example:

{"by_path": "/dev/disk/by-path/pci-0000:00:1f.2-ata-2.1"}.

It is recommended that the identifier will be 'by_path' |
| Enable Custom NICs | If enable you can edit ports names, if not enable global value taken. |
| NIC 1 Port 1 Interface Name | Change the name if you have deviation from the default configuration. NIC1 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |

Table 73: Monitor BM Parameters (continued)

| Parameters | Descriptions |
|-----------------------------|---|
| NIC 1 Port 2 Interface Name | Change the name if you have deviation from the default configuration. NIC1 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |
| NIC 2 Port 1 Interface Name | Change the name if you have deviation from the default configuration. NIC2 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |
| NIC 2 Port 2 Interface Name | Change the name if you have deviation from the default configuration. NIC2 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |
| NIC 3 Port 1 Interface Name | Change the name if you have deviation from the default configuration. NIC3 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |
| NIC 3 Port 2 Interface Name | Change the name if you have deviation from the default configuration. NIC3 is managing the infrastructure related traffic, others the workload related. Exception is storage node where there is no workload related network traffic. |

21.3.3 IPMI

21.3.3.1 Add IPMIs



Note: Ranges (like: 10.0.0.1 - 10.0.0.20) can be used, and/or comma separated values.



Note: You only have to provide the new IPMI address.

 **Attention:** If the IPMI Addresses list and/or the Availability Zones list are changed after assigning items from these lists to the node groups, all existing node assignments will be lost. In this case, it is necessary to re-assign the IPMIs and Availability Zones to the node groups.

Enter IPMI Addresses.

21.3.3.2 Multiple pools

Table 74: Multiple pools

| Parameters | Description |
|--|-----------------------------------|
| Multiple pools | Enable/Disable |
| Enter the multiple pools names | Enter name then select ADD |
| Enter storage disks paths (At the "Assign nodes" section you will be able to assign disks paths from storage nodes to created pools) | Enter value |

21.3.3.3 Define Rack Names

 **Note:** In order to enable Racks, Multiple Pools must be disabled.

Table 75: Rack names

| Parameters | Description |
|------------|------------------------------------|
| New Rack | Enter value then select ADD |

21.3.3.4 Define Zone Labels

The following are characteristics of Zone labels

- It is a logical division
- Containers are independent.

- Container technologies is not capable of having certain configurations, example: Geo-redundancy.



Note: For Zone labels, you can use the existing one or create a new label.

21.3.3.5 Assign Nodes



Note: Ranges (like: 10.0.0.1 - 10.0.0.20) can be used, and/or comma separated values.



Attention: If the IPMI Addresses list and/or the Availability Zones list are changed after assigning items from these lists to the node groups, all existing node assignments will be lost. In this case, it is necessary to re-assign the IPMIs and Availability Zones to the node groups.



Note: The Host Groups will only be enabled for assignment when all the entities are configured

22 Kubernetes configuration

22.1 Configure ImagePullSecret

About this task

The following procedure configures a secret to store login information when pulling images from a private Docker registry. It is recommended to add ImagePullSecrets to a serviceAccount resource.

1. Create a new secret labeled "myregistrykey".

Command syntax:

```
kubectl create secret docker-registry <name> --docker-server=<your_registry_server> --docker-username=<username> --docker-password=<password> --docker-email=<email>
```

Example command:

```
kubectl create secret docker-registry myregistrykey --docker-server=https://index.docker.io/v1 --docker-username=myusername --docker-password=mypassword --docker-email=user@mydomain.com
```

2. Verify the secret has been created.

```
kubectl get secrets myregistrykey
```

Example output:

```
NAME      TYPE      DATA      AGE
myregistrykey  kubernetes.io/.dockerconfigjson  1  1d
```

3. Modify the service account for the namespace to use this secret as an imagePullSecret.

Example command:

```
kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "myregistrykey"}]}'
```

4. Any new Pods created in the current namespace will have the following details added to their spec:

```
spec:
  imagePullSecrets:
    - name: myregistrykey
```

22.2 Topology Manager

Topology Manager is a Kubelet component that aims to co-ordinate the set of components that are responsible for these optimizations.

Before you begin<https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/#before-you-begin>

You need to have a Kubernetes cluster, and the `kubectl` command-line tool must be configured to communicate with your cluster. If you do not already have a cluster, you can create one by using `minikube` or you can use one of these Kubernetes playgrounds:

- [Katacoda](#)
- [Play with Kubernetes](#)

Your Kubernetes server must be at or later than version v1.18. To check the version, enter `kubectl version`.

22.2.1 How the Topology Manager works

Prior to the introduction of Topology Manager, the CPU and Device Manager in Kubernetes make resource allocation decisions independently of each other. This can result in undesirable allocations on multiple-socketed systems, performance/latency sensitive applications will suffer due to these undesirable allocations. Undesirable in this case meaning for example, CPUs and devices being allocated from different NUMA Nodes thus, incurring additional latency.

The Topology Manager is a Kubelet component, which acts as a source of truth so that other Kubelet components can make topology aligned resource allocation choices.

The Topology Manager provides an interface for components, called *Hint Providers*, to send and receive topology information. Topology Manager has a set of node level policies which are explained below.

The Topology manager receives Topology information from the *Hint Providers* as a bitmask denoting NUMA Nodes available and a preferred allocation indication. The Topology Manager policies perform a set of operations on the hints provided and converge on the hint determined by the policy to give the optimal result, if an undesirable hint is stored the preferred field for the hint will be set to false. In the current policies preferred is the narrowest preferred mask. The selected hint is stored as part of the Topology Manager. Depending on the policy configured the pod can be accepted or rejected from the node based on the selected hint. The hint is then stored in the Topology Manager for use by the *Hint Providers* when making the resource allocation decisions.

Enable the Topology Manager feature

Support for the Topology Manager requires `TopologyManager` *feature gate* to be enabled. It is enabled by default starting with Kubernetes 1.18.

Topology Manager Scopes and Policies

The Topology Manager currently:

- Aligns Pods of all QoS classes.

- Aligns the requested resources that Hint Provider provides topology hints for.

If these conditions are met, the Topology Manager will align the requested resources.

In order to customise how this alignment is carried out, the Topology Manager provides two distinct knobs: **scope** and **policy**.

The **scope** defines the granularity at which you would like resource alignment to be performed (e.g. at the pod or container level). And the **policy** defines the actual strategy used to carry out the alignment (e.g. best-effort, restricted, single-numa-node, etc.).

Details on the various **scopes** and **policies** available today can be found below.

Note: To align CPU resources with other requested resources in a Pod Spec, the CPU Manager should be enabled and proper CPU Manager policy should be configured on a Node. See [control CPU Management Policies](#).

22.2.1.1 Topology Manager Scopes

The Topology Manager can deal with the alignment of resources in a couple of distinct scopes:

- **container** (default)
- **pod**

Either option can be selected at a time of the kubelet startup, with `--topology-manager-scope` flag.

Container scope

The **container** scope is used by default.

Within this scope, the Topology Manager performs a number of sequential resource alignments, i.e., for each container (in a pod) a separate alignment is computed. In other words, there is no notion of grouping the containers to a specific set of NUMA nodes, for this particular scope. In effect, the Topology Manager performs an arbitrary alignment of individual containers to NUMA nodes.

The notion of grouping the containers was endorsed and implemented on purpose in the following scope, for example the **pod** scope.

Pod scope

To select the **pod** scope, start the kubelet with the command line option `--topology-manager-scope=pod`.

This scope allows for grouping all containers in a pod to a common set of NUMA nodes. That is, the Topology Manager treats a pod as a whole and attempts to allocate the entire pod (all containers) to either a single NUMA node or a common set of NUMA nodes. The following examples illustrate the alignments produced by the Topology Manager on different occasions:

- all containers can be and are allocated to a single NUMA node;
- all containers can be and are allocated to a shared set of NUMA nodes.

The total amount of particular resource demanded for the entire pod is calculated according to [effective requests/limits](#) formula, and thus, this total value is equal to the maximum of:

- the sum of all app container requests,
- the maximum of init container requests, for a resource.

Using the `pod` scope in tandem with `single-numa-node` Topology Manager policy is specifically valuable for workloads that are latency sensitive or for high-throughput applications that perform IPC. By combining both options, you are able to place all containers in a pod onto a single NUMA node; hence, the inter-NUMA communication overhead can be eliminated for that pod.

In the case of `single-numa-node` policy, a pod is accepted only if a suitable set of NUMA nodes is present among possible allocations. Reconsider the example above:

- a set containing only a single NUMA node - it leads to pod being admitted,
- whereas a set containing more NUMA nodes - it results in pod rejection (because instead of one NUMA node, two or more NUMA nodes are required to satisfy the allocation).

To recap, Topology Manager first computes a set of NUMA nodes and then tests it against Topology Manager policy, which either leads to the rejection or admission of the pod.

22.2.1.2 Topology Manager Policies

The Topology Manager supports four allocation policies. You can set a policy via a Kubelet flag, `--topology-manager-policy`. These four supported policies in NCS, are categorized as follows:

- **none (default)**
- **best-effort**
- **restricted**
- **strict**

 **Note:** If Topology Manager is configured with the `pod` scope, the container, which is considered by the policy, is reflecting requirements of the entire pod, and thus each container from the pod will result with **the same** topology alignment decision.

 **Important:** There are only two supported flags: `best-effort` and `single-numa-node` in NCS, out of the four available in Kubernetes.

none policy

This is the default policy and does not perform any topology alignment.

best-effort policy

For each container in a Pod, the kubelet, with `best-effort` topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager stores the preferred NUMA Node affinity for that container. If the affinity is not preferred, Topology Manager will store this and admit the pod to the node anyway.

The *Hint Providers* can then use this information when making the resource allocation decision.

restricted policy

For each container in a Pod, the kubelet, with **restricted** topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager stores the preferred NUMA Node affinity for that container. If the affinity is not preferred, Topology Manager will reject this pod from the node. This will result in a pod in a Terminated state with a pod admission failure.

Once the pod is in a Terminated state, the Kubernetes scheduler will **not** attempt to reschedule the pod. It is recommended to use a ReplicaSet or Deployment to trigger a redeploy of the pod. An external control loop could be also implemented to trigger a redeployment of pods that have the Topology Affinity error.

If the pod is admitted, the *Hint Providers* can then use this information when making the resource allocation decision.

strict policy (also known as single-numa-node)

For each container in a Pod, the kubelet, with **single-numa-node** topology management policy, calls each Hint Provider to discover their resource availability. Using this information, the Topology Manager determines if a single NUMA Node affinity is possible. If it is, Topology Manager will store this and the *Hint Providers* can then use this information when making the resource allocation decision. If, however, this is not possible then the Topology Manager will reject the pod from the node. This will result in a pod in a Terminated state with a pod admission failure.

Once the pod is in a Terminated state, the Kubernetes scheduler will **not** attempt to reschedule the pod. It is recommended to use a Deployment with replicas to trigger a redeploy of the Pod. An external control loop could be also implemented to trigger a redeployment of pods that have the Topology Affinity error.

22.2.2 Configurations of Kubernetes Topology Manager in NCS

The following table summarizes the Topology Manager related configuration combinations available in NCS 20FP2SU2:

Table 76: Topology Manager configurations

| NCS topology_manager_policy | k8s topologyManagerPolicy | resource_pooling |
|-----------------------------|---------------------------|------------------|
| disabled | N/A* | N/A* |
| best-effort | best-effort | static |
| best-effort | best-effort | dynamic |
| strict | single-numa-node | static |

Table 76: Topology Manager configurations (continued)

| NCS topology_manager_policy | k8s topologyManagerPolicy | resource_pooling |
|-----------------------------|---------------------------|------------------|
| strict | single-numa-node | dynamic |

N/A*: if the topology_manager_policy is 'disabled', Kubernetes TopologyManager feature gate is turned off.

For more details of the two available topology_manager_policy options ('best-effort' and 'strict'), refer to [How the Topology Manager works](#) on page 796.

For more information see [Resource Pooling](#) on page 807.

Resource pooling

The 'static' choice of 'resource_pooling' configuration item is the default value, it has no effect on the functionality of the cluster.

The 'dynamic' option of 'resource_pooling' configuration item has effect on the handling of device pools in kubernetes, namely on the 'exclusive' CPU pools of CPU Pooler, and the SR-IOV device pools of the SR-IOV device plugin.

Setting this configuration item to 'dynamic' will merge the requested CPUs of 'NUMA 0 Exclusive CPU Pool' and 'NUMA 0 Exclusive CPU Pool' to a new pool named 'exclusive_pool'. For example:

```
kubectl describe node:
[...]
nokia.k8s.io/exclusive numa_0_pool: 10
nokia.k8s.io/exclusive numa_1_pool: 10
[...]
```

to

```
kubectl describe node:?
[...]
nokia.k8s.io/exclusive_pool: 20
[...]
```

In the case of SR-IOV device pools, the PF specific pools will be merged into one common pool for all the SR-IOV enabled PFs. For example:

```
# kubectl describe nodes
[...]
nokia.k8s.io/sriov_ens1f0: 8
nokia.k8s.io/sriov_vfio_ens1f0: 2
nokia.k8s.io/sriov_ens1f1: 8
nokia.k8s.io/sriov_vfio_ens1f1: 2
```

[...]

to:

```
# kubectl describe nodes  
[ ... ]  
nokia.k8s.io/sriov:          16  
nokia.k8s.io/sriov_vfio:    4  
[ ... ]
```

Requesting these resources for the applications will require the usage of the merged pools in case 'dynamic' resource pooling is used.

22.2.2.1 Provisioning the Topology Manager policy on the NCS manager

Disabled:

K8s Topology Manager Policy

Best-effort:

K8s Topology Manager Policy

Resource Pooling

Strict:

K8s Topology Manager Policy

Resource Pooling



Note: Resource Pooling has two options: dynamic and static.

22.2.2.2 Examples for SRIOV and CPU usage with Topology Manager

Table 77: fsfsfsf

| | topology_manager_policy | resource_pooling | k8s cpu manager | sriov pool name | exclusive pool name |
|---|--------------------------------|-------------------------|------------------------|--|--|
| 1 | strict (single-numa-node) | dynamic | CPUManager | nokia.
k8s.io/
sriov | n/a |
| 2 | strict (single-numa-node) | static | CPUManager | nokia.
k8s.io/
sriov_<
pf_name1>
nokia.
k8s.io/
sriov_<
pf_name2> | n/a |
| 3 | strict (single-numa-node) | dynamic | CPU Pooler | nokia.
k8s.io/
sriov | nokia.
k8s.io/
exclusive_pool |
| 4 | strict (single-numa-node) | static | CPU Pooler | nokia.
k8s.io/
sriov_<
pf_name1>
nokia.
k8s.io/
sriov_<
pf_name2> | nokia.
k8s.io/
exclusive_numa_0_pool
nokia.
k8s.io/
exclusive_numa_1_pool |

Config 1 - resource_pooling 'dynamic', CPUManager ON

SRIOV pools are merged.

kubectl output

```
kubectl get node baremetal-baremetal-fi-805-allinone-0 -o json | jq  
'.status.allocatable'  
{  
    "cpu": "72",  
    "ephemeral-storage": "757864683085",  
    "hugepages-1Gi": "0",  
    "hugepages-2Mi": "0",  
    "memory": "356155840Ki",  
    "nokia.k8s.io/sriov": "20",  
    "pods": "110"  
}
```

example.yaml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: test-pod  
  labels:  
    app: test-app  
spec:  
  containers:  
    - name: test-container  
      image: bcmt-registry:5000/busybox:latest  
      imagePullPolicy: IfNotPresent  
      command: ["/bin/sh"]  
      args: ["-c", "sleep 10000"]  
      resources:  
        limits:  
          memory: 100Mi  
          cpu: 5  
          nokia.k8s.io/sriov: 2  
        requests:  
          memory: 100Mi  
          cpu: 5  
          nokia.k8s.io/sriov: 2  
  nodeSelector:  
    kubernetes.io/hostname: baremetal-baremetal-fi-805-allinone-0
```

Config 2 - resource_pooling 'static', CPUManager ON

No pools are merged

kubectl output

```
kubectl get node baremetal-baremetal-fi-805-allinone-0 -o json | jq  
'.status.allocatable'  
{  
    "cpu": "72",  
    "ephemeral-storage": "757864683085",  
    "hugepages-1Gi": "0",  
    "hugepages-2Mi": "0",  
    "memory": "356155840Ki",  
    "nokia.k8s.io/sriov_ens3f0": "10",  
    "nokia.k8s.io/sriov_ens3f1": "10",  
    "pods": "110"  
}
```

example.yaml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: test-pod  
  labels:  
    app: test-app  
spec:  
  containers:  
    - name: test-container  
      image: bcmt-registry:5000/busybox:latest  
      imagePullPolicy: IfNotPresent  
      command: ["/bin/sh"]  
      args: ["-c", "sleep 10000"]  
      resources:  
        limits:  
          memory: 100Mi  
          cpu: 5  
          nokia.k8s.io/sriov_ens3f0: 2  
          nokia.k8s.io/sriov_ens3f1: 2  
        requests:  
          memory: 100Mi  
          cpu: 5  
          nokia.k8s.io/sriov_ens3f0: 2  
          nokia.k8s.io/sriov_ens3f1: 2  
      nodeSelector:  
        kubernetes.io/hostname: baremetal-baremetal-fi-805-allinone-0
```

Config 3 - resource_pooling 'dynamic', CPUManager OFF, CPU Pooler enabled

Enable Exclusive CPU Pool

Allocated CPUs for NUMA 0 Exclusive CPU Pool

Allocated CPUs for NUMA 1 Exclusive CPU Pool

Allocated CPUs for Shared CPU Pool

The cpus of the two pools are merged into "nokia.k8s.io/exclusive_pool": "20", SRIOV pools are also merged.

kubectl output

```
kubectl get node baremetal-baremetal-fi-805-workerbm-0 -o json | jq  
'.status.allocatable'  
{  
  "cpu": "4",  
  "ephemeral-storage": "757864683085",  
  "hugepages-1Gi": "74Gi",  
  "memory": "294390208Ki",  
  "nokia.k8s.io/exclusive_pool": "20",  
  "nokia.k8s.io/lv-capacity": "937198188Ki",  
  "nokia.k8s.io/shared_pool": "32K",  
  "nokia.k8s.io/sriov": "20",  
  "pods": "110"  
}
```

example.yaml

```
apiVersion: v1  
kind: Pod  
metadata:  
  name: test-pod  
  labels:  

```

```
hugepages-1Gi: 1Gi
nokia.k8s.io/sriov: 2
nokia.k8s.io/exclusive_pool: 5
requests:
  memory: 100Mi
  hugepages-1Gi: 1Gi
  nokia.k8s.io/sriov: 2
  nokia.k8s.io/exclusive_pool: 5
nodeSelector:
  kubernetes.io/hostname: baremetal-baremetal-fi-805-workerbm-0
```

Config 4 - resource_pooling 'static', CPUManager OFF, CPU Pooler enabled

Enable Exclusive CPU Pool

Allocated CPUs for NUMA 0 Exclusive CPU Pool

5

Allocated CPUs for NUMA 1 Exclusive CPU Pool

15

Allocated CPUs for Shared CPU Pool

16

No pools are merged.

kubectl output

```
kubectl get node baremetal-baremetal-fi-805-edgebm-0 -o json | jq
'.status.allocatable'
{
  "cpu": "4",
  "ephemeral-storage": "757864683085",
  "hugepages-1Gi": "74Gi",
  "memory": "294390208Ki",
  "nokia.k8s.io/exclusive numa_0_pool": "5",
  "nokia.k8s.io/exclusive numa_1_pool": "15",
  "nokia.k8s.io/lv-capacity": "937198188Ki",
  "nokia.k8s.io/shared_pool": "32k",
  "nokia.k8s.io/sriov_ens3f0": "10",
  "nokia.k8s.io/sriov_ens3f1": "10",
  "pods": "110"
}
```

example.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
```

```

labels:
  app: test-app
spec:
  containers:
    - name: test-container
      image: bcmt-registry:5000/busybox:latest
      imagePullPolicy: IfNotPresent
      command: [ "/bin/sh" ]
      args: [ "-c", "sleep 10000" ]
      resources:
        limits:
          memory: 100Mi
          hugepages-1Gi: 1Gi
          nokia.k8s.io/sriov_ens3f0: 1
          nokia.k8s.io/sriov_ens3f1: 1
          nokia.k8s.io/exclusive numa_0_pool: 5
        requests:
          memory: 100Mi
          hugepages-1Gi: 1Gi
          nokia.k8s.io/sriov_ens3f0: 1
          nokia.k8s.io/sriov_ens3f1: 1
          nokia.k8s.io/exclusive numa_0_pool: 5
      nodeSelector:
        kubernetes.io/hostname: baremetal-baremetal-fi-805-edgebm-0

```

22.2.2.3 Resource Pooling

Static Policy

The static policy allows containers in Guaranteed pods with integer CPU requests access to exclusive CPUs on the node. This exclusivity is enforced using the *cpuset cgroup controller*.

 **Note:** System services such as the container runtime and the kubelet itself can continue to run on these exclusive CPUs. The exclusivity only extends to other pods.

 **Note:** CPU Manager doesn't support offline and online of CPUs at runtime. Also, if the set of online CPUs changes on the node, the node must be drained and CPU manager manually reset by deleting the state file `cpu_manager_state` in the kubelet root directory.

This policy manages a shared pool of CPUs that initially contains all CPUs in the node. The amount of exclusively allocatable CPUs is equal to the total number of CPUs in the node minus any CPU reservations by the kubelet `--kube-reserved` or `--system-reserved` options.

Exclusive Policy

From 1.17, the CPU reservation list can be specified explicitly by kubelet `--reserved-cpus` option.

The explicit CPU list specified by `--reserved-cpus` takes precedence over the CPU reservation specified by `--kube-reserved` and `--system-reserved`. CPUs reserved by these options are taken, in integer quantity, from the initial shared pool in ascending order by physical core ID. This shared pool is the set of CPUs on which any containers in `BestEffort` and `Burstable` pods run.

Containers in `Guaranteed` pods with fractional CPU requests also run on CPUs in the shared pool. Only containers that are both part of a `Guaranteed` pod and have integer CPU requests are assigned exclusive CPUs.

First, *enable CPU manager with the Static policy* in the Kubelet running on the compute nodes of your cluster. Then configure your pod to be in the *Guaranteed Quality of Service (QoS) class*. Request whole numbers of CPU cores (e.g., `1000m`, `4000m`) for containers that need exclusive cores. Create your pod in the same way as before (e.g., `kubectl create -f pod.yaml`). And *voilà*, the CPU manager will assign exclusive CPUs to each of container in the pod according to their CPU requests.

 **Note:** The kubelet requires a CPU reservation greater than zero be made using either `--kube-reserved` and/or `--system-reserved` or `--reserved-cpus` when the static policy is enabled. This is because zero CPU reservation would allow the shared pool to become empty.

The 'static' choice of 'resource_pooling' configuration item is the default value, it has no effect on the functionality of the cluster.

The 'dynamic' option of 'resource_pooling' configuration item has effect on the handling of device pools in kubernetes, namely on the 'exclusive' CPU pools of CPU Pooler, and the SR-IOV device pools of the SR-IOV device plugin.

Setting this configuration item to 'dynamic' will merge the requested CPUs of 'NUMA 0 Exclusive CPU Pool' and 'NUMA 0 Exclusive CPU Pool' to a new pool named 'exclusive_pool'. For example:

```
kubectl describe node:  
[...]  
nokia.k8s.io/exclusive numa_0_pool: 10  
nokia.k8s.io/exclusive numa_1_pool: 10  
[...]
```

to

```
kubectl describe node:  
[...]  
nokia.k8s.io/exclusive_pool: 20  
[...]
```

In the case of SR-IOV device pools, the PF specific pools will be merged into one common pool for all the SR-IOV enabled PFs. For example:

```
# kubectl describe nodes  
[...]  
nokia.k8s.io/sriov_ens1f0: 8  
nokia.k8s.io/sriov_vfio_ens1f0: 2
```

```
nokia.k8s.io/sriov_ens1f1:      8  
nokia.k8s.io/sriov_vfio_ens1f1:  2  
[...]
```

to:

```
# kubectl describe nodes  
[...]  
nokia.k8s.io/sriov:          16  
nokia.k8s.io/sriov_vfio:    4  
[...]
```

Requesting these resources for the applications will require the usage of the merged pools in case 'dynamic' resource pooling is used.

22.2.3 Known Limitations

1. The maximum number of NUMA nodes that Topology Manager allows is 8. With more than 8 NUMA nodes, there will be a state explosion when trying to enumerate the possible NUMA affinities and generating their hints.
2. The scheduler is not topology-aware, so it is possible to be scheduled on a node then fail on the node due to the Topology Manager.
3. The Device Manager and the CPU Manager are the only components to adopt the Topology Manager's HintProvider interface. This means that NUMA alignment can only be achieved for resources managed by the CPU Manager and the Device Manager. Memory or HugePages are not considered by the Topology Manager for NUMA alignment.
4. The topologyManagerScope is not available in the kubelet version shipped with NCS, so Topology Manager works with the default *container* scope.
5. Kube have a limitation to support HugePage per NUMA awareness as described in the following <https://kubernetes.io/docs/tasks/administer-cluster/topology-manager/>



Note: The last mentioned limitation should be considered during the design to avoid code changes in future releases when kube will support HugePages per NUMA awareness.

23 Appendix: Example for cpro-values.yaml files

 **Note:** The following registry URLs are provided as examples, the user should add their own.

```
global:
  registry: "csf-docker-delivered.[]" in the appropriate repository.
# registry1 repo is used to pull alertmanager, nodeExporter, server,
pushgateway and restserver components images
  registry1: "csf-docker-delivered.[]" in the appropriate repository.
# To configure annotations and labels for CPRO components resources
  annotations: {}
  labels: {}

## Define serviceAccount name for prometheus at global level.
## serviceAccount priority order is
## 1. serviceAccountName
## 2. global.serviceAccountName
## 3. If we are not using customized resources set rbac.enabled to true then
  resources will be created on helm install
## 4. If both serviceAccounts are not set and rbac.enabled is set to false
  then default serviceAccount will be used
##
  serviceAccountName:

## istio verion in X.Y format eg 1.4/1.5/1.6
  istioVersion: 1.4
  podNamePrefix: ""
  containerNamePrefix: ""

customResourceNames:
  resourceNameLimit: 63
  alertManagerPod:
    alertManagerContainer: ""
    configMapReloadContainer: ""
    cproUtil: ""
  restServerPod:
    restServerContainer: ""
    configMapReloadContainer: ""
  restServerHelmTestPod:
    name: ""
    testContainer: ""
  toolsPod:
    initContainer: ""
  serverPod:
    inCntInitChownData: ""
```

```

inCntInitConfigFile: ""
configMapReloadContainer: ""
serverContainer: ""
monitoringContainer: ""
cproUtil: ""

pushGatewayPod:
  pushGatewayContainer: ""

kubeStateMetricsPod:
  kubeStateMetricsContainer: ""

hooks:
  postDeleteJobName: ""
  postDeleteContainer: ""

webhook4fluentd:
  webhookContainer: ""

nodeExporter:
  nodeExporterContainer: ""

zombieExporter:
  zombieExporterContainer: ""

migrate:
  preUpgradePodName: ""
  preUpgradeContainer: ""
  postUpgradePodName: ""
  postUpgradeContainer: ""

serverHelmTestPod:
  name: ""
  testContainer: ""

custom:
# To configure customized annotations and labels for CPRO components psp
  psp:
    annotations:
      seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
  default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
      apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
  default
      apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    labels: {}

# To configure customized annotations and labels for CPRO components pods
  pod:
    annotations:
      seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
  default

```

```
    seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
        apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
        apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    labels: {}

rbac:
    enabled: true
    pspUseAppArmor: false

deployOnComPaaS: false

# These certs are currently used for fetching metrics from etcd.
# By default this will be disabled. user need to enable it
#
certManager:
    used: false
    duration: "8760h" # 365d
    renewBefore: "360h" # 15d
    keySize: "2048"
    api: "cert-manager.io/v1alpha2"
    #servername: "bcmt.domain"
    # As per the tests in sandbox and environment dnsNames: localhost was
    # sufficient for scrapping etcd metrics
    dnsNames:
        - localhost
    domain:
        #ipAddresses:
        # - 127.0.0.1
    issuerRef:
        name: ncms-ca-issuer
        # We can reference ClusterIssuers by changing the kind here.
        # The default value is Issuer (i.e. a locally namespaced Issuer)
        kind: ClusterIssuer

## If the flag is set to true, then server component scrape the metrics within
## listed namespaces
## else scrape the metrics at cluster level.
## Namespace listed in values_restricted.yaml file.
## If the flag is set to true, then restserver component is forbidden to
## access
## the configmap or any other resources of prometheus server from different
## namespaces and role/rolebinding is created
```

```
## else clusterrole/clusterrolebinding is created and able to access the
## configmaps
## and other resources of prometheus server across the different namespaces.
restrictedToNamespace: false

selinuxOptions:
  enabled: false
  level: ""
  role: ""
  type: ""
  user: ""

## Define serviceAccount name for alertmanager, kubeStateMetrics, pushgateway,
## server, webhook4fluentd, restserver and migrate components.
## Defaults to component's fully qualified name.
##
serviceAccountName:

## Define serviceAccount name for nodeExporter and zombieExporter components.
## Defaults to component's fully qualified name.
##
exportersServiceAccountName:

## If true, high availability feature will be enabled
## altermanger and server could create 2 instances
## If false, altermanger and server could create only 1 instance
##
ha:
  enabled: false

helmDeleteImage:
  imageRepo: tools/kubectl
  imageTag: v1.19.5-nano-20201210
  imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 100m
      memory: 100Mi
    requests:
      cpu: 50m
      memory: 32Mi

## If true, pvc of alertmanager and server will be reserved
## it's only useful when ha.enabled == true
persistence:
```

```
reservePvc: false

## Whether the chart will deploy on istio
istio:
  enable: false
  mtls_enable: true
  cni_enable: true
  test_timeout: 60

##sensitive details in prometheus config file
# sensitive details like username, password , token will be given in the
# secret format
# please refer to the confluence page for details on secret format and
# specific name for the key.

sensitiveDataSecretName: ""

alertmanager:
  ## If false, alertmanager will not be installed
  ##
  enabled: true

  ## Will you config DNS for pod? Assume true by default.
  ## work only if ha is true
  dnsConfig: true
  ##soft means preferredDuringSchedulingIgnoredDuringExecution
  ##hard means requiredDuringSchedulingIgnoredDuringExecution
  ##you could refer k8s api for more detail
  ##
  antiAffinityMode: "soft"

  ## alertmanager container name
  ##
  name: alertmanager

  ## alertmanager distroless container image
  ##
  distroImage:
    imageRepo: cpro/registry4/alertmanager
    imageTag: v0.21.0-6
    imagePullPolicy: IfNotPresent

  ## Additional alertmanager container arguments
  ##
```

```
extraArgs: {}

## The URL prefix at which the container can be accessed. Useful in the case
the '-web.external-url' includes a slug
## so that the various internal URLs are still able to access as they are in
the default case.
## (Optional)
## prefixURL: "alertmanager"
prefixURL: ""

## External URL which can access alertmanager
## Maybe same with Ingress host name
## Alertmanager validates the 'web.external-url', configure valid baseUrl if
not alertmanager installation
## will fail with error. So "/" is removed.
##
## when istio is enabled: baseURL path and Contextroot path should match
## baseURL: "http://localhost:31380/alertmanager"
baseURL: ""

## Additional alertmanager container environment variable
## For instance to add a http_proxy
##
extraEnv: {}

## ConfigMap override where fullname is {{.Release.Name}}-
{{.Values.alertmanager.configMapOverrideName}}
## Defining configMapOverrideName will cause templates/alertmanager-
configmap.yaml
## to NOT generate a ConfigMap resource
##
configMapOverrideName: ""

## Alertmanager TLS for outbound messages
## e.g. if connected to CNOT this should be set to CNOT's wildfly cert.
outboundTLS:
  enabled: true
  cert:

"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNzRENDQVpnQ0NRRFQ1ZVM5UnFyR3ZEQU5CZ2txaGtpRz1

ingress:
  ## If true, alertmanager Ingress will be created
  ##
  enabled: false
```

```

## alertmanager Ingress annotations
##
annotations: {}
#   kubernetes.io/ingress.class: nginx
#   kubernetes.io/tls-acme: 'true'

## alertmanager Ingress additional labels
##
extraLabels: {}

## alertmanager Ingress hostnames with optional path
## Must be provided if Ingress is enabled
##
hosts: []
#   - alertmanager.domain.com
#   - domain.com/alertmanager

## alertmanager Ingress TLS configuration
## Secrets must be manually created in the namespace
##
tls: []
#   - secretName: prometheus-alerts-tls
#     hosts:
#       - alertmanager.domain.com
## Whether use istio ingress gateway(Envoy) alertmanager /
istioIngress:
  enabled: true
## when istio is enabled: baseURL path and Contextroot path should match
  Contextroot: alertmanager
  selector: {istio: ingressgateway}
## the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
## Keep gatewayName to empty to create kubernetes gateway resource. If new
## virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
## tls section will be used if gatewayName is ""
  tls:
    enabled: true
    httpsPort: 443
## mode could be SIMPLE, MUTUAL, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "am-gateway"
## Alertmanager Deployment Strategy type
# strategy:

```

```
# type: Recreate

## Node tolerations for alertmanager scheduling to nodes with taints
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
##
tolerations: []
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for alertmanager pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Use an alternate scheduler, e.g. "stork".
## ref: https://kubernetes.io/docs/tasks/administer-cluster/configure-multiple-schedulers/
##
# schedulerName:

persistentVolume:
## If true, alertmanager will create/use a Persistent Volume Claim
## If false, use emptyDir
##
enabled: true

## alertmanager data Persistent Volume access modes
## Must match those of existing PV or dynamic provisioner
## Ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
##
accessModes:
- ReadWriteOnce

## alertmanager data Persistent Volume Claim annotations
##
annotations: {}

## alertmanager data Persistent Volume existing claim name
## Requires alertmanager.persistentVolume.enabled: true
## If defined, PVC must be created manually before volume will be bound
existingClaim: ""

## alertmanager data Persistent Volume mount root path
```

```
##  
mountPath: /data  
  
## alertmanager data Persistent Volume size  
##  
size: 2Gi  
  
## alertmanager data Persistent Volume Storage Class  
## If defined, storageClassName: <storageClass>  
## If set to "-", storageClassName: "", which disables dynamic  
provisioning  
## If undefined (the default) or set to null, no storageClassName spec is  
## set, choosing the default provisioner. (gp2 on AWS, standard on  
## GKE, AWS & OpenStack)  
##  
storageClass: ""  
  
## Subdirectory of alertmanager data Persistent Volume to mount  
## Useful if the volume's root directory is not empty  
##  
subPath: ""  
  
## Annotations to be added to alertmanager pods  
##  
podAnnotations: {}  
  
## it's only used when ha.enabled true  
## when ha.enabled is false, replicaCount will be hard coded to 1  
replicaCount: 2  
  
## alertmanager resource requests and limits  
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/  
##  
resources:  
  limits:  
    cpu: 500m  
    memory: 1Gi  
  requests:  
    cpu: 10m  
    memory: 32Mi  
  
## Security context to be added to alertmanager pods  
##  
securityContext:  
  runAsUser: 65534
```

```
fsGroup: 65534

retention:
  time: 120h

service:
  annotationsForAlertmanagerCluster: {}
  annotationsForScrape:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: http
  annotations: {}
  labels: {}
  clusterIP: ""

## Enabling peer mesh service end points for enabling the HA alert manager
## Ref: https://github.com/prometheus/alertmanager/blob/master/README.md
# enableMeshPeer : true

## List of IP addresses at which the alertmanager service is available
## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips
##
externalIPs: []

loadBalancerIP: ""
loadBalancerSourceRanges: []
servicePort: 80
clusterPort: 8001
# nodePort: 30000
type: ClusterIP

## Configurations about alertmanager probes
## Including livenessProbe and readinessProbe
livenessProbe:
  initialDelaySeconds: 30
  timeoutSeconds: 30
  failureThreshold: 3
  periodSeconds: 10
readinessProbe:
  initialDelaySeconds: 30
  timeoutSeconds: 30
  failureThreshold: 3
  periodSeconds: 10

## Monitors ConfigMap changes and POSTs to a URL
## Ref: https://github.com/jimmidyson/configmap-reload
```

```
##  
configmapReload:  
  ## configmap-reload container name  
  ##  
  name: configmap-reload  
  
  ## configmap-reload distroless container image  
  ##  
  distroImage:  
    imageRepo: cpro/registry4/cproconfigmap-reload  
    imageTag: v0.0.8  
    imagePullPolicy: IfNotPresent  
  restapiConfigmapReloadImage:  
    imageRepo: cpro/registry4/configmap-reload-distro  
    imageTag: v0.2.1-4.2.3  
    imagePullPolicy: IfNotPresent  
  
  ## Additional configmap-reload container arguments  
  ##  
  extraArgs: {}  
  
  ## Additional configmap-reload mounts  
  ##  
  extraConfigmapMounts: []  
    # - name: prometheus-alerts  
    #   mountPath: /etc/alerts.d  
    #   configMap: prometheus-alerts  
    #   readOnly: true  
  
  ## configmap-reload resource requests and limits  
  ## Ref: http://kubernetes.io/docs/user-guide/compute-resources/  
  ##  
  resources:  
    limits:  
      cpu: 10m  
      memory: 32Mi  
    requests:  
      cpu: 10m  
      memory: 32Mi  
  
  tools:  
    image:  
      imageRepo: cpro/registry4/tools-image  
      imageTag: 1.8.0
```

```
imagePullPolicy: IfNotPresent

helmtest:
  CPROconfigmapname:
  deletepolicy: hook-succeeded,before-hook-creation

## helmtest resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
resources:
  limits:
    cpu: 10m
    memory: 32Mi
  requests:
    cpu: 10m
    memory: 32Mi

initConfigFile:

##initConfigFile container name
#
name: init-config-file

##initConfigFile container image
#
image:
  imageRepo: os_base/centos-nano
  imageTag: 7.9-20201202
  imagePullPolicy: IfNotPresent

##initConfigFile resource requests and limits
##Ref: http://kubernetes.io/docs/user-guide/compute-resources/
#
resources:
  limits:
    cpu: 10m
    memory: 32Mi
  requests:
    cpu: 10m
    memory: 32Mi

initChownData:
  ## If false, data ownership will not be reset at startup
  ## This allows the prometheus-server to be run with an arbitrary user
  ##
```

```
enabled: false

## initChownData container name
##
name: init-chown-data

## initChownData container image
##
image:
  imageRepo: os_base/centos-nano
  imageTag: 7.9-20201202
  imagePullPolicy: IfNotPresent

## initChownData resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 10m
    memory: 32Mi
  requests:
    cpu: 10m
    memory: 32Mi

kubeStateMetrics:
  ## If false, kube-state-metrics will not be installed
  ##
  enabled: true

  ## kube-state-metrics container name
  ##
  name: kube-state-metrics

  ## kube-state-metrics container image
  ##
  image:
    imageRepo: cpro/registry4/kube-state-metrics
    imageTag: v1.9.7-1.2.0
    imagePullPolicy: IfNotPresent

  ## kube-state-metrics container arguments
  ##
  args: {}

  ## Node tolerations for kube-state-metrics scheduling to nodes with taints
```

```
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
##
tolerations: []
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for kube-state-metrics pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Annotations to be added to kube-state-metrics pods
##
podAnnotations: {}

pod:
  labels: {}

replicaCount: 1

## kube-state-metrics resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 100m
    memory: 200Mi
  requests:
    cpu: 10m
    memory: 32Mi

## Security context to be added to kube-state-metrics pods
##
securityContext:
  runAsUser: 65534

service:
  annotations: {}
  annotationsForScrape:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: http
  labels: {}
```

```
# Exposed as a headless service:  
# https://kubernetes.io/docs/concepts/services-networking/service/  
#headless-services  
clusterIP: None  
  
## List of IP addresses at which the kube-state-metrics service is  
available  
## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips  
##  
externalIPs: []  
  
loadBalancerIP: ""  
loadBalancerSourceRanges: []  
servicePort: 80  
type: ClusterIP  
  
nodeExporter:  
## If false, node-exporter will not be installed  
##  
enabled: true  
  
## node-exporter container name  
##  
name: node-exporter  
  
## node-exporter distroless container image  
##  
distroImage:  
imageRepo: cpro/registry4/node_exporter  
imageTag: v1.0.1-6  
imagePullPolicy: IfNotPresent  
  
## Custom Update Strategy  
##  
updateStrategy:  
type: RollingUpdate  
  
## Additional node-exporter container arguments  
## Update "web.listen-address" If you have updated podHostPort &  
podContainerPort  
## web.listen-address value should be same as podHostPort &  
podContainerPort  
extraArgs:  
web.listen-address: ":9100"
```

```

## Additional node-exporter hostPath mounts
##
extraHostPathMounts: []
# - name: textfile-dir
#   mountPath: /srv/txt_collector
#   hostPath: /var/lib/node-exporter
#   readOnly: true

extraConfigmapMounts: []
# - name: certs-configmap
#   mountPath: /prometheus
#   configMap: certs-configmap
#   readOnly: true

## Node tolerations for node-exporter scheduling to nodes with taints
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
##
tolerations:
  - operator: Exists
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for node-exporter pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Annotations to be added to node-exporter pods
##
podAnnotations: {}

## Labels to be added to node-exporter pods
##
pod:
  labels: {}

## node-exporter resource limits & requests
## Ref: https://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 500m
    memory: 500Mi

```

```
requests:
  cpu: 100m
  memory: 30Mi

## Security context to be added to node-exporter pods
## Need to set SYS_TIME to enable timex collector, since v0.16.0
nodeExporterContainerSecurityContext:
  capabilities:
    add:
      - SYS_TIME

service:
  annotations:
    prometheus.io/probe: node-exporter
  labels: {}

  # Exposed as a headless service:
  # https://kubernetes.io/docs/concepts/services-networking/service/
#headless-services
  clusterIP: None

  ## List of IP addresses at which the node-exporter service is available
  ## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips
  ##
  externalIPs: []

  loadBalancerIP: ""
  loadBalancerSourceRanges: []
  servicePort: 9101
  type: ClusterIP

  podContainerPort: 9100
  podHostPort: 9100

zombieExporter:
  ## If false, zombie-exporter will not be installed
  ##
  enabled: false

  ## zombie-exporter container name
  ##
  name: zombie-exporter

  ## zombie-exporter container image
  ##
```

```
image:
  imageRepo: cpro/registry4/zombie-process-exporter
  imageTag: 1.7.0
  imagePullPolicy: IfNotPresent

## Custom Update Strategy
##
updateStrategy:
  type: RollingUpdate

## Additional zombie-exporter container arguments
##
extraArgs: {}

## Additional zombie-exporter hostPath mounts
##
extraHostPathMounts: []
# - name: textfile-dir
#   mountPath: /srv/txt_collector
#   hostPath: /var/lib/zombie-exporter
#   readOnly: true

extraConfigmapMounts: []
# - name: certs-configmap
#   mountPath: /prometheus
#   configMap: certs-configmap
#   readOnly: true

## Node tolerations for node-exporter scheduling to nodes with taints
## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
##
tolerations: []
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for node-exporter pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Annotations to be added to zombie-exporter pods
##
podAnnotations: {}
```

```
## Labels to be added to zombie-exporter pods
##
pod:
  labels: {}

## zombie-exporter resource limits & requests
## Ref: https://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 200m
    memory: 50Mi
  requests:
    cpu: 100m
    memory: 30Mi

## Security context to be added to zombie-exporter pods
##
securityContext:
  runAsUser: 65534
  fsGroup: 65534

service:
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: http
  labels: {}

  # Exposed as a headless service:
  # https://kubernetes.io/docs/concepts/services-networking/service/
#headless-services
  clusterIP: None

  ## List of IP addresses at which the node-exporter service is available
  ## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips
  ##
  externalIPs: []

hostPort: 8002
loadBalancerIP: ""
loadBalancerSourceRanges: []
servicePort: 8003
type: ClusterIP
scrapeInterval: 15
```

```
## logForwardToConsole indicates the location where log printed
## True: log printed to console
## False: log printed to logFile
logForwardToConsole: "True"
##enable when logForwardToConsole is not True
logFile: "/var/log/zombieprocessexporter.log"
##log_level : 0 - DEBUG, 1 - INFO, 2 - WARN, 3 - ERROR
logLevel: 1

## Cpro Util is Debug container image for AlertManager, Prometheus
cproUtil:

##cpro Util container name
name: util

## cpro util container image information
imageRepo: cpro/registry4/cpro-util
imageTag: v0.0.5
imagePullPolicy: IfNotPresent

##cpro util resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
resources:
  limits:
    cpu: 100m
    memory: 200Mi
  requests:
    cpu: 10m
    memory: 32Mi

server:
## Prometheus server container name
##
name: server

etcdCertMountPath: /etc/etcd/ssl

##soft means preferredDuringSchedulingIgnoredDuringExecution
##hard means requiredDuringSchedulingIgnoredDuringExecution
##you could refer k8s api for more detail
##
antiAffinityMode: "soft"

## Prometheus server distroless container image
##
```

```
distroImage:
  imageRepo: cpro/registry4/prometheus
  imageTag: v2.23.0-4
  imagePullPolicy: IfNotPresent

## The URL prefix at which the container can be accessed. Useful in the case
the '-web.external-url' includes a slug
## so that the various internal URLs are still able to access as they are in
the default case.
## (Optional)
## prefixURL: "prometheus"
prefixURL: ""

## External URL which can access prometheus server
## Maybe same with Ingress host name
## when istio is enabled: baseURL path and Contextroot path should match
## baseURL: "http://localhost:31380/prometheus"
baseURL: ""

## This flag controls access to the administrative HTTP API which includes
functionality such as deleting time
## series. This is disabled by default.
## To enable Backup/Restore feature(need to take Prometheus snapshot), this
must be set to "true"
enableAdminApi: true

## Uncomment the param if the restrictedToNamespace flag is set to true,
## then list the namespaces to monitor in comma-separated value
## Example: namespaceList: ['test1','test2']
#
#namespaceList: []

## Additional Prometheus server container arguments
## This flag controls the capacity of the queue for pending Alertmanager
notifications.

##
extraArgs: {}
  # alertmanager.notification-queue-capacity: 10000
  # storage.tsdb.min-block-duration: 2h

## Additional Prometheus server container only key arguments
## Since there is an enableAdminApi separate flag, no need to add here
## wal compression flag is added, flag enables compression of the write-
ahead log (WAL).
##
```

```

extraKeys: []
  # - storage.tsdb.wal-compression

## Additional Prometheus server hostPath mounts
##
extraHostPathMounts: []
  # - name: certs-dir
  #   mountPath: /etc/kubernetes/certs
  #   hostPath: /etc/kubernetes/certs
  #   readOnly: true

extraConfigmapMounts: []
  # - name: certs-configmap
  #   mountPath: /prometheus
  #   configMap: certs-configmap
  #   readOnly: true

## Additional Prometheus server Secret mounts
# Defines additional mounts with secrets. Secrets must be manually created
in the namespace.
extraSecretMounts: []
  # - name: secret-files
  #   mountPath: /etc/secrets
  #   secretName: prom-secret-files
  #   readOnly: true

## ConfigMap override where fullname is {{.Release.Name}}-
{{.Values.server.configMapOverrideName}}
## Defining configMapOverrideName will cause templates/server-configmap.yaml
## to NOT generate a ConfigMap resource
##
configMapOverrideName: ""

## Enable this if you are previously deploy in non-HA (K8S Deployment) and
want to migrate to HA (K8S Statefulset)
## It will migrate scraped data to new created PersistentVolumes mounted to
each Statefulset pod.
## Invalid in BCMT 19.06, needs enhancement
migrate:
  enabled: false
  name: migrate

## The file name you backup in non-HA. This file will be copied to CBUR
STATEFULSET folder so it could be backup when cpro is in HA mode
fileName: "20190603030217_e01_LOCAL_june-cpro-server.tar.gz"

```

```
## migrate resource requests and limits
# ## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
resources:
  requests:
    cpu: 100m
    memory: 64Mi
  limits:
    cpu: 200m
    memory: 128Mi

## Pointing to cbur glusterfs repo. By default it is mounted to BCMT
control node.
cbur:
  path: "192.168.199.10:cbur-glusterfs-repo"
  endpoint: "glusterfs-cluster"

## Time for migrating data from CBUR DEPLOYMENT to STATEFULSET.
moveDuration: 30

#Container is experimental and should not be enabled in production. Flag is
set as false by default
monitoringContainer:
  enabled: false

## monitoringContainer container name
name: monitoringContainer

## monitoringContainer container image
image:
  imageRepo: cpro/registry4/monitoring-container
  imageTag: v0.5.0
  imagePullPolicy: IfNotPresent

## monitoringContainer resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
resources:
  limits:
    cpu: 500m
    memory: 512Mi
  requests:
    cpu: 200m
    memory: 128Mi
```

```
## loglevel can be INFO,ERROR,DEBUG. Not mandatory. Taken as INFO as
default
loglevel: INFO

##removal of .tmp directories
removetmp:
#to be given as true/false
enable: true

##deltatime indicates buffer time for WAL compaction after min-block-
duration
##addition of deltatime is mandatory and should be a prime number near 60
minutes. Most preferable as 53 .
deltatime: 53

##comparing the WAL segments to it itself , to ensure rewriting of closed
segments is not happening.
checkwalfiles:
enable: true
compareInterval: 23

##compaction freqeucy is checked. Ensuring the compaction process is
correct. Checking happens once in 2hrs + delta by default.
checkpoint:
enable: true

cbur:
enabled: true

image:
imageRepo: cbur/cbura
imageTag: 1.0.3-1665
imagePullPolicy: IfNotPresent

resources:
limits:
cpu: 200m
memory: 200Mi
requests:
cpu: 100m
memory: 64Mi

backendMode: "local"
```

```

##autoEnableCron = true indicates that the cron job is immediately
scheduled when the BrPolicy is created,
##while autoEnableCron = false indicates that scheduling of the cron job
should be done on a subsequent backup request.
##This option only works when k8swatcher.enabled is true
autoEnableCron: false
##Indicate if subsequent update of cronjob will be done via brpolicy
update.
##true means cronjob must be updated via brpolicy update,
##false means cronjob must be updated via manual "helm backup -t app -a
enable/disable" command.
autoUpdateCron: false
## It is used for scheduled backup task. Empty string is allowed for no
scheduled backup.
## Here means every 5 minutes of every day
cronJob: "*/5 * * * *"

## This value only applies to statefulset, when ha.enabled is true. The
value can be 0,1 or 2.
## 0: backup any one of the PODs and restore to any one of PODs
## 1: backup any one of the POD, and restore it to all PODs one by one by
its index
## 2: backup all PODs and restore them one by one based by its index
##
brOption: 2

## Limit the number of copies that can be saved. Once it is reached, the
newer backup will overwritten the oldest one.
maxCopy: 5

ingress:
## If true, Prometheus server Ingress will be created
##
enabled: false

## Prometheus server Ingress annotations
##
annotations: {}
#   kubernetes.io/ingress.class: nginx
#   kubernetes.io/tls-acme: 'true'

## Prometheus server Ingress additional labels
##
extraLabels: {}

```

```

## Prometheus server Ingress hostnames with optional path
## Must be provided if Ingress is enabled
##
hosts: []
#   - prometheus.domain.com
#   - domain.com/prometheus

## Prometheus server Ingress TLS configuration
## Secrets must be manually created in the namespace
##
tls: []
#   - secretName: prometheus-server-tls
#     hosts:
#       - prometheus.domain.com

## Whether use istio ingress gateway(Envoy) for /graph
istioIngress:
  enabled: true
  ## when istio is enabled: baseURL path and Contextroot path should match
  Contextroot: prometheus
  selector: {istio: ingressgateway}
  # the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
  ## Keep gatewayName to empty to create kubernetes gateway resource. If new
  virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
  tls:
    enabled: true
    httpsPort: 443
    ## mode could be SIMPLE, MUTUAL, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "am-gateway"
  ## Server Deployment Strategy type
  # strategy:
  #   type: Recreate

  ## Node tolerations for server scheduling to nodes with taints
  ## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
  ##
  tolerations: []
    # - key: "key"
    #   operator: "Equal|Exists"
    #   value: "value"
    #   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

```

```
## Node labels for Prometheus server pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Use an alternate scheduler, e.g. "stork".
## ref: https://kubernetes.io/docs/tasks/administer-cluster/configure-multiple-schedulers/
##
# schedulerName:

persistentVolume:
    ## If true, Prometheus server will create/use a Persistent Volume Claim
    ## If false, use emptyDir
    ##
    enabled: true

    ## Prometheus server data Persistent Volume access modes
    ## Must match those of existing PV or dynamic provisioner
    ## Ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
    ##
    accessModes:
        - ReadWriteOnce

    ## Prometheus server data Persistent Volume annotations
    ##
    annotations: {}

    ## Prometheus server data Persistent Volume existing claim name
    ## Requires server.persistentVolume.enabled: true
    ## If defined, PVC must be created manually before volume will be bound
    existingClaim: ""

    ## Prometheus server data Persistent Volume mount root path
    ##
    mountPath: /data

    mountPath2: /data

    ## Prometheus server data Persistent Volume size
    ##
    size: 16Gi

    ## Prometheus server data Persistent Volume Storage Class
```

```
## If defined, storageClassName: <storageClass>
## If set to "-", storageClassName: "", which disables dynamic
provisioning
## If undefined (the default) or set to null, no storageClassName spec is
## set, choosing the default provisioner. (gp2 on AWS, standard on
## GKE, AWS & OpenStack)
##
storageClass: ""

## Subdirectory of Prometheus server data Persistent Volume to mount
## Useful if the volume's root directory is not empty
##
subPath: ""

## Annotations to be added to Prometheus server pods
##
podAnnotations: {}
# iam.amazonaws.com/role: prometheus
# Add below annotation when istio is enabled and prometheus needs to
scrape metrics from kubernetes-apiservers
#traffic.sidecar.istio.io/excludeOutboundPorts: "8443"

## it's only used when ha.enabled true
## when ha.enabled is false, replicaCount will be hard coded to 1
replicaCount: 2

## Prometheus server resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
limits:
cpu: 2
memory: 4Gi
requests:
cpu: 500m
memory: 512Mi

## Security context to be added to server pods
##
securityContext:
runAsUser: 65534
fsGroup: 65534

service:
annotations: {}
```

```
annotationsForScrape:  
  prometheus.io/probe: prometheus  
labels: {}  
clusterIP: ""  
  
## List of IP addresses at which the Prometheus server service is  
available  
## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips  
##  
externalIPs: []  
  
loadBalancerIP: ""  
loadBalancerSourceRanges: []  
servicePort: 80  
#nodePort: 31000  
type: ClusterIP  
  
## Prometheus server pod termination grace period  
##  
terminationGracePeriodSeconds: 10  
  
## Prometheus data retention  
##  
retention: {}  
## How long to retain samples in storage. Units supported: s, m, h, d, w, y.  
Default is 15d.  
# time: ""  
## Maximum number of bytes that can be stored for blocks. Units supported:  
KB, MB, GB, TB, PB. Default is disabled. For example 200GB.  
## If both time and size retention policies are specified, whichever policy  
triggers first will be used at that instant.  
# size: ""  
  
## Configurations about server probes  
## Including livenessProbe and readinessProbe  
livenessProbe:  
  initialDelaySeconds: 30  
  timeoutSeconds: 30  
  failureThreshold: 3  
  periodSeconds: 10  
readinessProbe:  
  initialDelaySeconds: 30  
  timeoutSeconds: 30  
  failureThreshold: 3  
  periodSeconds: 10
```

```
pushgateway:
    ## If false, pushgateway will not be installed
    ##
    enabled: true

    ## pushgateway container name
    ##
    name: pushgateway

    ## soft means preferredDuringSchedulingIgnoredDuringExecution
    ## hard means requiredDuringSchedulingIgnoredDuringExecution
    ## you could refer k8s api for more detail
    #
    antiAffinityMode: "soft"

    ## pushgateway container image
    ##
    image:
        imageRepo: cpro/registry4/pushgateway
        imageTag: v1.3.0-1.2.0
        imagePullPolicy: IfNotPresent

    ## Additional pushgateway container arguments
    ## For arguments where no value is needed give it as "".
    ## eg push.disable-consistency-check: ""
    extraArgs:
        push.disable-consistency-check: ""

    ## External URL which can access pushgateway
    ## when istio is enabled: baseURL path and Contextroot path should match
    ## baseURL: "http://localhost:31380/pushgateway"
    baseURL: ""

    ## The URL prefix at which the container can be accessed. Useful in the case
    ## the '-web.external-url' includes a slug
    ## so that the various internal URLs are still able to access as they are in
    ## the default case.
    ## (Optional)
    ## prefixURL: "pushgateway"
    prefixURL: ""

    ingress:
        ## If true, pushgateway Ingress will be created
        ##
```

```
enabled: false

## pushgateway Ingress annotations
##
annotations: {}
#   kubernetes.io/ingress.class: nginx
#   kubernetes.io/tls-acme: 'true'

## pushgateway Ingress hostnames with optional path
## Must be provided if Ingress is enabled
##
hosts: []
#   - pushgateway.domain.com
#   - domain.com/pushgateway

## pushgateway Ingress TLS configuration
## Secrets must be manually created in the namespace
##
tls: []
#   - secretName: prometheus-alerts-tls
#     hosts:
#       - pushgateway.domain.com

## Whether use istio ingress gateway(Envoy) pushgate /
istioIngress:
  enabled: true
  selector: {istio: ingressgateway}
  ## when istio is enabled: baseURL path and Contextroot path should match
  Contextroot: pushgateway
  # the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
  ## Keep gatewayName to empty to create kubernetes gateway resource. If new
  virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
  tls:
    enabled: false
    httpsPort: 443
    ## mode could be SIMPLE, MUTUAL, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "pushgateway-secret"
  ## Node tolerations for pushgateway scheduling to nodes with taints
  ## Ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
  ##
  tolerations: []
```

```
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for pushgateway pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Annotations to be added to pushgateway pods
##
podAnnotations: {}

replicaCount: 1

## pushgateway resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 100m
    memory: 200Mi
  requests:
    cpu: 10m
    memory: 32Mi

## Security context to be added to push-gateway pods
##
securityContext:
  runAsUser: 65534

service:
  annotations:
    prometheus.io/probe: pushgateway
  labels: {}
  clusterIP: ""

## List of IP addresses at which the pushgateway service is available
## Ref: https://kubernetes.io/docs/user-guide/services/#external-ips
##
externalIPs: []

loadBalancerIP: ""
loadBalancerSourceRanges: []
```

```
servicePort: 9091
type: ClusterIP

webhook4fluentd:
## If false, webhook4fluentd will not be installed
# ##
enabled: false

## webhook4fluentd container name
##
name: webhook4fluentd

##soft means preferredDuringSchedulingIgnoredDuringExecution
##hard means requiredDuringSchedulingIgnoredDuringExecution
##you could refer k8s api for more detail
##
antiAffinityMode: "soft"

## webhook4fluentd container image
# ##
image:
  imageRepo: cpro/registry4/webhook4fluentd
  imageTag: 3.3.0
  imagePullPolicy: IfNotPresent

tolerations: []
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

## Node labels for webhook4fluentd pod assignment
## Ref: https://kubernetes.io/docs/user-guide/node-selection/
##
nodeSelector: {}

## Annotations to be added to webhook4fluentd pods
##
podAnnotations: {}

replicaCount: 2

## webhook4fluentd resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
```

```

resources:
  limits:
    cpu: 100m
    memory: 200Mi
  requests:
    cpu: 10m
    memory: 32Mi

## Security context to be added to push-gateway pods
##
securityContext:
  runAsUser: 65534

service:
  annotations: {}
  annotationsForScrape:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: http
  labels: {}
  clusterIP: ""

  servicePort: 8005
  type: ClusterIP

## alertmanager ConfigMap entries
##

#this config is used when webhook4fluentd.enabled is false
#alertmanagerFiles, alertmanagerWebhookFiles are mutually exclusive
#only one of them will be used
alertmanagerFiles:
  alertmanager.yml:
    global: {}
    # slack_api_url: ''

receivers:
  - name: default-receiver
  ##
  ## uncomment following lines when need to send to CNOT
  ## suppose CNOT has service name 'tls-cnot' in namespace 'default'
  ##
  # webhook_configs:
  #   - url: 'http://tls-cnot.default.svc.cluster.local/api/cnot/v1/notif'

```

```

##  

## uncomment following lines when need to send to CNOT in TLS  

## also need to change url above from HTTP to HTTPS  

## please notice 'http_config' is under 'webhook_configs'  

##  

#   http_config:  

#     tls_config:  

#       ca_file: /etc/config/alertmanager/tls/alertmanager.cert  

route:  

  group_wait: 10s  

  group_interval: 5m  

  receiver: default-receiver  

  repeat_interval: 3h  

#this config is used when webhook4fluentd.enabled is true  

#the following is default value, it could be changed.  

#'release' in the url should be changed to the right value  

#port in url should be webhook4fluentd.service.servicePort if the value is not  

  8005  

alertmanagerWebhookFiles:  

  alertmanager.yml:  

    global: {}  

    receivers:  

      - name: "webhook"  

        webhook_configs:  

          - url: 'http://release-cpro-webhook4fluentd:8005'  

    route:  

      group_wait: 10s  

      group_interval: 5m  

      receiver: "webhook"  

## Prometheus server ConfigMap entries  

##  

serverFiles:  

  alerts:  

    groups:  

      - name: pre-defined-alert-rules  

        rules:  

          - alert: NodeFilesystemUsageMinor  

            expr: ((node_filesystem_size_bytes{device="rootfs"}  

- node_filesystem_free_bytes{device="rootfs"}) /  

node_filesystem_size_bytes{device="rootfs"} * 100 >  

70) and ((node_filesystem_size_bytes{device="rootfs"})

```

```
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 <= 80)
    for: 1m
  labels:
    severity: MINOR
    name: FileSystemExceedThresholdMinor
    text: Usage of file system is greater than the threshold value
    id: 3001200
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC001
  annotations:
    summary: "{{\$labels.instance}}: High Filesystem usage detected"
    description: "{{\$labels.instance}}: Filesystem usage is above 70%"
- alert: NodeFilesystemUsageMajor
  expr: ((node_filesystem_size_bytes{device="rootfs"} /
node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 >
80) and ((node_filesystem_size_bytes{device="rootfs"} /
node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 <= 90)
    for: 1m
  labels:
    severity: MAJOR
    name: FileSystemExceedThresholdMajor
    text: Usage of file system is greater than the threshold value
    id: 3001201
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC002
  annotations:
    summary: "{{\$labels.instance}}: High Filesystem usage detected"
    description: "{{\$labels.instance}}: Filesystem usage is above 80%"
- alert: NodeFilesystemUsageCritical
  expr: (node_filesystem_size_bytes{device="rootfs"} /
node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 > 90
    for: 1m
  labels:
    severity: CRITICAL
    name: FileSystemExceedThresholdCritical
```

```

text: Usage of file system is greater than the threshold value
id: 3001202
source: CPRO
host: "{$labels.kubernetes_io_hostname}"
eventType: 11
probableCause: 351
key: MOC003
annotations:
  summary: "{$labels.instance}: High Filesystem usage detected"
  description: "{$labels.instance}: Filesystem usage is above 90%"
- alert: NodeCPUUsageMinor
  expr: (100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m]) * 100 > 70) and (100 - avg
by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m]) * 100 <= 80)
for: 1m
labels:
  severity: MINOR
  name: CPUExceedThresholdMinor
  text: Usage of CPU is greater than the threshold value
  id: 3001203
  source: CPRO
  host: "{$labels.kubernetes_io_hostname}"
  eventType: 11
  probableCause: 351
  key: MOC004
  annotations:
    summary: "{$labels.instance}: High CPU usage detected"
    description: "{$labels.instance}: CPU usage is above 70%"
- alert: NodeCPUUsageMajor
  expr: (100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m]) * 100 > 80) and (100 - avg
by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m]) * 100 <= 90)
for: 1m
labels:
  severity: MAJOR
  name: CPUExceedThresholdMajor
  text: Usage of CPU is greater than the threshold value
  id: 3001204
  source: CPRO
  host: "{$labels.kubernetes_io_hostname}"
  eventType: 11
  probableCause: 351
  key: MOC005
  annotations:
    summary: "{$labels.instance}: High CPU usage detected"

```

```

        description: "{{{$labels.instance}}}: CPU usage is above 80%"
- alert: NodeCPUUsageCritical
  expr: 100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: CPUExceedThresholdCritical
    text: Usage of CPU is greater than the threshold value
    id: 3001205
    source: CPRO
    host: "{{{$labels.kubernetes_io_hostname}}}"
    eventType: 11
    probableCause: 351
    key: MOC006
  annotations:
    summary: "{{{$labels.instance}}}: High CPU usage detected"
    description: "{{{$labels.instance}}}: CPU usage is above 90%"
- alert: NodeSwapUsageMinor
  expr: (100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 70) and (100 -
(node_memory_SwapFree_bytes + 1) / (node_memory_SwapTotal_bytes + 1) * 100 <=
80)
  for: 1m
  labels:
    severity: MINOR
    name: SwapExceedThresholdMinor
    text: Usage of swap space is greater than the threshold value
    id: 3001206
    source: CPRO
    host: "{{{$labels.kubernetes_io_hostname}}}"
    eventType: 11
    probableCause: 351
    key: MOC007
  annotations:
    summary: "{{{$labels.instance}}}: High swap space usage detected"
    description: "{{{$labels.instance}}}: Swap space usage is above 70%"
- alert: NodeSwapUsageMajor
  expr: (100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 80) and (100 -
(node_memory_SwapFree_bytes + 1) / (node_memory_SwapTotal_bytes + 1) * 100 <=
90)
  for: 1m
  labels:
    severity: MAJOR

```

```

name: SwapExceedThresholdMajor
text: Usage of swap space is greater than the threshold value
id: 3001207
source: CPRO
host: "{$labels.kubernetes_io_hostname}"
eventType: 11
probableCause: 351
key: MOC008
annotations:
  summary: "{$labels.instance}: High swap space usage detected"
  description: "{$labels.instance}: Swap space usage is above 80%"
- alert: NodeSwapUsageCritical
  expr: 100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: SwapExceedThresholdCritical
    text: Usage of swap space is greater than the threshold value
    id: 3001208
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC009
    annotations:
      summary: "{$labels.instance}: High swap space usage detected"
      description: "{$labels.instance}: Swap space usage is above 90%"
- alert: NodeMemoryUsageMinor
  expr: (100 - (node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes * 100 > 70) and (100 -
(node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100 <= 80)
  for: 1m
  labels:
    severity: MINOR
    name: MemoryExceedThresholdMinor
    text: Usage of memory is greater than the threshold value
    id: 3001209
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC010
    annotations:
      summary: "{$labels.instance}: High memory usage detected"

```

```

        description: "{{\$labels.instance}}: Memory usage is above 70%"
- alert: NodeMemoryUsageMajor
  expr: (100 - (node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes * 100 > 80) and (100 -
(node_memory_MemAvailable_bytes) / node_memory_MemTotal_bytes * 100 <= 90)
  for: 1m
  labels:
    severity: MAJOR
    name: MemoryExceedThresholdMajor
    text: Usage of memory is greater than the threshold value
    id: 3001210
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC011
  annotations:
    summary: "{{\$labels.instance}}: High memory usage detected"
    description: "{{\$labels.instance}}: Memory usage is above 80%"
- alert: NodeMemoryUsageCritical
  expr: 100 - (node_memory_MemAvailable_bytes) /
node_memory_MemTotal_bytes * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: MemoryExceedThresholdCritical
    text: Usage of memory is greater than the threshold value
    id: 3001211
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC012
  annotations:
    summary: "{{\$labels.instance}}: High memory usage detected"
    description: "{{\$labels.instance}}: Memory usage is above 90%"
- alert: NodeZombieCountMinor
  expr: (zombieCount > 5) and (zombieCount <= 10)
  for: 1m
  labels:
    severity: MINOR
    name: ZombieCountExceedThresholdMinor
    text: Zombie process count is greater than the threshold value
    id: 3001212
    source: CPRO

```

```

host: "{$labels.kubernetes_io_hostname}"
eventType: 11
probableCause: 351
key: MOC013
annotations:
  summary: "{$labels.instance}: Too many zombie processes detected"
  description: "{$labels.instance}: Zombie process count is above 5"
- alert: NodeZombieCountMajor
  expr: (zombieCount > 10) and (zombieCount <= 15)
  for: 1m
  labels:
    severity: MAJOR
    name: ZombieCountExceedThresholdMajor
    text: Zombie process count is greater than the threshold value
    id: 3001213
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC014
  annotations:
    summary: "{$labels.instance}: Too many zombie processes detected"
    description: "{$labels.instance}: Zombie process count is above
10"
- alert: NodeZombieCountCritical
  expr: zombieCount > 15
  for: 1m
  labels:
    severity: CRITICAL
    name: ZombieCountExceedThresholdCritical
    text: Zombie process count is greater than the threshold value
    id: 3001214
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC015
  annotations:
    summary: "{$labels.instance}: Too many zombie processes detected"
    description: "{$labels.instance}: Zombie process count is above
15"
- alert: NodeSysUpTimeCritical
  expr: node_time_seconds - node_boot_time_seconds < 5 * 24 * 3600
  for: 1m
  labels:

```

```
severity: CRITICAL
name: NodeSysUpTimeLessThresholdCritical
text: The number of seconds since last node reboot is less than the
threshold value
id: 3001215
source: CPRO
host: "{$labels.kubernetes_io_hostname}"
eventType: 11
probableCause: 351
key: MOC016
annotations:
  summary: "{$labels.instance}": Node uptime is less than threshold"
  description: "{$labels.instance}": Node uptime is less than 5 days"
- alert: NetworkRetransMajor
  expr: sum(rate(node_netstat_Tcp_RetransSegs[2m])) > 1500000
  for: 1m
  labels:
    severity: MAJOR
    name: NetworkRetransExceedThresholdMajor
    text: The number of segments re-transmitted is greater than the
threshold value
    id: 3001216
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC017
    annotations:
      summary: "{$labels.instance}": The number of re-trans segments is
more than threshold"
      description: "{$labels.instance}": The number of re-trans segments
is more than 1500000"
- alert: PrometheusTsdbWalCorruptions
  expr: rate(prometheus_tsdb_wal_corruptions_total[4m]) > 0
  labels:
    severity: MAJOR
    text: The number of TSDB corruptions is {{ $value }}
    id: 3001297
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC018
    annotations:
```

```
summary: "{$labels.instance}": The number of Prometheus TSDB WAL corruptions"
description: "{$labels.instance}": Prometheus encountered TSDB WAL corruptions"
rules: {}

prometheus.yml:
global:
## How frequently to scrape targets by default
## scrape_interval: 1m
## How long until a scrape request times out
## scrape_timeout: 10s
## How frequently to evaluate rules
## evaluation_interval: 1m
## external label to deduplicate data from replicas in a high-availability configuration
## A label(prometheus) whose value identifies the name of a high-availability cluster or group of Prometheus servers.
## prometheus_replica label is internally populated. Don't change it's value
## uncomment below 3 lines to add external label with proper value of "prometheus" label
##external_labels:
##  prometheus: cpro
##  prometheus_replica: $(HOSTNAME)

rule_files:
- /etc/config/rules
- /etc/config/alerts

scrape_configs:
- job_name: prometheus
  static_configs:
    - targets:
      - localhost:9090
  honor_labels: true

  kubernetes_sd_configs:
    - role: endpoints

  relabel_configs:
```

```

        - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_probe ]
            action: keep
            regex: prometheus
        - source_labels: [ __meta_kubernetes_pod_container_name ]
            action: drop
            regex: istio-proxy
        - source_labels: [ __meta_kubernetes_namespace ]
            action: replace
            target_label: namespace
        - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_path ]
            action: replace
            target_label: __metrics_path__

# Uncomment the below lines if etcd metrics are to be scrapped in BCMT
environment.
#- job_name: bcmt-etcd
#  kubernetes_sd_configs:
#    - role: node
#  metrics_path: /metrics
#  scheme: https
#  tls_config:
#    ca_file: /etc/etcd/ssl/ca.crt
#    cert_file: /etc/etcd/ssl/tls.crt
#    key_file: /etc/etcd/ssl/tls.key
#    insecure_skip_verify: true
#  relabel_configs:
#    - action: keep
#      regex: true
#      source_labels:
#        - __meta_kubernetes_node_label_is_control
#    - action: replace
#      regex: (.*)
#      replacement: $1:2379
#      source_labels:
#        - __meta_kubernetes_node_address_InternalIP
#        target_label: __address__
#    - action: replace
#      regex: (.*)
#      replacement: $1
#      source_labels:
#        - job
#        target_label: component

```

```
- job_name: 'kubernetes-apiservers'

  kubernetes_sd_configs:
    - role: endpoints

      # Default to scraping over https. If required, just disable this or
      change to
      # `http`.
      scheme: https

      # This TLS & bearer token file config is used to connect to the actual
      scrape
      # endpoints for cluster components. This is separate to discovery auth
      # configuration because discovery & scraping are two separate concerns
      in
      # Prometheus. The discovery auth config is automatic if Prometheus
      runs inside
      # the cluster. Otherwise, more config options have to be provided
      within the
      # <kubernetes_sd_config>.
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        # If your node certificates are self-signed or use a different CA to
        the
        # master CA, then disable certificate verification below. Note that
        # certificate verification is an integral part of a secure
        infrastructure
        # so this should only be disabled in a controlled environment. You
        can
        # disable certificate verification by uncommenting the line below.
        #
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token

      # Keep only the default/kubernetes service endpoints for the https
      port. This
      # will add targets for each API server which Kubernetes adds an
      endpoint to
      # the default/kubernetes service.
      relabel_configs:
        - source_labels: [__meta_kubernetes_namespace,
        __meta_kubernetes_service_name, __meta_kubernetes_endpoint_port_name]
          action: keep
          regex: default;kubernetes;https
        - source_labels: [__meta_kubernetes_namespace]
```

```

    action: replace
    target_label: namespace

- job_name: 'kubernetes-nodes'

    # Default to scraping over https. If required, just disable this or
    change to
    # `http`.
    scheme: https

    # This TLS & bearer token file config is used to connect to the actual
    scrape
    # endpoints for cluster components. This is separate to discovery auth
    # configuration because discovery & scraping are two separate concerns
    in
    # Prometheus. The discovery auth config is automatic if Prometheus
    runs inside
    # the cluster. Otherwise, more config options have to be provided
    within the
    # <kubernetes_sd_config>.
    tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        # If your node certificates are self-signed or use a different CA to
        the
        # master CA, then disable certificate verification below. Note that
        # certificate verification is an integral part of a secure
        infrastructure
        # so this should only be disabled in a controlled environment. You
        can
        # disable certificate verification by uncommenting the line below.
        #
        insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token

    kubernetes_sd_configs:
        - role: node

    relabel_configs:
        - action: labelmap
            regex: __meta_kubernetes_node_label_(.+)
        - target_label: __address__
            replacement: kubernetes.default.svc:443
        - source_labels: [__meta_kubernetes_node_name]
            regex: (.+)
            target_label: __metrics_path__

```

```
replacement: /api/v1/nodes/${1}/proxy/metrics
- source_labels: [__meta_kubernetes_namespace]
  action: replace
  target_label: namespace

- job_name: 'kubernetes-nodes-cadvisor'

  # Default to scraping over https. If required, just disable this or
  change to
  # `http`.
  scheme: https

  # This TLS & bearer token file config is used to connect to the actual
  scrape
  # endpoints for cluster components. This is separate to discovery auth
  # configuration because discovery & scraping are two separate concerns
  in
  # Prometheus. The discovery auth config is automatic if Prometheus
  runs inside
  # the cluster. Otherwise, more config options have to be provided
  within the
  # <kubernetes_sd_config>.
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    # If your node certificates are self-signed or use a different CA to
    the
    # master CA, then disable certificate verification below. Note that
    # certificate verification is an integral part of a secure
    infrastructure
    # so this should only be disabled in a controlled environment. You
    can
    # disable certificate verification by uncommenting the line below.
    #
    insecure_skip_verify: true
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token

  kubernetes_sd_configs:
    - role: node

  # This configuration will work only on kubelet 1.7.3+
  # As the scrape endpoints for cAdvisor have changed
  # if you are using older version you need to change the replacement to
  # replacement: /api/v1/nodes/${1}:4194/proxy/metrics
```

```

# more info here https://github.com/coreos/prometheus-operator/
issues/633

    relabel_configs:
        - action: labelmap
            regex: __meta_kubernetes_node_label_(.+)
        - target_label: __address__
            replacement: kubernetes.default.svc:443
        - source_labels: [__meta_kubernetes_node_name]
            regex: (.+)
            target_label: __metrics_path__
            replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor
        - source_labels: [__meta_kubernetes_namespace]
            action: replace
            target_label: namespace

    # Scrape config for service endpoints.
    #
    # The relabeling allows the actual service scrape endpoint to be
    # configured
    # via the following annotations:
    #
    # * `prometheus.io/scrape`: Only scrape services that have a value of
    # `true`
    # * `prometheus.io/scheme`: If the metrics endpoint is secured then you
    # will need
    #     to set this to `https` & most likely set the `tls_config` of the
    #     scrape config.
    # * `prometheus.io/path`: If the metrics path is not `/metrics` override
    #     this.
    # * `prometheus.io/port`: If the metrics are exposed on a different port
    #     to the
    #     service then set this appropriately.
    - job_name: 'kubernetes-service-endpoints-insecure'

        kubernetes_sd_configs:
            - role: endpoints

    relabel_configs:
        - source_labels: [__meta_kubernetes_pod_container_name]
            action: drop
            regex: istio-proxy
        - source_labels:
            [__meta_kubernetes_service_annotation_prometheus_io_scrape]
            action: keep
            regex: true

```

```

      - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_scheme ]
        action: drop
        target_label: __scheme__
        regex: https
      - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_path ]
        action: replace
        target_label: __metrics_path__
        regex: (.+)
      - source_labels: [ __address__,
[ __meta_kubernetes_service_annotation_prometheus_io_port ]
        action: replace
        target_label: __address__
        regex: ([^:]+)(?::\d+)?;(\d+)
        replacement: $1:$2
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
      - source_labels: [ __meta_kubernetes_namespace ]
        action: replace
        target_label: namespace
      - source_labels: [ __meta_kubernetes_service_name ]
        action: replace
        target_label: kubernetes_name
      - source_labels: [ __meta_kubernetes_pod_node_name ]
        target_label: kubernetes_io_hostname
        action: replace

      - job_name: 'prometheus-pushgateway'
        honor_labels: true
        scheme: http
        kubernetes_sd_configs:
          - role: service

        relabel_configs:
          - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_probe ]
            action: keep
            regex: pushgateway
          - source_labels: [ __meta_kubernetes_namespace ]
            action: replace
            target_label: namespace
          - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_path ]
            action: replace

```

```

    target_label: __metrics_path__

  - job_name: 'kubernetes-pods-insecure'
    kubernetes_sd_configs:
      - role: pod
        relabel_configs: # If first two labels are present, pod should be
          scraped by the istio-secure job.
        - source_labels:
          [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
            action: keep
            regex: true
        - source_labels:
          [__meta_kubernetes_pod_annotation_sidecar_istio_io_status,
           __meta_kubernetes_pod_annotation_istio_mtls]
            action: drop
            regex: (([^;]+);([^;]*))|(([^\;]*);(true))
        - source_labels:
          [__meta_kubernetes_pod_annotation_prometheus_io_scheme]
            action: drop
            regex: https
        - source_labels:
          [__meta_kubernetes_pod_annotation_prometheus_io_path]
            action: replace
            target_label: __metrics_path__
            regex: (.)
            - source_labels: [__address__,
              __meta_kubernetes_pod_annotation_prometheus_io_port]
                action: replace
                regex: ([^:]+)(?::\d+)?;(\d+)
                replacement: $1:$2
                target_label: __address__
            - action: labelmap
              regex: __meta_kubernetes_pod_label_(.+)
            - source_labels: [__meta_kubernetes_namespace]
              action: replace
              target_label: namespace
            - source_labels: [__meta_kubernetes_pod_name]
              action: replace
              target_label: kubernetes_pod_name
            - source_labels: [__meta_kubernetes_pod_node_name]
              action: replace
              target_label: instance

# Example scrape config for probing services via the Blackbox Exporter.
#

```

```
# The relabeling allows the actual service scrape endpoint to be
configured
# via the following annotations:
#
# * `prometheus.io/probe`: Only probe services that have a value of
`true`
#
#
- job_name: 'prometheus-nodeexporter'
  honor_labels: true

kubernetes_sd_configs:
  - role: endpoints

  relabel_configs:
    - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_probe]
      action: keep
      regex: node-exporter
    - source_labels: [__meta_kubernetes_namespace]
      action: replace
      target_label: namespace
    - source_labels: [__meta_kubernetes_pod_node_name]
      target_label: kubernetes_io_hostname
      action: replace

    - job_name: 'kubernetes-services'

      metrics_path: /probe
      params:
        module: [http_2xx]

      kubernetes_sd_configs:
        - role: service

        relabel_configs:
          - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_probe]
            action: keep
            regex: true
          - source_labels: [__address__]
            target_label: __param_target
          - target_label: __address__
            replacement: blackbox
          - source_labels: [__param_target]
```

```

        target_label: instance
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
      - source_labels: [__meta_kubernetes_namespace]
        target_label: namespace
      - source_labels: [__meta_kubernetes_service_name]
        target_label: kubernetes_name

# Example scrape config for pods
#
# The relabeling allows the actual pod scrape endpoint to be configured
via the
# following annotations:
#
# * `prometheus.io/scrape`: Only scrape pods that have a value of `true`
# * `prometheus.io/path`: If the metrics path is not `/metrics` override
this.
# * `prometheus.io/port`: Scrape the pod on the indicated port instead
of the default of `9102`.
# Mixer scrapping. Defaults to Prometheus and mixer on same namespace.
- job_name: 'istio-mesh'
  kubernetes_sd_configs:
    - role: endpoints
      namespaces:
        names:
          - istio-system
    relabel_configs:
      - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
        action: keep
        regex: istio-telemetry;prometheus

# Scrape config for envoy stats
- job_name: 'envoy-stats'
  metrics_path: /stats/prometheus
  kubernetes_sd_configs:
    - role: pod

    relabel_configs:
      - source_labels: [__meta_kubernetes_pod_container_port_name]
        action: keep
        regex: '.*-envoy-prom'
      - source_labels: [__address__,
__meta_kubernetes_pod_annotation_prometheus_io_port]
        action: replace

```

```
        regex: ([^:])(?:::\d+)?;(\d+)
        replacement: $1:15090
        target_label: __address__
    - action: labeldrop
        regex: __meta_kubernetes_pod_label_(.+)
    - source_labels: [__meta_kubernetes_namespace]
        action: replace
        target_label: namespace
    - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: pod_name

    - job_name: 'istio-policy'
        kubernetes_sd_configs:
        - role: endpoints
            namespaces:
                names:
                - istio-system

        relabel_configs:
        - source_labels: [__meta_kubernetes_service_name,
        __meta_kubernetes_endpoint_port_name]
            action: keep
            regex: istio-policy;http-policy-monitoring

    - job_name: 'istio-telemetry'
        kubernetes_sd_configs:
        - role: endpoints
            namespaces:
                names:
                - istio-system

        relabel_configs:
        - source_labels: [__meta_kubernetes_service_name,
        __meta_kubernetes_endpoint_port_name]
            action: keep
            regex: istio-telemetry;http-monitoring

    - job_name: 'pilot'
        kubernetes_sd_configs:
        - role: endpoints
            namespaces:
                names:
                - istio-system
```

```
relabel_configs:
- source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
  action: keep
  regex: istiod;http-monitoring
- source_labels: [__meta_kubernetes_service_label_app]
  target_label: app

- job_name: 'galley'
  kubernetes_sd_configs:
  - role: endpoints
    namespaces:
      names:
      - istio-system

  relabel_configs:
  - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
    action: keep
    regex: istio-galley;http-monitoring

- job_name: 'citadel'
  kubernetes_sd_configs:
  - role: endpoints
    namespaces:
      names:
      - istio-system

  relabel_configs:
  - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
    action: keep
    regex: istio-citadel;http-monitoring

- job_name: 'sidecar-injector'

  kubernetes_sd_configs:
  - role: endpoints
    namespaces:
      names:
      - istio-system

  relabel_configs:
```

```

        - source_labels: [__meta_kubernetes_service_name,
__meta_kubernetes_endpoint_port_name]
          action: keep
          regex: istio-sidecar-injector;http-monitoring

serverFilesForComPaaS:
  alerts: {}
  rules: {}

prometheus.yml:
  global:
    scrape_interval: 1m
    scrape_timeout: 10s
    evaluation_interval: 1m
    ## external label to deduplicate data from replicas in a high-
    availability configuration
    ## A label(prometheus) whose value identifies the name of a high-
    availability cluster or group of Prometheus servers.
    ## prometheus_replica label is internally populated. Don't change it's
    value
    ## uncomment below 3 lines to add external label with proper value of
    "prometheus" label
    ##external_labels:
    ##  prometheus: cpro
    ##  prometheus_replica: $(HOSTNAME)

  rule_files:
    - /etc/config/rules
    - /etc/config/alerts

  scrape_configs:
    - job_name: prometheus
      static_configs:
        - targets:
          - localhost:9090

## Define custom scrape job here for Prometheus.
## These jobs will be appended to prometheus.yml
customScrapeJobs: []
#   - job_name: cnot
#     metrics_path: /api/cnot/v1/metrics
#     scheme: http
#     static_configs:
#       - targets:

```

```

#           - cnot.default.svc.cluster.local
#           - job_name: grafana
#             scheme: https
#             tls_config:
#               insecure_skip_verify: true
#             static_configs:
#               - targets:
#                 - grafana-cpro-grafana.default.svc.cluster.local:80

networkPolicy:
  ## Enable creation of NetworkPolicy resources.
  ##
  enabled: false

restserver:
  enabled: false

  name: restserver

  podAnnotations: {}

BCMT:
  serverURL: https://k8s-apiserver.bcmt.cluster.local:8443

  ## soft means preferredDuringSchedulingIgnoredDuringExecution
  ## hard means requiredDuringSchedulingIgnoredDuringExecution
  ## you could refer k8s api for more detail
  #
  antiAffinityMode: "soft"

image:
  imageRepo: cpro/registry4/prometheus-restapi
  imageTag: 3.5.7
  imagePullPolicy: IfNotPresent

replicaCount: 1

service:
  # Options: ClusterIP or NodePort
  type: ClusterIP
  servicePort: 8888
  # Note the range of a valid nodePort is 30000-32767.
  nodePort: 32766

ingress:

```

```
enabled: false
annotations: {}
# kubernetes.io/ingress.class: nginx
# kubernetes.io/tls-acme: "true"
tls: []
# - secretName: chart-example-tls
#   hosts:
#     - chart-example.local
## Whether use istio ingress gateway(Envoy)
istioIngress:
  enabled: true
  selector: {istio: ingressgateway}
  Contextroot: restserver
  # the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
  ## Keep gatewayName to empty to create kubenetes gateway resource. If new
  virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
  tls:
    enabled: false
    httpsPort: 443
    ## mode could be SIMPLE, MUTUAL, PASSTHROUGH, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "restserver-gateway"
  resources:
    # We usually recommend not to specify default resources and to leave this as
    # a conscious
    # choice for the user. This also increases chances charts run on
    environments with little
    # resources, such as Minikube. If you do want to specify resources,
    uncomment the following
    # lines, adjust them as necessary, and remove the curly braces after
'resources':'.
  limits:
    cpu: 1
    memory: 512Mi
  requests:
    cpu: 100m
    memory: 128Mi

  nodeSelector: {}

  tolerations: []
    # - key: "key"
```

```

#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

configs:
  ncmsUsername: user-input
  ncmsPassword: user-input
  ncmsPassPhrase: user-input
  httpsEnabled: false
  restCACert: |+
    -----BEGIN CERTIFICATE-----
    MIIEETCAvmgAwIBAgIJIAIQFm91ZPjOsMA0GCSqGSib3DQEBCwUAMIGeMQswCQYD
    VQQGEwJDTjEQMA4GA1UECAwHQMvamluZzERMA8GA1UEBwwIV2FuZ2ppbmcxDjAM
    BgNVBAoMBU5va21hMQwwCgYDVQQLANDU0YxHjAcBgNVBAMMFW1pYW9mZW5rLWNh
    Lm5va21hLmNvbTEsMCoGCSqGSib3DQEJARYdbWhbZlrbmcua2FuZ0Bub2tpYS1z
    YmVsbC5jb20wHhcNMTgwNTIxMDIwNjIyWhcNMjgwNTE4MDIwNjIyWjCBnjELMAkG
    A1UEBhMCQ04xEADAOBgNVBAgMB0JlaWppbmcxETAPBgNVBAcMCFdhbmdqaW5nMQ4w
    DAYDVQQKDAVOb2tpYTEMMAoGA1UECwwDQ1NGMR4wHAYDVQQDBVtaWFvZmVuay1j
    YS5ub2tpYS5jb20xLDAqBgkqhkiG9w0BCQEWWh1pYW9mZW5nLmthbmdAbm9raWEt
    c2J1bGwuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA6NyT+S/i
    rFotAivpTUb2WjHVvvREgiC5f4pSqwHk/sVYv1+sTaaHPsd/wWa9yc1C9nQ3iHvX
    9PYo8dRKYsPKxoFcS18C1DLoVYCUDrpVS2HbpBxXlUrBow46LMFDAUQAH8e4fi4K
    +1/H5kxvnQRXPkKnx2uc13vGW5mRXTGfLGLHUwoVdighnRNAhqUUZqwafveBMXwf
    6mBTdOi6JUg+Hz6UrIsZlTciYI7t1weFm944YSKeF7L8LcqqnZxBfvbttyw8XmWd
    M0XbT8SJ3G1om4y8nbFyMHa4Fm8eN0TyxM71Ua2r1GAeda2rMNf+RSOreryEXbYY
    pNIzbUPyFFKkYQIDAQAB01AwTjAdBgNVHQ4EFgQUne4mdychSlHrO2W/uP+25A8dU
    x7wwHwYDVR0jBBgwFoAUu4mdychSlHrO2W/uP+25A8dUx7wwDAYDVR0TBAUwAwEB
    /zANBgkqhkiG9w0BAQsFAAOCAQEA4iMY4kEKlcOmihQnEn6a3Jwn+7ajEJ5MQOWw
    UONUH6T/EzUWJL74g2uoXFz2ykqv/eSSWjk6T0F+hJVKh2yXMeGS6+WvJlrakPNF
    sylpjRR7rYqFXLqtWpLGoXE/mTHj0PeakvCN9Fo+kJ8PmkdYuIcVxdM11Lv0judJ
    yrNL/3v2MLkEKF9xmtbjeyIYoRV1kf2ehdMVvWrpeWbY1PSNE2BSM60qFb5pCd9G
    O1yQ15b4smvgx1kcIerZ6abMsLJUF9Z+8crHKja1/4e9/xoP5n/CSjzPNWnaGQM1
    r/IGGTDpw6eym9yPH0nbv76/Mse3xqGGSZJA0eS1T6B2aXtUSg==
    -----END CERTIFICATE-----
  restServerKey: |+
    -----BEGIN PRIVATE KEY-----
    MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDICfMXbmZrWcOj
    081L1wHhTJgZ6ZdkUXAj/Abkkk3PLwPoZ9KxYwKhZ54Zy1N6owy+raYa0bGZWNzf
    b4P1cFMoGX7+vJYGOmkorF9AYBCh6jbeVoHfEYw1ozejFRo64Eud2g3MKBFLwcVB
    Za+bebm7QXLn0j3LURzN2Vembj7FZETg/weTGZC2Nw2RK+vACEFIeXkeDx2js1dQ
    hCquOAYvRV0emAZIm7uf800r6LoBojkBkTUXGLWDbvSkzxUPGXgbZT15ZpeS+jpc
    JH/gdjHVqoa+XDTzT8xZOrTXJp9HJUGg/Jy6P6MFRHJtkaFQ7Ci8j51teuBh2xFf
    /fxhdPF7AgMBAAECggEAXdmkhRsGz0qXJHS90b2YvsFbEf7iCHFs/Rw6qfiqf9A2
    1zFNYArIp4PZbaBatLf09q5dcH8wFWmX7dVLxrZR6RuO73yjDcV5iTaz3nNcNkNw
    b0e3xRb1PAPwv+XYgyqTnRQEk8tK1dqjHgybXUNwJ/Hr+AjjL/gJcYRK2r0RnN+S
  
```

```

FCPX4dpjNet7O+vSZ4nK+51qMJ20EvLdzhvIJkmKbjRr0/S6b+Ze1BKzo9DGfkvg
6B207As1ReHX5ahqSX1GPRGldcvW3VI4okHxtbIL30Y59c2+Prqjtg79fk/C100M
Te4juYm16wSW1RHMwtD8BweMCN/0A3nLvKhOekV0EQKBgQDyoKMXHJu2WrMsTtjY
1lwuvFeD/gCaHnS9W9m61TkvcnjZkwTXN16k16EmRgtbNaqDojnTU18LbOOR206
8w7ai79diL0G5bCst+uXpjIzld4tnf6n8TZYSuiG8EI0u4CpmUkRXU7s0X8cu0q0
pw22stvsVzhUvWRHG0H2QZtygwKBgQDTfiN1RmjafyqrLWzch0AUEhtijtfndBuE
NMaM1IBSUQnDJoJf0rMX+UEttj9VohRmRM8p8aAMEN6IjLh7Y9oYjoXGyYxzk2Pz
TacsueV477KkVBoS0snEAIjcrGxsCp2eCX/s1+4fJAYPS0WNDEDRMjoE7dBf6f6t
LnvXYgzzqQKBgAGDoSDuy8X6k02w3EeVwKOGB2HKfwR3NjFMVnKFDCNQ3Jqvs7/X
L4apTsizD+SQrlJHXvFamSYyPtGffm7XP3t7rckOpmdZnZ2mVDERF3Uc9VMBjmpL
5hPtoefdrfwYQ3hLfZo/I9P0hr+OJ6v2PO6r9RVngFF9cRfEgsffpvGzAoGAJw6p
b7QEEy3e7GPkMbaXt90sL4RfvP/FQSIZ9NIdrJYIroCDHT01E+1VKyL4CVF4YPae
J4nW28OVxTPvseHb2iMf83kvNfznPXx+vhTKmw3xOMXLVuSuNfzY6Z/yGfXP6+qn
NE8gS6H0eIiXHJhBtCCJdHWSwNPO0569Aia6a5kCgYEAlB0FdO4Ok8z3ut39aZpw
S1JooCCwdx5udyXtQvNPrmyMRRpTqFezYCYIGYVquR5KyZjISQJswCUzCMbJc3hi
nqY/F+5nre9HURT1py66VDxX6rOXldiR1CbDS6VacV0JnfYX6FwtRjGosgdhv6nV
NDZmqe58yC7pYp77yFHJOPM=
-----END PRIVATE KEY-----
restServerCert: |
  -----BEGIN CERTIFICATE-----
MIIDtDCCApwCCQDIFRUb+zmnGDANBgkqhkiG9w0BAQUFADCnjenELMAkGA1UEBhMC
Q04xE DAOBgNVBAgMB0JlaWppbmcxETAPBgNVBAcMCFdhbmdqaW5nMQ4wDAYDVQQK
DAVOb2tpYTEMMAoGA1UECwwDQ1NGMR4wHAYDVQQDBVtaWFvZmVuay1jYS5ub2tp
YS5jb20xLDAqBgkqhkiG9w0BCQEWHW1pYW9mZW5nLmthbmdAbm9raWEtc2JlbGwu
Y29tMB4XDTE4MDUyMTAyMDYyM1oXDTE5MDUyMTAyMDYyMlowgZgxCzAJBgNVBAYT
AkNOMRAwDgYDVQQIDAxCZWLqaW5nMREwDwYDVQQHDAhXYW5namluZzEOMAwGA1UE
CgwFTm9raWExDAAKBgNVBAsMA0NTRjEYMBYGA1UEAwPbWlhzb2lbumsuc2VydmlVY
MSwwKgYJKoZIhvcNAQkBFh1taWFvZmVuZy5rYW5nQG5va2lhLXNiZWxsLmNvbTCC
ASIwDQYJKoZIhvcNAQEBQADggEPADCCAQoCggEBAMhx8xduZmtZw6PTzUvXAeFM
mBnp12RRcCP8BuSSTc8vA+hn0rFjAqFnnhnLU3qjDL6tphrRsZ1Y119vg/VwUygZ
fv681gY6aSisX0BgEKHqNt5Wgd8RjDWjN6MVGjrgS53aDcwoEUvBxF1r5t5ubtB
cufSPctRHm3ZV6ZuPsVkrOD/B5MzkLY3DZEer68AIQuh5eR4PHaNKV1CEKq44Bi9F
XR6YBkibu5/zQ6vouge6MoGRNTEYtYNu9KTPFQ8ZeBt1Oxlml5L6Olwkf+B2MdWq
hr5cNPNPzFk6tNcmn0clQaD8nLo/owVEcm2RoVdsKLynPnW164GHbEV/9/GF08XsC
AwEAATANBgkqhkiG9w0BAQUFAAOCAQEAKNOkBdLhTwxDGDE8941CIiLHZ7pyYCP7
h+Ov0zfLRzEwbIQwcrkAfQYy007xWCmNfcHaSdTayaHV/i3QktSdQFwcI3i8+l
i/XkmGX+IvKG0KiQ/JwClkjO6B/DUKqU6fyg51HWPuYovLdNo0kKEVe30Fb9wNr+
8dMHAW/drTbxngUtNArPM8xsas5gQId887bHRzXJS7Pd9tojWMeyny8A0bXrshrdF
812eZNla7nBiQTipb9nJwjzW31B6V5jp95R4UnlvIgdhj2/UxEyrb/6yj/lfy05H
h8BzBBalgf2pvJfxh9ids4cHC0Bqc+Etqble6tGNNuUTmFzfx2+rgg==
-----END CERTIFICATE-----
# Log Level: DEBUG < INFO < WARN < ERROR < FATAL < OFF
loglevel: INFO

```

**Note:**

CPRO chart Istio Version parameter default value is 1.4. Thus, the user needs to update the parameter value with NCS support Istio Version before installation:

1.5 for BCMT versions < NCS 20 FP1

1.6 in NCS 20 FP1

1.7 in NCS 20 FP2

24 Appendix: Example for grafana-values.yaml files



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```
rbac:  
  enabled: true  
  pspUseAppArmor: false  
serviceAccountName:  
  
#replicas: 2  
  
deployOnCompass: false  
  
deploymentStrategy: Recreate  
  
=: &registry: "csf-docker-delivered.[...]" in the appropriate repository.  
  
global:  
  registry: *registry  
# registry2 repo is used in cmdb chart  
  registry2: "csf-docker-delivered.[...]" in the appropriate repository.  
# registry3 repo is used to pull grafana-tenant image  
  registry3: "csf-docker-delivered.[...]" in the appropriate repository.  
# registry4 repo is used to pull download-dashboards image  
  registry4: "csf-docker-delivered.[...]" in the appropriate repository.  
# registry5 repo is used to pull sane image  
  registry5: in the appropriate repository.  
# registry1 repo is used to pull grafana-plugins image  
  registry1: "csf-docker-delivered.[...]" in the appropriate repository.  
  
  annotations: {}  
  labels: {}  
# Define serviceAccount name for grafana at global level.  
## serviceAccount priority order is  
## 1. serviceAccountName  
## 2. global.serviceAccountName  
## 3. If we are not using customized resources set rbac.enabled to true then  
  resources will be created on helm install  
## 4. If both serviceAccounts are not set and rbac.enabled is set to false  
  then default serviceAccount will be used  
##  
  serviceAccountName:
```

```
## istio verion in X.Y format eg 1.4/1.5/1.6
istioVersion: 1.4
podNamePrefix: ""
containerNamePrefix: ""

custom:
  psp:
    annotations:
      seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
      seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
      apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
      apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    labels: {}

pod:
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
  apparmorAnnotations:
    apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
    apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
  labels: {}

name: grafana
helm3: true
appTitle: "Performance Monitoring"

HA:
  enabled: false

## Whether the chart will deploy on istio
istio:
  enable: false
  mtls_enable: true
  cni_enable: true
  createKeycloakServiceEntry:
    enabled: false
    extCkeyHostname: ""      # Ex. extCkeyHostname: "ckey.io"
    # Port on which ckey is externally accessible
    extCkeyPort: ""         # Ex. extCkeyPort: 31390
```

```

# Protocol on which ckey is externally accessible
extCkeyProtocol: ""          # accepted values: HTTP, HTTPS
## FQDN of ckey k8s service name internally accessible within k8s cluster
ckeyK8sSvcName: ""          # Ex. keycloak-ckey.default.svc.cluster.local
# Port on which ckey k8s service is accessible
ckeyK8sSvcPort: ""          # Ex. ckeyK8sSvcPort: 8443
hostAlias: ""                # If the host name of ckey is not resolvable then edge
node ip has to be given here
location: "MESH_INTERNAL"    # Location specifies whether the service
is part of Istio mesh or outside the mesh Ex. MESH_EXTERNAL/MESH_INTERNAL

image:
  imageRepo: cpro/grafana-registry1/grafana-tenant
  imageTag: 7.1.3-1.3.1
  imagePullPolicy: IfNotPresent

distroImage:
  imageRepo: cpro/grafana-registry1/grafana-tenant-distro
  imageTag: 7.1.3-1.3.1
  imagePullPolicy: IfNotPresent

runAsUser: 65534
fsGroup: 65534
supplementalGroups: [65534]
seLinuxOptions:
  enabled: false
  level: ""
  role: ""
  type: ""
  user: ""

helmDeleteImage:
  imageRepo: tools/kubectl
  imageTag: v1.19.5-nano-20201210
  imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 100m
      memory: 100Mi
    requests:
      cpu: 50m
      memory: 32Mi

  need_dbupdate: false
  sqllitetomdb: false

```

```
## Optionally specify an array of imagePullSecrets.  
## Secrets must be manually created in the namespace.  
## ref: https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-  
private-registry/  
##  
# pullSecrets:  
#   - myRegistrKeySecretName  
  
hookImage:  
  imageRepo: cpro/grafana-registry1/grafana-lcm-hook  
  imageTag: "1.10.0"  
  imagePullPolicy: IfNotPresent  
  resources:  
    limits:  
      cpu: 500m  
      memory: 1Gi  
    requests:  
      cpu: 100m  
      memory: 128Mi  
  
mdbToolImage:  
  imageRepo: cpro/grafana-registry1/grafana-mdb-tool  
  imageTag: "3.11.0"  
  imagePullPolicy: IfNotPresent  
  resources:  
    limits:  
      cpu: 500m  
      memory: 1Gi  
    requests:  
      cpu: 100m  
      memory: 128Mi  
  
grafanaFileMerge:  
  name: file-merge  
  imageRepo: cpro/registry4/cproconfigmap-reload  
  imageTag: v0.0.7  
  imagePullPolicy: IfNotPresent  
  resources:  
    limits:  
      cpu: 100m  
      memory: 200Mi  
    requests:  
      cpu: 10m  
      memory: 32Mi
```

```
## Grafana Util is Debug container image for Grafana
cproUtil:
  name: util
  imageRepo: cpro/registry4/cpro-util
  imageTag: v0.0.5
  imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
    requests:
      cpu: 10m
      memory: 32Mi

pluginsSideCar:
  ## If true, will install Pie chart and Bar chart and Alertmanager datasource
  plugins
  enabled: true
  imageRepo: cpro/grafana-registry1/grafana-plugins
  imageTag: "2.4.1"
  imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 128Mi

## Currently, sane integration should be enabled only by Network Operations
## Master
sane:
  enabled: false
  port: ""
  servicePort: ""
  env:
    - name: PORT
      value: ""
    - name: CSANE_SSO_PROXY_URL
      value: ""
  imageRepo: neo-docker-release/grafana-sane
  imageTag: "0.0.11"
  imagePullPolicy: IfNotPresent
```

```
ingress:
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
  labels: {}
  path: /
  hosts:
    - ""
  tls: []
#   - secretName: grafana-sane-server-tls
#     hosts:
#       - chart-example.local
resources:
  limits:
    cpu: 300m
    memory: 512Mi
  requests:
    cpu: 100m
    memory: 64Mi

cbur:
  enabled: true
## bkup are configurations about helm cbur functions that grafana provides.
## Please keep the values unchanged if you do not know exactly what is
needed.
  image:
    imageRepo: cbur/cbura
    imageTag: 1.0.3-1665
    imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 500m
      memory: 500Mi
    requests:
      cpu: 100m
      memory: 128Mi
## not other mode allowed/supported now
  backendMode: "local"
##autoEnableCron = true indicates that the cron job is immediately scheduled
when the BrPolicy is created,
##while autoEnableCron = false indicates that scheduling of the cron job
should be done on a subsequent backup request.
##This option only works when k8swatcher.enabled is true
  autoEnableCron: false
```

```
##Indicate if subsequent update of cronjob will be done via brpoilicy
update.
##true means cronjob must be updated via brpolicy update,
##false means cronjob must be updated via manual "helm backup -t app -a
enable/disable" command.
autoUpdateCron: false
## cronjob frequency, here means very 5 minutes of every day
cronJob: "*/5 * * * *"
## the maximum copy you want to saved.
maxCopy: 5
## data Encryption needs to be done on backed up data or not.
dataEncryptionEnable: true
## CBUR will get the passphrase from the secret and use it to encrypt/
decrypt the tarballs during backup/restore.
## Refer to CBUR user guide for format of secret
dataEncryptionSecretName: ""

### custom name to override default pod and container name
## pod and container name should be unique
customResourceNames:
resourceNameLimit: 63
grafanaPod:
  inCntChangeDbSchema: ""
  inCntChangeMariadbSchema: ""
  inCntWaitforMariadb: ""
  inCntGrafanaFileMerge: ""
  inCntDownloadDashboard: ""
  pluginSidecarContainer: ""
  grafanaSidecarDashboard: ""
  grafanaSaneAuthproxy: ""
  grafanaMdbtool: ""
  grafanaDatasource: ""
  grafanaContainer: ""
  cproUtil: ""
deleteDatasourceJobPod:
  name: ""
  deleteDatasourceContainer: ""
setDatasourceJobPod:
  name: ""
  setDatasourceContainer: ""
postUpgradeJobPod:
  name: ""
  postUpgradeJobContainer: ""
postDeleteJobPod:
  name: ""
```

```
deletedbContainer: ""
deletesecretsContainer: ""

importDashboardJobPod:
  name: ""
  importDashboardJobContainer: ""

downloadDashboardsImage:
  enabled: false
  imageRepo: cpro/grafana-registry1/grafana-curl
  tag: "1.18.0"
  pullPolicy: IfNotPresent

## Pod Annotations
podAnnotations: {}

## Deployment annotations
# annotations: {}

## Expose the grafana service to be accessed from outside the cluster
## (LoadBalancer service).
## or access it from within the cluster (ClusterIP service). Set the service
## type and the port to serve it.
## ref: http://kubernetes.io/docs/user-guide/services/
## 

#not supported for sensitivedata. to be uncommented when sensitivedata not
#enabled
# Database IP address (the IP address should be filed, if need_deployed:
#   false; it should be ignored, if need_deployed: true)
#dbIP: grafanadb-cmdb-mysql
# Database information (all items below are mandatory)
#dbName: grafana
#dbUser: grafana
#dbPassword: grafana

##sensitive details in Grafana config file
# sensitive details like username, password , token will be given in the
# secret format
# please refer to the confluence page for details on secret format and
# specific name for the key.

sensitiveDataSecretName: ""

#-----
# CMDB values
#-----
```

```
cmdb:
  enabled: false
  # If cmdb.need_deployed is True, mariadb will be deployed with grafana
  deployment
  need_deployed: false
  # If retain_data is true, will retain grafana data in mariadb when deleting
  grafana instance
  retain_data: false

  rbac_enabled: true

  ## Cluster Type is one of master-slave, master-master, galera, standalone
  cluster_type: "simplex"
  cluster_name: "my-cluster"

  istio:
    enabled: false
# CMDB TLS cert and key

  cacert:
"LS0tLS1CRUDJTIBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lKQU1ycEhLekRmZWdPTUEwR0N
  clientcert:
"LS0tLS1CRUDJTIBDRVJUSUZJQ0FURS0tLS0tCk1JSUNwVENDQVkwQ0FRRXdEUVlKS29aSWh2Y05BUUVMQ1FBd0d
  clientkey:
"LS0tLS1CRUDJTIBSU0EgUFJJVkJURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBUVBewVEchjsVXVjUFZINzZKbhJ

mariadb:
  # if cmdb.enabled, make sure that MYSQL_DATABASE is the same as dbName
  above!
  #r00tr00t
  root_password:

  ## If root user should be allowed from all hosts
  allow_root_all: false

  ## The number of MariaDB pods to create
  count: 1

  ## If automatic rollback of the database should be performed on pod
  restarts
  auto_rollback: true

  ## Use TLS for data in flight to/from client
  ## If using certificates, populate the certificate names in the certificates
```

```
## section in the secret that is created and specify the secret resource
## name in the certificates.secret name. Note that the secret resource
## containing all the certificate files must be pre-populated before
## CMDB deployment.
use_tls: false
certificates:
  ca_cert:      ca-cert.pem
  ca_key:       ca-key.pem
  client_cert:  client-cert.pem
  client_key:   client-key.pem
  client_req:   client-req.pem
  server_cert:  server-cert.pem
  server_key:   server-key.pem
  server_req:   server-req.pem
secret:

# if cmdb.need_deployed, make sure that MYSQL_DATABASE is the same as
dbName above!
databases:
  - name: grafana
    character_set: utf8
    collate: utf8_general_ci

users:
  - name: grafana
    # base64 encoded:grafana
    password: Z3JhZmFuYQ==
    host: "%"
    privilege: ALL
    object: "grafana.*"
    # if use_tls set, require SSL/X509 or not
    requires: ""

persistence:
  enabled: true
  accessMode: ReadWriteOnce
  size: 20Gi
  storageClass: ""
  resourcePolicy: delete
  preserve_pvc: false
  backup:
    enabled: true
    storageClass: ""
    accessMode: ReadWriteOnce
    size: 20Gi
```

```

resourcePolicy: delete
preserve_pvc: false
dir: /mariadb/backup

## A customized mysqld.conf to import
mysqld_site_conf: |-
  [mysqld]
  userstat = on

## Resources per MariaDB container (default values from CMDB chart are
used)
resources:
  requests:
    memory: 256Mi
    cpu: 250m
  #limits:
  #memory:
  #cpu:

service:
  type: ClusterIP
  port: 80
  annotations: {}
  annotationsForScrape:
    prometheus.io/scrape: "true"
  labels: {}

ingress:
  enabled: true
  annotations:
    ingress.citm.nokia.com/sticky-route-services: $cookie_JSESSIONID|JSESSIONID ip_cookie
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    ### when ingress version is or lower than 1.14.30, use below value
    #nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/secure-backends: "true"
    nginx.ingress.kubernetes.io/ssl-passthrough: "true"
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
  labels: {}
  path: /grafana/?(.*)
  ### when ingress version is or lower than 1.14.30, use below value
  #path: /grafana
  hosts:
    - ""

```

```
tls: []
# - secretName: chart-example-tls
#   hosts:
#     - chart-example.local

## Whether use istio ingress gateway(Envoy)
istioIngress:
  enabled: true
  # when istio is enabled: root_url path and Contextroot path should match
  Contextroot: grafana
  selector: {istio: ingressgateway}
  # the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
  ## Keep gatewayName to empty to create kubernetes gateway resource. If new
  virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
  ## tls section will be used if gatewayName is ""
  tls:
    enabled: true
    httpsPort: 443
    ## mode could be SIMPLE, MUTUAL, PASSTHROUGH, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "am-gateway"

resources:
  limits:
    cpu: 500m
    memory: 1Gi
  requests:
    cpu: 100m
    memory: 128Mi

## Node labels for pod assignment
## ref: https://kubernetes.io/docs/user-guide/node-selection/
#
nodeSelector: {}

## Tolerations for pod assignment
## ref: https://kubernetes.io/docs/concepts/configuration/taint-and-
toleration/
##
tolerations: []

## Affinity for pod assignment
```

```
## ref: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/
#affinity-and-anti-affinity
##
affinity: {}
nodeAntiAffinity: hard

## Enable persistence using Persistent Volume Claims
## ref: http://kubernetes.io/docs/user-guide/persistent-volumes/
##
persistence:
  enabled: true
  ##storageClassName: default
  accessModes:
    - ReadWriteOnce
  size: 1Gi
  annotations: {}
  # subPath: ""
  # existingClaim:

## below variables adminUser and adminPassword are mandatory when sensitive
data is enabled.
adminUser: admin
#adminPassword: admin

## Use an alternate scheduler, e.g. "stork".
## ref: https://kubernetes.io/docs/tasks/administer-cluster/configure-
multiple-schedulers/
##
# schedulerName:

## Extra environment variables that will be pass onto deployment pods
env: {}

## The name of a secret in the same kubernetes namespace which contain values
to be added to the environment
## This can be useful for auth tokens, etc
envFromSecret: ""

## Additional grafana server secret mounts
# Defines additional mounts with secrets. Secrets must be manually created in
the namespace.
extraSecretMounts: []
  # - name: secret-files
  #   mountPath: /etc/secrets
  #   secretName: grafana-secret-files
```

```
# readOnly: true

# Pass the plugins you want installed as a comma separated list.
# plugins: "digrich-bubblechart-panel,grafana-clock-panel"
plugins: ""

# Pass the plugin urls as a list. the file should be in tar.gz format
# These files should be downloaded with out needing proxy or certs
# If there are any issues in grafana due to new plugins, then CPRO team will
# ask these plugins to be removed during debugging.
pluginUrls: []
 #- [local repository]/~chalapat/vonage-status-panel.tar.gz

## Uncomment below datasources section to add alertmanager datasource.
## Added alertmanager datasource cannot be modified in Grafana UI
## Datasource can only be modified through restapi requests.
#
#datasources:
#  datasources.yaml:
#    apiVersion: 1
#    datasources:
#      - name: alertmanager
#        type: cAMPtoCAMP-prometheus-alertmanager-datasource
#        url: http://prometheus-cpro-alertmanager
#        access: proxy
#        isDefault: false

## Configure grafana datasources
## ref: http://docs.grafana.org/administration/provisioning/#datasources
##
SetDatasource:
  ## If true, an initial Grafana Datasource will be set
  ## Default: false
  ##
  enabled: true

  ## How long should it take to commit failure
  ## Default: 300
  ##
  #activeDeadlineSeconds: 300

  ## Curl Docker image
  ## Default: appropriate/curl:latest
  ##
  imageRepo: cpro/grafana-registry1/grafana-curl
```

```
imageTag: "1.17.0"
imagePullPolicy: IfNotPresent

resources:
  limits:
    cpu: 200m
    memory: 128Mi
  requests:
    cpu: 100m
    memory: 64Mi

datasource:
  ## The datasource name.
  ## Default: default
  name: prometheus

  ## Type of datasource
  ## Default: prometheus
  ##
  type: prometheus

  ## The url of the datasource. To set correctly you need to know
  ## the right datasource name and its port ahead. Check kubernetes
  ## dashboard or describe the service should fulfill the requirements.
  ## Syntax like `http://<release name>-<server name>:<port number>
  ## Default: "http://limping-tiger-server"
  ##
  url: "http://prometheus-cpro-server"

  ## The name of the database at the datasource.
  ## Required parameter when used with elasticsearch, which refers to the
index_name
  ## Default: <empty>
  # database:

  ## Additional JSON data to be passed to the configuration of the
datasource.
  ## The JSON data is passed to curl, therefore it needs proper quoting and
  ## escaping and needs to be on a single line. For example:
  ## '\"esVersion\": 2, \"interval\": \"Daily\", \"timeField\":
  \"@timestamp\"'
  # jsonData: null

  ## Specify if Grafana has to go thru proxy to reach datasource
  ## Default: proxy
```

```
##  
access: proxy  
  
## Specify should Grafana use this datasource as default  
## Default: true  
##  
isDefault: true  
  
## Specify the job policy  
## Default: OnFailure  
##  
restartPolicy: OnFailure  
  
SetDashboard:  
  enabled: true  
  backoffLimit: 10  
## When upgrade if overwrite = true, dashboards in old release will be  
overwritten by dashboards in new chart.  
  overwrite: true  
  tinytools:  
    imageRepo: cpro/grafana-registry1/grafana-tiny-tools  
    imageTag: "1.10.1"  
    imagePullPolicy: IfNotPresent  
  resourcesTinytools:  
    limits:  
      cpu: 200m  
      memory: 128Mi  
    requests:  
      cpu: 100m  
      memory: 64Mi  
  
## Configure grafana dashboard providers  
## ref: http://docs.grafana.org/administration/provisioning/#dashboards  
##  
dashboardProviders: {}  
# dashboardproviders.yaml:  
#   apiVersion: 1  
#   providers:  
#     - name: 'default'  
#       orgId: 1  
#       folder: ''  
#       type: file  
#       disableDeletion: false  
#       editable: true  
#       options:
```

```
#           path: /var/lib/grafana/dashboards

## Configure grafana dashboard to import
## NOTE: To use dashboards you must also enable/configure dashboardProviders
## ref: https://grafana.com/dashboards
## 

dashboards: {}

# some-dashboard:
#   json: |
#     $RAW_JSON
# prometheus-stats:
#   gnetId: 2
#   revision: 2
#   datasource: Prometheus
# local-dashboard:
#   url: https://example.com/repository/test.json

livenessProbe:
  scheme: HTTPS
  initialDelaySeconds: 60
  timeoutSeconds: 1
  failureThreshold: 10
  periodSeconds: 3

readinessProbe:
  scheme: HTTPS
  initialDelaySeconds: 60
  timeoutSeconds: 30
  failureThreshold: 10
  periodSeconds: 10

keycloak:
# Update ckeyUrl with the deployed keyclock baseurl, ex: 10.76.84.192:32443,
# ckey.example.com, ckeyistio.example.com:31390i(istio ingress), 10.76.84.192/
ckey (ingress)
  url: "10.76.84.192:32443"
  protocol: https
  realm: cpro
## if secret is null, then keycloak use cert. If secret is not null and it is
# a existing secret name, then use secret.
  cert:
    "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURrakNDQW5xZ0F3SUJBZ0lFRjJFNFNEQU5CZ2txaGtpRzl
  secret:
## please notice if use upper secret, please fill existing secret name, not
# base64 code.
  scheme: https
```

```
grafana:
  ##mounted to file /etc/grafana/ssl/server.crt
  server_cert:
    "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM5ekNDQWQrz0F3SUJBZ0lKQU1BMTBFMmdUNH11TUEwR

  ##mounted to file /etc/grafana/ssl/server.key
  server_key:
    "LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCk1JSUV2d01CQURBTkJna3Foa2lHOXcwQkFRRUZBQVNDQktP

## Grafana's primary configuration
## NOTE: values in map will be converted to ini format
## ref: http://docs.grafana.org/installation/configuration/
## 

grafana_ini:
  paths:
    data: /var/lib/grafana/data
    logs: /var/log/grafana
    plugins: /var/lib/grafana/plugins
    provisioning: /etc/grafana/provisioning
  analytics:
    check_for_updates: false
    reporting_enabled: false
  log:
    mode: console
  grafana_net:
    url: https://grafana.net
  server:
    protocol: https
    # when istio is enabled: root_url path and Contextroot path should match
    root_url: "%(protocol)s://%(domain)s:%(http_port)s/grafana"
    cert_file: /etc/grafana/ssl/server.crt
    cert_key: /etc/grafana/ssl/server.key
    #set to true when istio is enabled
    serve_from_sub_path: true
  #to be uncommented only when sensitive data is enabled, and to be passed as
  secret
  #smtp:
  #  user: $(smtp_user)
  #  password: $(smtp_password)
  users:
    allow_sign_up: true
    allow_org_create: true
    auto_assign_org: true
    auto_assign_org_role: Viewer
```

```

auth:
  disable_login_form: false
  disable_signout_menu: false
  #OAuth state max age cookie duration. Defaults to 60 seconds.
  #oauth_state_cookie_max_age is only available from Grafana v7.0+.
  oauth_state_cookie_max_age: 60
  ;signout_redirect_url: "{{ .Values.keycloak.protocol }}://{{ .Values.keycloak.url }}/auth/realms/{{ .Values.keycloak.realm }}/protocol/openid-connect/logout?redirect_uri=https://10.76.62.42"
  auth.generic_oauth:
    enabled: false
    name: "{{ .Values.keycloak.realm }}"
    client_id: grafana
    client_secret: 1ala7188-b5b7-4c19-8459-c45c32a64437
    scopes: openid
    auth_url: "{{ .Values.keycloak.protocol }}://{{ .Values.keycloak.url }}//auth/realms/{{ .Values.keycloak.realm }}/protocol/openid-connect/auth"
    token_url: "{{ .Values.keycloak.protocol }}://{{ .Values.keycloak.url }}//auth/realms/{{ .Values.keycloak.realm }}/protocol/openid-connect/token"
    api_url: "{{ .Values.keycloak.protocol }}://{{ .Values.keycloak.url }}//auth/realms/{{ .Values.keycloak.realm }}/protocol/openid-connect/userinfo"
    introspect_url: "{{ .Values.keycloak.protocol }}://{{ .Values.keycloak.url }}//auth/realms/{{ .Values.keycloak.realm }}/protocol/openid-connect/token/introspect"
    allow_sign_up: true
    tls_client_ca: /etc/grafana/keycloak/keycloak.crt
    tls_skip_verify_insecure: false
    # role_attribute_path is only available from Grafana v6.5+.
    ;role_attribute_path:
database:
  type: sqlite3
  host: grafanadb-cmdb-mysql:3306
  name: grafana
  user: grafana
  password: grafana
  ;ssl_mode: true
  ;ca_cert_path: /etc/grafana/cmdbtls/ca.crt
  ;client_key_path: /etc/grafana/cmdbtls/client.key
  ;client_cert_path: /etc/grafana/cmdbtls/client.crt
  ;server_cert_name: grafanadb-cmdb-mysql.default.svc.cluster.local
  #session has been removed since grafana 6.2(chart 3.0.x)
  #session:
    #provider:
      #provider_config:
        #provider: mysql

```

```
#provider_config: '`grafana:grafana@tcp(grafanadb-cmdb-mysql:3306)/grafana`'

## LDAP Authentication can be enabled with the following values on grafana.ini
## NOTE: Grafana will fail to start if the value for ldap.toml is invalid

# auth.ldap:
#   enabled: true
#   allow_sign_up: true
#   config_file: /etc/grafana/ldap.toml

security:
  #to be uncommented only when sensitive data is enabled.
  #recomended to not change. admin_user and admin_password can be set in
  adminUser and adminPassword section of values.yaml
  #admin_user: "{{ .Values.adminUser }}"
  #admin_password: "{{ .Values.adminPassword }}"
  # Set to true if you host Grafana behind HTTPS. Default is false
  cookie_secure: false

# key value pair and only key can be extraArgs values
# extraArgs values fomat should be as shown below
# key: value
# key:
extraArgs: {}

## Grafana's LDAP configuration
## Templated by the template in _helpers.tpl
## NOTE: To enable the grafana.ini must be configured with auth.ldap.enabled
## ref: http://docs.grafana.org/installation/configuration/#auth-ldap
## ref: http://docs.grafana.org/installation/ldap/#configuration

ldap:
  # `existingSecret` is a reference to an existing secret containing the ldap
  configuration
  # for Grafana in a key `ldap-toml`.
  existingSecret: ""
  # `config` is the content of `ldap.toml` that will be stored in the created
  secret
  config: ""
  # config: |-
  #   verbose_logging = true

  # [[servers]]
  #   host = "my-ldap-server"
  #   port = 636
  #   use_ssl = true
  #   start_tls = false
  #   ssl_skip_verify = false
```

```
# bind_dn = "uid=%s,ou=users,dc=myorg,dc=com"

## Grafana's SMTP configuration
## NOTE: To enable, grafana.ini must be configured with smtp.enabled
## ref: http://docs.grafana.org/installation/configuration/#smtp
smtp:
  # `existingSecret` is a reference to an existing secret containing the smtp
  configuration
  # for Grafana in keys `user` and `password`.
  existingSecret: ""

## Sidecars that collect the configmaps with specified label and stores the
## included files them into the respective folders
## Requires at least Grafana 5 to work and can't be used together with
## parameters dashboardProviders, datasources and dashboards
sidecar:
  imageRepo: cpro/grafana-registry1/csf-grafana-kiwigrid
  imageTag: 0.1.144-1.2.0
  imagePullPolicy: IfNotPresent
  resources:
    limits:
      cpu: 100m
      memory: 100Mi
    requests:
      cpu: 50m
      memory: 50Mi
  SKIP_TLS_VERIFY: false
  enableUniqueFilenames: false
  dashboards:
    enabled: false
    # label that the configmaps with dashboards are marked with
    label: "grafana_dashboard"
    # folder in the pod that should hold the collected dashboards
    folder: "/home/dashboards"
    folderAnnotation: "grafana_folderpath"
    resource: "both"

  provider:
    # name of the provider, should be unique
    name: sidecarProvider
    # orgid as configured in grafana
    orgid: 1
    # folder in which the dashboards should be imported in grafana
    folder: ''
    # type of the provider
```

```
type: file
# disableDelete to activate a import-only behaviour
disableDelete: false
# allow updating provisioned dashboards from the UI
allowUiUpdates: false
# allow Grafana to replicate dashboard structure from filesystem
foldersFromFilesStructure: true

datasources:
  enabled: false
  # label that the configmaps with datasources are marked with
  label: grafana_datasource
```

25 Appendix: Example of cpro-rest-api-srv-values.yaml



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```
global:
  registry: "csf-docker-delivered.[]" in the appropriate repository.
# registry1 repo is used to pull restserver image
  registry1: "csf-docker-delivered.[]" in the appropriate repository.
# To configure annotations and labels for restserver resources
  annotations: {}
  labels: {}
## Define serviceAccount name for restserver component at global level.
## serviceAccount priority order is
## 1. serviceAccountName
## 2. global.serviceAccountName
## 3. If we are not using customized resources set rbac.enabled to true then
  resources will be created on helm install
## 4. If both serviceAccounts are not set and rbac.enabled is set to false
  then default serviceAccount will be used
##
  serviceAccountName:
## istio verion in X.Y format eg 1.4/1.5/1.6
  istioVersion: 1.4
  podNamePrefix: ""
  containerNamePrefix: ""

    ### custom name to override default of the pod and container name
    ## restServerContainer and ConfigmapReloadContainerName should be unique
customResourceNames:
  resourceNameLimit: 63
  restServerPod:
    restServerContainer: ""
    configMapReloadContainer: ""
  toolsPod:
    initContainer: ""
  helmTestPod:
    name: ""
    testContainer: ""

custom:
# To configure customized annotations and labels for restserver psp
  psp:
    annotations:
```

```
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
        apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
        apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    labels: {}

# To configure customized annotations and labels for restserver pod
pod:
    annotations:
        seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
        seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
        apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
        apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    labels: {}

rbac:
    enabled: true
    pspUseAppArmor: false
deployOnComPaaS: false

istio:
    enable: false
    mtls_enable: true
    cni_enable: true

sensitiveDataSecretName: ""

## Define serviceAccount name for restserver component.
##
serviceAccountName:

restserver:
    enabled: true
    ##soft means preferredDuringSchedulingIgnoredDuringExecution
    ####hard means requiredDuringSchedulingIgnoredDuringExecution
    #####you could refer k8s api for more detail
    ##
    antiAffinityMode: "soft"
    name: restserver
```

```
podAnnotations: {}

BCMT:
  serverURL: https://k8s-apiserver.bcmt.cluster.local:8443

image:
  imageRepo: cpro/registry4/prometheus-restapi
  imageTag: 3.5.7
  imagePullPolicy: IfNotPresent

replicaCount: 1

## If the flag is set to true, then restserver component is forbidden to
access
## the configmap or any other resources of prometheus server from different
namespaces and role/rolebinding is created
## else clusterrole/clusterrolebinding is created and able to access the
configmaps
## and other resources of prometheus server across the different namespaces.
##
restrictedToNamespace: false

seLinuxOptions:
  enabled: false
  level: ""
  role: ""
  type: ""
  user: ""

service:
  # Options: ClusterIP or NodePort
  type: ClusterIP
  servicePort: 8888
  # Note the range of a valid nodePort is 30000-32767.
  nodePort: 32766

ingress:
  enabled: false
  annotations: {}
  # kubernetes.io/ingress.class: nginx
  # kubernetes.io/tls-acme: "true"
  tls: []
  # - secretName: chart-example-tls
```

```

#     hosts:
#       - chart-example.local
## Whether use istio ingress gateway(Envoy)
istioIngress:
  enabled: true
  selector: {istio: ingressgateway}
  Contextroot: restserver
  # the host used to access the management GUI from istio ingress gateway
  host: "*"
  httpPort: 80
## Keep gatewayName to empty to create kubernetes gateway resource. If new
virtualservice needs to refer to existing gateway then mention that name here
  gatewayName: "istio-system/single-gateway-in-istio-system"
## tls section will be used if gatewayName is ""
  tls:
    enabled: false
    httpsPort: 443
    ## mode could be SIMPLE, MUTUAL, PASSTHROUGH, ISTIO_MUTUAL
    mode: SIMPLE
    credentialName: "resetserver-gateway"
resources:
  limits:
    cpu: 1
    memory: 512Mi
  requests:
    cpu: 100m
    memory: 128Mi

nodeSelector: {}

tolerations: []
# - key: "key"
#   operator: "Equal|Exists"
#   value: "value"
#   effect: "NoSchedule|PreferNoSchedule|NoExecute(1.6 only)"

configs:
  ncmsUsername: user-input
  ncmsPassword: "user-input"
  ncmsPassPhrase: user-input
  httpsEnabled: false
  restCACert: |+
    -----BEGIN CERTIFICATE-----
    MIIEETCCAvmgAwIBAgIJAIQFm91ZPjOsMA0GCSqGSIb3DQEBCwUAMIGeMQswCQYD
    VQQGEwJDTjEQMA4GA1UECAwHQmVpamluzZERMA8GA1UEBwwIV2FuZ2ppbmcxDjAM

```

```

B9NVBAoMBU5va21hMQwwCgYDVQQLDANDU0YxHjAcBgNVBAMMFw1pYW9mZW5rLWNh
Lm5va21hLmNvbTEsMCoGCSqGSIB3DQEJARYdbWlh2Z1bmcua2FuZ0Bub2tpYS1z
YmVsbC5jb20wHhcNMTgwNTIxMDIwNjIyWhcNMjgwNTE4MDIwNjIyWjCBnjELMAkG
A1UEBhMCQ04xE DAOBgNVBAgMB0J1aWppbmcxETAPBgNVBACmCFdhbmdqaW5nMQ4w
DAYDVQQKDAVOb2tpYTEMMaoGA1UECwwDQ1NGMR4wHAYDVQQDBBtaWFvZmVuay1j
YS5ub2tpYS5jb20xLDAqBgkqhkiG9w0BCQEWHLpYW9mZW5nLmthbmdAbm9raWEt
c2J1bGwuY29tMIIIBIjANBgkqhkiG9w0BAQEFAOCQ8AMIIIBCgKCAQEA6NyT+S/i
rFotAivpTUb2WjHVVvREgiC5f4pSqwHk/sVYv1+sTaaHPSd/wWa9yc1C9nQ3iHvX
9PYo8dRKYsPKxoFcSl8C1DLovYCUDrpVS2HbpBxXlUrBow46LMFDAUQAH8e4fi4K
+1/H5kxvnQRXPkKnx2uc13vGW5mRXTGfLGLHUwoVdighnRNAhqUUZqwafveBMXwf
6mBTdOi6JUg+Hz6UrIsZ1TciYI7t1weFm944YSKeF7L8LcqgnZxBfvbtyw8XmWd
M0Xbt8SJ3G1om4y8nbFyMHa4Fm8eN0TyxM71Ua2r1GAeda2rMNf+RSOreryExbYY
pNIzbUPyFFKkYQIDAQABo1AwTjAdBgNVHQ4EFgQUne4mdychS1HrO2W/uP+25A8dU
x7wwHwYDVR0jBBgwFOAU4mdychS1HrO2W/uP+25A8dUx7wwDAYDVR0TBAUwAwEB
/zANBgkqhkiG9w0BAQsFAAOCAQEA4iMY4kEK1c0miHQnEn6a3Jwn+7ajeJ5MQOWw
UONUH6T/EzUWJL74g2uoXFz2ykqv/eSSWjk6T0F+hJVKh2yXMeGS6+WvJlrakPNF
sylpjRR7rYqFXLqtWpLGoxE/mTHj0PeakvCN9Fo+kJ8PmkdYuIcvxdM11Lv0judJ
yrNL/3v2MLkEKF9xmtpjeyIYoRV1kf2ehdMVvWrpeWbY1PSNE2BSM60qFb5pCd9G
O1yQ15b4smvgx1kcIerZ6abMsLJUF9Z+8crHKja1/4e9/xoP5n/CSjzPNWnaGQM1
r/IGGTDPw6eym9yPH0nbv76/Mse3xqGGSZJA0eS1T6B2aXtUSg==
-----END CERTIFICATE-----
restServerKey: | +
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDICfMXbmZrWcOj
081L1wHhTJgZ6ZdkUXAj/Abkkk3PLwPoZ9KxYwKhZ54Zy1N6owy+raYa0bGZWNzf
b4P1cFMoGX7+vJYGOmkorF9AYBCh6jbeVoHfEYw1ozejfRo64Eud2g3MKBFwcvB
Za+bebm7QXLn0j3LURzN2Vembj7FZETg/weTGZC2Nw2RK+vACEFIeXkeDx2jsldQ
hCquOAYvRV0emAZIm7uf80Or6LoBOjKBkTUXGLWDbvSkzxUPGXgbZT15ZpeS+jpc
JH/gdjHVqoa+XDTzT8xZOrTXJp9HJUGg/Jy6P6MFRHJtkaFQ7Ci8j51teuBh2xFf
/fxhdPF7AgMBAECggEAXdmkhRsGz0qXJHS90b2YvsFbEf7iCHFs/Rw6qfiqf9A2
1zFNYArIp4PZbaBatLf09q5dcH8wFWmX7dVLxrZR6RuO73yjDcV5iTaz3nNcNkNw
b0e3xRb1PAPwv+XYgyqTnRQEktK1dqjHgybXUNwJ/Hr+AjjL/gJcYRK2r0RnN+S
FCPX4dpjNet70+vSz4nK+51qMJ20EvLdzhvIJkmKbjRr0/S6b+Ze1BKzo9DGfkvg
6B207As1ReHX5ahqSX1GPRGldcvW3VI4okHXtbIL30Y59c2+Prqjtg79fk/clo0M
Te4jUYm16wSW1RHMwtD8BweMCN/0A3nLvKhOekV0EQKBgQDyoKMXHJu2WrMsTtjY
1lwuvFeD/gCaHnS9W9m61TkvcnJZkwTXN16k16EmRgtbNaqDojnTU18LbOOR206
8w7ai79diL0G5bCst+uXpjIzld4tnf6n8TZysuiG8EI0u4CpmUkRXU7s0X8cu0qO
pw22stvsVzhUvWRHG0H2QZtygwKBgQDTfiN1RmjafyqrLWzch0AUEhtijtfndBuE
NMaM1IBSUQnDJoJf0rMX+UEttj9VohRmRM8p8aAMEN6IjLh7Y9oYjoXGyYzk2Pz
TacsueV477KkVBoS0snEAIjcrGxsCp2eCX/s1+4fJAYPS0WNDEDRMjoE7dBf6f6t
LnvXYgzzqQKBgAGDoSDuy8X6k02w3EeVwKOGB2HKfwR3NjFMVnKFDCNQ3Jqvs7/X
L4apTsizD+SQrlJHXvFamSYyPtGffm7XP3t7rckOpmdZnZ2mVDERF3Uc9VMBjmpL
5hPtoefdrfwYQ3hLfzo/I9P0hr+OJ6v2P06r9RVngFF9cRfEgsffpvGzAoGAJw6p
b7QEEy3e7GPkMbaXt90sL4RfvP/FQSIZ9NIdrJYIroCDHT01E+1VKyL4CVF4YPae
J4nW280VxTPvseHb2iMf83kvNfznPx+xvhTKmw3xOMXLVuSuNfZy6Z/yGfxP6+qn

```

```

NE8gS6H0eIiXHJhBtCCJdHWSwNPO0569Aia6a5kCgYEA1B0FdO4Ok8z3ut39aZpw
S1JooCCwdx5udyXtQvNPrmyMRRpTqFezCYIGYVquR5KyZjISQJswCUzCMbJc3hi
nqY/F+5nre9HURT1py66VDxX6rOXldiR1CbDS6VacV0JnfYX6FwtRjGosgdhv6nV
NDZmqe58yC7pYp77yFHJOPM=
-----END PRIVATE KEY-----
restServerCert: |+
-----BEGIN CERTIFICATE-----
MIIDtDCCApwCCQDIFRUb+zmnGDANBgkqhkiG9w0BAQUFADCn jELMAkGA1UEBhMC
Q04xEDAOBgNVBAgMB0JlaWppbmcxETAPBgNVBAcMCFdhbmdqaW5nMQ4wDAYDVQQK
DAVOb2tpYTEMMAoGA1UECwwDQ1NGMR4wHAYDVQQDBVtaWFvZmVuay1jYS5ub2tp
YS5jb20xLDAqBgkqhkiG9w0BCQEWHW1pYW9mZW5nLmthbmdAbm9raWEtc2J1bGwu
Y29tMB4XDTE4MDUyMTAyMDYyM1oXDTE5MDUyMTAyMDYyM1owgZgxCzAJBgNVBAYT
AkNOMRAwDgYDVQQIDAdCZWLqaW5nMREwDwYDVQQHDAhXYW5namluZzEOMAwGA1UE
CgwFTm9raWExDDAKBgNVBAsMA0NTRjEYMBYGA1UEAwPBwlhb2Zlbumsuc2VydmlV
MSwwKgYJKoZIhvcNAQkBFh1taWFvZmVuZy5rYW5nQG5va2lhLXNiZWxsLmNvbTCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMhx8xdUzmtZw6PTzUvXAeFM
mBnp12RRcCP8BuSSTc8vA+hn0rFjAqFnnhnLU3qjDL6tphRsZ1Y119vg/VwUygZ
fv681gY6aSisX0BgEKHqNt5Wgd8RjDWjn6MVGjrgS53aDcwoEUvBxF1r5t5ubtB
cufSPctRHm3ZV6ZuPsvkROD/B5MzkLY3DZEr68AIQUh5eR4PHaNKV1CEKq44Bi9F
XR6YBkibu5/zQ6vouge6MoGRNTEYtYNu9KTPFQ8ZeBt1oXlm15L60lwkf+B2MdWq
hr5cNPnPzFk6tNcmn0clQaD8nLo/owVEcm2RoVDsKLyPnW164GHbEV/9/GF08XsC
AwEAATANBgkqhkiG9w0BAQUFAAOCAQEAKNOkBdLhTwxDGE8941CIIHZ7pyYCP7
h+Ov0zfkLRzEwbIQwcrkAfQYy007xWCmNfcHaSdFTayaHV/i3QktSdQFwcI3i8+L
i/XkmGX+IvKG0KiQ/JwClKj06B/DUKqU6fyg51HWPuYovLdNo0kKEVe30Fb9wNr+
8dMHAW/drTbxngUtNArPM8xsa5gQId887bHRzXJs7Pd9tojWMeyny8A0bXrshrdF
812eZNla7nBiQTIPb9nJwjzW31B6V5jP95R4UnlvIgdhj2/UxEyrb/6yj/lfyO5H
h8BzBBalgf2pvJfxh9idS4cHC0Bqc+Etqble6tGNNuUTmFzfx2+rgg==
-----END CERTIFICATE-----
# Log Level: DEBUG < INFO < WARN < ERROR < FATAL < OFF
loglevel: INFO

## Ref: https://github.com/jimmidyson/configmap-reload
##
configmapReload:
## configmap-reload container name
##
name: configmap-reload

## configmap-reload container image
##
restapiConfigmapReloadImage:
imageRepo: cpro/registry4/configmap-reload-distro
imageTag: v0.2.1-4.2.3
imagePullPolicy: IfNotPresent

```

```
## Additional configmap-reload container arguments
##
extraArgs: {}

## Additional configmap-reload mounts
##
extraConfigmapMounts: []
# - name: prometheus-alerts
#   mountPath: /etc/alerts.d
#   configMap: prometheus-alerts
#   readOnly: true

## configmap-reload resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/
##
resources:
  limits:
    cpu: 10m
    memory: 32Mi
  requests:
    cpu: 10m
    memory: 32Mi

tools:
  image:
    imageRepo: cpro/registry4/tools-image
    imageTag: 1.8.0
    imagePullPolicy: IfNotPresent

## CPRO configmap name for helm test, default value is null, and it will use
# the default configmap.
helmtest:
  CPROconfigmapname:
    deletepolicy: hook-succeeded,before-hook-creation

## helmtest resource requests and limits
## Ref: http://kubernetes.io/docs/user-guide/compute-resources/

resources:
  limits:
    cpu: 10m
    memory: 32Mi
```

```
requests:  
  cpu: 10m  
  memory: 32Mi
```

26 Appendix: Example of cpro-gen3gpp-values.yaml



Note: The following repository/registry URLs are provided as examples, the user should add their own.

```
# Default values for gen3gppxml.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
replicaCount: 1

global:
  # For pull gen3gppxml, proftpd & ConfigmapReload image
  registry: "csf-docker-delivered.[...]" in the appropriate repository.
  # For pull Cbur and kubectl image
  registry2: "csf-docker-delivered.[...]" in the appropriate repository.
  annotations: {}
  labels: {}

## Define serviceAccount name for Gen3gppxml component at global level.
## serviceAccount priority order is
## 1. serviceAccountName
## 2. global.serviceAccountName
## 3. If we are not using customized resources set rbac.enabled to true then
##    resources will be created on helm install
## 4. If both serviceAccounts are not set and rbac.enabled is set to false
##    then default serviceAccount will be used
##
  serviceAccountName:
## istio verion in X.Y format eg 1.4/1.5/1.6
  istioVersion: 1.4
  podNamePrefix: ""
  containerNamePrefix: ""

persistence:
  pvc_auto_delete: true

custom:
  psp:
    annotations:
      seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
      seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
    apparmorAnnotations:
      apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
      apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
```

```
labels:

pod:
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: runtime/
default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: runtime/default
  apparmorAnnotations:
    apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/
default
    apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
  labels:

customResourceNames:
  resourceNameLimit: 63
  gen3gppxmlPod:
    gen3gppxmlContainer: ""
    sftpContainer: ""
    configMapReloadContainer: ""
  postDeleteJobPod:
    name: ""
    postDeletePvcContainer: ""

name: gen3gppxml
## soft means preferredDuringSchedulingIgnoredDuringExecution
## hard means requiredDuringSchedulingIgnoredDuringExecution
## you could refer k8s api for more detail
antiAffinityMode: "soft"
helm3: true
rbac:
  enabled: true
  pspUseAppArmor: false
serviceAccountName:

lcm:
  scale:
    timeout: 180
image:
  imageRepo: "cpro-gen3gppxml"
  imageTag: 3.0.0-2.3.2
  pullPolicy: IfNotPresent
resources:
  requests:
    memory: 256Mi
```

```
cpu: 250m
limits:
  memory: 1024Mi
  cpu: 500m
seLinuxOptions:
  enabled: false
  level: ""
  role: ""
  type: ""
  user: ""
service:
  name: gen3gppxml
  # Set service type to NodePort when istio.enable is set to 'false'
  serviceType: ClusterIP
  sftpPort: 2309
  sftpNodePort: 30022
  restHttpPort: 8080
  restHttpsPort: 8081

kubectl:
  image:
    repo: tools/kubectl
    tag: v1.19.5-nano-20201210
  jobResources:
    requests:
      cpu: 200m
      memory: 500Mi
    limits:
      cpu: 1
      memory: 1Gi

cbur:
  enabled: true
  image:
    repository: cbur/cbur
    tag: 1.0.3-1665
    pullPolicy: IfNotPresent
  resources:
    requests:
      memory: 256Mi
      cpu: 250m
    limits:
      memory: 1024Mi
      cpu: 500m
```

```
## defines the backup storage options.
backendMode: "local"

##autoEnableCron = true indicates that the cron job is immediately scheduled
when the BrPolicy is created,
##while autoEnableCron = false indicates that scheduling of the cron job
should be done on a subsequent backup request.
##This option only works when k8swatcher.enabled is true
autoEnableCron: false
##Indicate if subsequent update of cronjob will be done via brpoilicy
update.
##true means cronjob must be updated via brpolicy update,
##false means cronjob must be updated via manual "helm backup -t app -a
enable/disable" command.
autoUpdateCron: false
## It is used for scheduled backup task. Empty string is allowed for no
scheduled backup.
## Here means every 10 minutes of every day
cronJob: "*/10 * * * *"
## Limit the number of copies that can be saved. Once it is reached, the
newer backup will overwritten the oldest one.
maxCopy: 5
## Whether the chart will deploy on istio
istio:
  enable: false
  mtls_enable: true
  cni_enable: true

## Whether use istio ingress gateway(Envoy)
istioIngress:
  enabled: true
  selector: {istio: ingressgateway}
  Contextroot: gen3gppxml
  # the host used to access the management GUI from istio ingress gateway(for
  http/https)
  host: "*"
  httpPort: 80
  ## Keep gatewayName to empty to create kubernetes gateway resource.
  ## If new virtualservice needs to refer to existing gateway then mention
  that name here
  ## gatewayName is used for http/https
  gatewayName: "istio-system/single-gateway-in-istio-system"
  ## tls section will be used if gatewayName is ""(within http gateway only)
  tls:
    enabled: false
    httpsPort: 443
```

```
## mode could be SIMPLE, MUTUAL, PASSTHROUGH
mode: PASSTHROUGH
credentialName: "am-gateway"
## Keep gatewayName to empty to create kubernetes gateway resource.
## If new virtualservice needs to refer to existing gateway then mention
that name here
## tcpGatewayName is used for sftp
tcpGatewayName: ""
sftpPort: 31400
# the host used to access the management GUI from istio ingress gateway(for
tcp)
tcpHost: "*"

configmapReload:
image:
repository: cpro/registry4/configmap-reload
tag: v0.2.1-4.2.3
pullPolicy: IfNotPresent
resources:
requests:
memory: 256Mi
cpu: 250m
limits:
memory: 1024Mi
cpu: 500m
containerSecurityContext:
runAsUser: 1001
sftp:
image:
repository: cpro-gen3gppxml-proftpd
tag: 3.0.0-2.3.2
pullPolicy: IfNotPresent
resources:
requests:
memory: 256Mi
cpu: 250m
limits:
memory: 1024Mi
cpu: 500m
user: sftp
#MaxLoginAttempts are the number of max wrong login attempts before there is
no ban on the user.
#banEngine tells whether to have banEngine enabled or not.
#banTime is the time for which user will be banned.
```

```

#banWithinTime is the time for which MaxLoginAttempts is counted. So if user
enters wrong login credentials within "ban #WithinTime" period. Then user is
banned for "banTime". Time should be given in hh:mm:ss format.
#debugLevel is the level of debug logs for proftpd container. Value ranges
from 1 to 10.

proftpdconfigs:
  MaxLoginAttempts: 3
  banEngine: "ON"
  banTime: 00:05:00
  banWithinTime: 00:01:00
  debugLevel: 5
  logLevel: 3
  authentication:
    passwd:
      enabled: true
      passwd: jAA628Mc5sokT9Gkf77QI5vhZGh1qJqKo7ZqO/
ndjpldoIZIjSeYghrkuqsKy4rrxXOWhzHW9YciLfu4Ly6zTy/
z7dr7I0NvNJdIi9Sxfw5NDwXqDDRiir7cRcwRmIJy9vmBtx3Rm0dSatWjs9zk2hL2SU9E/
rJwdmlJI3PbkZWC4E47SixLqElFQhi8mdSvqkBO/oHj6hc9uTKG89nzfw59fQg1Tisb986Kjh
+iI0SolfT1AcFXAKFVbM9Dx3exu+BTm/sUKrwPUBuFicxzjHuso
+Mh8LX5nScFcKj6IZxX0LSuAvOajvb1khvcu4Tzcl0tFXRiTd+1JXpLCSai2w==

rsaKey: |+
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAOCKPFnOzIsNAizi+uMWuKBdwEzrMVO+avnv5eEzP7zDUAms1
LanvgXfGnx9JSld04YiEKQeNk7g+WZn/VW5yNU63s3a93RhWCQupyfkIbmstfs6x
lest/15Yy8V6g1lTekz4lGNMNDNp03kHY3HJBEMFVYMC0+BAvgOpUf/iyqztGSrL
1Ueme9SxKg5OotIAD5SGJ175soh6wtTSOz2YXThymUWg7J6+RKmti3cwu8ANuP7P
h2Z5xsBTZ6UBC0B1KZxx/cXUzHS7M06hh4n0qPU4dQrl1rOLjYptIzNRp9CN8LqG
4B43aAH9AXwipkm7PiG2+jrzKyZpjpFGm+Mi jwIDAQABoIBADycY8jpBpiv/YLL
rgJG5vGb6bJf5BMz939TQRhsZH1GXecYhr52h13PkJLUHJa+ozXueHYuUedbtySG
uRma4Tlu4AYuJRpWt09LazV7BR6KraQI8yQyaaG9SuSEbKoFgE7UTWuBAzcMZdwE
mcc24pAQL/dDJcitTI1JKazMdLIKZQXV9YT7P9IKzrPlwSf6jxDComsiROuNgQuR
qj4625C10qGl46CFC4h5cP0bwUb52JCBDSsieBN9XIPFSvZ7uMK+VluzM9RikuNZ
mgHccSYxL2xzXWK6ynmw3CH1FfrJi0y4A1h+S94r0nesjrP1rr7ANDMQXws1+ydP
To096tECgyEA9hW3taGM8kuP1O9xNZQSzB+6kM4Qb9t/X9PWJcByATe4rNFs25jo
DBQAtPgne6t+UjtoWPgn2p4V+/PK4/5ys5JZ6LlA1iqjbdk4PKz63A/9vlq52Kde
HpxpPwc9o2PzNf4Wt7nbp//MTE69MDVxhxD6NL4XBs7TJoL/8JZH+qcCgYEA2IVm
x9hUCcpshyZko5W04t+yjhXqZ5V2vawy7YmTZz1NnXdbk8kD0JrwOU8lgK/MIJ//
vPbitLqFFA965LPBK8D9EL7Kgrjbw9SWVJqEHDKCMWRjxzkGq84kyhFVtcdI6Bne
5UMGERUPksrgcwgms/gViCN4oWDZkdtQU3HXdkCgYEA2jwYV7OVU6s/Ow+9zk4q
/WuGALU+TnTSWSF8YK5ybDIGTPzNFVJGKN/0YI1S0pKzzClzOK2/p4la5b4myFr3
TdkUcxXMNdPC/i/rs+zCWAFH7NF7aVACtWbXyEXwLb7EX7slrL04eAwRdvuW7qJ
m8aqneuWcCsrbhpPWdn5QBzECgYBGzOhc9NoWEYEHeIB0FVWRzkAewf/SO0eGoiZN
Ei2HFr6ofP0PCC4dKmze8Ih82f9Ps8H2UmyzWEJn2t/+pODHo+WccibaNAR4tq0p
oRUN+14jNIaEAJuz1kalZKG4PSAT9VGzegUkXNxozbozIyJIVSzIPQDot2w2J6kM

```

```

L5i/kQKBgQCXVteCrb1BL2HFiXbZ8kn/KHdBQUfMJ7JfWHobEtR8JZmayvOhfkOm
qK8KPCI1YnZpw0oSxgILxyP2jFnvdo8EMu36ragA7S52pMSQMS0D9BV/QXtzNUYB
LRn+jze+W0YgvZEgocxvFoRMol6Oh8K4X6ZA/Q6v8GX/OdWdPbQ1WQ==
-----END RSA PRIVATE KEY-----

key:
  enabled: true
  #ssh2 key to be used
  key: |+
    ----- BEGIN SSH2 PUBLIC KEY -----
    Comment: "2048-bit RSA, converted by root@vm-10-76-154-108 from
OpenSS"
    AAAAB3NzaC1yc2EAAAADAQABAAQDQIo8Wc7Miw0CLMj64xa4oF3ATOsxU75q+e/14TM
    /vMNQCZLUtqe+Bd8afH01KV3ThiIQpB42TuD5Zmf9VbnIlTrezdr3dGFYJC6nJ+Qhuayl+
    zrGV6y3/X1jLxXqDWVN6TPiUY0w0M2nTeQdjccxEQx9XIwKj4EC+A6lR/+LKrO0ZKsuVR6
    Z71LEqDk6i0gAP1IYnXvmyiHrC1NI7PZhdOHKZRaDsnr5Eoy2LdzC7wA24/s+HZnnGwFNn
    pQELQHUpnHH9xdTMdLsw7qGHifSo9Th1CuWWs4uNg+0jM1Gn0I3wuobgHjdoAf0BfCKmSb
    s+Ibb6OvMrJmmOkUab4yKP
    ----- END SSH2 PUBLIC KEY -----


#User can use the configOverride to override the configuration in configs/
Gen3GPPXML
#for example: OVERRIDE_XXX_YYY_ZZZ = 11111
#it is the same as below configuration:
#{{
#    "XXX" :
#    {
#        "YYY" :
#        {
#            "ZZZ" : "11111"
#        }
#    }
#}
#NOTE: It could not override the value in an array.
configOverride: |+
  OVERRIDE_prometheus_url = http://cpro-server.default.svc.cluster.local:80/
api/v1
configs:
  Gen3GPPXML: '{
    "measCollecFile" :
    {
      "xmlns" : "http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec",
      "xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-instance",

```

```
"xmlns:schemaLocation" : "http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec",
    "fileHeader" :
    {
        "fileFormatVersion" : "32.435 V9.1.0",
        "vendorName" : "NOKIA",
        "dnPrefix" :"dn",
        "elementType" : "eletype"
    }
},
"REST" :
{
    "HTTP" :
    {
        "enable":true,
        "httpPort":8080
    },
    "HTTPS" :
    {
        "enable":false,
        "httpsPort" : 8081,
        "verifyClient" : false,
        "securityFiles" :
        {
            "caCert":"/etc/gen3gppxml/secret/restCa",
            "cert":"/etc/gen3gppxml/secret/restCert",
            "key":"/etc/gen3gppxml/secret/restKey"
        }
    },
    "SSO" :
    {
        "enable":false,
        "validationUrl":"http://localhost:8280/auth/realm/myrealm/
protocol/openid-connect/userinfo",
        "HTTPS": false,
        "securityFiles" :
        {
            "caCert":"/etc/gen3gppxml/secret/ssoca",
            "cert":"/etc/gen3gppxml/secret/ssoCert",
            "key":"/etc/gen3gppxml/secret/ssoKey"
        }
    }
},
"prometheus" :
```

```
{
    "url" : "http://localhost:9090/api/v1",
    "HTTPS" : false,
    "securityFiles" :
    {
        "caCert": "/etc/gen3gppxml/secret/prometheusCa",
        "cert": "/etc/gen3gppxml/secret/prometheusCert",
        "key": "/etc/gen3gppxml/secret/prometheusKey"
    },
    "authentication" :
    {
        "enable": "false",
        "username" : "root",
        "password" : "P8KaOyx504Vpbv+hNFUGHicuO9TKArmVhGXAlCIBut2R7i/
DDd0Hx2W6OG2Q/XjJ8/NNxb1IljPSwErOIrPrHB30uvvH7vOhA2BhuMBFgYb4n/
+uKJaVqFS08s7dH8c0tKPbPGxhei9m64Nb/
fMhCnacD2yn5p04w173z1Y7fPuJVbzV1luXhuaiG42UqCntZTKjxhqEFczLqKcb7JAVABBMjiRmb91iYd44tKP73I
        "key": "/etc/gen3gppxml/secret/prometheusAuthKey"
    }
},
"plugins" :
{
    "vesAgent" :
    {
        "enable" : "false",
        "connectString" : "127.0.0.1:9991",
        "appName" : "csf"
    }
},
"postFix" : "gen3gppxmlDefaultPostFix",
"logLevel" : "INFO",
"reportLocation" : "/var/3gppxml/",
"expiredDays" : "14",
"notUseColonInQuery" : "false",
"threadPoolSize" : 20,
"misfireGraceTime" : 1,
"filter" : ["job='prometheus-nodeexporter'", "component='node-
exporter'"],
"defaultGranPeriod" : [ "5m" ],
"defaultRule" : "avg",
"defaultValueType" : "integer",
"orientation" : "vertical",
"hideNonExistentMetrics" : "true",
"invalidValue" : "0",
"FastPass" : {
}
}
```

```

"enable" : "true",
"maxAttributeValueEachLength" : 80,
"maxAttributeTypeEachLength" : 6,
"maxAttributeTypeNum" : 10,
"replaceInvalidCharInMeasObjLdnWith" : "_"
},
"measGroups" :
[
{
  "measInfoId" : "M1",
  "objects" : { "PLTFM": "1" },
  "metrics" :
  [
    {
      "metric": "node_boot_time_seconds"
    }
  ],
  "measInfoId" : "M2",
  "objects" : { "PLTFM": "1" },
  "metrics" :
  [
    {
      "metric": "node_ipvs_incoming_bytes_total", "rule": "max",
      "metric": "node_ipvs_outgoing_bytes_total", "rule": "max"
    }
  ],
  "measInfoId" : "M3",
  "objects" : { "PLTFM": "1" },
  "dnLabels" : [ "CPU" ],
  "labelRename": { "CPU": "cpu" },
  "metrics" :
  [
    {
      "metric" : "node_cpu_seconds_total",
      "rule" : "max",
      "label" :
      {
        "name" : "mode",
        "value" : [ "idle", "iowait", "irq", "nice", "softirq",
"steal", "system", "user" ]
      }
    }
  ]
}
]

```

```
"measInfoId" : "M4",
"objects" : { "PLTFM": "1" },
"dnLabels" : [ "DISK" ],
"labelRename": { "DISK": "device" },
"metrics" :
[
    { "metric": "node_disk_read_bytes_total", "rule": "max" },
    { "metric": "node_disk_written_bytes_total", "rule": "max" },
    { "metric": "node_disk_io_now" },
    { "metric": "node_disk_io_time_seconds_total" }
]
},
{
    "measInfoId" : "M5",
    "objects" : { "PLTFM": "1" },
    "metrics" :
    [
        { "metric": "node_entropy_available_bits" }
    ]
},
{
    "measInfoId" : "M6",
    "objects" : { "PLTFM": "1" },
    "metrics" :
    [
        { "metric": "node_load1" },
        { "metric": "node_load5" },
        { "metric": "node_load15" }
    ]
},
{
    "measInfoId" : "M7",
    "objects" : { "PLTFM": "1" },
    "metrics" :
    [
        { "metric": "node_memory_AnonPages_bytes" },
        { "metric": "node_memory_Buffers_bytes" },
        { "metric": "node_memory_Cached_bytes" },
        { "metric": "node_memory_Dirty_bytes" },
        { "metric": "node_memory_HardwareCorrupted_bytes" },
        { "metric": "node_memory_Mapped_bytes" },
        { "metric": "node_memory_MemFree_bytes" },
        { "metric": "node_memory_MemTotal_bytes" },
        { "metric": "node_memory_Shmem_bytes" },
        { "metric": "node_memory_SwapFree_bytes" },
    ]
}
```

```

        {
            "metric": "node_memory_SwapTotal_bytes" ,
            "metric": "node_memory_Writeback_bytes" ,
            "metric": "node_memory_MemAvailable_bytes"
        }
    },
    {
        "measInfoId" : "M8" ,
        "objects" : { "PLTFM": "1" } ,
        "metrics" :
        [
            {
                "metric": "node_ipvs_incoming_packets_total" , "rule": "max" ,
                "metric": "node_ipvs_outgoing_packets_total" , "rule": "max"
            }
        ]
    },
    {
        "measInfoId" : "M9" ,
        "objects" : { "PLTFM": "1" } ,
        "metrics" :
        [
            {
                "metric": "node_procs_blocked" ,
                "metric": "node_procs_running" ,
                "metric": "node_context_switches_total" , "rule": "max"
            }
        ]
    },
    {
        "measInfoId" : "M10" ,
        "objects" : { "PLTFM": "1" } ,
        "metrics" :
        [
            {
                "metric": "node_netstat_Tcp_AttemptFails" ,
                "metric": "node_netstat_Tcp_InSegs" ,
                "metric": "node_netstat_Tcp_OutSegs" ,
                "metric": "node_netstat_Tcp_RetransSegs" ,
                "metric": "node_netstat_TcpExt_ListenDrops"
            }
        ]
    },
    {
        "measInfoId" : "M11" ,
        "objects" : { "PLTFM": "1" } ,
        "metrics" :
        [
            {
                "metric": "node_netstat_Udp_InDatagrams" ,
                "metric": "node_netstat_Udp_InErrors" ,
                "metric": "node_netstat_Udp_OutDatagrams"
            }
        ]
    }
}

```

```

} ,
{
  "measInfoId" : "M12",
  "objects" : { "PLTFM": "1" },
  "metrics" :
  [
    { "metric": "node_vmstat_pgpgin" },
    { "metric": "node_vmstat_pgpfout" }
  ]
},
{
  "measInfoId" : "M13",
  "objects" : { "PLTFM": "1" },
  "dnLabels" : [ "IF" ],
  "labelRename": { "IF": "device" },
  "metrics" :
  [
    { "metric": "node_network_receive_bytes_total" },
    { "metric": "node_network_receive_drop_total" },
    { "metric": "node_network_receive_errs_total" },
    { "metric": "node_network_receive_packets_total" }
  ]
},
{
  "measInfoId" : "M14",
  "objects" : { "PLTFM": "1" },
  "dnLabels" : [ "IF" ],
  "labelRename": { "IF": "device" },
  "metrics" :
  [
    { "metric": "node_network_transmit_bytes_total" },
    { "metric": "node_network_transmit_drop_total" },
    { "metric": "node_network_transmit_errs_total" },
    { "metric": "node_network_transmit_packets_total" }
  ]
},
{
  "measInfoId" : "M15",
  "objects" : { "PLTFM": "1" },
  "dnLabels" : [ "DISK", "FS", "MP" ],
  "labelRename": { "DISK": "device", "FS": "fstype", "MP": "mountpoint" },
  "metrics" :
  [
    { "metric": "node_filesystem_free_bytes" },
    { "metric": "node_filesystem_free_inodes" }
  ]
}

```

```

        {"metric": "node_filesystem_size_bytes" }
    ]
}
]
}

secrets:
prometheusCa: |+
-----BEGIN CERTIFICATE-----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FX11bWwdtGVzdC5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBDoqc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+fuu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAsGA1UdDwQEAwIFoDATBqNVHSUEDDAKbgrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguapkzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==
-----END CERTIFICATE-----
prometheusCert: |+
-----BEGIN CERTIFICATE-----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FX11bWwdtGVzdC5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBDoqc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+fuu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAsGA1UdDwQEAwIFoDATBqNVHSUEDDAKbgrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguapkzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==
-----END CERTIFICATE-----

```

```

prometheusKey: | +
----BEGIN CERTIFICATE----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FX11bWwtdGVzdC5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBD0qc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+fuu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyzedswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAwIFoDATBgNVHSUEDDAKBgggrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAQM+e+8peUh49cdpKCsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguapkzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==

----END CERTIFICATE----

prometheusAuthKey: | +
----BEGIN CERTIFICATE----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FX11bWwtdGVzdC5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBD0qc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+fuu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyzedswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAwIFoDATBgNVHSUEDDAKBgggrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAQM+e+8peUh49cdpKCsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguapkzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==

----END CERTIFICATE----

restCa: | +
----BEGIN CERTIFICATE----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FX11bWwtdGVzdC5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBD0qc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzAm

```

```
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB  
UuOLlUbei8HqVCRFFp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/  
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb  
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w  
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAWIFoDATBgNVHSUEDDAKBgggrBgeFBQcDAjAN  
BqkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2  
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir  
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4  
04AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI  
3fGCSzLVZ1N2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH  
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==  
-----END CERTIFICATE-----  
restCert: |+  
-----BEGIN CERTIFICATE-----  
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1  
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM  
FX11bWwtGVzdc5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq  
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79  
CtaHcU8RrKAd7zBDoc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzaAm  
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB  
UuOLlUbei8HqVCRFFp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/  
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb  
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w  
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAWIFoDATBgNVHSUEDDAKBgggrBgeFBQcDAjAN  
BqkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2  
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir  
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4  
04AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI  
3fGCSzLVZ1N2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH  
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==  
-----END CERTIFICATE-----  
restKey: |+  
-----BEGIN CERTIFICATE-----  
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1  
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM  
FX11bWwtGVzdc5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq  
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79  
CtaHcU8RrKAd7zBDoc8p89dWOIfsn1E/lv1G1zvFfpNgtpzeNnzJVLvIZJAzaAm  
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB  
UuOLlUbei8HqVCRFFp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/  
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb  
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w  
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAWIFoDATBgNVHSUEDDAKBgggrBgeFBQcDAjAN  
BqkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2  
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
```

```
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==

-----END CERTIFICATE-----

ssoCa: | +
-----BEGIN CERTIFICATE-----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FXl1bWwtdGVzdc5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBDoc8p89dWOIfsn1E/lvlg1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAwIFoDATBgNVHSUEDDAKBggrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==

-----END CERTIFICATE-----

ssoCert: | +
-----BEGIN CERTIFICATE-----
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM
FXl1bWwtdGVzdc5sb2NhbGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/HwlaIwGeuf6PhI79
CtaHcU8RrKAd7zBDoc8p89dWOIfsn1E/lvlg1zvFfpNgtpzeNnzJVLvIZJAzAm
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaX11Y2VRVuX+3XX6IGle5olazlte/k+kHB
UuOLlUbei8HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb
vXkdzvjyteswNDEPbO1ER5efTQ1IIpAqIe5Iz0phyT7tbh1OM7EiDQIDAQABoy8w
LTAJBgNVHRMEAjAAAMAsGA1UdDwQEAwIFoDATBgNVHSUEDDAKBggrBgeFBQcDAjAN
BgkqhkiG9w0BAQsFAAOCAQEAQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir
nE9jklhKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4
O4AyB2JqwBJRj7Wvgymtfqc1QVhuQD+iAUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI
3fGCSzLVZlN2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH
kJdafMTyYWPXygCdsaSHeggbr6oTpwd395bsA7lcE6d6xw==

-----END CERTIFICATE-----

ssoKey: | +
-----BEGIN CERTIFICATE-----
```

```
MIIC7jCCAdagAwIBAgIBAjANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAhNeVR1  
c3RDQTAeFw0xODAzMjIxMDA4MTFaFw0xOTAzMjIxMDA4MTFaMDExHjAcBgNVBAMM  
FX11bWwtdGVzdC5sb2NhGRvbWFpbjEPMA0GA1UECgwGY2xpZW50MIIBIjANBgkq  
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA01FXQ7ItSD5zsKZ/Hw1aIwGeuf6PhI79  
CtaHcU8RrKAd7zBDoqc8p89dWOIfsn1E/lv1G1zvFfpNgtPzeNnzJVLvIZJAzAm  
0jJtXaxYDnKiLrR9hzNeJXuLgp6FYjaXlly2VRVuX+3XX6IGle5olazlte/k+kHB  
UuOLlUbe18HqVCRFfp+UWkWz85u+QrQlz/EmfB/O8uZ4LJmO6yBDyB123gRyVmi/  
7c9jIyVclz9+f2uu3jo0Eg5NMuRKkHv8Hh8v9Nx0FrIaz/yr3MX2wqzVuVbEQXYb  
vXkdzvjyzedNDEPb01ER5efTQ1IIpAqIe5Iz0phyT7tbh10M7EiDQIDAQABoy8w  
LT AJBgNVHRMEAjaAMAsGA1UdDwQEAvIFoDATBgNVHSUEDDAKBgggrBqEFBQcDAjAN  
BgkqhkiG9w0BAQsFAAOCAQEAEQM+e+8peUh49cdpKCxsGKShkcSw8DOPWbTb26Z2  
6js+gE2XhWvvP2qk6i/A0SLvFtpPC2sf+cilA9ZonrH08jQAY/mw0spzG41lQ7Ir  
nE9jk1hKrb8azrorNXMsYI7D7wKhGX4oL8aq7UND2A0cxQhYRvIg9pn/DIFiSAY4  
04AyB2JqwBJRj7Wvgymtfqc1QVhuQD+i AUGcfBMZQctbpiQqZKi4wyzj6AnEn+pI  
3fGCSzLXVZ1N2m4aXbhZRzd4mok1YcTalyalxJymMoguakpzVGNO1UXyPQIXxeQH  
kJdafMTyYWPXygCdsASHeeggbr6oTpwd395bsA71cE6d6xw==  
-----END CERTIFICATE-----  
  
persistentVolume:  
  size: 1Gi  
  #storageClass: default  
  
tolerations: []  
nodeSelector: {}
```

27 Appendix: Example Telemetry profile yaml

```

btel:
lcm:
  install:
    flags: --timeout 1000
  upgrade:
    flags: --force --reuse-values

global:
  registry: "bcmt-registry:5000"
  registry1: "bcmt-registry:5000"
  registry2: "bcmt-registry:5000"
  registry3: "bcmt-registry:5000"
  registry4: "bcmt-registry:5000"

tags:
  belk-fluentd: true
  belk-elasticsearch: false
  belk-kibana: false
  belk-curator: false
  belk-elasticsearch-exporter: false

belk:
  belk-fluentd:
    fluentd:
      EnvVars:
        system: "BCMT"
        systemId: "123456"
      nodeSelector:
        is_btel_all: 'true'
      tolerations:
        - key: 'is_btel'
          operator: 'Equal'
          value: 'true'
          effect: 'NoSchedule'
        - key: 'is_edge'
          operator: 'Equal'
          value: 'true'
          effect: 'NoExecute'
        - key: 'is_control'
          operator: 'Equal'
          value: 'true'

```

```
effect: 'NoExecute'  
configFile: |  
  <source>  
    @type tail  
    path /var/log/messages,/var/log/bcmt/apiserver/audit.log  
    pos_file /var/log/td-agent/non-containers.json.access.pos  
    tag kubernetes.non-container  
    #read_from_head true  
  <parse>  
    @type regexp  
    expression /^(?<header>[^{\}]+)?(?<message>\{ .+\})$)|(^(?  
<log>[^{\}].+))/  
  </parse>  
</source>  
  
<source>  
  @type tail  
  path /var/log/containers/*.log  
  pos_file /var/log/td-agent/containers.json.access.pos  
  tag kubernetes.*  
  #read_from_head true  
  <parse>  
    @type json  
    time_key time  
    time_format %iso8601  
    keep_time_key true  
  </parse>  
</source>  
  
<match kubernetes.non-container>  
  @type rewrite_tag_filter  
  <rule>  
    key header  
    pattern \sjournal:( \[stdout\])* \s*$  
    tag kubernetes.non-container.clear  
  </rule>  
  <rule>  
    key message  
    pattern ^\{ .+\}$  
    tag kubernetes.non-container.json  
  </rule>  
  <rule>  
    key log  
    pattern .+  
    tag kubernetes.non-container.legacy
```

```

        </rule>
    </match>

    <match kubernetes.non-container.clear>
        @type null
    </match>

    <filter kubernetes.non-container.legacy>
        @type record_modifier
        <record>
            log ${{"message":record['log']}}
            type log
        </record>
    </filter>

    <filter kubernetes.non-container.json>
        @type parser
        key_name message
        format json
        time_parse false
    </filter>

    <filter kubernetes.non-container.*>
        @type record_modifier
        remove_keys _dummy1_,_dummy2_,_dummy3_
        <record>
            _dummy1_ ${if !record.has_key?("host"); record["host"] = ENV["HOST"]; end; "UNAVAILABLE"}
            _dummy2_ ${if !record.has_key?("system"); record["system"] = ENV["SYSTEM"]; end; "UNAVAILABLE"}
            _dummy3_ ${if !record.has_key?("systemid"); record["systemid"] = ENV["SYSTEMID"]; end; "UNAVAILABLE"}
        </record>
    </filter>

    #this filter is used for C API which remove "[stdout]" from log
    #if CLOG Unified Logging C API won't be used, this filter can be
removed

    <filter kubernetes.var.log.containers.**.log>
        @type parser
        format /^(\[stdout\])*(<log>.+)$/i
        key_name log
        suppress_parse_error_log true
    </filter>

```

```
<filter kubernetes.var.log.containers.**.log>
    @type record_modifier
    <record>
        message ${record["log"]}
    </record>
</filter>

<filter kubernetes.var.log.containers.**.log>
    @type record_modifier
    remove_keys log
</filter>

<filter kubernetes.var.log.containers.**.log>
    @type kubernetes_metadata
    kubernetes_url https://kubernetes.default.svc:443
    de_dot false
</filter>

<match kubernetes.var.log.containers.**.log>
    @type rewrite_tag_filter
    <rule>
        key message
        pattern ^\s*\{(.+,\)?\s*"type":.+}\}\s*$
        tag nokia.logging.json
    </rule>
    <rule>
        key message
        pattern .+
        tag nokia.logging.legacy
    </rule>
</match>

<filter nokia.logging.json>
    @type parser
    key_name message
    reserve_data true
    format json
    time_parse false
</filter>

<filter kubernetes.non-container.legacy>
    @type record_transformer
    enable_ruby true
    <record>
        time ${time.strftime('%Y-%m-%dT%H:%M:%S.%LZ')}
    </record>

```

```
</record>
</filter>

<filter nokia.logging.*>
    @type record_modifier
    remove_keys _dummy1_
    <record>
        _dummy1_ ${record.has_key?( "kubernetes" ) ? record[ "namespace" ]=record[ "kubernetes" ]
        [ "namespace_name" ]:record[ "namespace" ] = 'UNAVAILABLE'
        tenant_msctype ${record[ "namespace" ]}.${record[ "type" ]}
    </record>
</filter>

<match kubernetes.non-container.json>
    @type rewrite_tag_filter
    <rule>
        key type
        pattern ^(.+)$
        tag nokia.logging.default.$1
    </rule>
</match>

<match kubernetes.non-container.legacy>
    @type rewrite_tag_filter
    <rule>
        key log
        pattern ^(.+)$
        tag nokia.logging.default.legacy
    </rule>
</match>

<filter nokia.logging.legacy>
    @type record_modifier
    <record>
        log ${{ "message":record[ "message" ]}}
        type log
    </record>
</filter>

<match nokia.logging.json>
    @type rewrite_tag_filter
    <rule>
        key "type"
        pattern ^(.+)$
```

```
    tag "nokia.logging.$1"
  </rule>
</match>

##Here is a sample for fluent-plugin-brevity-control
##this plugin can be used to remove reduplicative records
##for more details, please refer to CLOG User Guide
#<filter nokia.logging.log>
#  @type brevity_control
#  interval 10
#  num 2
#  attr_keys log.message, level
#  max_slot_num 100000
#  stats_msg_fields kubernetes
#</filter>

<filter nokia.logging.**>
  @type clog
</filter>

<match nokia.logging.*>
  @type rewrite_tag_filter
  <rule>
    key tenant_msctype
    pattern ^(.+)\.log$ 
    tag nokia.logging.$1.tmp
  </rule>
  <rule>
    key tenant_msctype
    pattern ^(.+\..+)$
    tag nokia.logging.$1
  </rule>
  <rule>
    key tenant_msctype
    pattern ^(.+)\.$
    tag nokia.logging.$1.legacy
  </rule>
</match>

<match nokia.logging.*.tmp>
  @type rewrite_tag_filter
  <rule>
    key facility
    pattern 24
    tag nokia.logging.${tag_parts[2]}.authlog
  </rule>
</match>
```

```

</rule>
<rule>
    key message
    pattern ^\s*\{(.+)?\s*"event-type":.+}\s*\$tag nokia.logging.\${tag_parts[2]}.auditlog
</rule>
<rule>
    key message
    pattern ^\s*\{(.+)?\s*"message":.+}\s*\$tag nokia.logging.\${tag_parts[2]}.log
</rule>
<rule>
    key message
    pattern .+
    tag nokia.logging.\${tag_parts[2]}.log2
</rule>
</match>

<filter nokia.logging.*.*>
    @type record_modifier
    remove_keys priority, _uid, _gid, _systemd_slice,
    _machine_id, _transport, _cap_effective, _comm, _exe, _cmdline, _hostname,
    _systemd_cgroup, _systemd_unit, _selinux_context, _boot_id, _pid, message,
    container_id_full, container_name, container_tag, _source_realtime_timestamp,
    docker, kubernetes, tenant_msgtype
</filter>

#####
## prometheus config start ##
#####
<source>
    @type prometheus
    #uncomment below line "bin" in IPv6 Env
    #bind ::

</source>

<filter nokia.logging.*.counter>
    @type record_modifier
    <record>
        counter_value ${record["counter"]["value"]}
        counter_mid ${record["counter"]["mid"]}
        counter_object ${record["counter"]["object"]}
        counter_id ${record["counter"]["id"]}
        namespace ${record["namespace"]}
    </record>

```

```

</filter>

#
# WARNING: Don't use the match directive here. Otherwise, the
messages of counter
# cannot be processed further by other parts behind this block.
#
<filter nokia.logging.*.counter>
  @type prometheus
  <metric>
    name meas_gauge
    type gauge
    desc "measurement exported via fluent-plugin-prometheus"
    key counter_value
    append_timestamp true
    metric_expiration 300
    group_keys mid, host, namespace
    audit_interval 30
    <labels>
      mid ${counter_mid}
      object ${counter_object}
      id ${counter_id}
      host ${host}
      namespace ${namespace}
    </labels>
  </metric>
</filter>

#####
## prometheus config end  ##
#####

<match nokia.logging.*.*>
  @type relabel
  @label @NOKIA-LOGGING-ROUTING
</match>

<label @NOKIA-LOGGING-ROUTING>
  <filter nokia.logging.*.alarm>
    @type record_transformer
    <record>
      @class
com.nsn.cam.alma.own.api.event.UnifiedLoggingAlarmEvent
    </record>
  </filter>

```

```

#
# tag: nokia.logging.${namespace}.${type}
#
<match nokia.logging.*.alarm>
    @type copy
#####
## amqp config start ##
#####
<store>
    @type amqp
    host crmq-ext.btel.svc.cluster.local:5672
    vhost /
    user alma_mquser
    pass alma_mqpasswd
    key event
    exchange cfw
    exchange_type direct
    content_type application/json
    durable true
    <buffer>
        @type file
        path /var/log/td-agent/rabbitmq-buffer/
nokia.logging.all.alarm
    flush_mode immediate
    overflow_action block
    total_limit_size 1024m
    </buffer>
</store>
#####
## amqp config end ##
#####

#####
## kafka config start ##
#####
# <store>
# @type kafka_buffered
# output_data_type json
# buffer_type file
# ssl_ca_certs_from_system false
# get_kafka_client_log false
# max_send_retries 2
# buffer_path /var/log/td-agent/kafka-buffer/
nokia.logging.all.alarm

```

```

        # brokers kf-btel-releasename:9092
        # default_topic alma
        # flush_interval 3s
        # </store>
#####
## kafka config end ##
#####

<store>
    @type rewrite_tag_filter
    <rule>
        key type
        pattern .+
        tag ${tag}.es
    </rule>
</store>
</match>

<match nokia.logging.**>
    @type null
</match>
</label>

belk-elasticsearch:
    elasticsearch_master:
        replicas: 3
        no_of_masters: 2
        nodeSelector:
            is_btel_worker: "true"
        tolerations:
            - key: 'is_btel'
              operator: 'Equal'
              value: 'true'
              effect: 'NoSchedule'
    elasticsearch_client:
        replicas: 3
        nodeSelector:
            is_btel_worker: "true"
        tolerations:
            - key: 'is_btel'
              operator: 'Equal'
              value: 'true'
              effect: 'NoSchedule'
    esdata:
        replicas: 2

```

```
nodeSelector:
  is_btel_worker: "true"
tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
persistence:
  storageClassName: "cinder-az-nova"
belk-kibana:
  kibana:
    nodeSelector:
      is_btel_worker: "true"
    tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'

cnot:
  cnot:
    replicaCount: 1
    nodeSelector:
      is_btel_worker: "true"
    tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
  cnot-configmap:
    cnotFiles:
      app_conf.yaml:
        CHANNELS:
          ALARM:
            servers:
- id: AlarmServer
          default: AlarmServer
        EMAIL:
          servers:
- id: SMTPServer1
          host: mail.emea.nsn-intra.net
          port: 25
          auth: false
          userName: user1
          password: '{2-5590228828990040}oIZvbnI='
```

```
        from: from@nokia.com
        to: to@nokia.com
        sendAsHtml: false
        tls: false
        starttls: false
        trustCert: EMAIL_SMTPServer1.crt
        description: one SMTP relay server
        default: SMTPServer1

TRIGGERS:
  - id: TriggerForCPRO
    condition:
      body: $.alerts[?(@.labels.source == 'CPRO' || @.labels.source
== 'NCM')]
    action:
      type: accept
      profile: ProfileForCPRO
  - id: TriggerForCALM
    condition:
      body: $.alerts[?(@.labels.source == 'CALM')]
    action:
      type: accept
      profile: ProfileForCALM

calm:
  ha: true
  snmpNodePort: 31161
  snmpTrapNodePort: 31162
  restNodePort: 30182
  heartBeatInterval: 300
  servers: 2
  persistence:
    storageClass: "cinder-az-nova"
  nodeSelector:
    is_btel_worker: "true"
  tolerations:
    - key: 'is_btel'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'
  url: jdbc:mariadb:failover//cmdb-mysql.btel.svc.cluster.local:3306/
calm_alma
  cnot:
    serverURL: cnot.btel.svc.cluster.local:443
  mq:
    hostname: crmq-ext.btel.svc.cluster.local:5671
```

```

clientKey: | +
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAlFga17gGSGAOFst7mxBKeeom7HF1AVPkqFWWu7rDr8VJQ4ib
iqS2Cj4EXymalyncOh9olrBBmMz9FokgYqooy3v9rkvvdkqlcRE93HX8KFA8xTgT
MqsgqINwGTRiXjnIKcpvNQAAjhyORJPPexCUPUaMk2ECYg6n51iEpjpaOxgaAOHM
Z06H+UjyWFvL3/PfvUz4J6GR0Cmu8LhfHZVXxwuH94EGPaxTjQmw7B5dVjrolANC
Yv0x/TBcFRGdKgCETYhc+Tozja6p+LbCrNA7Vhs36FVBL+gM6S2Kqg9w9gNAEBPm
dfzdA/0BfM5s0w10TiUIjAppMkWRNhP0Ug1UQIDAQABoIBAHKghQ0CSFH1mGSR
Lo6MgsfBQPXOYW0wDnVYbBO3RD+0b1Jlgj6bn7FzvQRp/y47aKjvn5QI3cBQmCb8
K0FXveHqswzN7RycOycIOa6y+kYA2m1UUfi+LEkLew4DnYNkCcuvf4Vg4vToMVyP
ticm0f8qAGTmr1SjuLs9+Y2KXn8bDWsxMbRQ9IxZbeBT086t3xagATeQa50zkiNT
glWtD06F5uGIWLXTB2r4AfUSmAidxeVjt5ifBMqHg/PXpM527UT/TaCjFiOMhs6y
go4WiN0ipqxuoD2lQQ2GKtPnVYKKkMV/YqvRjFQMoIKNkUoBIGbFb0PjOhIKg8jW
v/7KuoECgYE9/S05tEOi8y14vky3q3SRCikWZ5+gtR6VUjcxKVZPRl+HP0qqjP+
3NsGRt+Gr4Y8csvNYRDLvtRCBuXczKqH/YmINMBtUhZRp41p/nP7Sns8XJkyA1A
5L+qgSuFFNBjHa5LibYnOTu1L1V71fHDVbFgis571t+3+46zcoErIvkCgYE2zuj
1oBGU6Kk3hEDAhrGffPOUhilh6qIa8SE3u/HunC/lU6bT5c1W4CCAs+Hrdjd7Db5
D5F52CpGUheMDHD4tZHkYxf6FYDlarTLpRtiQE3UmDFgA1jVbN1E7JYDN14DRQN
C+X/99jyGfyg/J+F+5cYeTct5qD1m+GukIIIs4xkCgYAEjS5g6kd9E3WcE/z2D3TD
oYdayckCt8nDFQJQbcSU1Yo2F9bzJt8vyK111ZaQVEN5y/nqJTFIOfc4TzHdY+f5
JHQmm721+xjnFQOBINnLPbDGwdXNwjW5vTbxBg58U1LvcYqbbYeMymbKAfGy/Kwc
P5RJF/82xsaGzJjIqkxTaQKBgQCZtgEzhCxBzJNOORBD4DoQo8AZKN6sYig9tKq
dLg28mKBTvH6JeFdflarbdKVg551xkluKvOhK1LPsA7TkknvFF95ci0p/vTaFUGo
QQ1jp+Sf75XKKJjnjN7rAyqSNsKJBmTNkikikAAoPSl+dgEOymMfiBVw7fdn3Dg8U
5xCN0QKBgQC+WvJFhoPr+3cu+QOnTDyiqv0R5/4QIs7gWLWLQbqt9+3x6BolARPp
5RGkf+3/ApTeeZM4XEE1RN1ef3o0wxOdVAbAyzk8obK+R1kI+o74vQm6buR6HFr
i454fx11vDvFgJr+U5BzCbF9Mnmwc8/+2Zz4vMZpAuZTMAfGiKMQ==
-----END RSA PRIVATE KEY-----
clientCert: | +
-----BEGIN CERTIFICATE-----
MIIDc jCCAlqgAwIBAgIRANEJiuWI5+7FudZa8Cge92YwDQYJKoZIhvcNAQELBQAw
UDELMAkGA1UEBhMCQUExCzAJBgNVBAgMAkFBMQswCQYDVQQHDAJBQTELMAkGA1UE
CgwCQUExCzAJBgNVBAsMAkFBMQ0wCwYDVQQDDARCQ01UMB4XDTE5MTEyNTAzNDAx
MVoXDTIwMDgwNzE5NDAxM沃wLTERMBMGA1UEChMMY2VydC1tYW5hZ2VyMRQwEgYD
VQQDEwtleGFtcGx1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
ANRYGte4BkhgDhUre5sQSnnqJuxxdQFT5KhVVru6w6/FSUOIm4qktgo+BF8pmtp
3DofaJawQZjM/RaJIGKqKMT7/a5L73ZkpXERPdx1/ChQPMU4EzKriKiDcBk64145
yCnKcjuAGo4cjkSTz3sQ1D1GjJNhAmIOP+ZYhKY6WjsYggDhzGTuh/lI8lhby9/z
371M+CehkTgprvCx3x2VV8cLh/eBBj2sU40JsOweXVY66JQDXGL9Mf0wXBURnSoA
hE2IXPk6M42uqfi2wqzQO1YbN+hVQS/oDOktiqoKMPYDQBAT5nX2XQP9AXzObNMJ
dE41CIwKaTJFkTYT9FIFNVECAwEAAaNqMGgwDgYDVR0PAQH/BAQDAgWgMAwGA1Ud
EwEB/wQCMAAwHwYDVR0jBBgwFoAU1hOMHX6BCH74kPkS8EcCU3rn3vowJwYDVR0R
BCAwHoILZXhhbXBsZS5jb22CD3d3dy5leGFtcGx1LmNvbTANBgkqhkiG9w0BAQsF
AAOCAQEAYJlbkOgEXSMqu6FiUbDjv/Iu80TLT4+CRWz1VS1clVe1+rW2ahuCRo16
VjJozhp3KO1hADBji9y+Wh1AlQ0QxxCd+e0S2CKDRHmKfne9x25Y6dB43hbi62U

```

```

EF3QSWzcylbs8Pox5QyuAtU8zx9iWKWp82T+gDzXcB2s3Vm5fp/p/n0ruR6IOL0w
0NgXDpwWm4ULGfId/glzaVt/qR948BWpiwkQwxrp5M0ce0mh4bJ0tgeq2McTtHwd
gF5Ebstoj4wBUdrhYCDIJF/fNEuSJlgmH4OtqgPn954YxOx7g0C62t/rZCpneF2t
zJe6BGkuI8AlS8eF+WZNv9ESxb0USQ==
-----END CERTIFICATE-----
serverCert: | +
-----BEGIN CERTIFICATE-----
MIIDcjCCAlqgAwIBAgIRANEJiuWI5+7FudZa8Cge92YwDQYJKoZIhvcNAQELBQAw
UDELMAkGA1UEBhMCQUExCzAJBgNVBAgMAkFBMQswCQYDVQQHDAJBQTELMAkGA1UE
CgwCQUExCzAJBgNVBAAsMAkFBMQ0wCwYDVQQDDARCQ01UMB4XDTE5MTEyNTAzNDAx
MVoXDTIwMDgwNzE5NDAxMVoWLTENVMBMGA1UEChMMY2VydC1tYW5hZ2VyMRQwEgYD
VQQDEwtleGFtcGx1LmNvbTCCASiWQYJKoZIhvcNAQEBQADggEPADCCAQoCggEB
ANRYGte4BkhgDhUre5sQSnnqJuxxdQFT5KhVvru6w6/FSUOIm4qktgo+BF8pmtcp
3DofaJawQZjM/RaJIGKqKMt7/a5L73ZKpXERPdx1/ChQPMU4EzKrIKiDcBk64145
yCnKcjUAGo4ckjStz3sQ1D1GjJNhAmIOp+ZYhKY6WjsYGgDhzGTuh/1I81hby9/z
371M+CehkTgprvCx3x2VV8cLh/eBBj2sU40JsOweXVY66JQDXGL9Mf0wXBURnSoA
hE2IXPk6M42uqfi2wqzQ01YbN+hVQS/oDOktiqoKMPYDQBAT5nX2XQP9AXzObNMJ
dE4lCIwKaTJFkTYT9FIFNVECAwEAaNaqMGgwDgYDVR0PAQH/BAQDAgWgMAwGA1UD
EwEB/wQCMAAwHwYDVR0jBBgwFoAU1hOMHX6BCH74kPkS8EcCU3rn3vowJwYDVR0R
BCAwHoILZXhhbXBsZS5jb22CD3d3dy5leGFtcGx1LmNvbTANBgkqhkiG9w0BAQsF
AAOCAQEAYJ1bk0gEXSMqu6FiUbDjv/Iu8OTLT4+CRWz1VS1clVe1+rW2ahuCRo16
VjJozhp3KO1hADBji9y+Wh1AlQ0QxxCd+e0S2CKDRHmKfne9x25Y6dB43hbi62U
EF3QSWzcylbs8Pox5QyuAtU8zx9iWKWp82T+gDzXcB2s3Vm5fp/p/n0ruR6IOL0w
0NgXDpwWm4ULGfId/glzaVt/qR948BWpiwkQwxrp5M0ce0mh4bJ0tgeq2McTtHwd
gF5Ebstoj4wBUdrhYCDIJF/fNEuSJlgmH4OtqgPn954YxOx7g0C62t/rZCpneF2t
zJe6BGkuI8AlS8eF+WZNv9ESxb0USQ==
-----END CERTIFICATE-----
snmpv3UserConfiguration:
- userName: test1
  authProtocol: MD5
  authPassword: U2FsdGVkX1+0gy24BhfY4s8Pu1z9Jsk817UOA6+3/rE=
  privProtocol: DES
  privPassword: U2FsdGVkX1+0gy24BhfY4s8Pu1z9Jsk817UOA6+3/rE=
  passPhrase: mypassphrase
  context: contexta
- userName: test2
  authProtocol: MD5
  authPassword: U2FsdGVkX1+0gy24BhfY4s8Pu1z9Jsk817UOA6+3/rE=
  privProtocol: DES
  privPassword: U2FsdGVkX1+0gy24BhfY4s8Pu1z9Jsk817UOA6+3/rE=
  passPhrase: mypassphrase
  context: contextb
- userName: test3
  authProtocol: SHA

```

```
authPassword: U2FsdGVkX19sdXC5I7SnDoRbir1VgidfuJVanA2LSuo=
privProtocol: AES
privPassword: U2FsdGVkX19sdXC5I7SnDoRbir1VgidfuJVanA2LSuo=
passPhrase: mypassphrase
context: admin

snmpv2cUserConfiguration:
- securityName: sec1
  communityName: test11
  context: contexta
- securityName: sec2
  communityName: test12
  context: contextb
- securityName: sec3
  communityName: test13
  context: admin

groupConfiguration:
- groupName: v2cgroup1
  securityModel: SNMPv2c
  userName: sec1
  context: contexta
- groupName: v2cgroup2
  securityModel: SNMPv2c
  userName: sec2
  context: contextb
- groupName: v2cgroup3
  securityModel: SNMPv2c
  userName: sec3
  context: admin
- groupName: v3group1
  securityModel: USM
  userName: test1
  context: contexta
- groupName: v3group2
  securityModel: USM
  userName: test2
  context: contextb
- groupName: v3group3
  securityModel: USM
  userName: test3
  context: admin

viewConfiguration:
- viewName: fullReadView
```

```
oid: 1.3
context: contexta
- viewName: fullWriteView
  oid: 1.3
  context: contexta
- viewName: fullNotifyView
  oid: 1.3
  context: contexta
- viewName: fullReadView
  oid: 1.3
  context: contextb
- viewName: fullWriteView
  oid: 1.3
  context: contextb
- viewName: fullNotifyView
  oid: 1.3
  context: contextb
- viewName: fullReadView
  oid: 1.3
  context: admin
- viewName: fullWriteView
  oid: 1.3
  context: admin
- viewName: fullNotifyView
  oid: 1.3
  context: admin

accessConfiguration:
- groupName: v2cgroup1
  securityModel: SNMPv2c
  securityLevel: noAuthNoPriv
  readView: fullReadView
  writeView: fullWriteView
  notifyView: fullNotifyView
  context: contexta
- groupName: v2cgroup2
  securityModel: SNMPv2c
  securityLevel: noAuthNoPriv
  readView: fullReadView
  writeView: fullWriteView
  notifyView: fullNotifyView
  context: contextb
- groupName: v2cgroup3
  securityModel: SNMPv2c
  securityLevel: noAuthNoPriv
```

```
readView: fullReadView
writeView: fullWriteView
notifyView: fullNotifyView
context: admin
- groupName: v3group1
  securityModel: USM
  securityLevel: authPriv
  readView: fullReadView
  writeView: fullWriteView
  notifyView: fullNotifyView
  context: contexta
- groupName: v3group2
  securityModel: USM
  securityLevel: authPriv
  readView: fullReadView
  writeView: fullWriteView
  notifyView: fullNotifyView
  context: contextb
- groupName: v3group3
  securityModel: USM
  securityLevel: authNoPriv
  readView: fullReadView
  writeView: fullWriteView
  notifyView: fullNotifyView
  context: admin

notificationTargetsConfiguration:
- notifyName: notify1
  notifyIP: 10.32.230.206
  notifyPort: 162
  notifyParams: v2cParams1
  notifyType: trap
  context: contexta
- notifyName: notify2
  notifyIP: 10.92.132.246
  notifyPort: 162
  notifyParams: v2cParams2
  notifyType: trap
  context: contextb
- notifyName: notify3
  notifyIP: 127.0.0.1
  notifyPort: 162
  notifyParams: v2cParams3
  notifyType: trap
  context: admin
```

```
- notifyName: notify6
  notifyIP: 10.92.132.246
  notifyPort: 162
  notifyParams: v3Params1
  notifyType: trap
  context: contexta
- notifyName: notify7
  notifyIP: 127.0.0.1
  notifyPort: 162
  notifyParams: v3Params2
  notifyType: trap
  context: contextb
- notifyName: notify8
  notifyIP: 127.0.0.1
  notifyPort: 162
  notifyParams: v3Params3
  notifyType: trap
  context: admin

notificationParametersConfiguration:
- notifyParamsName: v2cParams1
  notifyUser: sec1
  notifySecurityModel: SNMPv2c
  notifySecurityLevel: noAuthNoPriv
  context: contexta
- notifyParamsName: v2cParams2
  notifyUser: sec2
  notifySecurityModel: SNMPv2c
  notifySecurityLevel: noAuthNoPriv
  context: contextb
- notifyParamsName: v2cParams3
  notifyUser: sec3
  notifySecurityModel: SNMPv2c
  notifySecurityLevel: noAuthNoPriv
  context: admin
- notifyParamsName: v3Params1
  notifyUser: test1
  notifySecurityModel: USM
  notifySecurityLevel: authPriv
  context: contexta
- notifyParamsName: v3Params2
  notifyUser: test2
  notifySecurityModel: USM
  notifySecurityLevel: authPriv
  context: contextb
```

```
- notifyParamsName: v3Params3
  notifyUser: test3
  notifySecurityModel: USM
  notifySecurityLevel: authPriv
  context: admin

cpro:
  alertmanager:
    replicaCount: 2
    nodeSelector:
      is_btel_worker: "true"
    tolerations:
      - key: 'is_btel'
        operator: 'Equal'
        value: 'true'
        effect: 'NoSchedule'
    persistentVolume:
      storageClass: "cinder-az-nova"
  kubeStateMetrics:
    replicaCount: 1
    nodeSelector:
      is_btel_worker: "true"
    tolerations:
      - key: 'is_btel'
        operator: 'Equal'
        value: 'true'
        effect: 'NoSchedule'
  nodeExporter:
    nodeSelector:
      is_btel_all: "true"
    tolerations:
      - key: 'is_btel'
        operator: 'Equal'
        value: 'true'
        effect: 'NoSchedule'
      - key: 'is_edge'
        operator: 'Equal'
        value: 'true'
        effect: 'NoExecute'
      - key: 'is_control'
        operator: 'Equal'
        value: 'true'
        effect: 'NoExecute'
  server:
    replicaCount: 2
```

```

extraHostPathMounts:
  - name: certs-dir
    mountPath: /etc/etcd
    hostPath: /etc/etcd
    readOnly: true
nodeSelector:
  is_btel_worker: "true"
tolerations:
  - key: 'is_btel'
    operator: 'Equal'
    value: 'true'
    effect: 'NoSchedule'
persistentVolume:
  storageClass: "cinder-az-nova"
pushgateway:
  replicaCount: 1
  nodeSelector:
    is_btel_worker: "true"
  tolerations:
    - key: 'is_btel'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'
ha:
  enabled: true
alertmanagerFiles:
  alertmanager.yml:
    receivers:
      - name: default-receiver
        webhook_configs:
          - url: 'http://cnot.btel.svc.cluster.local:80/api/cnot/v1/notif'
            http_config:
              tls_config:
                ca_file: /etc/config/alertmanager/tls/alertmanager.cert
serverFiles:
  alerts:
    groups:
      - name: pre-defined-alert-rules
        rules:
          - alert: NodeFilesystemUsageMinor
            expr: ((node_filesystem_size_bytes{device="rootfs"})
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 >
70) and ((node_filesystem_size_bytes{device="rootfs"})

```

```
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 <= 80)
    for: 1m
    labels:
        severity: MINOR
        name: FileSystemExceedThreshold
        text: Usage of file system is greater than the threshold value
        id: 3001200
        source: CPRO
        host: "{$labels.kubernetes_io_hostname}"
        eventType: 11
        probableCause: 351
        key: MOC001
    annotations:
        summary: "{$labels.instance}: High Filesystem usage detected"
        description: "{$labels.instance}: Filesystem usage is above
70%"

- alert: NodeFilesystemUsageMajor
    expr: ((node_filesystem_size_bytes{device="rootfs"})
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 >
80) and ((node_filesystem_size_bytes{device="rootfs"}
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 <= 90)
    for: 1m
    labels:
        severity: MAJOR
        name: FileSystemExceedThreshold
        text: Usage of file system is greater than the threshold value
        id: 3001201
        source: CPRO
        host: "{$labels.kubernetes_io_hostname}"
        eventType: 11
        probableCause: 351
        key: MOC002
    annotations:
        summary: "{$labels.instance}: High Filesystem usage detected"
        description: "{$labels.instance}: Filesystem usage is above
80%"

- alert: NodeFilesystemUsageCritical
    expr: (node_filesystem_size_bytes{device="rootfs"})
- node_filesystem_free_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} * 100 > 90
    for: 1m
    labels:
```

```

severity: CRITICAL
name: FileSystemExceedThreshold
text: Usage of file system is greater than the threshold value
id: 3001202
source: CPR0
host: "{{\$labels.kubernetes_io_hostname}}"
eventType: 11
probableCause: 351
key: MOC003
annotations:
  summary: "{{\$labels.instance}}: High Filesystem usage detected"
  description: "{{\$labels.instance}}: Filesystem usage is above
90%"
  - alert: NodeCPUUsageMinor
    expr: (100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 > 70) and (100 - avg
by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 <= 80)
    for: 1m
    labels:
      severity: MINOR
      name: CPUExceedThreshold
      text: Usage of CPU is greater than the threshold value
      id: 3001203
      source: CPR0
      host: "{{\$labels.kubernetes_io_hostname}}"
      eventType: 11
      probableCause: 351
      key: MOC004
      annotations:
        summary: "{{\$labels.instance}}: High CPU usage detected"
        description: "{{\$labels.instance}}: CPU usage is above 70%"
      - alert: NodeCPUUsageMajor
        expr: (100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 > 80) and (100 - avg
by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 <= 90)
        for: 1m
        labels:
          severity: MAJOR
          name: CPUExceedThreshold
          text: Usage of CPU is greater than the threshold value
          id: 3001204
          source: CPR0
          host: "{{\$labels.kubernetes_io_hostname}}"
          eventType: 11
          probableCause: 351

```

```
key: MOC005
annotations:
  summary: "{$labels.instance}": High CPU usage detected"
  description: "{$labels.instance}": CPU usage is above 80%
- alert: NodeCPUUsageCritical
  expr: 100 - avg by (instance)
(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: CPUExceedThreshold
    text: Usage of CPU is greater than the threshold value
    id: 3001205
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC006
  annotations:
    summary: "{$labels.instance}": High CPU usage detected"
    description: "{$labels.instance}": CPU usage is above 90%
- alert: NodeSwapUsageMinor
  expr: (100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 70) and (100 -
(node_memory_SwapFree_bytes + 1) / (node_memory_SwapTotal_bytes + 1) * 100 <=
80)
  for: 1m
  labels:
    severity: MINOR
    name: SwapExceedThreshold
    text: Usage of swap space is greater than the threshold value
    id: 3001206
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC007
  annotations:
    summary: "{$labels.instance}": High swap space usage detected"
    description: "{$labels.instance}": Swap space usage is above
70%
- alert: NodeSwapUsageMajor
  expr: (100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 80) and (100 -
```

```
(node_memory_SwapFree_bytes + 1) / (node_memory_SwapTotal_bytes + 1) * 100 <= 90)
  for: 1m
  labels:
    severity: MAJOR
    name: SwapExceedThreshold
    text: Usage of swap space is greater than the threshold value
    id: 3001207
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC008
  annotations:
    summary: "{$labels.instance}: High swap space usage detected"
    description: "{$labels.instance}: Swap space usage is above 80%"
  - alert: NodeSwapUsageCritical
    expr: 100 - (node_memory_SwapFree_bytes + 1) /
(node_memory_SwapTotal_bytes + 1) * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: SwapExceedThreshold
    text: Usage of swap space is greater than the threshold value
    id: 3001208
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC009
  annotations:
    summary: "{$labels.instance}: High swap space usage detected"
    description: "{$labels.instance}: Swap space usage is above 90%"
  - alert: NodeMemoryUsageMinor
    expr: (100 - (node_memory_MemFree_bytes +
node_memory_Buffers_bytes + node_memory_Cached_bytes) /
node_memory_MemTotal_bytes * 100 > 70) and (100 - (node_memory_MemFree_bytes +
node_memory_Buffers_bytes + node_memory_Cached_bytes) /
node_memory_MemTotal_bytes * 100 <= 80)
  for: 1m
  labels:
    severity: MINOR
    name: MemoryExceedThreshold
```

```

text: Usage of memory is greater than the threshold value
id: 3001209
source: CPRO
host: "{$labels.kubernetes_io_hostname}"
eventType: 11
probableCause: 351
key: MOC010
annotations:
  summary: "{$labels.instance}: High memory usage detected"
  description: "{$labels.instance}: Memory usage is above 70%"
- alert: NodeMemoryUsageMajor
  expr: (100 - (node_memory_MemFree_bytes +
node_memory_Buffers_bytes + node_memory_Cached_bytes) /
node_memory_MemTotal_bytes * 100 > 80) and (100 - (node_memory_MemFree_bytes +
node_memory_Buffers_bytes + node_memory_Cached_bytes) /
node_memory_MemTotal_bytes * 100 <= 90)
  for: 1m
  labels:
    severity: MAJOR
    name: MemoryExceedThreshold
    text: Usage of memory is greater than the threshold value
    id: 3001210
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC011
  annotations:
    summary: "{$labels.instance}: High memory usage detected"
    description: "{$labels.instance}: Memory usage is above 80%"
- alert: NodeMemoryUsageCritical
  expr: 100 - (node_memory_MemFree_bytes + node_memory_Buffers_bytes +
node_memory_Cached_bytes) / node_memory_MemTotal_bytes * 100 > 90
  for: 1m
  labels:
    severity: CRITICAL
    name: MemoryExceedThreshold
    text: Usage of memory is greater than the threshold value
    id: 3001211
    source: CPRO
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 11
    probableCause: 351
    key: MOC012
  annotations:

```

```

summary: "{{\$labels.instance}}: High memory usage detected"
description: "{{\$labels.instance}}: Memory usage is above 90%"
- alert: NodeZombieCountMinor
  expr: (zombieCount > 5) and (zombieCount <= 10)
  for: 1m
  labels:
    severity: MINOR
    name: ZombieCountExceedThreshold
    text: Zombie process count is greater than the threshold value
    id: 3001212
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC013
  annotations:
    summary: "{{\$labels.instance}}: Too many zombie processes
detected"
    description: "{{\$labels.instance}}: Zombie process count is
above 5"
- alert: NodeZombieCountMajor
  expr: (zombieCount > 10) and (zombieCount <= 15)
  for: 1m
  labels:
    severity: MAJOR
    name: ZombieCountExceedThreshold
    text: Zombie process count is greater than the threshold value
    id: 3001213
    source: CPRO
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 351
    key: MOC014
  annotations:
    summary: "{{\$labels.instance}}: Too many zombie processes
detected"
    description: "{{\$labels.instance}}: Zombie process count is
above 10"
- alert: NodeZombieCountCritical
  expr: zombieCount > 15
  for: 1m
  labels:
    severity: CRITICAL
    name: ZombieCountExceedThreshold
    text: Zombie process count is greater than the threshold value
  
```

```
id: 3001214
source: CPRO
host: "{{\$labels.kubernetes_io_hostname}}"
eventType: 11
probableCause: 351
key: MOC015
annotations:
  summary: "{{\$labels.instance}}: Too many zombie processes
detected"
  description: "{{\$labels.instance}}: Zombie process count is
above 15"
- alert: NodeSysUpTimeCritical
  expr: node_time_seconds - node_boot_time_seconds < 5 * 24 * 3600
  for: 1m
  labels:
    severity: CRITICAL
    name: NodeSysUpTimeLessThreshold
    text: The number of seconds since last node reboot is less than
the threshold value
  id: 3001215
  source: CPRO
  host: "{{\$labels.kubernetes_io_hostname}}"
  eventType: 11
  probableCause: 351
  key: MOC016
  annotations:
    summary: "{{\$labels.instance}}: Node uptime is less than
threshold"
    description: "{{\$labels.instance}}: Node uptime is less than 5
days"
- alert: NetworkRetransMajor
  expr: node_netstat_Tcp_RetransSegs > 1500000
  for: 1m
  labels:
    severity: MAJOR
    name: NetworkRetransExceedThreshold
    text: The number of segments re-transmitted is greater than the
threshold value
  id: 3001216
  source: CPRO
  host: "{{\$labels.kubernetes_io_hostname}}"
  eventType: 11
  probableCause: 351
  key: MOC017
  annotations:
```

```
summary: "{{\$labels.instance}}: The number of re-trans segments  
is more than threshold"  
description: "{{\$labels.instance}}: The number of re-trans  
segments is more than 1500000"  
- name: alertmanager.rules  
  rules:  
    - alert: alertmanager_reload_failed  
      expr: alertmanager_config_last_reload_successful == 0  
      for: 10m  
      labels:  
        severity: Warning  
        name: Alert Manager Reload Failed  
        text: Alert Manager failed to reload the config for the last 10  
minutes.  
      id: 3001720  
      source: NCS  
      host: "{{\$labels.kubernetes_io_hostname}}"  
      eventType: 10  
      probableCause: 307  
      annotations:  
        description: Alert Manager failed to reload the config for the  
last 10 minutes.  
        summary: Alertmanager configuration reload has failed.  
    - alert: alertmanager_down  
      expr: up{component="alertmanager"} == 0  
      for: 2m  
      labels:  
        severity: Warning  
        name: Alert Manager Down  
        text: Alert Manager has been down for 2 minutes.  
      id: 3001722  
      source: NCS  
      host: "{{\$labels.kubernetes_io_hostname}}"  
      eventType: 10  
      probableCause: 347  
      annotations:  
        description: Alert Manager has been down for 2 minutes.  
        summary: Alertmanager is down.  
- name: etcd.rules  
  rules:  
    - alert: insufficient_members  
      expr: count(up{job="etcd_file_sd"} == 0) >  
(count(up{job="etcd_file_sd"}) / 2 - 1)  
      for: 3m  
      labels:
```

```
severity: Critical
name: Etcd Insufficient Members
text: ETCD cluster has insufficient members to perform auto
recover in the last 3 minutes.
id: 3001735
source: NCS
host: "{{\$labels.kubernetes_io_hostname}}"
eventType: 10
probableCause: 348
annotations:
  description: ETCD cluster has insufficient members to perform
auto recover in the last 3 minutes.
  summary: etcd cluster insufficient members
- alert: no_leader
  expr: etcd_server_has_leader{job="etcd_file_sd"} == 0
  for: 1m
  labels:
    severity: Minor
    name: Etcd No Leader
    text: There is no leader in the past minute.
    id: 3001734
    source: NCS
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 308
  annotations:
    description: etcd member {{ \$labels.instance }} has no leader
for 1m.
    summary: etcd member has no leader for 1m.
- alert: no_leader
  expr: etcd_server_has_leader{job="etcd_file_sd"} == 0
  for: 3m
  labels:
    severity: Major
    name: Etcd No Leader
    text: There is no leader in the past 3 minutes.
    id: 3001734
    source: NCS
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 308
  annotations:
    description: etcd member {{ \$labels.instance }} has no leader
for 3m.
    summary: etcd member has no leader for 3m.
```

```
- alert: no_leader
  expr: etcd_server_has_leader{job="etcd_file_sd"} == 0
  for: 5m
  labels:
    severity: Critical
    name: Etcd No Leader
    text: There is no leader in the past 5 minutes.
    id: 3001734
    source: NCS
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 11
    probableCause: 308
  annotations:
    description: etcd member {{ \$labels.instance }} has no leader
for 5m.
    summary: etcd member has no leader for 5m.
- alert: EtcdDown
  expr: up{job="etcd_file_sd"} == 0
  for: 2m
  labels:
    severity: Critical
    name: Etcd Down
    text: Prometheus could not fetch any metrics from etcd in the
last 2 minutes.
    id: 3001736
    source: NCS
    host: "{{\$labels.kubernetes_io_hostname}}"
    eventType: 10
    probableCause: 347
  annotations:
    description: Prometheus can't scape metrics from etcd(s).
    summary: Etcd down
- name: general.rules
  rules:
    - alert: TooManyOpenFileDescriptors
      expr: 100 * (process_open_fds{job="kubernetes-nodes"} /
process_max_fds{job="kubernetes-nodes"}) > 95
      for: 10m
      labels:
        severity: Critical
        name: Too Many Open File Descriptors
        text: The instance consumed more than 95% of the file/socket
descriptors in the last 10 minutes.
      id: 3001737
      source: NCS
```

```
host: "{$labels.kubernetes_io_hostname}"  
eventType: 10  
probableCause: 317  
annotations:  
    description: The instance {{ $labels.instance }} consumed more  
than 95% of the file/socket descriptors in the last 10 minutes.  
    summary: Too many open file descriptors  
- name: kube-apiserver.rules  
    rules:  
        - alert: K8SApiserverDown  
            expr: absent(up{job="kubernetes-apiservers"} == 1)  
            for: 5m  
            labels:  
                severity: Critical  
                name: K8S Api Server Down  
                text: Prometheus cannot scrape any metrics from Kubernetes API  
server for the last 5 minutes.  
            id: 3001738  
            source: NCS  
            host: "{$labels.kubernetes_io_hostname}"  
            eventType: 10  
            probableCause: 347  
            annotations:  
                description: Prometheus failed to scrape API server(s), or all  
API servers have  
                    disappeared from service discovery.  
                summary: API server unreachable  
- name: kube-state-metrics.rules  
    rules:  
        - alert: KubeDeploymentGenerationMismatch  
            expr: kube_deployment_status_observed_generation !=  
kube_deployment_metadata_generation  
            for: 15m  
            labels:  
                severity: Major  
                name: Kube Deployment Generation Mismatch  
                text: Deployment generation for does not match, this indicates  
that the Deployment has failed but has not been rolled back. Report after 15  
minutes.  
            id: 3001752  
            source: NCS  
            host: "{$labels.kubernetes_io_hostname}"  
            eventType: 10  
            probableCause: 307  
            annotations:
```

```

        description: Observed deployment generation does not match
expected one for
        deployment {{$labels.namespace}}/{{$labels.deployment}}
        summary: Deployment is outdated
- alert: KubeDeploymentReplicasMismatch
        expr: (kube_deployment_status_replicas_available != kube_deployment_spec_replicas)
        unless (kube_deployment_spec_paused == 1)
        for: 1h
        labels:
            severity: Major
            name: Kube Deployment Replicas Mismatch
            text: Deployment has not matched the expected number of replicas
for longer than an hour.
        id: 3001753
        source: NCS
        host:="{{$labels.kubernetes_io_hostname}}"
        eventType: 10
        probableCause: 307
        annotations:
            description: Replicas are not updated and available for
deployment {{$labels.namespace}}/{{$labels.deployment}} for 1 hour.
            summary: Deployment replicas are outdated
- alert: KubeDaemonSetNotScheduled
        expr: kube_daemonset_status_number_ready /
kube_daemonset_status_desired_number_scheduled
        * 100 < 100
        for: 10m
        labels:
            severity: Major
            name: Kube DaemonSet Not Scheduled
            text: Pods of DaemonSet are not scheduled. Report after 10
minutes.
        id: 3001756
        source: NCS
        host:="{{$labels.kubernetes_io_hostname}}"
        eventType: 10
        probableCause: 356
        annotations:
            description: Only {{$value}}% of desired pods scheduled and
ready for daemonset {{$labels.namespace}}/{{$labels.daemonset}}.
            summary: Daemonsets are not scheduled correctly
- alert: DaemonSetsMissScheduled
        expr: kube_daemonset_status_number_missscheduled > 0
        for: 10m

```

```

labels:
  severity: Major
  name: Kube DaemonSet Mis-scheduled
  text: Pods of DaemonSet are running where they are not supposed
to run. Report after 10 minutes.
  id: 3001757
  source: NCS
  host: "{$labels.kubernetes_io_hostname}"
  eventType: 10
  probableCause: 302
  annotations:
    description: Pods of daemonset {$labels.namespace}/
{$labels.daemonset} are running where they are not supposed to run.
    summary: Daemonsets are not scheduled correctly
  - alert: KubePodCrashLooping
    expr: increase(kube_pod_container_status_restarts_total[1h]) > 12
    for: 10m
  labels:
    severity: Critical
    name: Kube Pod Crash Looping
    text: Pod kept restarting within every 5 minutes in the past 1
hour.
    id: 3001750
    source: NCS
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 10
    probableCause: 307
    annotations:
      description: Pod {$labels.namespace}/{$labels.pod} is was
restarted {$value}
      times within the last hour
      summary: Pod is restarting frequently
  - alert: K8SHalfOfNodesAreNotReady
    expr:
count(kube_node_status_condition{condition="Ready",status="true"} == 0)
      > 0 and
(count(kube_node_status_condition{condition="Ready",status="true"} ==
0) /
count(kube_node_status_condition{condition="Ready",status="true"})) > 0.5
      for: 1m
  labels:
    severity: Critical
    name: K8S Half Of Nodes Are Not Ready
    text: More than half of the nodes are not ready for the last 1
minute.

```

```
id: 3001739
source: NCS
host: "{{\$labels.kubernetes_io_hostname}}"
eventType: 10
probableCause: 356
annotations:
  description: '{{ \$value }} Kubernetes nodes (more than 50%) are
in the NotReady
  state).'
  summary: Many Kubernetes nodes are Not Ready
- name: kubelet.rules
  rules:
    - alert: K8SHalfOfKubeletAreDown
      expr: absent(up{job="kubelet"} == 1) or (count(up{job="kubelet"} ==
0) / count(up{job="kubelet"}) > 0.5)
      for: 1h
  labels:
    severity: Critical
    name: K8S Half Of Kubelet Are Down
    text: More than half of the kubelets are down for the last 1
hour.

id: 3001740
source: NCS
host: "{{\$labels.kubernetes_io_hostname}}"
eventType: 10
probableCause: 347
annotations:
  description: Prometheus failed to scrape {{ \$value }}% of
kubelets, or all Kubelets
  have disappeared from service discovery.
  summary: Many Kubelets cannot be scraped
- name: node.rules
  rules:
    - alert: NodeRunOutOfDisk
      expr: (node_filesystem_size_bytes{device="rootfs"} -
node_filesystem_avail_bytes{device="rootfs"}) /
node_filesystem_size_bytes{device="rootfs"} == 1
  labels:
    service: k8s
    severity: Critical
    name: Node Run Out Of Disk
    text: Node ran out the disk space disk of /data0
  id: 3001741
  source: NCS
```

```
host: "{$labels.kubernetes_io_hostname}"  
eventType: 10  
probableCause: 151  
annotations:  
    description: '{{ $labels.kubernetes_io_hostname }} has run out  
the disk space of /data0.'  
    summary: Node ran out of disk space.  
- alert: NodeMemoryUnderPressure  
  expr:  
kube_node_status_condition{condition="MemoryPressure",status="true"} == 1  
  labels:  
    service: k8s  
    severity: Warning  
    name: Node Memory Under Pressure  
    text: Node memory is under pressure  
    id: 3001743  
    source: NCS  
    host: "{$labels.kubernetes_io_hostname}"  
    eventType: 10  
    probableCause: 332  
  annotations:  
    description: '{{ $labels.node }} is under memory pressure.'  
    summary: Node is under memory pressure.  
- alert: NodeDiskUnderPressure  
  expr:  
kube_node_status_condition{condition="DiskPressure",status="true"} == 1  
  labels:  
    service: k8s  
    severity: Warning  
    name: Node Disk Under Pressure  
    text: Node disk space is under pressure  
    id: 3001742  
    source: NCS  
    host: "{$labels.kubernetes_io_hostname}"  
    eventType: 10  
    probableCause: 151  
  annotations:  
    description: '{{ $labels.node }} disk space is under pressure.'  
    summary: Node is under disk pressure.  
- alert: NodeHighCpuUsage  
  expr: (100 - (avg by (kubernetes_io_hostname)  
(irate(node_cpu_seconds_total{job="prometheus-nodeexporter",mode="idle"}  
[5m])) * 100)) > 90  
  for: 30m  
  labels:
```

```

severity: Critical
name: Node High Cpu Usage
text: High CPU usage is detected, above 90% for the last 30
minutes.
id: 3001745
source: NCS
host: "{$labels.kubernetes_io_hostname}"
eventType: 10
probableCause: 310
annotations:
  summary: "{$labels.kubernetes_io_hostname}: High CPU usage
detected"
    description: "{$labels.kubernetes_io_hostname}: CPU usage is
above 90% (current value is: {{ $value }})"
  - alert: NodeHighMemoryUsage
    expr: (((node_memory_MemTotal_bytes-node_memory_MemFree_bytes-
node_memory_Cached_bytes-node_memory_Buffers_bytes)/
(node_memory_MemTotal_bytes)*100)) > 90
    for: 30m
  labels:
    severity: Critical
    name: Node High Memory Usage
    text: High memory usage is detected, above 90% for the last 30
minutes.
    id: 3001744
    source: NCS
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 10
    probableCause: 332
    annotations:
      summary: "{$labels.kubernetes_io_hostname}: High memory usage
detected"
        description: "{$labels.kubernetes_io_hostname}: Memory usage
is above 90% (current value is: {{ $value }})"
      - name: prometheus.rules
        rules:
          - alert: PrometheusReloadFailed
            expr: prometheus_config_last_reload_successful == 0
            for: 10m
        labels:
          severity: Warning
          name: Prometheus Reload Failed
          text: Prometheus failed to reload the config for the last 10
minutes.
          id: 3001721

```

```

source: NCS
host: "{$labels.kubernetes_io_hostname}"
eventType: 10
probableCause: 307
annotations:
  description: Reloading Prometheus' configuration has failed.
  summary: Prometheus configuration reload has failed
- name: pod-service-network.rules
  rules:
    - alert: PodServiceNetworkDown
      expr: (sum(up{job="calico_file_sd"}) == 1) < sum(kube_node_info))
or (sum(up{job="weave_file_sd"}) == 1) < sum(kube_node_info))
      for: 5m
  labels:
    severity: Warning
    name: Pod Service Network Down
    text: Pod Service Network has been down for 5 minutes.
    id: 3001724
    source: NCS
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 10
    probableCause: 347
  annotations:
    description: 'Pod Service Network has been down for more than 5
minutes'
    summary: 'Pod Service Network has been down'
- name: kube-dns.rules
  rules:
    - alert: KubednsDown
      expr: absent(up{k8s_app="kube-dns"} == 1)
      for: 5m
  labels:
    severity: Warning
    name: Kubedns Down
    text: Kubedns has been down for 5 minutes.
    id: 3001726
    source: NCS
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 10
    probableCause: 347
  annotations:
    description: 'Kube-DNS is down for more than 5 minutes'
    summary: 'Kube-DNS is down'
- name: heapster.rules
  rules:

```

```
- alert: HeapsterDown
  expr: absent(up{instance="heapster.kube-system.svc:80"} == 1)
  for: 5m
  labels:
    severity: Warning
    name: Heapster Down
    text: Heapster has been down for 5 minutes.
    id: 3001728
    source: NCS
    host: "{$labels.kubernetes_io_hostname}"
    eventType: 10
    probableCause: 347
  annotations:
    description: 'Heapster is down for more than 5 minutes'
    summary: 'Heapster is down'
- name: kube-dashboard.rules
  rules:
  - alert: KubeDashboardDown
    expr: absent(up{instance="kube-dashboard.kube-system.svc:81"} ==
1)
    for: 5m
    labels:
      severity: Warning
      name: Kube Dashboard Down
      text: Kube dashboard has been down for 5 minutes.
      id: 3001730
      source: NCS
      host: "{$labels.kubernetes_io_hostname}"
      eventType: 10
      probableCause: 347
    annotations:
      description: 'kube-dashboard is down for more than 5 minutes'
      summary: 'kube-dashboard is down'
- name: coredns.rules
  rules:
  - alert: CorednsDown
    expr: coredns_panic_count_total{k8s_app="kube-dns"} > 0
    for: 5m
    labels:
      severity: Warning
      name: Coredns Down
      text: Coredns on {{ $labels.kubernetes_io_hostname }} has been
down for 5 minutes.
      id: 3001733
      source: NCS
```

```
host: "{$labels.kubernetes_io_hostname}"  
eventType: 10  
probableCause: 347  
annotations:  
    description: 'Coredns on {{ $labels.kubernetes_io_hostname }} has been down for 5 minutes.'  
    summary: 'Coredns is down'  
- name: storage.rules  
    rules:  
        - alert: LocalVolumeProvisionerDown  
            expr: absent(up{instance="local-volume-provisioner.kube-system.svc:8080"} == 1)  
            for: 5m  
            labels:  
                severity: Warning  
                name: Local Volume Provisioner Down  
                text: Local storage provisioner is not running for the last 5 minutes.  
                id: 3001731  
                source: NCS  
            host: "{$labels.kubernetes_io_hostname}"  
            eventType: 10  
            probableCause: 347  
            annotations:  
                description: 'Local storage provisioner is not running for the last 5 minutes.'  
                summary: 'Local storage provisioner is not running.'  
                - alert: RookCephManagerNotHealth  
                    expr: kube_pod_status_phase{pod=~".*rook-ceph-mgr.*", phase!="Running"} == 1  
                    for: 5m  
                    labels:  
                        severity: Warning  
                        name: Rook Ceph Manager Not Health  
                        text: Rook ceph manager has been unhealthy for the last 5 minutes.  
                        id: 3001732  
                        source: NCS  
                    host: "{$labels.kubernetes_io_hostname}"  
                    eventType: 10  
                    probableCause: 347  
                    annotations:  
                        description: 'Rook ceph manager has been unhealthy for the last 5 minutes.'  
                        summary: 'Rook ceph manager has been unhealthy.'
```

```
- name: grafana.rules
  rules:
    - alert: GrafanaDown
      expr:
        kube_deployment_status_replicas_unavailable{deployment=~".*grafana.*"} > 0
        for: 10m
      labels:
        severity: Warning
        name: Grafana Down
        text: Grafana has been down for 10 minutes.
        id: 3001723
        source: NCS
        host: "{$labels.kubernetes_io_hostname}"
        eventType: 10
        probableCause: 347
      annotations:
        description: Prometheus can't scrape metrics from a Grafana pod
        for at least 10 minutes.
        summary: No Grafana instance is running, no dashboards
        available.
    - name: additional.rules
      rules:
        - alert: ControlNodesDown
          annotations:
            description: '{$labels.instance} is down'
            summary: '{$labels.instance} is down'
          expr: up{job="kubernetes-nodes", is_control="true"} == 0
          for: 1m
          labels:
            eventType: 10
            host: '{$labels.kubernetes_io_hostname}'
            id: 3001746
            name: Control Node Down
            probableCause: 356
            severity: Critical
            source: NCS
            text: '{$labels.instance} is down'
        - alert: WorkerNodesDown
          annotations:
            description: '{$labels.instance} is down'
            summary: '{$labels.instance} is down'
          expr: up{job="kubernetes-nodes", is_worker="true"} == 0
          for: 1m
          labels:
            eventType: 10
```

```
host: '$labels.kubernetes_io_hostname}'
id: 3001747
name: Worker Node Down
probableCause: 356
severity: Critical
source: NCS
text: '$labels.instance} is down'
- alert: EdgeNodesDown
  annotations:
    description: '$labels.instance} is down'
    summary: '$labels.instance} is down'
  expr: up{job="kubernetes-nodes", is_edge="true"} == 0
  for: 1m
  labels:
    eventType: 10
    host: '$labels.kubernetes_io_hostname}'
    id: 3001748
    name: Edge Node Down
    probableCause: 356
    severity: Critical
    source: NCS
    text: '$labels.instance} is down'
- alert: StorageNodesDown
  annotations:
    description: '$labels.instance} is down'
    summary: '$labels.instance} is down'
  expr: up{job="kubernetes-nodes", is_storage="true"} == 0
  for: 1m
  labels:
    eventType: 10
    host: '$labels.kubernetes_io_hostname}'
    id: 3001749
    name: Storage Node Down
    probableCause: 356
    severity: Critical
    source: NCS
    text: '$labels.instance} is down'
- alert: KubePodNotReady
  annotations:
    description: '$labels.namespace}/{$labels.pod} has been in
a non-ready state for longer than an hour.'
    summary: 'Kube Pod Not Ready'
  expr: kube_pod_status_phase{phase !~ "Running|Succeeded"} == 1
  for: 1h
  labels:
```

```

eventType: 10
host: '$labels.kubernetes_io_hostname}'
id: 3001751
name: Kube Pod Not Ready
probableCause: 307
severity: Critical
source: NCS
text: '$labels.namespace}/{$labels.pod} has been in a non-ready state for longer than an hour.'
- alert: PodScheduleFailure
  annotations:
    description: '$labels.namespace}/{$labels.pod} cannot be scheduled to node due to lack of resources, cluster node scale out might be needed.'
    summary: 'Pod Schedule Failure'
  expr: kube_pod_status_scheduled{condition != "true"} == 1
  for: 1m
  labels:
    eventType: 11
    host: '$labels.kubernetes_io_hostname}'
    id: 3001707
    name: Pod Schedule Failure
    probableCause: 343
    severity: Critical
    source: NCS
    text: 'Pod {$labels.namespace}/{$labels.pod} cannot be scheduled to node due to lack of resources, cluster node scale out might be needed.'
- alert: FluentdDown
  annotations:
    description: 'Fluentd has been down for 5 minutes.'
    summary: 'Fluentd Down'
  expr:
  (absent(kube_daemonset_status_desired_number_scheduled{daemonset=~".*fluentd.*"}) == 1) or (kube_daemonset_status_number_unavailable{daemonset=~".*fluentd.*"} > 0)
  for: 5m
  labels:
    eventType: 10
    host: '$labels.kubernetes_io_hostname}'
    id: 3001727
    name: Fluentd Down
    probableCause: 347
    severity: Warning
    source: NCS

```

```
text: 'Fluentd has been down for 5 minutes.'
- alert: MetricsServerDown
  annotations:
    description: 'Metrics Server has been down for 5 minutes.'
    summary: 'Metrics Server Down'
  expr:
absent(kube_deployment_status_replicas_available{deployment="metrics-server"} > 0)
  for: 5m
  labels:
    eventType: 10
    host: '$labels.kubernetes_io_hostname}'
    id: 3001729
    name: Metrics Server Down
    probableCause: 347
    severity: Warning
    source: NCS
    text: 'Metrics Server has been down for 5 minutes.'
- alert: KubeStatefulSetGenerationMismatch
  annotations:
    description: 'Observed statefulset generation does not match expected one for
      statefulset {{$labels.namespace}}/{{$labels.statefulset}}'
    summary: 'Kube StatefulSet Generation Mismatch'
    expr: kube_statefulset_status_observed_generation != kube_statefulset_metadata_generation
  for: 15m
  labels:
    eventType: 10
    host: '$labels.kubernetes_io_hostname}'
    id: 3001754
    name: Kube StatefulSet Generation Mismatch
    probableCause: 307
    severity: Major
    source: NCS
    text: 'StatefulSet {{$labels.namespace}}/{{$labels.statefulset}} generation for does not match, this indicates that the StatefulSet has failed but has not been rolled back. Report after 15 minutes.'
- alert: KubeStatefulSetReplicasMismatch
  annotations:
    description: 'StatefulSet {{$labels.namespace}}/{{$labels.statefulset}} has not matched the expected number of replicas for longer than 15 minutes.'
    summary: 'Kube StatefulSet Replicas Mismatch'
```

```
expr: kube_statefulset_status_replicas_ready !=  
kube_statefulset_status_replicas  
for: 15m  
labels:  
  eventType: 10  
  host: '$labels.kubernetes_io_hostname}'  
  id: 3001755  
  name: Kube StatefulSet Replicas Mismatch  
  probableCause: 307  
  severity: Major  
  source: NCS  
  text: 'StatefulSet {$labels.namespace}/{$labels.statefulset}  
has not matched the expected number of replicas for longer than 15 minutes.'  
- alert: KubeClientCertificateExpiration  
  annotations:  
    description: 'A client certificate used to authenticate to the  
apiserver is expiring in less than 24 hours.'  
    summary: 'Kube Client Certificate Expiration'  
  expr:  
    apiserver_client_certificate_expiration_seconds_count{job="kubernetes-  
apiservers"} > 0 and histogram_quantile(0.01, sum by (job, le)  
(rate(apiserver_client_certificate_expiration_seconds_bucket{job="kubernetes-  
apiservers"}[5m])) < 86400  
    for: 5m  
  labels:  
    eventType: 10  
    host: '$labels.kubernetes_io_hostname}'  
    id: 3001762  
    name: Kube Client Certificate Expiration  
    probableCause: 307  
    severity: Critical  
    source: NCS  
    text: 'A client certificate used to authenticate to the  
apiserver is expiring in less than 24 hours.'  
- alert: KubeJobFailed  
  annotations:  
    description: 'Kube job {$labels.namespace}/  
{$labels.job_name} failed to complete. Report after 1 hour.'  
    summary: 'Kube Job Failed'  
  expr: kube_job_status_failed > 0  
  for: 1h  
  labels:  
    eventType: 10  
    host: '$labels.kubernetes_io_hostname}'  
    id: 3001758
```

```
name: Kube Job Failed
probableCause: 302
severity: Major
source: NCS
text: 'Kube job {{\$labels.namespace}}/{{\$labels.job_name}} failed to complete. Report after 1 hour.'
- alert: KubeCronJobRunning
  annotations:
    description: 'Kube Cronjob {{ \$labels.namespace }}/{{ \$labels.cronjob }} is taking more than 1h to complete.'
    summary: 'Kube CronJob Running'
  expr: time() - kube_cronjob_next_schedule_time > 3600
  for: 1h
  labels:
    eventType: 11
    host: '{{\$labels.kubernetes_io_hostname}}'
    id: 3001759
    name: Kube Cronjob Running Too Long
    probableCause: 351
    severity: Major
    source: NCS
    text: 'Kube Cronjob {{ \$labels.namespace }}/{{ \$labels.cronjob }} is taking more than 1h to complete.'
- alert: KubePersistentVolumeErrors
  annotations:
    description: 'The persistent volume {{ \$labels.persistentvolume }} is Pending or Failed for 5 minutes.'
    summary: 'Kube Persistent Volume Errors'
  expr: kube_persistentvolume_status{phase=~"Failed|Pending"} > 0
  for: 5m
  labels:
    eventType: 10
    host: '{{\$labels.kubernetes_io_hostname}}'
    id: 3001761
    name: Kube Persistent Volume Errors
    probableCause: 151
    severity: Critical
    source: NCS
    text: 'The persistent volume {{ \$labels.persistentvolume }} is Pending or Failed for 5 minutes.'
- alert: KubePersistentVolumeUsagePressure
  annotations:
```

```
        description: 'The PersistentVolume {{ $labels.namespace }}/{{ $labels.persistentvolumeclaim }} in {{ $labels.namespace }} is less than 3% free.'
        summary: 'Kube Persistent Volume Usage Pressure'
        expr: kubelet_volume_stats_available_bytes{job="kubernetes-nodes"} / kubelet_volume_stats_capacity_bytes{job="kubernetes-nodes"} * 100 < 3
        for: 5m
        labels:
          eventType: 10
          host: '$labels.kubernetes_io_hostname}'
          id: 3001760
          name: Kube Persistent Volume Usage Pressure
          probableCause: 151
          severity: Critical
          source: NCS
          text: 'The PersistentVolume {{ $labels.namespace }}/{{ $labels.persistentvolumeclaim }} in Namespace is less than 3% free.'
prometheus.yml:
  scrape_configs:
    - job_name: prometheus
      static_configs:
        - targets:
          - localhost:9090
      honor_labels: true

      kubernetes_sd_configs:
        - role: endpoints

      relabel_configs:
        - source_labels:
[__meta_kubernetes_service_annotation_prometheus_io_probe]
          action: keep
          regex: prometheus
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
          target_label: namespace

# A scrape configuration for running Prometheus on a Kubernetes cluster.
# This uses separate scrape configs for cluster components (i.e. API server, node)
# and services to allow each to use different authentication configs.
#
```

```
# Kubernetes labels will be added as Prometheus labels on metrics
via the
    # `labelmap` relabeling action.

    # Scrape config for API servers.
    #
    # Kubernetes exposes API servers as endpoints to the default/
kubernetes
        # service so this uses `endpoints` role and uses relabelling to only
keep
            # the endpoints associated with the default/kubernetes service using
the
                # default named port `https`. This works for single API server
deployments as
                    # well as HA API server deployments.
                    - job_name: 'kubernetes-apiservers'

                    kubernetes_sd_configs:
                        - role: endpoints

                        # Default to scraping over https. If required, just disable this
or change to
                            # `http`.
                            scheme: https

                        # This TLS & bearer token file config is used to connect to the
actual scrape
                            # endpoints for cluster components. This is separate to discovery
auth
                                # configuration because discovery & scraping are two separate
concerns in
                                    # Prometheus. The discovery auth config is automatic if Prometheus
runs inside
                                        # the cluster. Otherwise, more config options have to be provided
within the
                                            # <kubernetes_sd_config>.
                                            tls_config:
                                                ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
                                                # If your node certificates are self-signed or use a different
CA to the
                                                    # master CA, then disable certificate verification below. Note
that
                                                        # certificate verification is an integral part of a secure
infrastructure
```

```

        # so this should only be disabled in a controlled environment.

You can
        # disable certificate verification by uncommenting the line
below.
        #
        insecure_skip_verify: true
bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/
token

        # Keep only the default/kubernetes service endpoints for the https
port. This
        # will add targets for each API server which Kubernetes adds an
endpoint to
        # the default/kubernetes service.
        relabel_configs:
            - source_labels: [__meta_kubernetes_namespace,
__meta_kubernetes_service_name, __meta_kubernetes_endpoint_port_name]
                action: keep
                regex: default;kubernetes;https
            - source_labels: [__meta_kubernetes_namespace]
                action: replace
                target_label: namespace

            - job_name: 'kubernetes-nodes'

        # Default to scraping over https. If required, just disable this
or change to
        # `http`.
        scheme: https

        # This TLS & bearer token file config is used to connect to the
actual scrape
        # endpoints for cluster components. This is separate to discovery
auth
        # configuration because discovery & scraping are two separate
concerns in
        # Prometheus. The discovery auth config is automatic if Prometheus
runs inside
        # the cluster. Otherwise, more config options have to be provided
within the
        # <kubernetes_sd_config>.
        tls_config:
            ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
            # If your node certificates are self-signed or use a different
CA to the

```

```
# master CA, then disable certificate verification below. Note
that
# certificate verification is an integral part of a secure
infrastructure
# so this should only be disabled in a controlled environment.
You can
# disable certificate verification by uncommenting the line
below.

#
insecure_skip_verify: true
bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/
token

kubernetes_sd_configs:
- role: node

relabel_configs:
- action: labelmap
  regex: __meta_kubernetes_node_label_(.+)
- target_label: __address__
  replacement: kubernetes.default.svc:443
- source_labels: [__meta_kubernetes_node_name]
  regex: (.)
  target_label: __metrics_path__
  replacement: /api/v1/nodes/${1}/proxy/metrics
- source_labels: [__meta_kubernetes_namespace]
  action: replace
  target_label: namespace

- job_name: 'kubernetes-nodes-cadvisor'

  # Default to scraping over https. If required, just disable this
or change to
  # `http`.
  scheme: https

  # This TLS & bearer token file config is used to connect to the
actual scrape
  # endpoints for cluster components. This is separate to discovery
auth
  # configuration because discovery & scraping are two separate
concerns in
  # Prometheus. The discovery auth config is automatic if Prometheus
runs inside
```

```
# the cluster. Otherwise, more config options have to be provided
within the
# <kubernetes_sd_config>.
tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    # If your node certificates are self-signed or use a different
CA to the
        # master CA, then disable certificate verification below. Note
that
        # certificate verification is an integral part of a secure
infrastructure
        # so this should only be disabled in a controlled environment.
You can
    # disable certificate verification by uncommenting the line
below.
    #
    insecure_skip_verify: true
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/
token

kubernetes_sd_configs:
    - role: node

    # This configuration will work only on kubelet 1.7.3+
    # As the scrape endpoints for cAdvisor have changed
    # if you are using older version you need to change the
replacement to
    # replacement: /api/v1/nodes/${1}:4194/proxy/metrics
    # more info here https://github.com/coreos/prometheus-operator/
issues/633
    relabel_configs:
        - action: labelmap
            regex: __meta_kubernetes_node_label_(.+)
        - target_label: __address__
            replacement: kubernetes.default.svc:443
        - source_labels: [__meta_kubernetes_node_name]
            regex: (.)
            target_label: __metrics_path__
            replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor
        - source_labels: [__meta_kubernetes_namespace]
            action: replace
            target_label: namespace

    # Scrape config for service endpoints.
    #
```

```
# The relabeling allows the actual service scrape endpoint to be
configured
# via the following annotations:
#
# * `prometheus.io/scrape`: Only scrape services that have a value
of `true`
# * `prometheus.io/scheme`: If the metrics endpoint is secured then
you will need
# to set this to `https` & most likely set the `tls_config` of the
scrape config.
# * `prometheus.io/path`: If the metrics path is not `/metrics`
override this.
# * `prometheus.io/port`: If the metrics are exposed on a different
port to the
# service then set this appropriately.
- job_name: 'kubernetes-service-endpoints'

  kubernetes_sd_configs:
    - role: endpoints

      relabel_configs:
        - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_scrape ]
          action: keep
          regex: true
          - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_scheme ]
          action: replace
          target_label: __scheme__
          regex: (https?)
          - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_path ]
          action: replace
          target_label: __metrics_path__
          regex: (.)
          - source_labels: [ __address__,
__meta_kubernetes_service_annotation_prometheus_io_port ]
          action: replace
          target_label: __address__
          regex: ([^:])(?:::\d+)?;(\d+)
          replacement: $1:$2
          - action: labelmap
            regex: __meta_kubernetes_service_label_(.+)
          - source_labels: [ __meta_kubernetes_namespace ]
            action: replace
```

```
        target_label: kubernetes_namespace
      - source_labels: [__meta_kubernetes_service_name]
        action: replace
        target_label: kubernetes_name
      - source_labels: [__meta_kubernetes_pod_node_name]
        target_label: kubernetes_io_hostname
        action: replace

    - job_name: 'prometheus-pushgateway'
      honor_labels: true

      kubernetes_sd_configs:
        - role: service

      relabel_configs:
        - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_probe ]
          action: keep
          regex: pushgateway
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
          target_label: namespace

      # Example scrape config for probing services via the Blackbox
      Exporter.
      #
      # The relabeling allows the actual service scrape endpoint to be
      configured
      # via the following annotations:
      #
      # * `prometheus.io/probe`: Only probe services that have a value of
      `true`
      #
      #
    - job_name: 'prometheus-nodeexporter'
      honor_labels: true

      kubernetes_sd_configs:
        - role: endpoints

      relabel_configs:
        - source_labels:
[ __meta_kubernetes_service_annotation_prometheus_io_probe ]
          action: keep
          regex: node-exporter
```

```
- source_labels: [__meta_kubernetes_namespace]
  action: replace
  target_label: namespace
- source_labels: [__meta_kubernetes_pod_node_name]
  target_label: kubernetes_io_hostname
  action: replace

# Example scrape config for pods
#
# The relabeling allows the actual pod scrape endpoint to be
configured via the
# following annotations:
#
# * `prometheus.io/scrape`: Only scrape pods that have a value of
`true`
# * `prometheus.io/path`: If the metrics path is not `/metrics`,
override this.
# * `prometheus.io/port`: Scrape the pod on the indicated port
instead of the default of `9102`.
- job_name: 'kubernetes-pods'

  kubernetes_sd_configs:
    - role: pod

    relabel_configs:
      - source_labels:
[ __meta_kubernetes_pod_annotation_prometheus_io_scrape ]
        action: keep
        regex: true
      - source_labels:
[ __meta_kubernetes_pod_annotation_prometheus_io_path ]
        action: replace
        target_label: __metrics_path__
        regex: (.)
      - source_labels: [ __address__,
__meta_kubernetes_pod_annotation_prometheus_io_port ]
        action: replace
        regex: ([^:]+)(?::\d+)?;(\d+)
        replacement: $1:$2
        target_label: __address__
      - action: labelmap
        regex: __meta_kubernetes_pod_label_(.+)
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
```

```
target_label: namespace
- source_labels: [__meta_kubernetes_pod_name]
  action: replace
  target_label: kubernetes_pod_name
- source_labels: [__meta_kubernetes_pod_node_name]
  action: replace
  target_label: instance
- job_name: kubelet
  scheme: https
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/
token
  kubernetes_sd_configs:
    - role: node
  relabel_configs:
    - action: labelmap
      regex: __meta_kubernetes_node_label_(.+)
    - target_label: __address__
      replacement: kubernetes.default.svc:443
    - source_labels: [__meta_kubernetes_node_name]
      regex: (.+)
      target_label: __metrics_path__
      replacement: /api/v1/nodes/${1}/proxy/metrics
    - job_name: kubernetes-service
      honor_labels: true
      scrape_interval: 1m
      scrape_timeout: 10s
      metrics_path: /metrics
      scheme: http
      kubernetes_sd_configs:
        - api_server: null
          role: service
          namespaces:
            names: []
  relabel_configs:
    - source_labels: [__meta_kubernetes_service_name]
      separator: ;
      regex: heapster|kube-dashboard|local-volume-provisioner
      replacement: $1
      action: keep
    - job_name: etcd_file_sd
      scrape_interval: 1m
      scrape_timeout: 10s
      metrics_path: /metrics
```

```
file_sd_configs:
  - files:
      - /etc/etcd/targets_etcd.json
    scheme: https
    tls_config:
      ca_file: /etc/etcd/ssl/ca.pem
      cert_file: /etc/etcd/ssl/etcd-client.pem
      key_file: /etc/etcd/ssl/etcd-client-key.pem
      insecure_skip_verify: true
  - job_name: calico_file_sd
    kubernetes_sd_configs:
      - role: node
    relabel_configs:
      - action: labelmap
        regex: __meta_kubernetes_node_label_(.+)
      - action: replace
        regex: (.*)
        replacement: $1:9091
        source_labels:
          - __meta_kubernetes_node_address_InternalIP
          target_label: __address__
      - action: replace
        regex: (.*)
        replacement: $1
        source_labels:
          - job
          target_label: component
  - job_name: weave_file_sd
    scrape_interval: 1m
    scrape_timeout: 10s
    metrics_path: /metrics
    file_sd_configs:
      - files:
          - /etc/etcd/targets_weave.json
restserver:
  nodeSelector:
    is_btel_worker: "true"
  tolerations:
    - key: 'is_btel'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'

grafana:
  replicas: 2
```

```

HA:
  enabled: true
nodeSelector:
  is_btel_worker: "true"
tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
persistence:
  storageClassName: "cinder-az-nova"

gen3gppxml:
  service:
    name: gen3gppxml
    serviceType: NodePort
    sftpPort: 22
    sftpNodePort: 30022
    restHttpPort: 8080
    restHttpsPort: 8081
  sftp:
    user: sftpuser
    authentication:
      passwd:
        enabled: true
        passwd: zBgDcqRJa3vLCBUCxPta2/OHULfwUgLq63BJnOhzH
+ZE02RjTDi2xm2qk0XtHquu0HgXshAloQrgs xv5/
gS5cEqS8AOL1RX3VXKehMqqroJQ5kRRX5RfFYdGnkPm
+8CFh2xuV6RBjHQsWwa221nD1b821H1c6LRmhX8QzxSNF3npJIhSQOsYFFcWW1ij7hxm4cPiKULBLip0mayvcqP5V
+s7XrDWHt1OZtyqJA==
      rsaKey: |+
        -----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQDnMVbkaPLnj63S
xd1bIsIf4Jz8zKqj4bW0b+ob0Qzn1RIR3Rn4JIyyDlob5GeRka6/mSYvsxEah6YG
1tWGbwHfQizF/tcEKGhD5uek15+FFL87IN7VmtQwdKWAWXNzzR6sUzaP1Io2UPzY
i37qqe5koltfu7VLplhD1bTCse01WUBZ9EZAzLbV1CjuvLXJAHP1/NsME4hleK3G
tF0pLkut5lwoSliyiQwhjaqz6dbQ+Du3qRDu5mJqnyjy7GjziGTSbMbBvQIfjdBz
thsAuVIL2zLGhqFy5/sH3X15f1Ngth4tyvja3DWOoS6Tm28xu3e9JgImPONpV0Ec
SYXGEh/3AgMBAEAEggEBAJ5dMcQsD5eG1+61ErfGy2rClUale43eonUC5D3Zinln
p5bcUVmkz6t4IUL34/bVvRv17S82E5v9M5igskeTBh+X+UQd4ClnSZGGQ6pvD+p
Ip8CGSIAeLDodixBIFCGhShMivHjCPnD6C78/ucHmADfJfAhvEIVpAhzbuqerm8R
smMOxrQ/bMgvrvwtFWK0iZEhLRWV1mJEg3qFjoH3Qf4PIIr0U2X8K4z/C7FY0ShmE
6FOeOgmKIdKb2wiMsDH63eqmfdFMDyRDexmR1/lpOo9SNmKgVdLUjsgmL1x5vA
1HNJqyyp9KBrYO+xyYDx6NIuYfeeZxHHH4Bia8DaeECgYEA/hshxQkMEz4OpTii
dv1Y3R0e4CKm5+j7Mu j3EmFSBnfa7tQwJA AugOIxHnNositq sRa6nD4pqby2hnGbX

```

```
7J79Xc/NsZx+i6u8dcxRqf2pRgTcz5wpOPg00qh2s4AWUk7fe3izVEO+DXgC9eC1
0MdZXe+aMLS+qbF7JiWhZiTbAnUCgYEA6Op8aJxszih6dLVHdKbxjgKaPyfz/pYJ
LVRiv09wcECnRoVxgkDUyRxNTqXvBX2iyh0ls4En4AFDdAXeozx3pQLzQEptQSFu
+mIba8smj3EWSbrfQlICxfrX6pZ7H6weUlw78WEYuGsbf0g6kUi38t1+vzqzW66E
15HehyPGczsCgYEA21Sf+WqtuY4/Je2Ujgz73x9bnytJ1bML4A219X97RYn+t1b/
lp1SVnz+8yx+dfv55sux94AKDpzcehuBhjRVjbvoV/40XGqH6UXK12fTzFQ2JB0u
17XmIk6250kQu0rJK9IrN/ig+w8zvveBK1xpXu0Ju7DHQcGoAm1Iutn1EGECgYEA
pwzmxbDLaOEnUuVhpozF2vCK6JkjKok5c8V+PXAgWKJGVxIGY0FUXBlsaOgiEnIV
e5Gzj//GNacyVdiP4cvQ1A0VohYV6BFYGUSchifKND8LmT3qOorPameKzLm4/PJZ
nQCCDZFyRXKseF2nA+R/6jRqX4Uedrcoaf04+PvzrVkcgYEAznugyzxW7+ng9BC2
50j1ZIr20iP+ySqJnuchSntxaHyi1MYVbKIyXIvh6qPiIqBM9Bge3EOjeqz8p+0t
4Lo6H9V4S2Fu7VcpPVgG3YYJOuxcEgt26soc/Sf1Jz1dBp4JWqI3T0Rgse3CvKv
hm+q9n+CS/Lj2hTxMRBoxuyCgi4=
-----END PRIVATE KEY-----
configs:
  Gen3GPPXML: '{
    "measCollecFile" :
    {
      "xmlns" : "http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec",
      "xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-instance",
      "xmlns:schemaLocation" : "http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec http://www.3gpp.org/ftp/specs/
archive/32_series/32.435#measCollec",
      "fileHeader" :
      {
        "fileFormatVersion" : "32.435 V9.1.0",
        "vendorName" : "NOKIA",
        "dnPrefix" : "dn",
        "elementType" : "eletype"
      }
    },
    "REST" :
    {
      "HTTP" :
      {
        "enable":true,
        "httpPort":8080
      },
      "HTTPS" :
      {
        "enable":false,
        "httpsPort" : 8081,
        "verifyClient" : false,
        "securityFiles" :

```

```

    {
        "caCert": "/etc/gen3gppxml/secret/rest/ca.crt",
        "cert": "/etc/gen3gppxml/secret/rest/gen3gppxml.crt",
        "key": "/etc/gen3gppxml/secret/rest/gen3gppxml.key"
    }
},
"SSO":
{
    "enable": false,
    "validationUrl": "http://localhost:8280/auth/realm/myrealm/protocol/openid-connect/userinfo",
    "HTTPS": false,
    "securityFiles" :
    {
        "caCert": "/etc/gen3gppxml/secret/sso/ca.crt",
        "cert": "/etc/gen3gppxml/secret/sso/gen3gppxml.crt",
        "key": "/etc/gen3gppxml/secret/sso/gen3gppxml.key"
    }
},
"prometheus" :
{
    "url" : "http://localhost:9090/api/v1",
    "HTTPS" : false,
    "securityFiles" :
    {
        "caCert": "/etc/gen3gppxml/secret/prometheus/ca.crt",
        "cert": "/etc/gen3gppxml/secret/prometheus/gen3gppxml.crt",
        "key": "/etc/gen3gppxml/secret/prometheus/gen3gppxml.key"
    },
    "authentication" :
    {
        "enable": "false",
        "username" : "root",
        "password" : "P8KaOyx504Vpbv+hNFUGHicuO9TKArmVhGXAlCIBut2R7i/DDd0Hx2W60G2Q/XjJ8/NNxb1l1jPSwEr0IrxFrHB30uvvH7vOhA2BhuMBFgYb4n/+uKJaVqFS08s7dH8c0tKPbPGxhei9m64Nb/fMhCnacD2yn5p04w173Z1Y7fPuJVbzV1luXhuaIG42UqCntZTKjxhqEFczLqKcb7JAVABBMjiRmb91iYd44tKP73I",
        "key": "/etc/gen3gppxml/secret/prometheus/auth.key"
    }
},
"plugins" :
{
    "vesAgent" :
    {

```

```

        "enable" : "false",
        "connectString" : "127.0.0.1:9991",
        "appName" : "csf"
    }
},
"postFix" : "gen3gppxmlDefaultPostFix",
"logLevel" : "INFO",
"reportLocation" : "/var/3gppxml/",
"expiredDays" : "14",
"threadPoolSize" : 20,
"defaultGranPeriod" : [ "5m" ],
"defaultRule" : "avg",
"defaultValueType" : "integer",
"orientation" : "vertical",
"hideNonExistentMetrics" : "true",
"invalidValue" : "0",
"FastPass" : {
    "enable" : "true",
    "maxAttributeValueEachLength" : 80,
    "maxAttributeTypeEachLength" : 6,
    "maxAttributeTypeNum" : 10,
    "replaceInvalidCharInMeasObjLdnWith" : "_"
},
"measGroups" :
[
{
    "measInfoId" : "M1",
    "metrics" :
    [
        { "metric": "alertmanager_config_last_reload_successful" }
    ]
},
{
    "measInfoId" : "M2",
    "metrics" :
    [
        { "metric": "etcd_server_has_leader" },
        { "metric": "etcd_debugging_mvcc_db_compaction_keys_total" },
        { "metric": "etcd_debugging_mvcc_db_total_size_in_bytes" },
        { "metric": "etcd_debugging_mvcc_delete_total" },
        { "metric": "etcd_debugging_mvcc_events_total" },
        { "metric": "etcd_debugging_mvcc_keys_total" },
        { "metric": "etcd_debugging_mvcc_pending_events_total" },
        { "metric": "etcd_debugging_mvcc_put_total" },
        { "metric": "etcd_debugging_mvcc_range_total" },
    ]
}
]
}

```

```

        {
            "metric": "etcd_debugging_mvcc_slow_watcher_total" },
            {
            "metric": "etcd_debugging_mvcc_txn_total" },
            {
            "metric": "etcd_debugging_mvcc_watch_stream_total" },
            {
            "metric": "etcd_debugging_mvcc_watcher_total" },
            {
            "metric": "etcd_debugging_server_lease_expired_total" },
            {
            "metric": "etcd_debugging_store_expires_total" },
            {
            "metric": "etcd_debugging_store_reads_total",
                "rule" : "max",
                "label" :
                {
                    "name" : "method",
                    "value" : [ "get", "getRecursive" ]
                }
            },
            {
            "metric": "etcd_debugging_store_watch_requests_total" },
            {
            "metric": "etcd_debugging_store_watchers" },
            {
            "metric": "etcd_debugging_store_writes_total",
                "rule" : "max",
                "label" :
                {
                    "name" : "method",
                    "value" : [ "set" ]
                }
            },
        },
        {
            "metric": "etcd_disk_backend_commit_duration_seconds_count" },
            {
            "metric": "etcd_disk_backend_defrag_duration_seconds_count" },
            {
            "metric": "etcd_disk_backend_snapshot_duration_seconds_count",
                {
                "metric": "etcd_disk_wal_fsync_duration_seconds_count" },
                {
                "metric": "etcd_grpc_proxy_cache_hits_total" },
                {
                "metric": "etcd_grpc_proxy_cache_keys_total" },
                {
                "metric": "etcd_grpc_proxy_cache_misses_total" },
                {
                "metric": "etcd_grpc_proxy_events_coalescing_total" },
                {
                "metric": "etcd_grpc_proxy_watchers_coalescing_total" },
                {
                "metric": "etcd_mvcc_db_total_size_in_bytes" },
                {
                "metric": "etcd_mvcc_db_total_size_in_use_in_bytes" },
                {
                "metric": "etcd_mvcc_hash_duration_seconds_count" },
                {
                "metric": "etcd_mvcc_hash_rev_duration_seconds_count" },
            },
            {
            "metric": "etcd_network_client_grpc_received_bytes_total",
                {
                "metric": "etcd_network_client_grpc_sent_bytes_total" },
                {
                "metric": "etcd_server_go_version" },
                {
                "metric": "etcd_server_health_failures" },
            }
        }
    }
}

```

```
{ "metric": "etcd_server_health_success" },
{ "metric": "etcd_server_heartbeat_send_failures_total" },
{ "metric": "etcd_server_id" },
{ "metric": "etcd_server_is_leader" },
{ "metric": "etcd_server_leader_changes_seen_total" },
{ "metric": "etcd_server_proposals_applied_total" },
{ "metric": "etcd_server_proposals_committed_total" },
{ "metric": "etcd_server_proposals_failed_total" },
{ "metric": "etcd_server_proposals_pending" },
{ "metric": "etcd_server_quota_backend_bytes" },
{ "metric": "etcd_server_read_indexes_failed_total" },
{ "metric": "etcd_server_slow_apply_total" },
{ "metric": "etcd_server_slow_read_indexes_total" },
{ "metric": "etcd_server_version" },
{ "metric": "etcd_snap_db_fsync_duration_seconds_count" },

{ "metric": "etcd_snap_db_save_total_duration_seconds_count" }
]
},
{
  "measInfoId" : "M3",
  "metrics" :
  [
    {
      "metric": "process_open_fds" },
      "metric": "process_max_fds" }
    ]
},
{
  "measInfoId" : "M4",
  "dnLabels" : [ "DPLMN" ],
  "labelRename": { "DPLMN": "deployment" },
  "metrics" :
  [
    {
      "metric": "kube_deployment_status_observed_generation" },
      "metric": "kube_deployment_metadata_generation" },
      "metric": "kube_deployment_status_replicas_available" },
      "metric": "kube_deployment_spec_replicas" },
      "metric": "kube_deployment_spec_paused" },
      "metric": "kube_deployment_status_replicas_unavailable" }
    ]
},
{
  "measInfoId" : "M5",
  "dnLabels" : [ "DAEMST" ],
  "labelRename": { "DAEMST": "daemonset" },
```

```
"metrics" :  
[  
    { "metric": "kube_daemonset_status_number_ready" },  
  
    { "metric": "kube_daemonset_status_desired_number_scheduled" },  
        { "metric": "kube_daemonset_status_number_misscheduled" },  
        { "metric": "kube_daemonset_status_number_unavailable" }  
    ]  
,  
{  
    "measInfoId" : "M6",  
    "dnLabels" : [ "POD" ],  
    "labelRename": { "POD": "pod" },  
    "metrics" :  
    [  
        { "metric": "kube_pod_container_status_restarts_total" },  
        { "metric": "kube_pod_status_phase" },  
        { "metric": "kube_pod_status_scheduled" }  
    ]  
,  
{  
    "measInfoId" : "M7",  
    "dnLabels" : [ "CDTN" ],  
    "labelRename": { "CDTN": "condition" },  
    "metrics" :  
    [  
        { "metric": "kube_node_status_condition" }  
    ]  
,  
{  
    "measInfoId" : "M8",  
    "metrics" :  
    [  
  
        { "metric": "prometheus_config_last_reload_successful" }  
    ]  
,  
{  
    "measInfoId" : "M9",  
    "metrics" :  
    [  
        { "metric": "coredns_panic_count_total" }  
    ]  
,  
{
```

```
"measInfoId" : "M10",
"dnLabels" : [ "STSET" ],
"labelRename": { "STSET": "statefulset" },
"metrics" :
[
    { "metric": "kube_statefulset_status_observed_generation" },
    { "metric": "kube_statefulset_metadata_generation" },
    { "metric": "kube_statefulset_status_replicas_ready" },
    { "metric": "kube_statefulset_status_replicas" }
]
},
{
"measInfoId" : "M11",
"metrics" :
[
    { "metric": "apiserver_audit_event_total" },
    { "metric": "apiserver_audit_requests_rejected_total" },
    { "metric": "get_token_count" },
    { "metric": "get_token_fail_count" }
]
},
{
"measInfoId" : "M12",
"metrics" :
[
    { "metric": "kube_job_status_failed" },
    { "metric": "kube_cronjob_next_schedule_time" }
]
},
{
"measInfoId" : "M13",
"dnLabels" : [ "PVC" ],
"labelRename": { "PVC": "persistentvolumeclaim" },
"metrics" :
[
    { "metric": "kubelet_volume_stats_available_bytes" },
    { "metric": "kubelet_volume_stats_capacity_bytes" }
]
},
{
"measInfoId" : "M14",
"metrics" :
[
    { "metric": "node_boot_time_seconds" }
]
```

```

} ,
{
    "measInfoId" : "M15",
    "metrics" :
    [
        {
            "metric": "node_ipvs_incoming_bytes_total", "rule": "max" },
            {"metric": "node_ipvs_outgoing_bytes_total", "rule": "max" }
        ]
},
{
    "measInfoId" : "M16",
    "dnLabels" : [ "CPU" ],
    "labelRename": { "CPU": "cpu" },
    "metrics" :
    [
        {
            "metric" : "node_cpu_seconds_total",
            "rule" : "max",
            "label" :
            {
                "name" : "mode",
                "value" : [ "idle", "iowait", "irq", "nice",
"softirq", "steal", "system", "user" ]
            }
        }
    ]
},
{
    "measInfoId" : "M17",
    "dnLabels" : [ "DISK" ],
    "labelRename": { "DISK": "device" },
    "metrics" :
    [
        {
            "metric": "node_disk_read_bytes_total", "rule": "max" },
            {"metric": "node_disk_written_bytes_total", "rule": "max" },
            {"metric": "node_disk_io_now" },
            {"metric": "node_disk_io_time_seconds_total" }
        ]
},
{
    "measInfoId" : "M18",
    "metrics" :
    [
        { "metric": "node_entropy_available_bits" }
    ]
}

```

```

} ,
{
  "measInfoId" : "M19",
  "metrics" :
  [
    { "metric": "node_load1" },
    { "metric": "node_load5" },
    { "metric": "node_load15" }
  ]
},
{
  "measInfoId" : "M20",
  "metrics" :
  [
    { "metric": "node_memory_AnonPages_bytes" },
    { "metric": "node_memory_Buffers_bytes" },
    { "metric": "node_memory_Cached_bytes" },
    { "metric": "node_memory_Dirty_bytes" },
    { "metric": "node_memory_HardwareCorrupted_bytes" },
    { "metric": "node_memory_Mapped_bytes" },
    { "metric": "node_memory_MemFree_bytes" },
    { "metric": "node_memory_MemTotal_bytes" },
    { "metric": "node_memory_Shmem_bytes" },
    { "metric": "node_memory_SwapFree_bytes" },
    { "metric": "node_memory_SwapTotal_bytes" },
    { "metric": "node_memory_Writeback_bytes" }
  ]
},
{
  "measInfoId" : "M21",
  "metrics" :
  [
    { "metric": "node_ipvs_incoming_packets_total" },
    { "metric": "node_ipvs_outgoing_packets_total" }
  ]
},
{
  "rule": "max"
},
{
  "rule": "max"
}
],
}
,
{
  "measInfoId" : "M22",
  "metrics" :
  [
    { "metric": "node_procs_blocked" },
    { "metric": "node_procs_running" },
    { "metric": "node_context_switches_total", "rule": "max" }
  ]
}

```

```

        ]
    } ,
{
    "measInfoId" : "M23" ,
    "metrics" :
    [
        {
            "metric": "node_netstat_Tcp_AttemptFails" ,
            "metric": "node_netstat_Tcp_InSegs" ,
            "metric": "node_netstat_Tcp_OutSegs" ,
            "metric": "node_netstat_Tcp_RetransSegs" ,
            "metric": "node_netstat_TcpExt_ListenDrops" }
        ]
    } ,
{
    "measInfoId" : "M24" ,
    "metrics" :
    [
        {
            "metric": "node_netstat_Udp_InDatagrams" ,
            "metric": "node_netstat_Udp_InErrors" ,
            "metric": "node_netstat_Udp_OutDatagrams" }
        ]
    } ,
{
    "measInfoId" : "M25" ,
    "metrics" :
    [
        {
            "metric": "node_vmstat_pgggin" ,
            "metric": "node_vmstat_pggout" }
        ]
    } ,
{
    "measInfoId" : "M26" ,
    "dnLabels" : [ "IF" ] ,
    "labelRename": { "IF": "device" } ,
    "metrics" :
    [
        {
            "metric": "node_network_receive_bytes_total" ,
            "metric": "node_network_receive_drop_total" ,
            "metric": "node_network_receive_errs_total" ,
            "metric": "node_network_receive_packets_total" }
        ]
    } ,
{
    "measInfoId" : "M27" ,
    "dnLabels" : [ "IF" ] ,

```

```
        "labelRename": { "IF": "device" },
        "metrics" :
        [
            {
                "metric": "node_network_transmit_bytes_total",
                "metric": "node_network_transmit_drop_total",
                "metric": "node_network_transmit_errs_total",
                "metric": "node_network_transmit_packets_total"
            }
        ]
    }
}
replicaCount: 1
configOverride: |+
    OVERRIDE_prometheus_url = http://cpro-
server.btel.svc.cluster.local:9090/api/v1
persistentVolume:
    storageClass: "cinder-az-nova"
tolerations:
- key: 'is_btel'
    operator: 'Equal'
    value: 'true'
    effect: 'NoSchedule'
nodeSelector:
    is_btel_worker: "true"

crmq:
persistence:
    data:
        storageClass: "cinder-az-nova"
    log:
        storageClass: "cinder-az-nova"
replicas: 3
nodeSelector:
    is_btel_worker: "true"
tolerations:
- key: 'is_btel'
    operator: 'Equal'
    value: 'true'
    effect: 'NoSchedule'
rabbitmq:
    username: alma_mquser
    password: alma_mqpasswd
    tls:
        cacert: |+
            -----BEGIN CERTIFICATE-----
```

```

MIIDcjCCAlqgAwIBAgIIM8YCiV7PUpEwDQYJKoZIhvCNAQELBQAwUDELMakGA1UE
BhMCQUExCzAJBgNVBAgMAkFBMQswCQYDVQQHDAJBQTELMAkGA1UECgwCQUExCzAJ
BgNVBAsMAkFBMQ0wCwYDVQQDARCQ01UMB4XDTE5MTEyMTA2MzE1NFoXDTIyMDIx
MTA2MzE1NFowUDELMakGA1UEBhMCQUExCzAJBgNVBAgMAkFBMQswCQYDVQQHDAJB
QTELMAkGA1UECgwCQUExCzAJBgNVBAsMAkFBMQ0wCwYDVQQDARCQ01UMIIBIjAN
BgkqhkiG9w0BAQEFAOCQ8AMIIIBCgKCAQEAvFcxt7EAuLFbi07QnFpsbUVAgan4
Ne1m9IQ1tryOvca/WHM84koDLq9polAyFqZF5+hrTKEBFXdOYXY3mqv7wtCfYS8o
PUUV58IyMH9CpdcsAw9JBrkeaHBE2hb4f5e801A/eb7ixGnGuWuFX6yTGMoq3Rdy
3i9MXH5MSTSZVWK7wtVACY8JqcSGqYsFR8AuDhS39rBJAKgnOii2Fuq73A5M6wQ
OA2LyWjoqADQ3rVxiSEf2fk1HxBCHDR4Ug391rwBud4WS9EABfnyxYmR6YHNXP8
Ln2kfsn7FPIuHXRR+bpJvDnfVkAFo/GGIAbeg0EaxbM+z62Cz/fPyRPk2wIDAQAB
o1AwTjAMBgNVHRMEBTADAQH/MB0GA1UdDgQWBBTWE4wdfoEIFviQ+RLwRwJTeufe
+jAfBgNVHSMEGDAwgbTWE4wdfoEIFviQ+RLwRwJTeufe+jANBgkqhkiG9w0BAQsf
AAOCAQEAhPMPycHfRMHG1tmCmuBHjWQArFDssaa/+jyt9h4VOr5LNvq2szinfj6
0+U/3p8VFGm/7E1C9joQrp6y6J30pgvx0m7wDgIFjwCHwtZQCIwtHXwyPuSO3326
moF8XxP0J7ASb4rHlgwnR4pPXXU4+b00XYn0OBom5nSIC4ffBVz3X/GScsagaSxj
9X4LLhzPyyEiv3+v0c4SZPhCDW16nrawfDO3d0fbte8aQDVM3fLkwj6Cdvcia4VR
+i0tqrx6bdagKFpn80mL3W7azErDf3LqKOckChZJZ53Svrtdc{jgFHI5ZczbYbEEZ
S4AkZIwFd+Qqjy1CM9c1QgI1ddMViQ==

-----END CERTIFICATE-----

cert: |+
-----BEGIN CERTIFICATE-----
MIIDcjCCAlqgAwIBAgIRANEJiuWI5+7FudZa8Cge92YwDQYJKoZIhvCNAQELBQAw
UDELMakGA1UEBhMCQUExCzAJBgNVBAgMAkFBMQswCQYDVQQHDAJBQTELMAkGA1UE
CgwCQUExCzAJBgNVBAsMAkFBMQ0wCwYDVQQDARCQ01UMB4XDTE5MTEyNTAzNDAX
MVoXDTIwMDgwNzE5NDAxMVoWLTExMBMGa1UEChMMY2VydC1tYW5hZ2VyMRQwEgYD
VQQDEwtleGFtcGx1LmNvbTCCASIwDQYJKoZIhvCNAQEBBQADggEPADCCAQoCggEB
ANRYGte4BkhgDhUre5sQSnnqJuxxdQFT5KhVVru6w6/FSUOIm4qktgo+BF8pmtcp
3DofaJawQZjM/RaJIGKqKmt7/a5L73ZKpXERPdx1/ChQPMU4EzKrIKiDcBk64145
yCnKcjUAGo4cjkSTz3sQ1D1GjJNhAmIOp+ZYhKY6WjsYGgDhzGTuh/1I8lhby9/z
371M+CehkTgprvCx3x2VV8cLh/eBBj2sU40JsOweXVY66JQDXGL9Mf0wXBURnSoA
hE2IXPk6M42uqfi2wqzQO1YbN+hVQS/oDoktiqoKMPYDQBAT5nX2XQP9AXzObNMJ
dE4lCIwKaTJFkTYT9F1FNVECAwEAAaNqMGgwDgYDVR0PAQH/BAQDAgWgMAwGA1Ud
EwEB/wQCMAAwHwYDVR0jBBgwFoAU1hOMHX6BCH74kPkS8EcCU3rn3vowJwYDVR0R
BCAwHoILZXhhbXBsZS5jb22CD3d3dy51eGFtcGx1LmNvbTANBgkqhkiG9w0BAQsf
AAOCAQEAYJlbkOgEXSMqu6FiUbDjv/Iu80TLT4+CRwz1VSIC1Ve1+rW2ahuCRo16
VjJozhp3K01hADBji9y+Wh1A1Q0QxxCd+e0S2CKDRHmKfne9x25Y6dBn43hbi62U
EF3QSWzcylbs8Pox5QyuAtU8zx9iWKwp82T+gDzXcB2s3Vm5fp/p/n0ruR6IOL0w
0NgXDpwWm4ULGFIId/glzaVt/qR948BWpiwkQwxrp5M0ce0mh4bJ0tgeq2McTtHwd
gF5Ebstoj4wBUdrhYCDIJF/fNEuSJgmH40tqgPn954YxOx7g0C62t/rZCpneF2t
zJe6BGkuI8Als8eF+WZNv9ESxb0USQ==

-----END CERTIFICATE-----

key: |+
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAI1Fga17gGSGAOSt7mxBKeeom7HF1AVPkqFVWu7rDr8VJQ4ib

```

```

iqS2Cj4EXymalyncOh9olrBBmMz9FokgYqooy3v9rkvvdkqlcRE93HX8KFA8xTgT
MqsgqINwGTriXjnIKcpyNQAAjhyORJPPexCUPUaMk2ECYg6n5liEpjpaOxgaAOHM
Z06H+UjyWFvL3/PfvUz4J6GROCMu8LHFHZVXxwuH94EGPaxTjQmw7B5dVjrolANC
Yv0x/TBcFRGdKgCETYhc+Toz ja6p+LbCrNA7Vhs36FVBL+gM6S2Kqgow9gNAEBPm
dfZdA/0BfM5s0w10TiUIjAppMkWRNhP0UgU1UQIDAQABAOIBAHKghQ0CSFH1mGSR
Lo6MgsfBQPXOYW0wDnVYbBO3RD+0b1J1gj6bn7FzvQRp/y47aKjvn5QI3cBQmCb8
K0FXveHqswzN7RycOycIOa6y+kYA2m1UUfi+LEkLew4DnYNkCcuVf4Vg4vToMVyP
ticm0f8qAGTmr1SjuLs9+Y2KXn8bDWsxMbRQ9IxZbeBT086t3xagATeQa50zkiNT
glWtDO6F5uGIWLXTB2r4AfUSmAidxeVjT5ifBMqHg/PXpM527UT/TaCjFiOMhs6y
go4WiN0ipqxuoD2lQO2GKTnPvYKKMV/YqvRjFQMoIKNkUoBIGbFb0PjOhIKg8jW
v/7KuoECgYEAE9/S05tEOi8y14vky3q3SRCikWZ5+gtR6VUjcxKVZPR1+hPOqqjp+
3NsGRt+Gr4Y8csvNYRDLvtRCBuXczKqH/YmINMBtUhZRp41p/nP7Sns8XJkyAlA
5L+qgSuFFNBjHa5LibYnOTu1Llv71fHDVbFgis57lt+3+46zcoErIvkCgYEAE2zuj
1oBGU6Kk3hEDAhrGffPOUhilh6qIa8SE3u/HunC/1U6bT5c1W4CCAs+Hrdjd7Db5
D5F52CpGUheMDHD4tZHKyxf6FYDlarTLpRtiQE3UmDFgA1jVbN1E7JYDN14DRQN
C+X/99jyGfyg/J+F+5cYeTct5qD1m+GukIIIs4xkCgYAEjs5g6kd9E3WcE/z2D3TD
oYdayckCt8nDFQJQbcSU1Yo2F9bzJt8vyK111zaQVEN5y/nqJTFIOfc4TzHdY+f5
JHQmm721+xjnFQOBINnLPbDGwdXNwjW5vTbxBg58U1LvcYqbbYeMymbKAfGy/Kwc
P5RJF/82xsaGzJjIqkxTaQKBgQCZtgEzhnCxBzJNOORBD4DoQo8AZKN6sYig9tKq
dLg28mKBTvH6JeFdflarbdKVg551xk1uKvOhK1LPsA7TkknvFF95ci0p/VTaFUGo
QQ1jp+SF75XKKJj jN7rAyqSNsKJBmTNkikikAAoPS1+dgEOymMfiBVw7fdN3Dg8U
5xCN0QKBgQC+WvJFhoPr+3cu+QQnTDyiqv0R5/4QIs7gWLWLQbqt9+3x6BolARPp
5RGkf+3/ApTeeZM4XEE1RN1ef3o0wxOdVAbAyzzk8obK+R1kI+o74vQm6buR6HFr
i454fx11vDvFgJr+U5BzCbF9Mnmwc8/+2Zz4vMZZpAuZTMAfGiKMQ==

-----END RSA PRIVATE KEY-----

verify_option: "verify_peer"
fail_if_no_peer_cert: "true"
ssl_port: 5671

cmdb:
  cluster_type: "simplex"
mariadb:
  count: 1
  heuristic_recover: none
  nodeAffinity:
    enabled: true
    key: is_btel_worker
    value: true
  nodeSelector: {}
  tolerations:
    - key: 'is_btel'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'
  persistence:

```

```
storageClass: "cinder-az-nova"
backup:
  storageClass: "cinder-az-nova"
maxscale:
  count: 0
nodeAffinity:
  enabled: true
  key: is_btel_edge
  value: true
nodeSelector: {}
tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
admin:
  nodeAffinity:
    enabled: true
    key: is_btel_worker
    value: true
  nodeSelector: {}
  tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
  persistence:
    storageClass: "cinder-az-nova"

citm-ingress:
  controller:
    tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
- key: 'is_edge'
  operator: 'Equal'
  value: 'true'
  effect: 'NoExecute'
  nodeSelector:
    is_btel_edge: 'true'
  replicaCount: 1
default404:
  nodeSelector:
```

```
is_btel_worker: 'true'
tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'

btel-oper:
global:
  registry: "bcmt-registry:5000"
replicas: 1
nodeSelector:
  is_btel_worker: "true"
tolerations:
- key: 'is_btel'
  operator: 'Equal'
  value: 'true'
  effect: 'NoSchedule'
```



Note: calico_file_sd job has (calico endpoints) targets that are in the following location:

/etc/etcd/targets_calico.json

When the issue of scaling in and scaling out nodes, the list is not getting updated:

- Update the nodes according to the existing nodes in the cluster.
- Use kubernetes_sd_configs to update the targets dynamically.
- Change the labeling
- Add the values.yaml file of Telemetry for NCS on Virtualized deployment with the original "calico_file_sd" job.

28 Appendix: Baremetal Security - Communications matrix

Table 78: Security Communications Matrix

Role	Service	Unidirectional Bidirectional	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
NCS Manager	SSH	Bidirectional	TCP	22	External-OAM	Administrative shell access, install automation	no
NCS Manager	HTTP	Bidirectional	TCP	80	External-OAM	Redirect to HTTPS tcp/443	no
NCS Manager	HTTPS	Bidirectional	TCP	443	External-OAM	NCS Manager UI	no
NCS Manager	DNS	Bidirectional	UDP/TCP	53	External-OAM	IPv4 HTTP address resolution	yes
NCS Manager	NTP	Bidirectional	UDP	123	External-OAM	Towards NTP server	yes
NCS Manager	RPC (Rpc Bind)		UDP	111	External	RPC is an IPMI interface.	no
NCS Manager	EPMD/ RabbitMQ on Manage		TCP	4369	External		yes

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
NCS Manager	mysqld (Mitigated: Password Protected)		TCP	4567	External		yes
NCS Manager	logstash/metric beat (Required)	Bidictional	TCP	5044	External		yes
NCS Manager	Gunicorn		TCP	5151	External	RPC is an IPMI interface.	no
NCS Manager	Mellanox sharp (Mitigated: Not configured or used.)		TCP	6126	External		yes
NCS Manager	HAProxy http proxy	Bidictional	TCP	8080	External	for Internal haproxy status monitoring	yes
NCS Manager	Nginx master	Bidictional	TCP	8082	External	Being used Internally as a deployer helper	no

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
						utility to install/uninstall ncs cluster	
NCS Manager	kube-apiserver	Bidirectional	TCP	8443	Internal	Exposes Kubernetes API	
NCS Manager	openstack-cloud	Bidirectional	TCP	10258/10259	External-OAM	Openstack API - not relevant for Baremetal	yes
NCS Manager	kibana	Bidirectional	TCP	5602	External-OAM	dashboard for elk	no
NCS Manager	Elasticsearch	Bidirectional	TCP	9200	External		yes
NCS Manager	Zabbix	Bidirectional	TCP	9443	External-OAM	Zabbix server for FM/PM	no
NCS Manager	Zabbix server (Required)	Bidirectional	TCP	10050	External		yes
NCS Manager	EPMD/RabbitMQ		TCP	25672	External		yes

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
NCS Manager	redis-sentinel		TCP	26379	External	Memory db for Masters	yes
NCS Manager	Not found by nmap or on Monitor. (dockerd on Manager)		TCP	52091	External		yes
NCS Manager	ironic nginx	Bidictional	TCP	61399	External-OAM	ironic service for deployment and LC.	no
NCS Master	RPC (rpcbind)	Bidictional	UDP/TCP	111	Internal	RPC program numbers into universal addresses	
NCS Master	Nginx/Deployer Service	Bidictional	TCP	8082	Internal	Being used internally as a deployer helper utility to install/	

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
						uninstall ncs cluster	
NCS Master	radosgw (optional if object storage is used)	Bidirectional	TCP	13808	External- OAM	for Object storage Access for Applications using s3/ swift Api's	no
NCS Master	ceph	Bidirectional	TCP	6800: 7300 , 3300, 6789	Internal	Software for shared Storage	
NCS Master	ceph	Bidirectional	TCP	7000	External	Ceph Information http. Harmless.	no
NCS Master	bird (tcp wrapped)	Bidirectional	TCP	60179	Internal	BGP routing daemon	
NCS Master	BGP	Bidirectional	TCP	179	Internal	Routing protocol used by Calico	
NCS Master	BGP	Bidirectional	TCP	179	External	Routing protocol used by Calico	no

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
NCS Master	logstash/ metric beat	Bidirectional	TCP	5044	Internal		
NCS Master	Zabbix_agent	Bidirectional	TCP	10050	Internal/External	Zabbix service	yes
NCS Master	haproxy		TCP	38765	External		yes
NCS Portal	HTTP (Redirects to HTTPS)	Bidirectional	TCP	80	External	NCS portal	no
NCS Portal	RPC (rpcbind)	Bidirectional	TCP	600	Internal	RPC program numbers into universal addresses	
NCS Portal	Nginx/ Cluster Portal	Bidirectional	TCP	8084	External-OAM	NCS portal	no
NCS Portal	Zabbix_agent/ Zabbix server	Bidirectional	TCP	10050/ 10051/ 10052	Internal	Zabbix service	

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
NCS Portal	pilot-agent??	Bidirectional	TCP	15020	Internal	Optional: when istio is installed	
NCS Portal	Envoy	Bidirectional	TCP	15021, 15090	Internal	Service mesh	
Worker/ Storage/ Edge	SSH	Bidirectional	TCP	22	Internal	administrative shell access, Accessed from the ncs manager	
Worker/ Storage/ Edge	bcmt-controll	Bidirectional	TCP	8086	Internal	an entry point for NCS task runner	
Worker/ Storage/ Edge	calico-node	Bidirectional	TCP	9091	Internal	calico K8s controller	
Worker/ Storage/ Edge	Kubelet	Bidirectional	TCP	10250	Internal	K8S node level agent	
Worker/ Storage/ Edge	ceph	Bidirectional	TCP	6800: 7300 , 3300, 6789	Internal	Software for shared Storage	

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
Worker/ Storage/ Edge	bird (tcp wrapped)	Bidirectional	TCP	60179	Internal	BGP routing daemon	
Worker/ Storage/ Edge	logstash/ metric beat	Bidirectional	TCP	5044	Internal		
Worker/ Storage/ Edge	Zabbix_ agent	Bidirectional	TCP	10050	Internal	Zabbix service	
Worker/ Storage	BGP	Bidirectional	TCP	179	Internal	Routing protocol used by Calico	
Edge	BGP	Bidirectional	TCP	179	External	Routing protocol used by Calico	no
Monitor	SSH	Bidirectional	TCP	22	External- OAM	administrative shell access, install automation	
Monitor	RPC (Rpc Bind)		UDP/TCP	111	External	RPC is an IPMI interface.	yes

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
Monitor	mysqld (Mitigated: Password Protected)		TCP	4567	External		yes
Monitor	Mellanox sharp (Mitigated: Not configured or used.)		TCP	6126	External		yes
Monitor	Not found by nmap. (Required: HAProxy http proxy)	Bidictional	TCP	8080	External	for Internal haproxy status monitoring	yes
Monitor	Keycloak/ snmp/ alarm/ alma/jmx (Required: Zabbix)		TCP	9099	External		yes
Monitor	Zabbix web	Bidictional	TCP	9443	External	Zabbix UI	no
Monitor	Alarm manager	Bidictional	UDP	1161	External	snmpget/ snmpwalk	no

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
Monitor	Alarm manager	Bidirectional	UDP	1162	External	receive traps on	no
Monitor	Elasticsearch (Required)	Bidirectional	TCP	9201/ 9300/ 9200	Internal		
Monitor	Kibana-nginx (Required)	Bidirectional	TCP	5602	External	kibana UI	no
Monitor	kibana	Bidirectional	TCP	5601	Internal		
Monitor	Zabbix server (Required)	Bidirectional	TCP	10050, 10051	External		yes
Monitor	logstash/metric beat (Required)	Bidirectional	TCP	5044	External		yes
Storage only node	SSH	Bidirectional	TCP	22	Internal	administrative shell access, Accessed from the ncs manager	

Table 78: Security Communications Matrix (continued)

Role	Service	Unidirection Bidirection	IP/ Protocol	Port Range	External/ Internal	Usage	For External Ports: Blocked by Hardening?
Storage only node	ceph	Bidictional	TCP	6800: 7300 , 3300, 6789	Internal	Software for shared Storage	

 **Note:** All services are **synchronous** and use **static IPs**.

 **Note:** For Baremetal deployment, please use port 8084 instead of 8082.

 **Note:** There are also 2 additional open ports:

- `tcp6 0 0 ::::8888 ::::* LISTEN 3236/kubectl`
- `tcp6 0 0 ::::9100 ::::* LISTEN 26875/node_exporter`

The following ports and services can be opened for Edge and Monitor nodes are available when having external access:

- on Edges for istio if used.
- on Edge nodes for nodePorts if used.
- on any node for DANM/Multus ports if used.

 **Note:**

1. Nginx 9090 is on Edges for CITM - from external.
2. 38765 is on master only, from masters only, on L2, cluster internal.

Firewall management information: the following table includes ports that require external access.

Table 79: Ports that require external access

Service	On nodes	External access	Protocol	IPv4/IPv6/ DualStack	Port
SSH	Manager, Master	On External (OAM) network	TCP	IPv4	22
DNS	Manager, Master Edge	On External (OAM) network On Edge external network	UDP/TCP	IPv6	53
NTP	Manager, Master	On External (OAM) network	UDP	IPv4	123
HTTP/HTTPS	Manager	On External (OAM) network	TCP	IPv4	80 / 443
HTTPS	Master	On External (OAM) network	TCP	IPv4	8084
Zabbix	Manager or Monitor	On External (OAM) network	TCP	IPv4	9443
ELK	Manager or Monitor	On External (OAM) network	TCP	IPv4	5602
Alarm manager	Manager or Monitor	On External (OAM) network	TCP	IPv4	1161
Per deployment					

Table 79: Ports that require external access (continued)

Service	On nodes	External access	Protocol	IPv4/IPv6/ DualStack	Port
BELK	Master	On External (OAM) network	TCP	IPv4	
CITM (Ingress)	Edge	On Edge external networks	TCP		9090
Istio Ingress	Edge	On Edge external networks			
BGP	Edge	On Edge external networks	TCP	IPv4	179

29 Appendix: Changing the BCMT server timezone from UTC to US (PST, CST, EST)

Changing a timezone in a living site with BCMT infra can be performed manually but may impact any application which is time sensitive.

When a host time zone and docker(POD) time zone are different, even after changing the host time zone, the POD still needs to mount the host file to line up the timezone with the host.

1. a. Each container in a POD should mount the host "/etc/localtime" to line up with the host timezone, otherwise, even when the host time zone has been changed, the container timezone will still be set to UTC(default).

```
{
  containers:
    name: bcmt-api
    ....
    volumeMounts:
      mountPath: /etc/pki
      - name: host-timezone
        mountPath: /etc/localtime
        readOnly: true

    volumes:
    ...
    - name: host-timezone
      mountPath: /etc/localtime
      readOnly: true
}
```

- b. If the containers do not mount the /etc/localtime file, PodPreset can be used to enforce it within a namespace:

```

apiVersion: settings.k8s.io/v1alpha1
kind: PodPreset
metadata:
  name: mount-local-time
  namespace: appns
spec:
  volumeMounts:
    - name: host-timezone
      mountPath: /etc/localtime
      readOnly: true
  volumes:
    - name: host-timezone
      hostPath:
        path: /etc/localtime
}
```

Make sure to edit the namespace value and apply the PodReset object by this command:

```
kubectl apply -f <podreset.yaml>
```

2. Change the host time zone as follows, on each node, change the timezone, and drain the pod to the other node:

a. ln -sf /usr/share/zoneinfo/CST6CDT /etc/localtime, check the time zone with "timedatectl".

b. Enter:

```
/usr/local/bin/kubectl drain --force --ignore-daemonsets --delete-local-data node-nam
```

c. Reboot the node.

d. Enter:

```
kubectl uncordon node_name  
kubectl get node node_name
```

This will ensure that the node is ready.

3. Update the BCMT configuration data on one of the control nodes as follows:

a. Get the current timezone as follows:

```
export ETCDCTL_API=3; etcdctl --endpoints=https://127.0.0.1:2379 --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem get  
/BCMTClusterManager/bcmt_config/cluster_config/timezone  
/BCMTClusterManager/bcmt_config/cluster_config/timezone  
"UTC"
```

b. Set the timezone, (CST6CDT as an example):

```
export ETCDCTL_API=3; etcdctl --endpoints=https://127.0.0.1:2379 --cert=/etc/etcd/ssl/etcd-client.pem --key=/etc/etcd/ssl/etcd-client-key.pem put  
/BCMTClusterManager/bcmt_config/cluster_config/timezone \"CST6CDT\"
```