# Contents

## Copyright notice

## Preface

- About this documentation set
- Intended audience
- Conventions
- Technology Preview support scope
- Documentation history

## Mirantis Container Cloud CLI

## Operate managed clusters

- Create and operate a baremetal-based managed cluster
  - Create a managed cluster
  - Add a bare metal host
    - Add a bare metal host using web UI
    - Add a bare metal host using CLI
  - Add a machine
    - Create a machine using web UI
    - Create a machine using CLI
    - Deploy a machine to a specific bare metal host
    - Override network interfaces naming and order
  - Add a Ceph cluster
- Delete a managed cluster
- Advanced networking configuration
  - Create subnets
  - Automate multiple subnet creation using SubnetPool
  - Create L2 templates
    - L2 template example with bonds and bridges
    - L2 template example for automatic multiple subnet creation
  - Create a custom bare metal host profile
  - Default configuration of the host system storage
  - Create a custom host profile
  - Enable huge pages in a host profile
Create and operate an OpenStack-based managed cluster 42
  Create a managed cluster 42
  Add a machine 45
  Delete a managed cluster 48
Create and operate an AWS-based managed cluster 49
  Create a managed cluster 49
  Add a machine 52
  Delete a managed cluster 54
Create and operate a VMWare vSphere-based managed cluster 55
  Create a managed cluster 55
  Add a machine 59
  Delete a managed cluster 61
Change a cluster configuration 62
Update a managed cluster 62
Delete a machine 64

**Operate management and regional clusters** 65
  Automatic upgrade workflow 65
  Remove a management cluster 66
  Remove a regional cluster 67

Attach an existing Mirantis Kubernetes Engine cluster 68

Connect to the Mirantis Kubernetes Engine web UI 72

Connect to a Mirantis Container Cloud cluster 74

Manage IAM 76
  IAM CLI 76
    Configure IAM CLI 76
    Available IAM CLI commands 76
  Role list 80

Manage StackLight 82
  Access StackLight web UIs 82
  View Grafana dashboards 82
  View Kibana dashboards 86
  Available StackLight alerts 87
  Alertmanager 87
AlertmanagerFailedReload 87
AlertmanagerMembersInconsistent 87
AlertmanagerNotificationFailureWarning 87
AlertmanagerAlertsInvalidWarning 88
Calico 88
CalicoDataplaneFailuresHigh 88
CalicoDataplaneAddressMsgBatchSizeHigh 88
CalicoDatapanelfaceMsgBatchSizeHigh 89
CalicoIPsetErrorsHigh 89
CalicoIptablesSaveErrorsHigh 89
CalicoIptablesRestoreErrorsHigh 89
Ceph 90
CephClusterHealthMinor 90
CephClusterHealthCritical 91
CephMonQuorumAtRisk 91
CephOSDDownMinor 91
CephOSDDiskNotResponding 91
CephOSDDiskUnavailable 92
CephClusterNearFull 92
CephClusterCriticallyFull 92
CephOSDPgNumTooHighWarning 92
CephOSDPgNumTooHighCritical 93
CephMonHighNumberOfLeaderChanges 93
CephNodeDown 93
CephDataRecoveryTakingTooLong 93
CephPGRepairTakingTooLong 94
CephOSDVersionMismatch 94
CephMonVersionMismatch 94
Docker Swarm 94
DockerSwarmLeadElectionLoop 95
DockerSwarmNetworkUnhealthy 95
DockerSwarmNodeFlapping 95
DockerSwarmServiceReplicasDown 96
<table>
<thead>
<tr>
<th>Alert Type</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DockerSwarmServiceReplicasFlapping</td>
<td>96</td>
</tr>
<tr>
<td>DockerSwarmServiceReplicasOutage</td>
<td>96</td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>96</td>
</tr>
<tr>
<td>ElasticHeapUsageCritical</td>
<td>97</td>
</tr>
<tr>
<td>ElasticHeapUsageWarning</td>
<td>97</td>
</tr>
<tr>
<td>ElasticClusterStatusCritical</td>
<td>97</td>
</tr>
<tr>
<td>ElasticClusterStatusWarning</td>
<td>98</td>
</tr>
<tr>
<td>NumberOfRelocationShards</td>
<td>98</td>
</tr>
<tr>
<td>NumberOfInitializingShards</td>
<td>98</td>
</tr>
<tr>
<td>NumberOfUnassignedShards</td>
<td>98</td>
</tr>
<tr>
<td>NumberOfPendingTasks</td>
<td>99</td>
</tr>
<tr>
<td>ElasticNoNewDataCluster</td>
<td>99</td>
</tr>
<tr>
<td>ElasticNoNewDataNode</td>
<td>99</td>
</tr>
<tr>
<td>etcd</td>
<td>99</td>
</tr>
<tr>
<td>etcdInsufficientMembers</td>
<td>100</td>
</tr>
<tr>
<td>etcdNoLeader</td>
<td>100</td>
</tr>
<tr>
<td>etcdHighNumberOfLeaderChanges</td>
<td>100</td>
</tr>
<tr>
<td>etcdGRPCRequestsSlow</td>
<td>101</td>
</tr>
<tr>
<td>etcdMemberCommunicationSlow</td>
<td>101</td>
</tr>
<tr>
<td>etcdHighNumberOfFailedProposals</td>
<td>101</td>
</tr>
<tr>
<td>etcdHighFsncDurations</td>
<td>101</td>
</tr>
<tr>
<td>etcdHighCommitDurations</td>
<td>102</td>
</tr>
<tr>
<td>External endpoint</td>
<td>102</td>
</tr>
<tr>
<td>ExternalEndpointDown</td>
<td>102</td>
</tr>
<tr>
<td>ExternalEndpointTCPFailure</td>
<td>102</td>
</tr>
<tr>
<td>General alerts</td>
<td>103</td>
</tr>
<tr>
<td>TargetDown</td>
<td>103</td>
</tr>
<tr>
<td>TargetFlapping</td>
<td>103</td>
</tr>
<tr>
<td>NodeDown</td>
<td>103</td>
</tr>
<tr>
<td>Watchdog</td>
<td>104</td>
</tr>
<tr>
<td>General node alerts</td>
<td>104</td>
</tr>
<tr>
<td>FileDescriptorUsageCritical</td>
<td>104</td>
</tr>
<tr>
<td>FileDescriptorUsageMajor</td>
<td>105</td>
</tr>
</tbody>
</table>
FileDescriptorUsageWarning 105
SystemCpuFullWarning 105
SystemLoadTooHighWarning 105
SystemLoadTooHighCritical 106
SystemDiskFullWarning 106
SystemDiskFullMajor 106
SystemMemoryFullWarning 106
SystemMemoryFullMajor 107
SystemDiskInodesFullWarning 107
SystemDiskInodesFullMajor 107
SystemDiskErrorsTooHigh 108
Ironic 108
IronicBmMetricsMissing 108
IronicBmApiOutage 108
Kubernetes applications 108
KubePodCrashLooping 109
KubePodNotReady 109
KubeDeploymentGenerationMismatch 109
KubeDeploymentReplicasMismatch 110
KubeStatefulSetReplicasMismatch 110
KubeStatefulSetGenerationMismatch 110
KubeStatefulSetUpdateNotRolledOut 110
KubeDaemonSetRolloutStuck 111
KubeDaemonSetNotScheduled 111
KubeDaemonSetMisScheduled 111
KubeCronJobRunning 112
KubeJobCompletion 112
KubeJobFailed 112
Kubernetes resources 112
KubeCPUMemOvercommitPods 113
KubeMemOvercommitPods 113
KubeCPUMemOvercommitNamespaces 113
KubeMemOvercommitNamespaces 113
KubeQuotaExceeded 114
CPUThrottlingHigh 114

Kubernetes storage 114
KubePersistentVolumeUsageCritical 115
KubePersistentVolumeFullInFourDays 115
KubePersistentVolumeErrors 115

Kubernetes system 115
KubeNodeNotReady 116
KubeVersionMismatch 116
KubeClientErrors 116
KubeletTooManyPods 117
KubeAPIDown 117
KubeAPIOutage 117
KubeAPILatencyHighWarning 117
KubeAPILatencyHighMajor 118
KubeAPIErrorsHighMajor 118
KubeAPIErrorsHighWarning 118
KubeAPIResourceErrorsHighMajor 118
KubeAPIResourceErrorsHighWarning 119
KubeClientCertificateExpirationInSevenDays 119
KubeClientCertificateExpirationInOneDay 119
ContainerScrapeError 120

Netchecker 120
NetCheckerAgentErrors 120
NetCheckerReportsMissing 120
NetCheckerTCP_Server_Delay 120
NetCheckerDNS_Slow 121

NGINX 121
NginxServiceDown 121
NginxDroppedIncomingConnections 121

Node network 122
SystemRxPacketsErrorTooHigh 122
SystemTxPacketsErrorTooHigh 122
SystemRxPacketsDroppedTooHigh  122
SystemTxPacketsDroppedTooHigh  123
NodeNetworkInterfaceFlapping  123
Node time  123
ClockSkewDetected  123
PostgreSQL  124
  PostgresqlDataPageCorruption  124
  PostgresqlDeadlocksDetected  124
  PostgresqlInsufficientWorkingMemory  125
  PostgresqlPatroniClusterSplitBrain  125
  PostgresqlPatroniClusterUnlocked  125
  PostgresqlPrimaryDown  125
  PostgresqlReplicaDown  126
  PostgresqlReplicationNonStreamingReplicas  126
  PostgresqlReplicationPaused  126
  PostgresqlReplicationSlowWalApplication  126
  PostgresqlReplicationSlowWalDownload  127
  PostgresqlReplicationWalArchiveWriteFailing  127
Prometheus  127
  PrometheusConfigReloadFailed  128
  PrometheusNotificationQueueRunningFull  128
  PrometheusErrorSendingAlertsWarning  128
  PrometheusErrorSendingAlertsMajor  128
  PrometheusNotConnectedToAlertmanagers  129
  PrometheusTSDBReloadingFailing  129
  PrometheusTSDBCompactionsFailing  129
  PrometheusTSDBWALCorruptions  129
  PrometheusNotIngestingSamples  130
  PrometheusTargetScrapesDuplicate  130
  PrometheusRuleEvaluationsFailed  130
Salesforce notifier  131
  SfNotifierAuthFailure  131
SMART disks  131
SystemSMARTDiskUDMACrcErrorsTooHigh
SystemSMARTDiskHealthStatus
SystemSMARTDiskReadErrorRate
SystemSMARTDiskSeekErrorRate
SystemSMARTDiskTemperatureHigh
SystemSMARTDiskReallocatedSectorsCount
SystemSMARTDiskCurrentPendingSectors
SystemSMARTDiskReportedUncorrectableErrors
SystemSMARTDiskOfflineUncorrectableSectors
SystemSMARTDiskEndToEndError
SSL certificates
  SSLCertExpirationWarning
  SSLCertExpirationMajor
  SSLProbesFailing
  MCCCSSLCertExpirationMajor
  MCCCSSLCertExpirationWarning
  MCCCSSLProbesFailing
Telemeter
  TelemeterClientFederationFailed
Mirantis Kubernetes Engine
  MKEAPIDown
  MKEAPIOutage
  MKEContainerUnhealthy
  MKENodeDiskFullCritical
  MKENodeDiskFullWarning
  MKENodeDown
Configure StackLight
StackLight configuration parameters
  Alerta
  Elasticsearch
  Grafana
  Logging
  High availability
Metric collector 143
Prometheus 144
Cluster size 147
Resource limits 148
Kubernetes tolerations 149
Storage class 150
NodeSelector 152
Salesforce reporter 153
Ceph monitoring 154
External endpoint monitoring 155
Ironic monitoring 158
SSL certificates monitoring 159
Workload monitoring 160
Mirantis Kubernetes Engine monitoring 161
Alerts configuration 162
Watchdog alert 163
Alertmanager integrations 164
Notifications to email 166
Notifications to Salesforce 169
Notifications to Slack 173
Verify StackLight after configuration 177
Enable generic metric scraping 182

Manage Ceph 183
Enable automated Ceph LCM 183
Add, remove, or reconfigure Ceph nodes 183
Replace a failed Ceph OSD 186
Update Ceph cluster 187
Enable Ceph RGW Object Storage 187
Verify Ceph components 191
  Verify the Ceph core services 191
  Verify rook-discover 191
  Verify Ceph cluster state 192

Troubleshooting 193
Collect cluster logs

**Mirantis Container Cloud API**

Public key resources

Bare metal resources

Cluster

  metadata
  spec:providerSpec
  spec:providerSpec common
  spec:providerSpec configuration
  status:providerStatus common
  status:providerStatus for cluster readiness
  status:providerStatus for Open ID Connect
  status:providerStatus for cluster releases

Machine

  metadata
  spec:providerSpec for instance configuration

Machine status

BareMetalHostProfile

  metadata
  spec

BareMetalHost

  BareMetalHost metadata
  BareMetalHost configuration
  BareMetalHost status

IpamHost

  IpamHost metadata
  IpamHost configuration
  IpamHost status

Subnet

  Subnet metadata
  Subnet spec
  Subnet status

SubnetPool
SubnetPool metadata  221
SubnetPool spec  222
SubnetPool status  223
IPAddr  224
IPAddr metadata  224
IPAddr spec  225
IPAddr status  225
L2Template  226
L2Template metadata  226
L2Template configuration  228
L2Template status  229
Preface

This documentation provides information on how to deploy and operate Mirantis Container Cloud.

- About this documentation set
- Intended audience
- Conventions
- Technology Preview support scope
- Documentation history

About this documentation set

The documentation is intended to help operators understand the core concepts of the product.

The information provided in this documentation set is being constantly improved and amended based on the feedback and kind requests from our software consumers. This documentation set outlines description of the features that are supported within two latest Cloud Container minor releases, with a corresponding note Available since release.

The following table lists the guides included in the documentation set you are reading:

<table>
<thead>
<tr>
<th>Guides list</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Guide</strong></td>
</tr>
<tr>
<td>Reference Architecture</td>
</tr>
<tr>
<td>Deployment Guide</td>
</tr>
<tr>
<td>Operations Guide</td>
</tr>
<tr>
<td>Release Compatibility Matrix</td>
</tr>
<tr>
<td>Release Notes</td>
</tr>
</tbody>
</table>

For your convenience, we provide all guides from this documentation set in HTML (default), single-page HTML, PDF, and ePUB formats. To use the preferred format of a guide, select the required option from the Formats menu next to the guide title on the Container Cloud documentation home page.

Intended audience

This documentation assumes that the reader is familiar with network and cloud concepts and is intended for the following users:
• Infrastructure Operator
  • Is member of the IT operations team
  • Has working knowledge of Linux, virtualization, Kubernetes API and CLI, and OpenStack to support the application development team
  • Accesses Mirantis Container Cloud and Kubernetes through a local machine or web UI
  • Provides verified artifacts through a central repository to the Tenant DevOps engineers
• Tenant DevOps engineer
  • Is member of the application development team and reports to line-of-business (LOB)
  • Has working knowledge of Linux, virtualization, Kubernetes API and CLI to support application owners
  • Accesses Container Cloud and Kubernetes through a local machine or web UI
  • Consumes artifacts from a central repository approved by the Infrastructure Operator

Conventions
This documentation set uses the following conventions in the HTML format:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface font</td>
<td>Inline CLI tools and commands, titles of the procedures and system response examples, table titles.</td>
</tr>
<tr>
<td>monospaced font</td>
<td>Files names and paths, Helm charts parameters and their values, names of packages, nodes names and labels, and so on.</td>
</tr>
<tr>
<td>italic font</td>
<td>Information that distinguishes some concept or term.</td>
</tr>
<tr>
<td>Links</td>
<td>External links and cross-references, footnotes.</td>
</tr>
<tr>
<td>Main menu &gt; menu item</td>
<td>GUI elements that include any part of interactive user interface and menu navigation.</td>
</tr>
<tr>
<td>Superscript</td>
<td>Some extra, brief information. For example, if a feature is available from a specific release or if a feature is in the Technology Preview development stage.</td>
</tr>
<tr>
<td>Note</td>
<td>Messages of a generic meaning that may be useful to the user.</td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.
Technology Preview support scope

This documentation set includes description of the Technology Preview features. A Technology Preview feature provide early access to upcoming product innovations, allowing customers to experience the functionality and provide feedback during the development process. Technology Preview features may be privately or publicly available and neither are intended for production use. While Mirantis will provide support for such features through official channels, normal Service Level Agreements do not apply. Customers may be supported by Mirantis Customer Support or Mirantis Field Support.

As Mirantis considers making future iterations of Technology Preview features generally available, we will attempt to resolve any issues that customers experience when using these features.

During the development of a Technology Preview feature, additional components may become available to the public for testing. Because Technology Preview features are being under development, Mirantis cannot guarantee the stability of such features. As a result, if you are using Technology Preview features, you may not be able to seamlessly upgrade to subsequent releases of that feature. Mirantis makes no guarantees that Technology Preview features will be graduated to a generally available product release.
The Mirantis Customer Success Organization may create bug reports on behalf of support cases filed by customers. These bug reports will then be forwarded to the Mirantis Product team for possible inclusion in a future release.

**Documentation history**

The documentation set refers to Mirantis Container Cloud GA as to the latest released GA version of the product. For details about the Container Cloud GA minor releases dates, refer to Container Cloud releases.
Mirantis Container Cloud CLI

The Mirantis Container Cloud APIs are implemented using the Kubernetes CustomResourceDefinitions (CRDs) that enable you to expand the Kubernetes API. For details, see Mirantis Container Cloud API.

You can operate Container Cloud using the kubectl command-line tool that is based on the Kubernetes API. For the kubectl reference, see the official Kubernetes documentation.

The Container Cloud Operations Guide mostly contains manuals that describe the Container Cloud web UI that is intuitive and easy to get started with. Some sections are divided into a web UI instruction and an analogous but more advanced CLI one. Certain Container Cloud operations can be performed only using CLI with the corresponding steps described in dedicated sections. For details, refer to the required component section of this guide.
Operate managed clusters

Note
This tutorial applies only to the Container Cloud web UI users with the writer access role assigned by the Infrastructure Operator. To add a bare metal host, the operator access role is also required.

After you deploy the Mirantis Container Cloud management cluster, you can start creating managed clusters that will be based on the same cloud provider type that you have for the management cluster: OpenStack, AWS, bare metal, or VMWare vSphere.

The deployment procedure is performed using the Container Cloud web UI and comprises the following steps:

1. Create an initial cluster configuration depending on the provider type.
2. For a baremetal-based managed cluster, create and configure bare metal hosts with corresponding labels for machines such as worker, manager, or storage.
3. Add the required amount of machines with the corresponding configuration to the managed cluster.
4. For a baremetal-based managed cluster, add a Ceph cluster.

Create and operate a baremetal-based managed cluster

After bootstrapping your baremetal-based Mirantis Container Cloud management cluster as described in Deployment Guide: Deploy a baremetal-based management cluster, you start creating the baremetal-based managed clusters using the Container Cloud web UI.

Create a managed cluster

This section instructs you on how to configure and deploy a managed cluster that is based on the baremetal-based management cluster through the Mirantis Container Cloud web UI.

To create a managed cluster on bare metal:

1. Recommended. Verify that you have successfully configured an L2 template for a new cluster as described in Advanced networking configuration. You may skip this step if you do not require L2 separation for network traffic.
2. Optional. Create a custom bare metal host profile depending on your needs as described in Create a custom bare metal host profile.
3. Log in to the Container Cloud web UI with the writer permissions.
4. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
5. In the SSH keys tab, click Add SSH Key to upload the public SSH key that will be used for the SSH access to VMs.
6. In the Clusters tab, click Create Cluster.

7. Configure the new cluster in the Create New Cluster wizard that opens:

   1. Define general and Kubernetes parameters:

   Create new cluster: General, Provider, and Kubernetes

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General settings</td>
<td>Cluster name</td>
<td>The cluster name.</td>
</tr>
<tr>
<td></td>
<td>Provider</td>
<td>Select Baremetal.</td>
</tr>
<tr>
<td></td>
<td>Region</td>
<td>From the drop-down list, select Baremetal.</td>
</tr>
<tr>
<td></td>
<td>Release version</td>
<td>The Container Cloud version.</td>
</tr>
<tr>
<td></td>
<td>SSH keys</td>
<td>From the drop-down list, select the SSH key name that you have previously added for SSH access to the bare metal hosts.</td>
</tr>
<tr>
<td>Provider</td>
<td>LB host IP</td>
<td>The IP address of the load balancer endpoint that will be used to access the Kubernetes API of the new cluster. This IP address must be from the same subnet as used for DHCP in Metal³.</td>
</tr>
<tr>
<td></td>
<td>LB address range</td>
<td>The range of IP addresses that can be assigned to load balancers for Kubernetes Services by MetaLLB.</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>Node CIDR</td>
<td>The Kubernetes worker nodes CIDR block. For example, 10.10.10.0/24.</td>
</tr>
<tr>
<td></td>
<td>Services CIDR blocks</td>
<td>The Kubernetes Services CIDR blocks. For example, 10.233.0.0/18.</td>
</tr>
<tr>
<td></td>
<td>Pods CIDR blocks</td>
<td>The Kubernetes pods CIDR blocks. For example, 10.233.64.0/18.</td>
</tr>
</tbody>
</table>

2. Configure StackLight:

StackLight configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.
<table>
<thead>
<tr>
<th>StackLight</th>
<th>Enable Monitoring</th>
<th>Selected by default. Deselect to skip StackLight deployment.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>&lt;br&gt;You can also enable, disable, or configure StackLight parameters after deploying a managed cluster. For details, see Change a cluster configuration or Configure StackLight.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enable Logging</th>
<th>Select to deploy the StackLight logging stack. For details about the logging components, see Reference Architecture: StackLight deployment architecture.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA Mode</td>
<td>Select to enable StackLight monitoring in the HA mode. For the differences between HA and non-HA modes, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Elasticsearch</th>
<th>Retention Time</th>
<th>The Elasticsearch logs retention period in Logstash.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent Volume Claim Size</td>
<td>The Elasticsearch persistent volume claim size.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prometheus</th>
<th>Retention Time</th>
<th>The Prometheus database retention period.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retention Size</td>
<td>The Prometheus database retention size.</td>
<td></td>
</tr>
<tr>
<td>Persistent Volume Claim Size</td>
<td>The Prometheus persistent volume claim size.</td>
<td></td>
</tr>
</tbody>
</table>

| Enable Watchdog Alert | Select to enable the Watchdog alert that fires as long as the entire alerting pipeline is functional. |
Custom Alerts

Specify alerting rules for new custom alerts or upload a YAML file in the following exemplary format:

- alert: HighErrorRate
  expr: job:request_latency_seconds:mean5m{job="myjob"} > 0.5
  for: 10m
  labels:
    severity: page
  annotations:
    summary: High request latency

For details, see Official Prometheus documentation: Alerting rules. For the list of the predefined StackLight alerts, see Operations Guide: Available StackLight alerts.

<table>
<thead>
<tr>
<th>StackLight Email Alerts</th>
<th>Enable Email Alerts</th>
<th>Select to enable the StackLight email alerts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Resolved</td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
<td></td>
</tr>
<tr>
<td>Require TLS</td>
<td>Select to enable transmitting emails through TLS.</td>
<td></td>
</tr>
</tbody>
</table>

Email alerts configuration for StackLight

Fill out the following email alerts parameters as required:
- To - the email address to send notifications to.
- From - the sender address.
- SmartHost - the SMTP host through which the emails are sent.
- Authentication username - the SMTP user name.
- Authentication password - the SMTP password.
- Authentication identity - the SMTP identity.
- Authentication secret - the SMTP secret.

<table>
<thead>
<tr>
<th>StackLight Slack Alerts</th>
<th>Enable Slack alerts</th>
<th>Select to enable the StackLight Slack alerts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Resolved</td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
<td></td>
</tr>
</tbody>
</table>

Slack alerts configuration for StackLight

Fill out the following Slack alerts parameters as required:
- API URL - The Slack webhook URL.
- Channel - The channel to send notifications to, for example, #channel-for-alerts.

8. Click Create.

To view the deployment status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the deployment is complete.
Now, proceed to Add a bare metal host.

Add a bare metal host

This section describes how to add a bare metal host to a newly created managed cluster using either the Container Cloud web UI or CLI for an advanced configuration.

Add a bare metal host using web UI

After you create a managed cluster as described in Create a managed cluster, proceed with adding a bare metal host through the Mirantis Container Cloud web UI using the instruction below.

Before you proceed with adding a bare metal host, verify that the physical network on the server has been configured correctly. See Reference Architecture: Network fabric for details.

To add a bare metal host to a baremetal-based managed cluster:

1. Log in to the Container Cloud web UI with the operator permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Baremetal tab, click Add BM host.
4. Fill out the Add new BM host form as required:
   - Baremetal host name
     Specify the name of the new bare metal host.
   - Username
     Specify the name of the user for accessing the BMC (IPMI user).
   - Password
     Specify the password of the user for accessing the BMC (IPMI password).
   - Boot MAC address
     Specify the MAC address of the PXE network interface.
   - IP Address
     Specify the IP address to access the BMC.
   - Label
     Assign the machine label to the new host that defines which type of machine may be deployed on this bare metal host. Only one label can be assigned to a host. The supported labels include:
     - Manager
       This label is selected and set by default. Assign this label to the bare metal hosts that can be used to deploy machines with the manager type. These hosts must match the CPU and RAM requirements described in Reference Architecture: Reference hardware configuration.
• **Worker**
  The host with this label may be used to deploy the worker machine type. Assign this label to the bare metal hosts that have sufficient CPU and RAM resources, as described in Reference Architecture: Reference hardware configuration.

• **Storage**
  Assign this label to the bare metal hosts that have sufficient storage devices to match Reference Architecture: Reference hardware configuration. Hosts with this label will be used to deploy machines with the storage type that run Ceph OSDs.

5. **Click Create**
   While adding the bare metal host, Container Cloud discovers and inspects the hardware of the bare metal host and adds it to BareMetalHost.status for future references.

Now, you can proceed to Create a machine using web UI.

**Add a bare metal host using CLI**

After you create a managed cluster as described in Create a managed cluster, proceed with adding bare metal hosts using the Mirantis Container Cloud CLI using the instruction below.

To add a bare metal host using API:

1. Log in to the host where your management cluster kubeconfig is located and where kubectl is installed.

2. Create a secret YAML file that describes the credentials of the new bare metal host.

   **Example of the bare metal host secret:**
   
   ```yaml
   apiVersion: v1
data:
  password: <credentials-password>
username: <credentials-user-name>
kind: Secret
metadata:
  labels:
    kaas.mirantis.com/credentials: "true"
    kaas.mirantis.com/provider: baremetal
    kaas.mirantis.com/region: region-one
    name: <credentials-name>
  namespace: <managed-cluster-project-name>
type: Opaque
   
In the data section, add the IPMI user name and password in the base64 encoding to access the BMC. To obtain the base64-encoded credentials, you can use the following command in your Linux console:

   ```bash
   echo -n <username|password> | base64
   ```
3. Apply this secret YAML file to your deployment:

```
kubectl apply -f ${<bmh-cred-file-name>}.yaml
```

4. Create a YAML file that contains a description of the new bare metal host. Example of the bare metal host configuration file with the worker role:

```yaml
apiVersion: metal3.io/v1alpha1
class: BareMetalHost
metadata:
  labels:
    kaas.mirantis.com/baremetalhost-id: <unique-bare-metal-host-hardware-node-id>
    hostlabel.bm.kaas.mirantis.com/worker: "true"
    kaas.mirantis.com/provider: baremetal
    kaas.mirantis.com/region: region-one
    name: <bare-metal-host-unique-name>
    namespace: <managed-cluster-project-name>
spec:
bmc:
  address: <ip_address_for-bmc-access>
  credentialsName: <credentials-name>
  bootMACAddress: <bare-metal-host-boot-mac-address>
  online: true
```

For a detailed fields description, see BareMetalHost.

5. Apply this configuration YAML file to your deployment:

```
kubectl apply -f ${<bare-metal-host-config-file-name>}.yaml
```

Now, proceed with Deploy a machine to a specific bare metal host.

### Add a machine

This section describes how to add a machine to a newly created managed cluster using either the Mirantis Container Cloud web UI or CLI for an advanced configuration.

Create a machine using web UI

After you add a bare metal host to the managed cluster as described in Add a bare metal host using web UI, you can create a Kubernetes machine in your cluster using the Mirantis Container Cloud web UI.

To add a Kubernetes machine to a baremetal-based managed cluster:

1. Log in to the Mirantis Container Cloud web UI with the operator or writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the required cluster name. The cluster page with the Machines list opens.

4. Click Create Machine button.

5. Fill out the Create New Machine form as required:

   • Count
     Specify the number of machines to add.

   • Manager
     Select Manager or Worker to create a Kubernetes manager or worker node. The required minimum number of machines is three for the manager nodes HA and two for the Container Cloud workloads.

   • BareMetal Host Label
     Assign the role to the new machine(s) to link the machine to a previously created bare metal host with the corresponding label. You can assign one role type per machine. The supported labels include:

     • Worker
       The default role for any node in a managed cluster. Only the kubelet service is running on the machines of this type.

     • Manager
       This node hosts the manager services of a managed cluster. For the reliability reasons, Container Cloud does not permit running end user workloads on the manager nodes or use them as storage nodes.

     • Storage
       This node is a worker node that also hosts Ceph OSDs and provides its disk resources to Ceph. Container Cloud permits end users to run workloads on storage nodes by default.

   • Node Labels
Select the required node labels for the machine to run certain components on a specific node. For example, for the StackLight nodes that run Elasticsearch and require more resources than a standard node, select the StackLight label. The list of available node labels is obtained from your current Cluster release.

Caution!

If you deploy StackLight in the HA mode (recommended), add the StackLight label to minimum three nodes.

Note

You can configure node labels after deploying a machine. On the Machines page, click the More action icon in the last column of the required machine field and select Configure machine.

6. Click Create.

At this point, Container Cloud adds the new machine object to the specified managed cluster. And the Bare Metal Operator controller creates the relation to BareMetalHost with the labels matching the roles.

Provisioning of the newly created machine starts when the machine object is created and includes the following stages:

1. Creation of partitions on the local disks as required by the operating system and the Container Cloud architecture.
2. Configuration of the network interfaces on the host as required by the operating system and the Container Cloud architecture.
3. Installation and configuration of the Container Cloud LCM agent.

Now, proceed to Add a Ceph cluster.

Create a machine using CLI

This section describes a bare metal host and machine configuration using Mirantis Container Cloud CLI.

Deploy a machine to a specific bare metal host

A Kubernetes machine requires a dedicated bare metal host for deployment. The bare metal hosts are represented by the BareMetalHost objects in Kubernetes API. All BareMetalHost objects are labeled by the Operator when created. A label reflects the hardware capabilities of a host. As a result of labeling, all bare metal hosts are divided into three types: Control Plane, Worker, and Storage.
In some cases, you may need to deploy a machine to a specific bare metal host. This is especially useful when some of your bare metal hosts have different hardware configuration than the rest.

To deploy a machine to a specific bare metal host:

1. Log in to the host where your management cluster kubeconfig is located and where kubectl is installed.
2. Identify the bare metal host that you want to associate with the specific machine. For example, host host-1.

   kubectl get baremetalhost host-1 -o yaml

3. Add a label that will uniquely identify this host, for example, by the name of the host and machine that you want to deploy on it.

   Caution!
   Do not remove any existing labels from the BareMetalHost resource. For more details about labels, see BareMetalHost.

   kubectl edit baremetalhost host-1

Configuration example:

```
kind: BareMetalHost
metadata:
  name: host-1
  namespace: myProjectName
labels:
  kaas.mirantis.com/baremetalhost-id: host-1-worker-HW11-cad5
```

4. Create a new text file with the YAML definition of the Machine object, as defined in Machine.
5. Add a label selector that matches the label you have added to the BareMetalHost object in the previous step.

   Example:

```
kind: Machine
metadata:
  name: worker-HW11-cad5
  namespace: myProjectName
spec:
```

providerSpec:
  value:
    apiVersion: baremetal.k8s.io/v1alpha1
    kind: BareMetalMachineProviderSpec
    ...
    hostSelector:
      matchLabels:
        kaas.mirantis.com/baremetalhost-id: host-1-worker-HW11-cad5

6. Specify the details of the machine configuration in the object created in the previous step. For example:

   • Add a reference to a custom BareMetalHostProfile object, as defined in Machine.
   • Specify an override for the ordering and naming of the NICs for the machine. For details, see Override network interfaces naming and order.
   • If you use a specific L2 template for the machine, set the unique name or label of the corresponding L2 template in the L2templateSelector section of the Machine object.

7. Add the configured machine to the cluster:

   kubectl create -f worker-HW11-cad5.yaml

   Once done, this machine will be associated with the specified bare metal host.

Override network interfaces naming and order

An L2 template contains the ifMapping field that allows you to identify Ethernet interfaces for the template. The Machine object API enables the Operator to override the mapping from the L2 template by enforcing a specific order of names of the interfaces when applied to the template.

The field I2TemplateIfMappingOverride in the spec of the Machine object contains a list of interfaces names. The order of the interfaces names in the list is important because the L2Template object will be rendered with NICs ordered as per this list.

Note

Changes in the I2TemplateIfMappingOverride field will apply only once when the Machine and corresponding IpamHost objects are created. Further changes to I2TemplateIfMappingOverride will not reset the interfaces assignment and configuration.

Caution!

The I2TemplateIfMappingOverride field must contain the names of all interfaces of the bare metal host.
The following example illustrates how to include the override field to the Machine object. In this example, we configure the interface eno1, which is the second on-board interface of the server, to precede the first on-board interface eno0.

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Machine
metadata:
  finalizers:
  - foregroundDeletion
  - machine.cluster.sigs.k8s.io
labels:
  cluster.sigs.k8s.io/cluster-name: kaas-mgmt
  cluster.sigs.k8s.io/control-plane: "true"
  kaas.mirantis.com/provider: baremetal
  kaas.mirantis.com/region: region-one
spec:
  providerSpec:
    value:
      apiVersion: baremetal.k8s.io/v1alpha1
      hostSelector:
        matchLabels:
          baremetal: hw-master-0
      image: {}
      kind: BareMetalMachineProviderSpec
    l2TemplateIfMappingOverride:
    - eno1
    - eno0
    - enp0s1
    - enp0s2
```

As a result of the configuration above, when used with the example L2 template for bonds and bridges described in Create L2 templates, the eno1 interface will be used for the bm-pxe bridge, while the eno0 interface will be used to create subinterfaces for Kubernetes networks.

See also
Delete a machine

Add a Ceph cluster
After you add machines to your new bare metal managed cluster as described in Add a machine, you can create a Ceph cluster on top of this managed cluster using the Mirantis Container Cloud web UI.

The procedure below enables you to create a Ceph cluster with minimum three Ceph nodes that provides persistent volumes to the Kubernetes workloads in the managed cluster.
To create a Ceph cluster in the managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the required cluster name. The Cluster page with the Machines and Ceph clusters lists opens.
4. In the Ceph Clusters block, click Create Cluster.
5. Configure the Ceph cluster in the Create New Ceph Cluster wizard that opens:

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General settings</td>
<td>Name</td>
<td>The Ceph cluster name.</td>
</tr>
<tr>
<td></td>
<td>Cluster Network</td>
<td>Replication network for Ceph OSDs. Must match the corresponding values of the cluster L2Template object or the environment network values.</td>
</tr>
<tr>
<td></td>
<td>Public Network</td>
<td>Public network for Ceph data. Must match the corresponding values of the cluster L2Template object or the environment network values.</td>
</tr>
<tr>
<td></td>
<td>Enable OSDs LCM</td>
<td>Select to enable LCM for Ceph OSDs.</td>
</tr>
<tr>
<td>Machines /</td>
<td>Select machine</td>
<td>Select the name of the Kubernetes machine that will host the corresponding Ceph node in the Ceph cluster.</td>
</tr>
<tr>
<td>Machine #1-3</td>
<td>Manager, Monitor</td>
<td>Select the required Ceph services to install on the Ceph node.</td>
</tr>
<tr>
<td></td>
<td>Devices</td>
<td>Select the disk that Ceph will use.</td>
</tr>
</tbody>
</table>

Warning
Do not select the device for system services, for example, sda.

6. To add more Ceph nodes to the new Ceph cluster, click + next to any Ceph Machine title in the Machines tab. Configure a Ceph node as required.
Warning
Do not add more than 3 Manager and/or Monitor services to the Ceph cluster.

7. After you add and configure all nodes in your Ceph cluster, click Create. Once done, verify your Ceph cluster as described in Verify Ceph components.

Delete a managed cluster
Deleting a managed cluster does not require a preliminary deletion of the machines running on the cluster.

To delete a baremetal-based managed cluster:

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required cluster and select Delete.
4. Verify the list of machines to be removed. Confirm the deletion.
5. Optional. If you do not plan to reuse the credentials of the deleted cluster, delete them:
   1. In the Credentials tab, click the Delete credential action icon next to the name of the credentials to be deleted.
   2. Confirm the deletion.

Warning
You can delete credentials only after deleting the managed cluster they relate to.

Deleting a cluster automatically frees up the resources allocated for this cluster, for example, instances, load balancers, networks, floating IPs, and so on.

Advanced networking configuration
By default, Mirantis Container Cloud configures a single interface on the cluster nodes, leaving all other physical interfaces intact.

With L2 networking templates, you can create advanced host networking configurations for your clusters. For example, you can create bond interfaces on top of physical interfaces on the host or use multiple subnets to separate different types of network traffic.

You can use several host-specific L2 templates per one cluster to support different hardware configurations. For example, you can create L2 templates with different number and layout of NICs to be applied to the specific machines of one cluster.
When you create a baremetal-based project, the exemplary templates with the ipam/PreInstalledL2Template label are copied to this project. These templates are preinstalled during the management cluster bootstrap.

Follow the procedures below to create L2 templates for your managed clusters.

Create subnets

Before creating an L2 template, ensure that you have the required subnets that can be used in the L2 template to allocate IP addresses for the managed cluster nodes. Where required, create a number of subnets for a particular project using the Subnet CR. A subnet has three logical scopes:

- **global** - CR uses the default namespace. A subnet can be used for any cluster located in any project.
- **namespaced** - CR uses the namespace that corresponds to a particular project where managed clusters are located. A subnet can be used for any cluster located in the same project.
- **cluster** - CR uses the namespace where the referenced cluster is located. A subnet is only accessible to the cluster that L2Template.spec.clusterRef refers to. The Subnet objects with the cluster scope will be created for every new cluster.

You can have subnets with the same name in different projects. In this case, the subnet that has the same project as the cluster will be used. One L2 template may often reference several subnets, those subnets may have different scopes in this case.

The IP address objects (IPaddr CR) that are allocated from subnets always have the same project as their corresponding IpamHost objects, regardless of the subnet scope.

To create subnets:

1. Log in to a local machine where your management cluster kubeconfig is located and where kubectl is installed.

```
Note
The management cluster kubeconfig is created during the last stage of the management cluster bootstrap.
```

2. Create the subnet.yaml file with a number of global or namespaced subnets:

   
   `kubectl --kubeconfig <pathToManagementClusterKubeconfig> apply -f <SubnetFileName.yaml>`

```
Note
In the command above and in the steps below, substitute the parameters enclosed in angle brackets with the corresponding values.
```
Example of a subnet.yaml file:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: Subnet
metadata:
  name: demo
  namespace: demo-namespace
spec:
  cidr: 10.11.0.0/24
  gateway: 10.11.0.9
includeRanges:
- 10.11.0.5-10.11.0.70
nameservers:
- 172.18.176.6
```

Specification fields of the Subnet object

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cidr (singular)</td>
<td>A valid IPv4 CIDR, for example, 10.11.0.0/24.</td>
</tr>
<tr>
<td>includeRanges (list)</td>
<td>A list of IP address ranges within the given CIDR that should be used in the allocation of IPs for nodes (excluding the gateway address). The IPs outside the given ranges will not be used in the allocation. Each element of the list can be either an interval 10.11.0.5-10.11.0.70 or a single address 10.11.0.77. In the example above, the addresses 10.11.0.5-10.11.0.70 (excluding the gateway address 10.11.0.9) will be allocated for nodes. The includeRanges parameter is mutually exclusive with excludeRanges.</td>
</tr>
<tr>
<td>excludeRanges (list)</td>
<td>A list of IP address ranges within the given CIDR that should not be used in the allocation of IPs for nodes. The IPs within the given CIDR but outside the given ranges will be used in the allocation (excluding gateway address). Each element of the list can be either an interval 10.11.0.5-10.11.0.70 or a single address 10.11.0.77. The excludeRanges parameter is mutually exclusive with includeRanges.</td>
</tr>
<tr>
<td>useWholeCidr (boolean)</td>
<td>If set to true, the subnet address (10.11.0.0 in the example above) and the broadcast address (10.11.0.255 in the example above) are included into the address allocation for nodes. Otherwise, (false by default), the subnet address and broadcast address will be excluded from the address allocation.</td>
</tr>
<tr>
<td>gateway (singular)</td>
<td>A valid gateway address, for example, 10.11.0.9.</td>
</tr>
<tr>
<td>nameservers (list)</td>
<td>A list of the IP addresses of name servers. Each element of the list is a single address, for example, 172.18.176.6.</td>
</tr>
</tbody>
</table>
Caution!

The subnet for the PXE network is automatically created during deployment and must contain the ipam/DefaultSubnet: "1" label. Each bare metal region must have only one subnet with this label.

3. Verify that the subnet is successfully created:

   kubectl get subnet kaas-mgmt -oyaml

In the system output, verify the status fields of the subnet.yaml file using the table below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statusMessage</td>
<td>Contains a short state description and a more detailed one if applicable.</td>
</tr>
<tr>
<td></td>
<td>The short status values are as follows:</td>
</tr>
<tr>
<td></td>
<td>• OK - operational.</td>
</tr>
<tr>
<td></td>
<td>• ERR - non-operational. This status has a detailed description,</td>
</tr>
<tr>
<td></td>
<td>for example, ERR: Wrong includeRange for CIDR....</td>
</tr>
<tr>
<td>cidr</td>
<td>Reflects the actual CIDR, has the same meaning as spec.cidr.</td>
</tr>
<tr>
<td>gateway</td>
<td>Reflects the actual gateway, has the same meaning as spec.gateway.</td>
</tr>
<tr>
<td>nameservers</td>
<td>Reflects the actual name servers, has same meaning as spec.nameservers.</td>
</tr>
<tr>
<td>ranges</td>
<td>Specifies the address ranges that are calculated using the fields from spec:</td>
</tr>
<tr>
<td></td>
<td>cidr, includeRanges, excludeRanges, gateway, useWholeCidr. These ranges</td>
</tr>
<tr>
<td></td>
<td>are directly used for nodes IP allocation.</td>
</tr>
<tr>
<td>lastUpdate</td>
<td>Includes the date and time of the latest update of the Subnet RC.</td>
</tr>
<tr>
<td>allocatable</td>
<td>Includes the number of currently available IP addresses that can be allocated</td>
</tr>
<tr>
<td></td>
<td>for nodes from the subnet.</td>
</tr>
<tr>
<td>allocatedIPs</td>
<td>Specifies the list of IPv4 addresses with the corresponding IPAddr object</td>
</tr>
<tr>
<td></td>
<td>IDs that were already allocated from the subnet.</td>
</tr>
<tr>
<td>capacity</td>
<td>Contains the total number of IP addresses being held by ranges that equals</td>
</tr>
<tr>
<td></td>
<td>to a sum of the allocatable and allocatedIPs parameters values.</td>
</tr>
<tr>
<td>versionIpam</td>
<td>Contains the version of the kaas-ipam component that made the latest changes</td>
</tr>
<tr>
<td></td>
<td>to the Subnet RC.</td>
</tr>
</tbody>
</table>

Example of a successfully created subnet:
apiVersion: ipam.mirantis.com/v1alpha1
kind: Subnet
metadata:
  labels:
    ipam/UID: 6039758f-23ee-40ba-8c0f-61c01b0ac863
    kaas.mirantis.com/provider: baremetal
    kaas.mirantis.com/region: region-one
  name: kaas-mgmt
  namespace: default
spec:
cidr: 10.0.0.0/24
excludeRanges:
  - 10.0.0.100
  - 10.0.0.101-10.0.0.120
gateway: 10.0.0.1
includeRanges:
  - 10.0.0.50-10.0.0.90
nameservers:
  - 172.18.176.6
status:
  allocatable: 38
allocatedIPs:
  - 10.0.0.50:0b50774f-ffed-11ea-84c7-0242c0a85b02
  - 10.0.0.51:1422e651-ffed-11ea-84c7-0242c0a85b02
  - 10.0.0.52:1d19912c-ffed-11ea-84c7-0242c0a85b02
capacity: 41
CIDR: 10.0.0.0/24
gateway: 10.0.0.1
lastUpdate: "2020-09-26T11:40:44Z"
nameservers:
  - 172.18.176.6
ranges:
  - 10.0.0.50-10.0.0.90
statusMessage: OK
versionIpam: v3.0.999-20200807-130909-44151f8

4. Proceed to creating an L2 template for one or multiple managed clusters as described in Create L2 templates.

See also
Automate multiple subnet creation using SubnetPool

Automate multiple subnet creation using SubnetPool
Caution!

This feature is available starting from the Container Cloud release 2.2.0.

Before creating an L2 template, ensure that you have the required subnets that can be used in the L2 template to allocate IP addresses for the managed cluster nodes. You can also create multiple subnets using the SubnetPool object to separate different types of network traffic. SubnetPool allows for automatic creation of Subnet objects that will consume blocks from the parent SubnetPool CIDR IP address range. The SubnetPool blockSize setting defines the IP address block size to allocate to each child Subnet. SubnetPool has a global scope, so any SubnetPool can be used to create the Subnet objects for any namespace and for any cluster.

To automate multiple subnet creation using SubnetPool:

1. Log in to a local machine where your management cluster kubeconfig is located and where kubectl is installed.

   Note
   The management cluster kubeconfig is created during the last stage of the management cluster bootstrap.

2. Create the subnetpool.yaml file with a number of subnet pools:

   Note
   You can define either or both subnets and subnet pools, depending on the use case. A single L2 template can use either or both subnets and subnet pools.

   kubectl --kubeconfig <pathToManagementClusterKubeconfig> apply -f <SubnetFileName.yaml>

   Note
   In the command above and in the steps below, substitute the parameters enclosed in angle brackets with the corresponding values.

Example of a subnetpool.yaml file:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: SubnetPool
metadata:
  name: kaas-mgmt
```
namespace: default
labels:
  kaas.mirantis.com/provider: baremetal
  kaas.mirantis.com/region: region-one
spec:
cidr: 10.10.0.0/16
blockSize: /25
nameservers:
  - 172.18.176.6
gatewayPolicy: first

For the specification fields description of the SubnetPool object, see SubnetPool spec.

3. Verify that the subnet pool is successfully created:

    kubectl get subnetpool kaas-mgmt -oyaml

    In the system output, verify the status fields of the subnetpool.yaml file. For the status fields description of the SubnetPool object, see SubnetPool status.

4. Proceed to creating an L2 template for one or multiple managed clusters as described in Create L2 templates. In this procedure, select the exemplary L2 template for multiple subnets that contains the l3Layout section.

    Caution!

    Using the l3Layout section, define all subnets of a cluster. Otherwise, do not use the l3Layout section. Defining only part of subnets is not allowed.
Create L2 templates

After you create subnets for one or more managed clusters or projects as described in Create subnets or Automate multiple subnet creation using SubnetPool, follow the procedure below to create L2 templates for a managed cluster. This procedure contains exemplary L2 templates for the following use cases:
L2 template example with bonds and bridges

This section contains an exemplary L2 template that demonstrates how to set up bonds and bridges on hosts for your managed clusters as described in Create L2 templates.

Example of an L2 template with interfaces bonding:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: L2Template
metadata:
  name: test-managed
  namespace: managed-ns
spec:
  clusterRef: child-cluster
  autoIfMappingPrio:
    - provision
    - eno
    - ens
    - enp
  npTemplate: |
    version: 2
    ethernets:
      ten10gbe0s0:
        dhcp4: false
        dhcp6: false
        match:
          macaddress: {{mac 2}}
          set-name: {{nic 2}}
      ten10gbe0s1:
        dhcp4: false
        dhcp6: false
        match:
          macaddress: {{mac 3}}
          set-name: {{nic 3}}
  bonds:
    bond0:
      interfaces:
      - ten10gbe0s0
      - ten10gbe0s1
  bridges:
    bm-ceph:
      interfaces: [bond0]
      addresses:
      - {{ip "bm-ceph:demo"}}
```
L2 template example for automatic multiple subnet creation

Caution!

This feature is available starting from the Container Cloud release 2.2.0.

This section contains an exemplary L2 template for automatic multiple subnet creation as described in Automate multiple subnet creation using SubnetPool. This template also contains the L3Layout section that allows defining the Subnet scopes and enables optional auto-creation of the Subnet objects from the SubnetPool objects.

For details on how to create L2 templates, see Create L2 templates.

Example of an L2 template for multiple subnets:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: L2Template
metadata:
  name: test-managed
  namespace: managed-ns
spec:
  clusterRef: child-cluster
  autoIfMappingPrio:
    - provision
    - eno
    - ens
    - enp
l3Layout:
  - subnetName: pxe-subnet
    scope: global
  - subnetName: subnet-1
    subnetPool: kaas-mgmt
    scope: namespace
  - subnetName: subnet-2
    subnetPool: kaas-mgmt
    scope: cluster
npTemplate:
  version: 2
  ethernets:
    onboard1gbe0:
      dhcp4: false
      dhcp6: false
      match:
        macaddress: {{mac 0}}
      set-name: {{nic 0}}
      addresses:
        - {{ip "0:pxe-subnet"}}
```
nameservers:
  addresses: {{nameservers_from_subnet "pxe-subnet"}}
gateway4: {{gateway_from_subnet "pxe-subnet"}}
onboard1gbe1:
dhcp4: false
dhcp6: false
match:
  macaddress: {{mac 1}}
  set-name: {{nic 1}}
ten10gbe0s0:
dhcp4: false
dhcp6: false
match:
  macaddress: {{mac 2}}
  set-name: {{nic 2}}
  addresses:
    - {{ip "2:subnet-1"}}
ten10gbe0s1:
dhcp4: false
dhcp6: false
match:
  macaddress: {{mac 3}}
  set-name: {{nic 3}}
  addresses:
    - {{ip "3:subnet-2"}}

In the template above, the following networks are defined in the l3Layout section:

- **pxe-subnet** - global PXE network that already exists. A subnet name must refer to the PXE subnet created for the region.
- **subnet-1** - unless already created, this subnet will be created from the kaas-mgmt subnet pool. The subnet name must be unique within the project. This subnet is shared between the project clusters.
- **subnet-2** - will be created from the kaas-mgmt subnet pool. This subnet has the cluster scope. Therefore, the real name of the Subnet CR object consists of the subnet name defined in l3Layout and the cluster UID. But the npTemplate section of the L2 template must contain only the subnet name defined in l3Layout. The subnets of the cluster scope are not shared between clusters.

**Caution!**

Using the l3Layout section, define all subnets of a cluster. Otherwise, do not use the l3Layout section. Defining only part of subnets is not allowed.
To create an L2 template for a new managed cluster:

1. Log in to a local machine where your management cluster kubeconfig is located and where kubectl is installed.

   Note
   The management cluster kubeconfig is created during the last stage of the management cluster bootstrap.

2. Inspect the existing L2 templates to select the one that fits your deployment:

   ```
   kubectl --kubeconfig <pathToManagementClusterKubeconfig> \
   get l2template -n <ProjectNameForNewManagedCluster>
   ```

3. Create an L2 YAML template specific to your deployment using one of the exemplary templates:
   - L2 template example with bonds and bridges
   - L2 template example for automatic multiple subnet creation

   Note
   You can create several L2 templates with different configurations to be applied to different nodes of the same cluster. In this case:
   - First create the default L2 template for a cluster. It will be used for machines that do not have L2templateSelector.
   - Verify that the unique ipam/DefaultForCluster label is added to the first L2 template of the cluster.
   - Set a unique name and add a unique label to the metadata section of each L2 template of the cluster.
   - To select a particular L2 template for a machine, use either the L2 template name or label in the L2templateSelector section of the corresponding machine configuration file.
     
     If you use an L2 template for only one machine, set name. For a group of machines, set label.
     
     For details about configuration of machines, see Deploy a machine to a specific bare metal host.

4. Add or edit the mandatory parameters in the new L2 template. The following tables provide the description of the mandatory and the l3Layout section parameters in the example templates mentioned in the previous step.
## L2 template mandatory parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterRef</td>
<td>References the Cluster object that this template is applied to. The default value is used to apply the given template to all clusters in the corresponding project, unless an L2 template that references a specific cluster name exists.</td>
</tr>
</tbody>
</table>

**Caution!**

- A cluster can be associated with only one template.
- An L2 template must have the same namespace as the referenced cluster.
- A project can have only one default L2 template.

<table>
<thead>
<tr>
<th>ifMapping or autoIfMappingPrio</th>
<th>• ifMapping is a list of interface names for the template. The interface mapping is defined globally for all bare metal hosts in the cluster but can be overridden at the host level, if required, by editing the IpamHost object for a particular host.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• autoIfMappingPrio is a list of prefixes, such as eno, ens, and so on, to match the interfaces to automatically create a list for the template. If you are not aware of any specific ordering of interfaces on the nodes, use the default ordering from Predictable Network Interfaces Names specification for systemd. You can also override the default NIC list per host using the IfMappingOverride parameter of the corresponding IpamHost. The provision value corresponds to the network interface that was used to provision a node. Usually, it is the first NIC found on a particular node. It is defined explicitly to ensure that this interface will not be reconfigured accidentally.</td>
</tr>
</tbody>
</table>
A netplan-compatible configuration with special lookup functions that defines the networking settings for the cluster hosts, where physical NIC names and details are parameterized. This configuration will be processed using Go templates. Instead of specifying IP and MAC addresses, interface names, and other network details specific to a particular host, the template supports use of special lookup functions. These lookup functions, such as nic, mac, ip, and so on, return host-specific network information when the template is rendered for a particular host. For details about netplan, see the official netplan documentation.

Caution!

All rules and restrictions of the netplan configuration also apply to L2 templates. For details, see the official netplan documentation.

For more details about the L2Template custom resource (CR), see the L2Template API section.

### I3Layout section parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subnetName</td>
<td>Name of the Subnet object that will be used in the npTemplate section to allocate IP addresses from. All Subnet names must be unique across a single L2 template.</td>
</tr>
<tr>
<td>subnetPool</td>
<td>Optional. Default: none. Name of the parent SubnetPool object that will be used to create a Subnet object with a given subnetName and scope. If a corresponding Subnet object already exists, nothing will be created and the existing object will be used. If no SubnetPool is provided, no new Subnet object will be created.</td>
</tr>
</tbody>
</table>
| scope         | Logical scope of the Subnet object with a corresponding subnetName. Possible values:  
  - global - the Subnet object is accessible globally, for any Container Cloud project and cluster in the region, for example, the PXE subnet.
  - namespace - the Subnet object is accessible within the same project and region where the L2 template is defined.
  - cluster - the Subnet object is only accessible to the cluster that L2Template.spec.clusterRef refers to. The Subnet objects with the cluster scope will be created for every new cluster. |

The following table describes the main lookup functions for an L2 template.
### Lookup function

<table>
<thead>
<tr>
<th>Lookup function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{nic N}</code></td>
<td>Name of a NIC number N. NIC numbers correspond to the interface mapping list.</td>
</tr>
<tr>
<td><code>{mac N}</code></td>
<td>MAC address of a NIC number N registered during a host hardware inspection.</td>
</tr>
<tr>
<td><code>{ip &quot;N:subnet-a&quot;}</code></td>
<td>IP address and mask for a NIC number N. The address will be auto-allocated from the given subnet if the address does not exist yet.</td>
</tr>
<tr>
<td><code>{ip &quot;br0:subnet-x&quot;}</code></td>
<td>IP address and mask for a virtual interface, “br0” in this example. The address will be auto-allocated from the given subnet if the address does not exist yet.</td>
</tr>
<tr>
<td><code>{gateway_from_subnet &quot;subnet-a&quot;}</code></td>
<td>IPv4 default gateway address from the given subnet.</td>
</tr>
<tr>
<td><code>{nameservers_from_subnet &quot;subnet-a&quot;}</code></td>
<td>List of the IP addresses of name servers from the given subnet.</td>
</tr>
</tbody>
</table>

#### Note

Every subnet referenced in an L2 template can have either a global or namespaced scope. In the latter case, the subnet must exist in the same project where the corresponding cluster and L2 template are located.

5. Add the L2 template to your management cluster:

   kubectl --kubeconfig <pathToManagementClusterKubeconfig> apply -f <pathToL2TemplateYamlFile>

6. Optional. Further modify the template:

   kubectl --kubeconfig <pathToManagementClusterKubeconfig> \
   -n <ProjectNameForNewManagedCluster> edit l2template <L2templateName>

7. Proceed with creating a managed cluster as described in Create a managed cluster. The resulting L2 template will be used to render the netplan configuration for the managed cluster machines.

The workflow of the netplan configuration using an L2 template is as follows:

1. The kaas-ipam service uses the data from BareMetalHost, the L2 template, and subnets to generate the netplan configuration for every cluster machine.

2. The generated netplan configuration is saved in the status.netconfigV2 section of the IpamHost resource. If the status.l2RenderResult field of the IpamHost resource is OK, the
configuration was rendered in the IpamHost resource successfully. Otherwise, the status contains an error message.

3. The baremetal-provider service copies data from the status.netconfigV2 of IpamHost to the Spec.StatetItemsOverwrites[‘deploy’][‘bm_ipam_netconfigv2’] parameter of LCMMachine.

4. The lcm-agent service on every host synchronizes the LCMMachine data to its host. The lcm-agent service runs a playbook to update the netplan configuration on the host during the pre-download and deploy phases.

See also
Create a custom bare metal host profile

Create a custom bare metal host profile
The bare metal host profile is a Kubernetes custom resource. It allows the operator to define how the storage devices and the operating system are provisioned and configured.

This section describes the bare metal host profile default settings and configuration of custom profiles for managed clusters using Mirantis Container Cloud API. This procedure also applies to a management cluster with a few differences described in Deployment Guide: Customize the default bare metal host profile.
Default configuration of the host system storage

The default host profile requires three storage devices in the following strict order:

1. **Boot device and operating system storage**
   This device contains boot data and operating system data. It is partitioned using the GUID Partition Table (GPT) labels. The root file system is an ext4 file system created on top of an LVM logical volume. For a detailed layout, refer to the table below.

2. **Local volumes device**
   This device contains an ext4 file system with directories mounted as persistent volumes to Kubernetes. These volumes are used by the Mirantis Container Cloud services to store its data, including monitoring and identity databases.

3. **Ceph storage device**
   This device is used as a Ceph datastore or Ceph OSD.

The following table summarizes the default configuration of the host system storage set up by the Container Cloud bare metal management.

<table>
<thead>
<tr>
<th>Device/partition</th>
<th>Name/Mount point</th>
<th>Recommended size, GB</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sda1</td>
<td>bios_grub</td>
<td>4 MiB</td>
<td>The mandatory GRUB boot partition required for non-UEFI systems.</td>
</tr>
<tr>
<td>/dev/sda2</td>
<td>UEFI -&gt; /boot/efi</td>
<td>0.2 GiB</td>
<td>The boot partition required for the UEFI boot mode.</td>
</tr>
<tr>
<td>/dev/sda3</td>
<td>config-2</td>
<td>64 MiB</td>
<td>The mandatory partition for the cloud-init configuration. Used during the first host boot for initial configuration.</td>
</tr>
<tr>
<td>/dev/sda4</td>
<td>lvm_root_part</td>
<td>100% of the remaining free space in the LVM volume group</td>
<td>The main LVM physical volume that is used to create the root file system.</td>
</tr>
<tr>
<td>/dev/sdb</td>
<td>lvm_lvp_part -&gt;</td>
<td>100% of the remaining free space in the LVM volume group</td>
<td>The LVM physical volume that is used to create the file system for LocalVolumeProvisioner.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>/dev/sdc</td>
<td>-</td>
<td>100% of the remaining free space in the LVM volume group</td>
<td>Clean raw disk that will be used for the Ceph storage back end.</td>
</tr>
</tbody>
</table>

If required, you can customize the default host storage configuration. For details, see Create a custom host profile.
Create a custom host profile

In addition to the default BareMetalHostProfile object installed with Mirantis Container Cloud, you can create custom profiles for managed clusters using Container Cloud API.

**Note**

The procedure below also applies to the Container Cloud management clusters.

To create a custom bare metal host profile:

1. Select from the following options:
   - For a management cluster, log in to the bare metal seed node that will be used to bootstrap the management cluster.
   - For a managed cluster, log in to the local machine where you management cluster kubeconfig is located and where kubectl is installed.

**Note**

The management cluster kubeconfig is created automatically during the last stage of the management cluster bootstrap.

2. Select from the following options:
   - For a management cluster, open templates/bm/baremetalhostprofiles.yaml.template for editing.
   - For a managed cluster, create a new bare metal host profile under the templates/bm directory.

3. Edit the host profile using the example template below to meet your hardware configuration requirements:

```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalHostProfile
metadata:
  name: <PROFILE_NAME>
  namespace: <PROJECT_NAME>
spec:
devices:
  # From the HW node, obtain the first device, which size is at least 60GiB
  - device:
      minSizeGiB: 60
      wipe: true
      partitions:
        - name: bios_grub
```

©2021, Mirantis Inc.
partflags:
  - bios_grub
  sizeGiB: 0.00390625
  wipe: true
- name: uefi
  partflags:
  - esp
    sizeGiB: 0.2
    wipe: true
  - name: config-2
    sizeGiB: 0.0625
    wipe: true
  - name: lvm_root_part
    sizeGiB: 0
    wipe: true

# From the HW node, obtain the second device, which size is at least 60Gib
# If a device exists but does not fit the size, 
# the BareMetalHostProfile will not be applied to the node
  - device:
    minSizeGiB: 30
    wipe: true
# From the HW node, obtain the disk device with the exact name
  - device:
    byName: /dev/nvme0n1
    minSizeGiB: 30
    wipe: true
  partitions:
  - name: lvm_lvp_part
    sizeGiB: 0
    wipe: true

# Example of wiping a device w/o partitioning it.
# Mandatory for the case when a disk is supposed to be used for Ceph back end 
# later
  - device:
    byName: /dev/sde
    wipe: true
fileSystems:
  - fileSystem: vfat
    partition: config-2
  - fileSystem: vfat
    mountPoint: /boot/efi
    partition: uefi
  - fileSystem: ext4
    logicalVolume: root
    mountPoint: /
  - fileSystem: ext4
    logicalVolume: lvp
    mountPoint: /mnt/local-volumes/
logicalVolumes:
- name: root
  sizeGiB: 0
  vg: lvm_root
- name: lvp
  sizeGiB: 0
  vg: lvm_lvp

postDeployScript: |
  #!/bin/bash -ex
  echo $(date) 'post_deploy_script done' >> /root/post_deploy_done

preDeployScript: |
  #!/bin/bash -ex
  echo $(date) 'pre_deploy_script done' >> /root/pre_deploy_done

volumeGroups:
- devices:
  - partition: lvm_root_part
    name: lvm_root
- devices:
  - partition: lvm_lvp_part
    name: lvm_lvp

grubConfig:
  defaultGrubOptions:
  - GRUB_DISABLE_RECOVERY="true"
  - GRUB_PRELOAD_MODULES=lvm
  - GRUB_TIMEOUT=20

kernelParameters:
  sysctl:
    kernel.panic: "900"
    kernel.dmesg_restrict: "1"
    kernel.core_uses_pid: "1"
    fs.file-max: "922372036854775807"
    fs.aio-max-nr: "1048576"
    fs.inotify.max_user_instances: "4096"
    vm.max_map_count: "262144"

4. Add or edit the mandatory parameters in the new BareMetalHostProfile object. For the parameters description, see API: BareMetalHostProfile spec.

5. Select from the following options:

- For a management cluster, proceed with the cluster bootstrap procedure as described in Deployment Guide: Bootstrap a management cluster.
- For a managed cluster:
  1. Add the bare metal host profile to your management cluster:

```
kubectl --kubeconfig <pathToManagementClusterKubeconfig> -n <projectName> apply -f <pathToBareMetalHostProfileFile>
```

  2. If required, further modify the host profile:

```
kubectl --kubeconfig <pathToManagementClusterKubeconfig> -n <projectName> edit baremetalhostprofile <hostProfileName>
```
3. Proceed with creating a managed cluster as described in Create a managed cluster.

Enable huge pages in a host profile

The BareMetalHostProfile API allows configuring a host to use the huge pages feature of the Linux kernel on managed clusters.

**Note**

Huge pages is a mode of operation of the Linux kernel. With huge pages enabled, the kernel allocates the RAM in bigger chunks, or pages. This allows a KVM (kernel-based virtual machine) and VMs running on it to use the host RAM more efficiently and improves the performance of VMs.

To enable huge pages in a custom bare metal host profile for a managed cluster:

1. Log in to the local machine where you management cluster kubeconfig is located and where kubectl is installed.

   **Note**
   
   The management cluster kubeconfig is created automatically during the last stage of the management cluster bootstrap.

2. Open for editing or create a new bare metal host profile under the templates/bm/ directory.

3. Edit the grubConfig section of the host profile spec using the example below to configure the kernel boot parameters and enable huge pages:

   ```yaml
   spec:
     grubConfig:
       defaultGrubOptions:
         - GRUB_DISABLE_RECOVERY="true"
         - GRUB_PRELOAD_MODULES=lvm
         - GRUB_TIMEOUT=20
         - GRUB_CMDLINE_LINUX_DEFAULT="hugepagesz=1G hugepages=N"
   ```

   The example configuration above will allocate N huge pages of 1 GB each on the server boot. The last hugepagesz parameter value is default unless default_hugepagesz is defined. For details about possible values, see official Linux kernel documentation.

4. Add the bare metal host profile to your management cluster:

   ```bash
   kubectl --kubeconfig <pathToManagementClusterKubeconfig> -n <projectName> apply -f <pathToBareMetalHostProfileFile>
   ```
5. If required, further modify the host profile:

```
kubectl --kubeconfig <pathToManagementClusterKubeconfig> -n <projectName> edit baremetalhostprofile <hostProfileName>
```

6. Proceed with creating a managed cluster as described in Create a managed cluster.

**Create and operate an OpenStack-based managed cluster**

After bootstrapping your OpenStack-based Mirantis Container Cloud management cluster as described in Deployment Guide: Deploy an OpenStack-based management cluster, you can create the OpenStack-based managed clusters using the Container Cloud web UI.

**Create a managed cluster**

This section describes how to create an OpenStack-based managed cluster using the Mirantis Container Cloud web UI of the OpenStack-based management cluster.

To create an OpenStack-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the SSH Keys tab, click Add SSH Key to upload the public SSH key that will be used for the OpenStack VMs creation.
4. In the Credentials tab:

   1. Click Add Credential to add your OpenStack credentials. You can either upload your OpenStack clouds.yaml configuration file or fill in the fields manually.
   2. Verify that the new credentials status is Ready. If the status is Error, hover over the status to determine the reason of the issue.
5. In the Clusters tab, click Create Cluster and fill out the form with the following parameters as required:

   1. Configure general settings and the Kubernetes parameters:

```
<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General settings</td>
<td>Name</td>
<td>Cluster name</td>
</tr>
<tr>
<td></td>
<td>Provider</td>
<td>Select OpenStack</td>
</tr>
<tr>
<td></td>
<td>Provider credential</td>
<td>From the drop-down list, select the OpenStack credentials name that you created in the previous step.</td>
</tr>
</tbody>
</table>
```
<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StackLight</td>
<td>Enable Monitoring</td>
<td>Selected by default. Deselect to skip StackLight deployment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can also enable, disable, or configure StackLight parameters after deploying a managed cluster. For details, see Change a cluster configuration or Configure StackLight.</td>
</tr>
<tr>
<td></td>
<td>Enable Logging</td>
<td>Select to deploy the StackLight logging stack. For details about the logging components, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td><strong>HA Mode</strong></td>
<td>Select to enable StackLight monitoring in the HA mode. For the differences between HA and non-HA modes, see Reference Architecture: StackLight deployment architecture.</td>
<td></td>
</tr>
<tr>
<td><strong>Elasticsearch Retention Time</strong></td>
<td>The Elasticsearch logs retention period in Logstash.</td>
<td></td>
</tr>
<tr>
<td><strong>Persistent Volume Claim Size</strong></td>
<td>The Elasticsearch persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td><strong>Prometheus Retention Time</strong></td>
<td>The Prometheus database retention period.</td>
<td></td>
</tr>
<tr>
<td><strong>Retention Size</strong></td>
<td>The Prometheus database retention size.</td>
<td></td>
</tr>
<tr>
<td><strong>Persistent Volume Claim Size</strong></td>
<td>The Prometheus persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td><strong>Enable Watchdog Alert</strong></td>
<td>Select to enable the Watchdog alert that fires as long as the entire alerting pipeline is functional.</td>
<td></td>
</tr>
<tr>
<td><strong>Custom Alerts</strong></td>
<td>Specify alerting rules for new custom alerts or upload a YAML file in the following exemplary format:</td>
<td></td>
</tr>
</tbody>
</table>
| | - **alert**: HighErrorRate  
| |  **expr**: job:request_latency_seconds:mean5m{job="myjob"} > 0.5  
| |  **for**: 10m  
| |  **labels**:  
| |  **severity**: page  
| |  **annotations**:  
| |  **summary**: High request latency |
| **StackLight Email Alerts** | Select to enable the StackLight email alerts. |
| **Send Resolved** | Select to enable notifications about resolved StackLight alerts. |
| **Require TLS** | Select to enable transmitting emails through TLS. |
| StackLight Slack Alerts | Email alerts configuration for StackLight | Fill out the following email alerts parameters as required:
- To - the email address to send notifications to.
- From - the sender address.
- SmartHost - the SMTP host through which the emails are sent.
- Authentication username - the SMTP user name.
- Authentication password - the SMTP password.
- Authentication identity - the SMTP identity.
- Authentication secret - the SMTP secret. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Slack alerts</td>
<td>Send Resolved</td>
<td>Select to enable the StackLight Slack alerts.</td>
</tr>
</tbody>
</table>
|                         | Slack alerts configuration for StackLight | Fill out the following Slack alerts parameters as required:
- API URL - The Slack webhook URL.
- Channel - The channel to send notifications to, for example, #channel-for-alerts. |

6. Click Create.
To view the deployment status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the deployment is complete.

7. Proceed with Add a machine.

See also
Delete a managed cluster

Add a machine
After you create a new OpenStack-based Mirantis Container Cloud managed cluster as described in Create a managed cluster, proceed with adding machines to this cluster using the Container Cloud web UI.

You can also use the instruction below to scale up an existing managed cluster.

To add a machine to an OpenStack-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.

3. In the Clusters tab, click the required cluster name. The cluster page with Machines list opens.

4. On the cluster page, click Create Machine.

5. Fill out the form with the following parameters as required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Specify the number of machines to create. The required minimum number of machines is three for the manager nodes HA and two for the Container Cloud workloads. Select Manager or Worker to create a Kubernetes manager or worker node.</td>
</tr>
<tr>
<td>Flavor</td>
<td>From the drop-down list, select the required hardware configuration for the machine. The list of available flavors corresponds to the one in your OpenStack environment. For the hardware requirements, see: Reference Architecture: Requirements for an OpenStack-based cluster.</td>
</tr>
<tr>
<td>Image</td>
<td>From the drop-down list, select the cloud image with Ubuntu 18.04. If you do not have this image in the list, add it to your OpenStack environment using the Horizon web UI by downloading the image from the Ubuntu official website.</td>
</tr>
<tr>
<td>Availability zone</td>
<td>From the drop-down list, select the availability zone from which the new machine will be launched.</td>
</tr>
</tbody>
</table>
Select the required node labels for the machine to run certain components on a specific node. For example, for the StackLight nodes that run Elasticsearch and require more resources than a standard node, select the StackLight label. The list of available node labels is obtained from your current Cluster release.

Caution!
If you deploy StackLight in the HA mode (recommended), add the StackLight label to minimum three nodes.

Note
You can configure node labels after deploying a machine. On the Machines page, click the More action icon in the last column of the required machine field and select Configure machine.

6. Click Create.
7. Repeat the steps above for the remaining machines.

You can monitor the machine status in the Managers or Workers columns on the Clusters page. Once the status changes to Ready, the deployment of the managed cluster components on this machine is complete.

The machine creation starts with the Provision status. During provisioning, the machine is not expected to be accessible since its infrastructure (VM, network, and so on) is being created.

Other machine statuses are the same as the LCMMachine object states described in Reference Architecture: LCM controller.

8. Verify the status of the cluster nodes as described in Connect to a Mirantis Container Cloud cluster.

Warning
The operational managed cluster should contain minimum 3 Kubernetes manager nodes and 2 Kubernetes worker nodes. To meet the etcd quorum and to prevent the deployment failure, scaling down of the manager nodes is prohibited.
Delete a managed cluster
Deleting a managed cluster does not require a preliminary deletion of VMs that run on this cluster.

To delete an OpenStack-based managed cluster:

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required cluster and select Delete.
4. Verify the list of machines to be removed. Confirm the deletion.
   Deleting a cluster automatically frees up the resources allocated for this cluster, for example, instances, load balancers, networks, floating IPs.
5. If the cluster deletion hangs and the The cluster is being deleted message does not disappear for a while:
   1. Expand the menu of the tab with your username.
   2. Click Download kubeconfig to download kubeconfig of your management cluster.
   3. Log in to any local machine with kubectl installed.
   4. Copy the downloaded kubeconfig to this machine.
   5. Run the following command:

```
kubectl --kubeconfig <KUBECONFIG_PATH> edit -n <PROJECT_NAME> cluster <MANAGED_CLUSTER_NAME>
```
   6. Edit the opened kubeconfig by removing the following lines:

```
finalizers:
- cluster.cluster.k8s.io
```

6. If you are going to remove the associated regional cluster or if you do not plan to reuse the credentials of the deleted cluster, delete them:

   1. In the Credentials tab, verify that the required credentials are not in the In Use status.
   2. Click the Delete credential action icon next to the name of the credentials to be deleted.
   3. Confirm the deletion.
Create and operate an AWS-based managed cluster

After bootstrapping your AWS-based Mirantis Container Cloud management cluster as described in Deployment Guide: Deploy an AWS-based management cluster, you can create the AWS-based managed clusters using the Container Cloud web UI.

Create a managed cluster

This section describes how to create an AWS-based managed cluster using the Mirantis Container Cloud web UI of the AWS-based management cluster.

To create an AWS-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the SSH Keys tab, click Add SSH Key to upload the public SSH key that will be configured on each AWS instance to provide user access.
4. In the Credentials tab:
   1. Click Add Credential and fill in the required fields to add your AWS credentials.
   2. Verify that the new credentials status is Ready. If the status is Error, hover over the status to determine the reason of the issue.
5. In the Clusters tab, click Create Cluster and fill out the form with the following parameters as required:

   1. Configure general settings and the Kubernetes parameters:

      Managed cluster configuration

      | Section                | Parameter         | Description                                                                 |
      |------------------------|-------------------|-----------------------------------------------------------------------------|
      | General settings       | Name              | Cluster name                                                                |
      |                        | Provider          | Select AWS                                                                  |
      |                        | Provider credential| From the drop-down list, select the previously created AWS credentials name. |
      |                        | Release version   | The Container Cloud version.                                                |
### SSH keys

From the drop-down list, select the SSH key name that you have previously added for SSH access to VMs.

### Provider

**AWS region**

From the drop-down list, select the AWS Region for the managed cluster. For example, `us-east-2`.

### Kubernetes

**Services CIDR blocks**

The Kubernetes Services CIDR block. For example, `10.233.0.0/18`.

**Pods CIDR blocks**

The Kubernetes Pods CIDR block. For example, `10.233.64.0/18`.

---

2. Configure StackLight:

#### StackLight configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StackLight</td>
<td>Enable Monitoring</td>
<td>Selected by default. Deselect to skip StackLight deployment.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable Logging</td>
<td>Select to deploy the StackLight logging stack. For details about the logging components, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td></td>
<td>HA Mode</td>
<td>Select to enable StackLight monitoring in the HA mode. For the differences between HA and non-HA modes, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>Retention Time</td>
<td>The Elasticsearch logs retention period in Logstash.</td>
</tr>
<tr>
<td></td>
<td>Persistent Volume Claim Size</td>
<td>The Elasticsearch persistent volume claim size.</td>
</tr>
</tbody>
</table>

**Note**

You can also enable, disable, or configure StackLight parameters after deploying a managed cluster. For details, see Change a cluster configuration or Configure StackLight.
<table>
<thead>
<tr>
<th>Prometheus</th>
<th>Retention Time</th>
<th>The Prometheus database retention period.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retention Size</td>
<td>The Prometheus database retention size.</td>
<td></td>
</tr>
<tr>
<td>Persistent Volume Claim Size</td>
<td>The Prometheus persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td>Enable Watchdog Alert</td>
<td>Select to enable the Watchdog alert that fires as long as the entire alerting pipeline is functional.</td>
<td></td>
</tr>
<tr>
<td>Custom Alerts</td>
<td>Specify alerting rules for new custom alerts or upload a YAML file in the following exemplary format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>alert</strong>: HighErrorRate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>expr</strong>: job:request_latency_seconds:mean5m{job=&quot;myjob&quot;} &gt; 0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>for</strong>: 10m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>labels</strong>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>severity</strong>: page</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>annotations</strong>:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <strong>summary</strong>: High request latency</td>
<td></td>
</tr>
<tr>
<td>StackLight Email Alerts</td>
<td>Enable Email Alerts</td>
<td>Select to enable the StackLight email alerts.</td>
</tr>
<tr>
<td>Send Resolved</td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
<td></td>
</tr>
<tr>
<td>Require TLS</td>
<td>Select to enable transmitting emails through TLS.</td>
<td></td>
</tr>
<tr>
<td>Email alerts configuration for StackLight</td>
<td>Fill out the following email alerts parameters as required:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To - the email address to send notifications to.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• From - the sender address.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SmartHost - the SMTP host through which the emails are sent.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication username - the SMTP user name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication password - the SMTP password.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication identity - the SMTP identity.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication secret - the SMTP secret.</td>
<td></td>
</tr>
</tbody>
</table>
6. Click Create.

To view the deployment status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the deployment is complete.

7. Proceed with Add a machine.

<table>
<thead>
<tr>
<th>StackLight Slack Alerts</th>
<th>Enable Slack alerts</th>
<th>Select to enable the StackLight Slack alerts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Resolved</td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
<td></td>
</tr>
<tr>
<td>Slack alerts configuration for StackLight</td>
<td>Fill out the following Slack alerts parameters as required:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• API URL - The Slack webhook URL.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Channel - The channel to send notifications to, for example, #channel-for-alerts.</td>
<td></td>
</tr>
</tbody>
</table>

See also
Delete a managed cluster

Add a machine

After you create a new AWS-based managed cluster as described in Create a managed cluster, proceed with adding machines to this cluster using the Mirantis Container Cloud web UI.

You can also use the instruction below to scale up an existing managed cluster.

To add a machine to an AWS-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the required cluster name. The cluster page with the Machines list opens.
4. Click Create Machine.
5. Fill out the form with the following parameters as required:
<table>
<thead>
<tr>
<th>Count</th>
<th>Specify the number of machines to create. The required minimum number of machines is three for the manager nodes HA and two for the Container Cloud workloads. Select Manager or Worker to create a Kubernetes manager or worker node.</th>
</tr>
</thead>
</table>
| Instance type | From the drop-down list, select the required AWS instance type. For production deployments, Mirantis recommends:  
- c5d.2xlarge for worker nodes  
- c5d.4xlarge for manager nodes  
- r5.4xlarge for nodes where the StackLight server components run  
For more details about requirements, see Reference architecture: AWS system requirements. |
| AMI ID | From the drop-down list, select the required AMI ID of Ubuntu 18.04. For example, ubuntu-bionic-18.04-amd64-server-20200729. |
| Root device size | Select the required root device size, 40 by default. |
| Node Labels | Select the required node labels for the machine to run certain components on a specific node. For example, for the StackLight nodes that run Elasticsearch and require more resources than a standard node, select the StackLight label. The list of available node labels is obtained from your current Cluster release. |

**Caution!**
If you deploy StackLight in the HA mode (recommended), add the StackLight label to minimum three nodes.

**Note**
You can configure node labels after deploying a machine. On the Machines page, click the More action icon in the last column of the required machine field and select Configure machine.

6. Click Create.
7. Repeat the steps above for the remaining machines.
You can monitor the machine status in the Managers or Workers columns on the Clusters page. Once the status changes to Ready, the deployment of the managed cluster components on this machine is complete.

The machine creation starts with the Provision status. During provisioning, the machine is not expected to be accessible since its infrastructure (VM, network, and so on) is being created.

Other machine statuses are the same as the LCMMachine object states described in Reference Architecture: LCM controller.

8. Verify the status of the cluster nodes as described in Connect to a Mirantis Container Cloud cluster.

Warning
The operational managed cluster should contain minimum 3 Kubernetes manager nodes and 2 Kubernetes worker nodes. To meet the etcd quorum and to prevent the deployment failure, scaling down of the manager nodes is prohibited.

See also
Delete a machine

Delete a managed cluster
Deleting a managed cluster does not require a preliminary deletion of VMs that run on this cluster.

To delete an AWS-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required cluster and select Delete.
4. Verify the list of machines to be removed. Confirm the deletion.

Deleting a cluster automatically removes the Amazon Virtual Private Cloud (VPC) connected with this cluster and frees up the resources allocated for this cluster, for example, instances, load balancers, networks, floating IPs.

5. If you are going to remove the associated regional cluster or if you do not plan to reuse the credentials of the deleted cluster, delete them:

   1. In the Credentials tab, verify that the required credentials are not in the In Use status.
2. Click the Delete credential action icon next to the name of the credentials to be deleted.
3. Confirm the deletion.

**Warning**
You can delete credentials only after deleting the managed cluster they relate to.

---

**Create and operate a VMWare vSphere-based managed cluster**

**Caution!**
This feature is available as Technology Preview. Use such configuration for testing and evaluation purposes only. For details about the Mirantis Technology Preview support scope, see the Preface section of this guide.

**Caution!**
This feature is available starting from the Container Cloud release 2.2.0.

After bootstrapping your VMWare vSphere-based Mirantis Container Cloud management cluster as described in Deployment Guide: Deploy a VMWare vSphere-based management cluster, you can create vSphere-based managed clusters using the Container Cloud web UI.

**Create a managed cluster**
This section describes how to create a VMWare vSphere-based managed cluster using the Mirantis Container Cloud web UI of the vSphere-based management cluster.

To create a vSphere-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the SSH Keys tab, click Add SSH Key to upload the public SSH key that will be used for the vSphere VMs creation.
4. In the Credentials tab:
1. Click Add Credential to add your vSphere credentials. You can either upload your vSphere vsphere.yaml configuration file or fill in the fields manually.

2. Verify that the new credentials status is Ready. If the status is Error, hover over the status to determine the reason of the issue.

5. In the RHEL Licenses tab, click Add RHEL License and fill out the form with the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL License Name</td>
<td>RHEL license name</td>
</tr>
<tr>
<td>Username</td>
<td>User name to access the RHEL license</td>
</tr>
<tr>
<td>Password</td>
<td>Password to access the RHEL license</td>
</tr>
<tr>
<td>Pool IDs</td>
<td>Optional. Specify the pool IDs for RHEL licenses for Virtual Datacenters. Otherwise, Subscription Manager will select a subscription from the list of available and appropriate for the machines.</td>
</tr>
</tbody>
</table>

6. In the Clusters tab, click Create Cluster and fill out the form with the following parameters as required:

1. Configure general settings and Kubernetes parameters:

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General settings</td>
<td>Name</td>
<td>Cluster name</td>
</tr>
<tr>
<td></td>
<td>Provider</td>
<td>Select vSphere</td>
</tr>
<tr>
<td></td>
<td>Provider Credential</td>
<td>From the drop-down list, select the vSphere credentials name that you have previously added.</td>
</tr>
<tr>
<td></td>
<td>SSH keys</td>
<td>From the drop-down list, select the SSH key name that you have previously added for SSH access to VMs.</td>
</tr>
<tr>
<td>Provider</td>
<td>LB Host IP</td>
<td>The IP address of the load balancer endpoint that will be used to access the Kubernetes API of the new cluster.</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>LB address range</td>
<td>The range of IP addresses that can be assigned to load balancers for Kubernetes Services.</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>Node CIDR</td>
<td>The Kubernetes nodes CIDR block. For example, 10.10.10.0/24.</td>
</tr>
<tr>
<td></td>
<td>Services CIDR Blocks</td>
<td>The Kubernetes Services CIDR block. For example, 10.233.0.0/18.</td>
</tr>
<tr>
<td></td>
<td>Pods CIDR blocks</td>
<td>The Kubernetes Pods CIDR block. For example, 10.233.64.0/18.</td>
</tr>
</tbody>
</table>

2. Configure StackLight:

**StackLight configuration**

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StackLight</td>
<td>Enable Monitoring</td>
<td>Selected by default. Deselect to skip StackLight deployment.</td>
</tr>
<tr>
<td></td>
<td>Enable Logging</td>
<td>Select to deploy the StackLight logging stack. For details about the logging components, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td></td>
<td>HA Mode</td>
<td>Select to enable StackLight monitoring in the HA mode. For the differences between HA and non-HA modes, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>Retention Time</td>
<td>The Elasticsearch logs retention period in Logstash.</td>
</tr>
<tr>
<td><strong>Persistent Volume Claim Size</strong></td>
<td>The Elasticsearch persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Prometheus Retention Time</strong></td>
<td>The Prometheus database retention period.</td>
<td></td>
</tr>
<tr>
<td><strong>Retention Size</strong></td>
<td>The Prometheus database retention size.</td>
<td></td>
</tr>
<tr>
<td><strong>Persistent Volume Claim Size</strong></td>
<td>The Prometheus persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td><strong>Enable Watchdog Alert</strong></td>
<td>Select to enable the Watchdog alert that fires as long as the entire alerting pipeline is functional.</td>
<td></td>
</tr>
<tr>
<td><strong>Custom Alerts</strong></td>
<td>Specify alerting rules for new custom alerts or upload a YAML file in the following exemplary format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- alert: HighErrorRate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>expr: job:request_latency_seconds:mean5m{job=&quot;myjob&quot;} &gt; 0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for: 10m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>labels:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>severity: page</td>
<td></td>
</tr>
<tr>
<td></td>
<td>annotations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>summary: High request latency</td>
<td></td>
</tr>
<tr>
<td><strong>StackLight Email Alerts</strong></td>
<td>Select to enable the StackLight email alerts.</td>
<td></td>
</tr>
<tr>
<td><strong>Send Resolved</strong></td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
<td></td>
</tr>
<tr>
<td><strong>Require TLS</strong></td>
<td>Select to enable transmitting emails through TLS.</td>
<td></td>
</tr>
<tr>
<td>Email alerts configuration for StackLight</td>
<td>Fill out the following email alerts parameters as required:</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• To - the email address to send notifications to.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• From - the sender address.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SmartHost - the SMTP host through which the emails are sent.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication username - the SMTP user name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication password - the SMTP password.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication identity - the SMTP identity.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication secret - the SMTP secret.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>StackLight Slack Alerts</th>
<th>Enable Slack alerts</th>
<th>Select to enable the StackLight Slack alerts.</th>
</tr>
</thead>
</table>

| Send Resolved | Select to enable notifications about resolved StackLight alerts. |

<table>
<thead>
<tr>
<th>Slack alerts configuration for StackLight</th>
<th>Fill out the following Slack alerts parameters as required:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• API URL - The Slack webhook URL.</td>
</tr>
<tr>
<td></td>
<td>• Channel - The channel to send notifications to, for example, #channel-for-alerts.</td>
</tr>
</tbody>
</table>

7. Click Create.

To view the deployment status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the deployment is complete.

8. Proceed with Add a machine.

**Add a machine**

After you create a new VMWare vSphere-based Mirantis Container Cloud managed cluster as described in Create a managed cluster, proceed with adding machines to this cluster using the Container Cloud web UI.

You can also use the instruction below to scale up an existing managed cluster.

To add a machine to a vSphere-based managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.

3. In the Clusters tab, click the required cluster name. The cluster page with Machines list opens.

4. On the cluster page, click Create Machine.

5. Fill out the form with the following parameters as required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Number of machines to create. The required minimum number of machines is three for the manager nodes HA and two for the Container Cloud workloads. Select Manager or Worker to create a Kubernetes manager or worker node.</td>
</tr>
<tr>
<td>Template Path</td>
<td>Path to the prepared OVF template.</td>
</tr>
<tr>
<td>SSH Username</td>
<td>SSH user name to access the node. Defaults to cloud-user.</td>
</tr>
<tr>
<td>RHEL License</td>
<td>From the drop-down list, select the RHEL license that you previously added for the cluster being deployed.</td>
</tr>
<tr>
<td>Node Labels</td>
<td>Select the required node labels for the machine to run certain components on a specific node. For example, for the StackLight nodes that run Elasticsearch and require more resources than a standard node, select the StackLight label. The list of available node labels is obtained from your current Cluster release.</td>
</tr>
</tbody>
</table>

Caution!

If you deploy StackLight in the HA mode (recommended), add the StackLight label to minimum three nodes.

Note
You can configure node labels after deploying a machine. On the Machines page, click the More action icon in the last column of the required machine field and select Configure machine.

6. Click Create.
7. Repeat the steps above for the remaining machines.

You can monitor the machine status in the Managers or Workers columns on the Clusters page. Once the status changes to Ready, the deployment of the managed cluster components on this machine is complete.

The machine creation starts with the Provision status. During provisioning, the machine is not expected to be accessible since its infrastructure (VM, network, and so on) is being created.

Other machine statuses are the same as the LCMMachine object states described in Reference Architecture: LCM controller.

8. Verify the status of the cluster nodes as described in Connect to a Mirantis Container Cloud cluster.

Warning
The operational managed cluster should contain minimum 3 Kubernetes manager nodes and 2 Kubernetes worker nodes. To meet the etcd quorum and to prevent the deployment failure, scaling down of the manager nodes is prohibited.

Seealso
Delete a machine

Delete a managed cluster
Deleting a managed cluster does not require a preliminary deletion of VMs that run on this cluster.

To delete a VMWare vSphere-based managed cluster:

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required cluster and select Delete.
4. Verify the list of machines to be removed. Confirm the deletion.
5. Deleting a cluster automatically turns the machines off. Therefore, clean up the hosts manually in the vSphere web UI. The machines will be automatically released from the RHEL subscription once the deletion succeeds.
6. If you are going to remove the associated regional cluster or if you do not plan to reuse the credentials of the deleted cluster, delete them:

   1. In the Credentials tab, verify that the required credentials are not in the In Use status.
2. Click the Delete credential action icon next to the name of the credentials to be deleted.

3. Confirm the deletion.

**Warning**
You can delete credentials only after deleting the managed cluster they relate to.

**Change a cluster configuration**

After deploying a managed cluster, you can enable or disable StackLight and configure its parameters if enabled. Alternatively, you can configure StackLight through kubeconfig as described in Configure StackLight.

To change a cluster configuration:

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Select the required project.
3. On the Clusters page, click the More action icon in the last column of the required cluster and select Configure cluster.
4. In the Configure cluster window, select or deselect StackLight and configure its parameters if enabled.
5. Click Update to apply the changes.

**Update a managed cluster**

A Mirantis Container Cloud management cluster automatically upgrades to a new available Container Cloud release version that supports new Cluster releases. Once done, a newer version of a Cluster release becomes available for managed clusters that you update using the Container Cloud web UI.

**Caution!**

Make sure to update the Cluster release version of your managed cluster before the current Cluster release version becomes unsupported by a new Container Cloud release version. Otherwise, Container Cloud stops auto-upgrade and eventually Container Cloud itself becomes unsupported.

This section describes how to update a managed cluster of any provider type using the Container Cloud web UI.
Caution!

For the bare metal clusters, before updating a managed cluster, set the noout flag for Ceph and unset it after the update. For details, see Release notes: Storage known issues.

To update a managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click More action icon in the last column for each cluster and select Update cluster where available.
4. In the Release Update window, select the required Cluster release to update your managed cluster to.
   
   The Description section contains the list of components versions to be installed with a new Cluster release. The release notes for each Container Cloud and Cluster release are available at Release Notes: Container Cloud releases and Release Notes: Cluster releases.
5. Click Update.
   
   Before the cluster update starts, Container Cloud performs a backup of MKE and Docker Swarm. The backup directory is located under:
   
   - /srv/backup/swarm on every Container Cloud node for Docker Swarm
   - /srv/backup/ucp on one of the controller nodes for MKE
   
   To view the update status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the update is complete.

Caution!

Due to the development limitations, the MCR upgrade to version 19.03.13 on existing Container Cloud clusters is not supported.

Note

In rare cases, after a managed cluster upgrade, Grafana may stop working due to the issues with helm-controller.

The development team is working on the issue that will be addressed in the upcoming release.
Delete a machine

This section instructs you on how to scale down an existing managed cluster through the Mirantis Container Cloud web UI.

Warning
A machine with the manager node role cannot be deleted manually. A machine with such role is automatically deleted during the managed cluster deletion.

To delete a machine from a managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click on the required cluster name to open the list of machines running on it.
4. Click the More action icon in the last column of the machine you want to delete and select Delete. Confirm the deletion.

Deleting a machine automatically frees up the resources allocated to this machine.

Warning
The operational managed cluster should contain minimum 3 Kubernetes manager nodes and 2 Kubernetes worker nodes. To meet the etcd quorum and to prevent the deployment failure, scaling down of the manager nodes is prohibited.
Operate management and regional clusters

The Mirantis Container Cloud web UI enables you to perform the following operations with the Container Cloud management and regional clusters:

- View the cluster details (such as cluster ID, creation date, nodes count, and so on) as well as obtain a list of the cluster endpoints including the StackLight components, depending on your deployment configuration.

To view generic cluster details, in the Clusters tab, click the More action icon in the last column of the required cluster and select Cluster info.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding more than 3 nodes or deleting nodes from a management or regional cluster is not supported.</td>
</tr>
<tr>
<td>Removing a management or regional cluster using the Container Cloud web UI is not supported. Use the dedicated cleanup script instead. For details, see Remove a management cluster and Remove a regional cluster.</td>
</tr>
<tr>
<td>Before removing a regional cluster, delete the credentials of the deleted managed clusters associated with the region.</td>
</tr>
</tbody>
</table>

- Verify the current release version of the cluster including the list of installed components with their versions and the cluster release change log.

To view a cluster release version details, in the Clusters tab, click the version in the Release column next to the name of the required cluster.

This section outlines the operations that can be performed with a management or regional cluster.

Automatic upgrade workflow

A management cluster upgrade to a newer version is performed automatically once a new Container Cloud version is released. Regional clusters also upgrade automatically along with the management cluster. For more details about the Container Cloud release upgrade mechanism, see: Reference Architecture: Container Cloud release controller.

Container Cloud remains operational during the management and regional clusters upgrade. Managed clusters are not affected during this upgrade. For the list of components that are updated during the Container Cloud upgrade, see the Components versions section of the corresponding Container Cloud release in Release Notes.

When Mirantis announces support of the newest versions of Mirantis Container Runtime (MCR) and Mirantis Kubernetes Engine (MKE), Container Cloud automatically upgrades these components as well. For the maintenance window best practices before upgrade of these components, see official MKE, MSR, and MCR Documentation.
Caution!

Due to the development limitations, the MCR upgrade to version 19.03.13 on existing Container Cloud clusters is not supported.

Note

MKE and Kubernetes API may return short-term 50x errors during the upgrade process. Ignore these errors.

Remove a management cluster

This section describes how to remove a management cluster.

To remove a management cluster:

1. Verify that you have successfully removed all managed clusters that run on top of the management cluster to be removed. For details, see the corresponding Delete a managed cluster section depending on your cloud provider in Operate managed clusters.

2. Log in to a local machine where your management cluster kubeconfig is located and where kubectl is installed.

   Note
   The management cluster kubeconfig is created during the last stage of the management cluster bootstrap.

3. Run the following script:

   bootstrap.sh cleanup

   Note
   Removing a management or regional cluster using the Container Cloud web UI is not supported.
Remove a regional cluster

This section describes how to remove a regional cluster.

To remove a regional cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the project with the managed clusters of the regional cluster to remove using the Switch Project action icon located on top of the main left-side navigation panel.
3. Verify that you have successfully deleted all managed clusters that run on top of the regional cluster to be removed. For details, see the corresponding Delete a managed cluster section depending on your cloud provider in Operate managed clusters.
4. Delete the credentials associated with the region:
   1. In the Credentials tab, click the first credentials name.
   2. In the window that opens, capture the Region Name field.
   3. Repeat two previous steps for the remaining credentials in the list.
   4. Delete all credentials with the name of the region that you are going to remove.
5. Log in to a local machine where your management and regional clusters kubeconfig files are located and where kubectl is installed.

   Note
   The management or regional cluster kubeconfig files are created during the last stage of the management or regional cluster bootstrap.

6. Run the following script with the corresponding values of your cluster:

   ```bash
   REGIONAL_CLUSTER_NAME=<regionalClusterName> REGIONAL_KUBECONFIG=<pathToRegionalClusterKubeconfig> KUBECONFIG=<mgmtClusterKubeconfig> ./bootstrap.sh destroy_regional
   ```

   Note
   Removing a management or regional cluster using the Container Cloud web UI is not supported.

Seealso

- Connect to a Mirantis Container Cloud cluster
- Configure StackLight
Attach an existing Mirantis Kubernetes Engine cluster

Starting from Mirantis Kubernetes Engine (MKE) 3.3.3, you can attach an existing MKE cluster that is not deployed by Mirantis Container Cloud to a management cluster. This feature allows for visualization of all your MKE clusters details in one place including clusters health, capacity, and usage.

For supported configurations of existing MKE clusters that are not deployed by Container Cloud, see Docker Enterprise Compatibility Matrix.

Note

Using the free Mirantis license, you can create up to three Container Cloud managed clusters with three worker nodes on each cluster. Within the same quota, you can also attach existing MKE clusters that are not deployed by Container Cloud. If you need to increase this quota, contact Mirantis support for further details.

Using the instruction below, you can also install StackLight to your existing MKE cluster during the attach procedure. For the StackLight system requirements, refer to the Reference Architecture: Requirements of the corresponding cloud provider.

You can also update all your MKE clusters to the latest version once your management cluster automatically updates to a newer version where a new MKE Cluster release with the latest MKE version is available. For details, see Update a managed cluster.

Caution!

- An MKE cluster can be attached to only one management cluster. Attaching a Container Cloud-based MKE cluster to another management cluster is not supported.
- Due to the development limitations, if you detach an MKE cluster that is not deployed by Container Cloud, Helm controller and OIDC integration are not deleted.
- Detaching a Container Cloud-based MKE cluster is not supported.

To attach an existing MKE cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, expand the Create Cluster menu and click Attach Existing MKE Cluster.
4. In the wizard that opens, fill out the form with the following parameters as required:
   1. Configure general settings:
MKE cluster configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Settings</td>
<td>Cluster Name</td>
<td>Specify the cluster name.</td>
</tr>
<tr>
<td></td>
<td>Region</td>
<td>Select the required cloud provider: OpenStack, AWS, or bare metal.</td>
</tr>
</tbody>
</table>

2. Upload the MKE client bundle or fill in the fields manually. To download the MKE client bundle, refer to MKE user access: Download client certificates.

3. Configure StackLight:

StackLight configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StackLight</td>
<td>Enable Monitoring</td>
<td>Selected by default. Deselect to skip StackLight deployment.</td>
</tr>
<tr>
<td></td>
<td>Enable Logging</td>
<td>Select to deploy the StackLight logging stack. For details, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td></td>
<td>HA Mode</td>
<td>Select to enable StackLight monitoring in the HA mode. For the differences between HA and non-HA modes, see Reference Architecture: StackLight deployment architecture.</td>
</tr>
<tr>
<td></td>
<td>Elasticsearch Retention Time</td>
<td>The Elasticsearch logs retention period in Logstash.</td>
</tr>
<tr>
<td></td>
<td>Persistent Volume Claim Size</td>
<td>The Elasticsearch persistent volume claim size.</td>
</tr>
<tr>
<td></td>
<td>Prometheus Retention Time</td>
<td>The Prometheus database retention period.</td>
</tr>
<tr>
<td></td>
<td>Retention Size</td>
<td>The Prometheus database retention size.</td>
</tr>
<tr>
<td>Persistent Volume Claim Size</td>
<td>The Prometheus persistent volume claim size.</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Enable Watchdog Alert</td>
<td>Select to enable the Watchdog alert that fires as long as the entire alerting pipeline is functional.</td>
<td></td>
</tr>
<tr>
<td>Custom Alerts</td>
<td>Specify alerting rules for new custom alerts or upload a YAML file in the following exemplary format:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- alert: HighErrorRate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>expr: job:request_latency_seconds:mean5m{job=&quot;myjob&quot;} &gt; 0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for: 10m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>labels:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>severity: page</td>
<td></td>
</tr>
<tr>
<td></td>
<td>annotations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>summary: High request latency</td>
<td></td>
</tr>
</tbody>
</table>

For details, see [Official Prometheus documentation: Alerting rules](#). For the list of the predefined StackLight alerts, see [Operations Guide: Available StackLight alerts](#).

<table>
<thead>
<tr>
<th>StackLight Email Alerts</th>
<th>Enable Email Alerts</th>
<th>Select to enable the StackLight email alerts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Resolved</td>
<td></td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
</tr>
<tr>
<td>Require TLS</td>
<td></td>
<td>Select to enable transmitting emails through TLS.</td>
</tr>
</tbody>
</table>

**Email alerts configuration for StackLight**

Fill out the following email alerts parameters as required:

- To - the email address to send notifications to.
- From - the sender address.
- SmartHost - the SMTP host through which the emails are sent.
- Authentication username - the SMTP user name.
- Authentication password - the SMTP password.
- Authentication identity - the SMTP identity.
- Authentication secret - the SMTP secret.

<table>
<thead>
<tr>
<th>StackLight Slack Alerts</th>
<th>Enable Slack alerts</th>
<th>Select to enable the StackLight Slack alerts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Resolved</td>
<td></td>
<td>Select to enable notifications about resolved StackLight alerts.</td>
</tr>
</tbody>
</table>
Slack alerts configuration for StackLight

<table>
<thead>
<tr>
<th>Slack alerts configuration for StackLight</th>
<th>Fill out the following Slack alerts parameters as required:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• API URL - The Slack webhook URL.</td>
</tr>
<tr>
<td></td>
<td>• Channel - The channel to send notifications to, for example, #channel-for-alerts.</td>
</tr>
</tbody>
</table>

5. Click Create.

To view the deployment status, verify the cluster status on the Clusters page. Once the orange blinking dot near the cluster name disappears, the deployment is complete.

See also
Connect to the Mirantis Kubernetes Engine web UI
Connect to the Mirantis Kubernetes Engine web UI

After you deploy a new or attach an existing Mirantis Kubernetes Engine (MKE) cluster to a management cluster, start managing your cluster using the MKE web UI.

To connect to the MKE web UI:

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required MKE cluster and select Cluster info.
4. In the dialog box with the cluster information, copy the MKE UI endpoint.
5. Paste the copied IP to a web browser and use the same credentials that you use to access the Container Cloud web UI.

**Warning**

To ensure the Container Cloud stability in managing the Container Cloud-based MKE clusters, a number of MKE API functions is not available for the Container Cloud-based MKE clusters as compared to the attached MKE clusters that are deployed not by Container Cloud. Use the Container Cloud web UI or CLI for this functionality instead.

See Reference Architecture: MKE API limitations for details.

**Caution!**

The MKE web UI contains help links that lead to the Docker Enterprise documentation suite. Besides MKE and Mirantis Container Runtime (MCR), which are integrated with Container Cloud, that documentation suite covers other Docker Enterprise components and cannot be fully applied to the Container Cloud-based MKE clusters. Therefore, to avoid any sort of misconceptions, before you proceed with MKE web UI documentation, read Reference Architecture: MKE API limitations and make sure you are using the documentation of the supported MKE version as per Release Compatibility Matrix.

**See also**

- Operate managed clusters
• Connect to a Mirantis Container Cloud cluster
• Docker Enterprise documentation: MKE user access
Connect to a Mirantis Container Cloud cluster

After you deploy a Mirantis Container Cloud management or managed cluster, connect to the cluster to verify the availability and status of the nodes as described below.

This section also describes how to SSH to a node of a cluster where Bastion host is used for SSH access. For example, on the OpenStack-based management cluster or AWS-based management and managed clusters.

To connect to a managed cluster:

1. Log in to the Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the required cluster name. The cluster page with the Machines list opens.
4. Verify the status of the manager nodes. Once the first manager node is deployed and has the Ready status, the Download Kubeconfig option for the cluster being deployed becomes active.
5. Open the Clusters tab.
6. Click the More action icon in the last column of the required cluster and select Download Kubeconfig:
   1. Enter your user password.
   2. Not recommended. Select Offline Token to generate an offline IAM token. Otherwise, for security reasons, the kubeconfig token expires every 30 minutes of the Container Cloud API idle time and you have to download kubeconfig again with a newly generated token.
   3. Click Download.
7. Verify the availability of the managed cluster machines:
   1. Export the kubeconfig parameters to your local machine with access to kubectl. For example:

```
export KUBECONFIG=~/Downloads/kubeconfig-test-cluster.yml
```

   2. Obtain the list of available Container Cloud machines:

```
kubectl get nodes -o wide
```

The system response must contain the details of the nodes in the READY status.

To connect to a management cluster:

1. Log in to a local machine where your management cluster kubeconfig is located and where kubectl is installed.
Note
The management cluster kubeconfig is created during the last stage of the management cluster bootstrap.

2. Obtain the list of available management cluster machines:

```bash
kubectl get nodes -o wide
```

The system response must contain the details of the nodes in the READY status.

To SSH to a Container Cloud cluster node if Bastion is used:

1. Obtain kubeconfig of the management or managed cluster as described in the procedures above.
2. Obtain the internal IP address of a node you require access to:

```bash
kubectl get nodes -o wide
```

3. Obtain the Bastion public IP:

```bash
kubectl get cluster -o jsonpath='{.status.providerStatus.bastion.publicIp}'
-n <project_name> <cluster_name>
```

4. Run the following command:

```bash
ssh -i <private_key> ubuntu@<node_internal_ip> -o "proxycommand ssh -W %h:%p \
-i <private_key> ubuntu@<bastion_public_ip>"
```

Substitute the parameters enclosed in angle brackets with the corresponding values of your cluster obtained in previous steps. The `<private_key>` for a management cluster is located at `~/.ssh/openstack_tmp`. For a managed cluster, this is the SSH Key that you added in the Container Cloud web UI before the managed cluster creation.

See also
- Connect to the Mirantis Kubernetes Engine web UI
Manage IAM

IAM CLI

IAM CLI is a user-facing command-line tool for managing scopes, roles, and grants. Using your personal credentials, you can perform different IAM operations through the iamctl tool. For example, you can verify the current status of the IAM service, request or revoke service tokens, verify your own grants within Mirantis Container Cloud as well as your token details.

Configure IAM CLI

The iamctl command-line interface uses the iamctl.yaml configuration file to interact with IAM.

To create the IAM CLI configuration file:

1. Log in to the management cluster.
2. Change the directory to one of the following:
   • $HOME/.iamctl
   • $HOME
   • $HOME/etc
   • /etc/iamctl
3. Create iamctl.yaml with the following exemplary parameters and values that correspond to your deployment:

```yaml
server: <IAM_API_ADDRESS>
timeout: 60
verbose: 99 # Verbosity level, from 0 to 99
tls:
  enabled: true
tls:
  ca: <PATH_TO_CA_BUNDLE>
auth:
  issuer: <IAM_REALM_IN_KEYCLOAK>
  ca: <PATH_TO_CA_BUNDLE>
  client_id: iam
  client_secret:

The <IAM_REALM_IN_KEYCLOAK> value has the <keycloak-url>/auth/realms/<realm-name> format, where <realm-name> defaults to iam.

Available IAM CLI commands

Using iamctl, you can perform different role-based access control operations in your managed cluster. For example:

• Grant or revoke access to a managed cluster and a specific user for troubleshooting
• Grant or revoke access to a Mirantis Container Cloud project that contains several managed clusters
• Create or delete tokens for the Container Cloud services with a specific set of grants as well as identify when a service token was used the last time

The iamctl command-line interface contains the following set of commands:

• General commands
• Account information commands
• Scope commands
• Role commands
• Grant commands
• Service token commands
• User commands

The following tables describe the iamctl commands with their descriptions.

### General commands

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iamctl --help, iamctl help</td>
<td>Output the list of available commands.</td>
</tr>
<tr>
<td>iamctl help &lt;command&gt;</td>
<td>Output the description of a specific command.</td>
</tr>
</tbody>
</table>

### Account information commands

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iamctl account info</td>
<td>Output detailed account information such as user email, user name, the details of their active and offline sessions, tokens statuses and expiration dates.</td>
</tr>
<tr>
<td>iamctl account login</td>
<td>Log in the current user. The system prompts to enter your authentication credentials. After a successful login, your user token is added to the $HOME/.iamctl directory.</td>
</tr>
<tr>
<td>iamctl account logout</td>
<td>Log out the current user. Once done, the user information is removed from $HOME/.iamctl.</td>
</tr>
</tbody>
</table>

### Scope commands
### Role commands

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iamctl role list &lt;scope&gt;</code></td>
<td>List the roles for the specified scope in IAM.</td>
</tr>
<tr>
<td><code>iamctl role show &lt;scope&gt; &lt;role&gt;</code></td>
<td>Output the details of the specified scope role including the role name (admin, viewer, reader), its description, and an example of the grant command. For example: <code>iamctl role show m:iam admin</code>.</td>
</tr>
</tbody>
</table>

### Grant commands

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iamctl grant give [username] [scope] [role]</code></td>
<td>Provide a user with a role in a scope. For example, the <code>iamctl grant give jdoe m:iam admin</code> command provides the IAM admin role in the m:iam scope to John Doe. For the list of supported IAM scopes and roles, see: Role list.</td>
</tr>
</tbody>
</table>

Note
To lock or disable a user, use LDAP or Google OAuth depending on the external provider integrated to your deployment.
**iamctl grant list <username>**

List the grants provided to the specified user. For example:

```
iamctl grant list jdoe.
```

Example output:

<table>
<thead>
<tr>
<th>SCOPE</th>
<th>ROLE</th>
<th>GRANT FQN</th>
</tr>
</thead>
<tbody>
<tr>
<td>m:iam</td>
<td>admin</td>
<td>m:iam@admin</td>
</tr>
<tr>
<td>m:sl</td>
<td>viewer</td>
<td>m:sl@viewer</td>
</tr>
<tr>
<td>m:kaas</td>
<td>writer</td>
<td>m:kaas@writer</td>
</tr>
</tbody>
</table>

- m:iam@admin - admin rights in all IAM-related applications
- m:sl@viewer - viewer rights in all StackLight-related applications
- m:kaas@writer - writer rights in Container Cloud

**iamctl grant revoke [username] [scope] [role]**

Revoke the grants provided to the user.

---

### Service token commands

<table>
<thead>
<tr>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>iamctl servicetoken list [--all]</strong></td>
<td>List the details of all service tokens created by the current user. The output includes the following service token details:</td>
</tr>
<tr>
<td></td>
<td>• ID</td>
</tr>
<tr>
<td></td>
<td>• Alias, for example, nova, jenkins-ci</td>
</tr>
<tr>
<td></td>
<td>• Creation date and time</td>
</tr>
<tr>
<td></td>
<td>• Creation owner</td>
</tr>
<tr>
<td></td>
<td>• Grants</td>
</tr>
<tr>
<td></td>
<td>• Last refresh date and time</td>
</tr>
<tr>
<td></td>
<td>• IP address</td>
</tr>
<tr>
<td><strong>iamctl servicetoken show [ID]</strong></td>
<td>Output the details of a service token with the specified ID.</td>
</tr>
<tr>
<td><strong>iamctl servicetoken create [alias] [service] [grant1 grants2...]</strong></td>
<td>Create a token for a specific service with the specified set of grants. For example, <code>iamctl servicetoken create new-token iam m:iam@viewer</code>.</td>
</tr>
<tr>
<td><strong>iamctl servicetoken delete [ID1 ID2...]</strong></td>
<td>Delete a service token with the specified ID.</td>
</tr>
</tbody>
</table>

---

### User commands
<table>
<thead>
<tr>
<th><strong>Usage</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>iamctl user list</td>
<td>List user names and emails of all current users.</td>
</tr>
<tr>
<td>iamctl user show &lt;username&gt;</td>
<td>Output the details of the specified user.</td>
</tr>
</tbody>
</table>

### Role list

Mirantis Container Cloud creates the IAM roles in scopes. For each application type, such as iam, k8s, or kaas, Container Cloud creates a scope in Keycloak. And every scope contains a set of roles such as admin, user, viewer. The default IAM roles can be changed during a managed cluster deployment. You can grant or revoke a role access using the IAM CLI. For details, see: IAM CLI.

Example of the structure of a cluster-admin role in a managed cluster:

```
m:k8s:kaas-tenant-name:k8s-cluster-name@cluster-admin
```

- m - prefix for all IAM roles in Container Cloud
- k8s - application type, Kubernetes
- kaas-tenant-name:k8s-cluster-name - a managed cluster identifier in Container Cloud (CLUSTER_ID)
- @ - delimiter between a scope and role
- cluster-admin - name of the role within the Kubernetes scope

The following tables include the scopes and their roles descriptions by Container Cloud components:

- Container Cloud
- Kubernetes
- StackLight

#### Container Cloud

<table>
<thead>
<tr>
<th><strong>Scope identifier</strong></th>
<th><strong>Role name</strong></th>
<th><strong>Grant example</strong></th>
<th><strong>Role description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>m:kaas</td>
<td>reader</td>
<td>m:kaas@reader ¹</td>
<td>List the managed clusters within the Container Cloud scope.</td>
</tr>
<tr>
<td></td>
<td>writer</td>
<td>m:kaas@writer ¹</td>
<td>Create or delete the managed clusters within the Container Cloud scope.</td>
</tr>
<tr>
<td>Scope identifier</td>
<td>Role name</td>
<td>Grant example</td>
<td>Role description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>m:k8s:&lt;CLUSTER_ID&gt;</td>
<td>cluster-admin</td>
<td>m:k8s:&lt;CLUSTER_ID&gt;@cluster-admin</td>
<td>Allow the super-user access to perform any action on any resource on the cluster level. When used in ClusterRoleBinding, provide full control over every resource in a cluster and all Kubernetes namespaces.</td>
</tr>
</tbody>
</table>

**StackLight**

<table>
<thead>
<tr>
<th>Scope identifier</th>
<th>Role name</th>
<th>Grant example</th>
<th>Role description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m:sl:$&lt;CLUSTER_ID&gt; or m:sl:$&lt;CLUSTER_ID&gt;:&lt;SERVICE_NAME&gt;</td>
<td>admin</td>
<td>• m:sl:$&lt;CLUSTER_ID&gt;@admin &lt;br&gt; • m:sl:$&lt;CLUSTER_ID&gt;:alerta@admin &lt;br&gt; • m:sl:$&lt;CLUSTER_ID&gt;:alertmanager@admin &lt;br&gt; • m:sl:$&lt;CLUSTER_ID&gt;:kibana@admin &lt;br&gt; • m:sl:$&lt;CLUSTER_ID&gt;:grafana@admin &lt;br&gt; • m:sl:$&lt;CLUSTER_ID&gt;:prometheus@admin</td>
<td>Access the specified web UI(s) within the scope. The m:sl:$&lt;CLUSTER_ID&gt;@admin grant provides access to all StackLight web UIs: Prometheus, Alerta, Alertmanager, Kibana, Grafana.</td>
</tr>
</tbody>
</table>
Manage StackLight

Using StackLight, you can monitor the components deployed in Mirantis Container Cloud and be quickly notified of critical conditions that may occur in the system to prevent service downtimes.

Access StackLight web UIs

StackLight provides five web UIs including Prometheus, Alertmanager, Alerta, Kibana, and Grafana. This section describes how to access any of these web UIs.

To access a StackLight web UI:

1. Log in to the Mirantis Container Cloud web UI.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. In the Clusters tab, click the More action icon in the last column of the required cluster and select Cluster info.
4. In the dialog box with the cluster information, copy the required endpoint IP from the StackLight Endpoints section.
5. Paste the copied IP to a web browser and use the default credentials to log in to the web UI. Once done, you are automatically authenticated to all StackLight web UIs.

See also

- Reference Architecture: Deployment architecture
- Reference Architecture: Authentication flow

View Grafana dashboards

Using the Grafana web UI, you can view the visual representation of the metric graphs based on the time series databases.

To view the Grafana dashboards:

1. Log in to the Grafana web UI as described in Access StackLight web UIs.
2. From the drop-down list, select the required dashboard to inspect the status and statistics of the corresponding service in your management or managed cluster:
<table>
<thead>
<tr>
<th>Component</th>
<th>Dashboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ceph Cluster</td>
<td>Ceph Cluster</td>
<td>Provides the overall health status of the Ceph cluster, capacity, latency, and recovery metrics.</td>
</tr>
<tr>
<td>Ceph Nodes</td>
<td>Ceph Nodes</td>
<td>Provides an overview of the host-related metrics, such as the number of Ceph Monitors, Ceph OSD hosts, average usage of resources across the cluster, network and hosts load.</td>
</tr>
<tr>
<td>Ceph OSD</td>
<td>Ceph OSD</td>
<td>Provides metrics for Ceph OSDs, including the Ceph OSD read and write latencies, distribution of PGs per Ceph OSD, Ceph OSDs and physical device performance.</td>
</tr>
<tr>
<td>Ceph Pools</td>
<td>Ceph Pools</td>
<td>Provides metrics for Ceph pools, including the client IOPS and throughput by pool and pools capacity usage.</td>
</tr>
<tr>
<td>Ironic Bare Metal</td>
<td>Ironic BM</td>
<td>Provides graphs on Ironic health, HTTP API availability, provisioned nodes by state and installed ironic-conductor back-end drivers.</td>
</tr>
<tr>
<td>Container Cloud clusters</td>
<td>Clusters Overview</td>
<td>Represents the main cluster capacity statistics for all clusters of a Mirantis Container Cloud deployment where StackLight is installed.</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Kubernetes Calico</td>
<td></td>
<td>Provides metrics of the entire Calico cluster usage, including the cluster status, host status, and Felix resources.</td>
</tr>
<tr>
<td>Kubernetes Cluster</td>
<td></td>
<td>Provides metrics for the entire Kubernetes cluster, including the cluster status, host status, and resources consumption.</td>
</tr>
<tr>
<td>Kubernetes Deployments</td>
<td></td>
<td>Provides information on the desired and current state of all service replicas deployed on a Container Cloud cluster.</td>
</tr>
<tr>
<td>Kubernetes Namespaces</td>
<td></td>
<td>Provides the pods state summary and the CPU, MEM, network, and IOPS resources consumption per name space.</td>
</tr>
<tr>
<td>Kubernetes Nodes</td>
<td></td>
<td>Provides charts showing resources consumption per Container Cloud cluster node.</td>
</tr>
<tr>
<td>Kubernetes Pods</td>
<td></td>
<td>Provides charts showing resources consumption per deployed pod.</td>
</tr>
<tr>
<td>Stack Light</td>
<td>NGINX</td>
<td>Provides the overall status of the NGINX cluster and information about NGINX requests and connections.</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Alertmanager</td>
<td>Provides performance metrics on the overall health status of the Prometheus Alertmanager service, the number of firing and resolved alerts received for various periods, the rate of successful and failed notifications, and the resources consumption.</td>
<td></td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>Provides information about the overall health status of the Elasticsearch cluster, including the resources consumption and the state of the shards.</td>
<td></td>
</tr>
<tr>
<td>Grafana</td>
<td>Provides performance metrics for the Grafana service, including the total number of Grafana entities, CPU and memory consumption.</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Provides PostgreSQL statistics, including read (DQL) and write (DML) row operations, transaction and lock, replication lag and conflict, and checkpoint statistics, as well as PostgreSQL performance metrics.</td>
<td></td>
</tr>
<tr>
<td>Prometheus</td>
<td>Provides the availability and performance behavior of the Prometheus servers, the sample ingestion rate, and system usage statistics per server. Also, provides statistics about the overall status and uptime of the Prometheus service, the chunks number of the local storage memory, target scrapes, and queries duration.</td>
<td></td>
</tr>
<tr>
<td>Pushgateway</td>
<td>Provides performance metrics and the overall health status of the service, the rate of samples received for various periods, and the resources consumption.</td>
<td></td>
</tr>
<tr>
<td>Prometheus Relay</td>
<td>Provides service status and resources consumption metrics.</td>
<td></td>
</tr>
<tr>
<td>Telemeter Server</td>
<td>Provides statistics and the overall health status of the Telemeter service.</td>
<td></td>
</tr>
<tr>
<td>System</td>
<td>Provides a detailed resource consumption and operating system information per Container Cloud cluster node.</td>
<td></td>
</tr>
</tbody>
</table>
MKE Cluster

Provides a global overview of an MKE cluster: statistics about the number of the worker and manager nodes, containers, images, Swarm services.

MKE Containers

Provides per container resources consumption metrics for the MKE containers such as CPU, RAM, network.

### View Kibana dashboards

Using the Kibana web UI, you can view the visual representation of logs and Kubernetes events of your deployment.

To view the Kibana dashboards:

1. Log in to the Kibana web UI as described in Access StackLight web UIs.
2. Click the required dashboard to inspect the visualizations or perform a search:

<table>
<thead>
<tr>
<th>Dashboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logs</td>
<td>Provides visualizations on the number of log messages per severity, source, and top log-producing host, namespaces, containers, and applications. Includes search.</td>
</tr>
<tr>
<td>Kubernetes events</td>
<td>Provides visualizations on the number of Kubernetes events per type, and top event-producing resources and namespaces by reason and event type. Includes search.</td>
</tr>
</tbody>
</table>
Available StackLight alerts

This section provides an overview of the available predefined StackLight alerts. To view the alerts, use the Prometheus web UI. To view the firing alerts, use Alertmanager or Alerta web UI.

Alertmanager

This section describes the alerts for the Alertmanager service.

- AlertmanagerFailedReload
- AlertmanagerMembersInconsistent
- AlertmanagerNotificationFailureWarning
- AlertmanagerAlertsInvalidWarning

AlertmanagerFailedReload

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Failure to reload the Alertmanager configuration.</td>
</tr>
<tr>
<td>Description</td>
<td>Reloading the Alertmanager configuration failed for the {{ $labels.namespace }}/ {{ $labels.pod }} Pod.</td>
</tr>
</tbody>
</table>

AlertmanagerMembersInconsistent

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Alertmanager cluster members are not found.</td>
</tr>
<tr>
<td>Description</td>
<td>Alertmanager has not found all other members of the cluster.</td>
</tr>
</tbody>
</table>

AlertmanagerNotificationFailureWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Alertmanager has failed notifications.</td>
</tr>
</tbody>
</table>
AlertmanagerAlertsInvalidWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Alertmanager has invalid alerts.</td>
</tr>
<tr>
<td>Description</td>
<td>An average of {{ $value }} Alertmanager {{ $labels.integration }} alerts on the {{ $labels.namespace }/{{ $labels.pod }} Pod are invalid for 2 minutes.</td>
</tr>
</tbody>
</table>

Calico

This section describes the alerts for Calico.

- CalicoDataplaneFailuresHigh
- CalicoDataplaneAddressMsgBatchSizeHigh
- CalicoDataplaneIfaceMsgBatchSizeHigh
- CalicoIPsetErrorsHigh
- CalicoIptablesSaveErrorsHigh
- CalicoIptablesRestoreErrorsHigh

CalicoDataplaneFailuresHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>High number of data plane failures within Felix.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }/{{ $labels.pod }} Felix Pod on the {{ $labels.node }} node has {{ $value }} data plane failures within the last hour.</td>
</tr>
</tbody>
</table>

CalicoDataplaneAddressMsgBatchSizeHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
</table>
### CalicoDataplaneIfaceMsgBatchSizeHigh

<table>
<thead>
<tr>
<th><strong>Summary</strong></th>
<th>Felix interface message batch size is higher than 5.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The size of the data plane interface message batch on the {{ $labels.namespace }}/{{ $labels.pod }} Felix Pod on the {{ $labels.node }} node is {{ $value }}.</td>
</tr>
</tbody>
</table>

### CalicoIPsetErrorsHigh

<table>
<thead>
<tr>
<th><strong>Summary</strong></th>
<th>More than 5 IPset errors occur in Felix per hour.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The {{ $labels.namespace }}/{{ $labels.pod }} Felix Pod on the {{ $labels.node }} node has {{ $value }} IPset errors within the last hour.</td>
</tr>
</tbody>
</table>

### CalicoIptablesSaveErrorsHigh

<table>
<thead>
<tr>
<th><strong>Summary</strong></th>
<th>More than 5 iptable save errors occur in Felix per hour.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The {{ $labels.namespace }}/{{ $labels.pod }} Felix Pod on the {{ $labels.node }} node has {{ $value }} iptable save errors within the last hour.</td>
</tr>
</tbody>
</table>

### CalicoIptablesRestoreErrorsHigh
Severi ty  |   Warning
---|---
Summ ary  |   More than 5 iptable restore errors occur in Felix per hour.
Descri ption  |   The `{ $labels.namespace }/{ $labels.pod }` Felix Pod on the `{ $labels.node }` node has `{ $value }` iptable restore errors within the last hour.

### Ceph

This section describes the alerts for the Ceph cluster.

- CephClusterHealthMinor
- CephClusterHealthCritical
- CephMonQuorumAtRisk
- CephOSDDownMinor
- CephOSDDiskNotResponding
- CephOSDDiskUnavailable
- CephClusterNearFull
- CephClusterCriticallyFull
- CephOSDPgNumTooHighWarning
- CephOSDPgNumTooHighCritical
- CephMonHighNumberOfLeaderChanges
- CephNodeDown
- CephDataRecoveryTakingTooLong
- CephPGRepairTakingTooLong
- CephOSDVersionMismatch
- CephMonVersionMismatch

---

#### CephClusterHealthMinor

<table>
<thead>
<tr>
<th>Severi ty</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summ ary</td>
<td>Ceph cluster health is WARNING.</td>
</tr>
<tr>
<td>Descri ption</td>
<td>The Ceph cluster is in the WARNING state. For details, run ceph -s.</td>
</tr>
</tbody>
</table>
CephClusterHealthCritical

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Ceph cluster health is CRITICAL.</td>
</tr>
<tr>
<td>Description</td>
<td>The Ceph cluster is in the CRITICAL state. For details, run ceph -s.</td>
</tr>
</tbody>
</table>

CephMonQuorumAtRisk

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Storage quorum is at risk.</td>
</tr>
<tr>
<td>Description</td>
<td>The storage cluster quorum is low.</td>
</tr>
</tbody>
</table>

CephOSDDownMinor

<table>
<thead>
<tr>
<th>Severity</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Ceph OSDs are down.</td>
</tr>
<tr>
<td>Description</td>
<td>{{ $value }} of Ceph OSDs in the Ceph cluster are down. For details, run ceph osd tree.</td>
</tr>
</tbody>
</table>

CephOSDDiskNotResponding

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Disk is not responding.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk device is not responding on the {{ $labels.host }} host.</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**CephOSDDiskUnavailable**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Disk is not accessible.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk device is not accessible on the {{ $labels.host }} host.</td>
</tr>
</tbody>
</table>

**CephClusterNearFull**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Storage cluster is nearly full. Expansion is required.</td>
</tr>
<tr>
<td>Description</td>
<td>The storage cluster utilization has crossed 85%.</td>
</tr>
</tbody>
</table>

**CephClusterCriticallyFull**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Storage cluster is critically full and needs immediate expansion.</td>
</tr>
<tr>
<td>Description</td>
<td>The storage cluster utilization has crossed 95%.</td>
</tr>
</tbody>
</table>

**CephOSDPgNumTooHighWarning**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summ ary</td>
<td>Some Ceph OSDs have more than 200 PGs.</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Descri ption</td>
<td>Some Ceph OSDs contain more than 200 PGs. This may have a negative impact on the cluster performance. For details, run ceph pg dump.</td>
</tr>
</tbody>
</table>

### CephOSDPgNumTooHighCritical

<table>
<thead>
<tr>
<th>Severi ty</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summ ary</td>
<td>Some Ceph OSDs have more than 300 PGs.</td>
</tr>
<tr>
<td>Descri ption</td>
<td>Some Ceph OSDs contain more than 300 PGs. This may have a negative impact on the cluster performance. For details, run ceph pg dump.</td>
</tr>
</tbody>
</table>

### CephMonHighNumberOfLeaderChanges

<table>
<thead>
<tr>
<th>Severi ty</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summ ary</td>
<td>Many leader changes occur in the storage cluster.</td>
</tr>
<tr>
<td>Descri ption</td>
<td>{{ $value }} leader changes per minute occur for the {{ $labels.instance }} instance of the {{ $labels.job }} Ceph Monitor.</td>
</tr>
</tbody>
</table>

### CephNodeDown

<table>
<thead>
<tr>
<th>Severi ty</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summ ary</td>
<td>Ceph node {{ $labels.node }} went down.</td>
</tr>
<tr>
<td>Descri ption</td>
<td>The {{ $labels.node }} Ceph node is down and requires immediate verification.</td>
</tr>
</tbody>
</table>

### CephDataRecoveryTakingTooLong
# Data Recovery is Slow

**CephPGRepairTakingTooLong**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Self-heal issues detected.</td>
</tr>
<tr>
<td>Description</td>
<td>The self-heal operations take an excessive amount of time.</td>
</tr>
</tbody>
</table>

**CephOSDVersionMismatch**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Multiple versions of storage services are running.</td>
</tr>
<tr>
<td>Description</td>
<td>{{ $value }} different versions of Ceph OSD components are running.</td>
</tr>
</tbody>
</table>

**CephMonVersionMismatch**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Multiple versions of storage services are running.</td>
</tr>
<tr>
<td>Description</td>
<td>{{ $value }} different versions of Ceph Monitor components are running.</td>
</tr>
</tbody>
</table>

# Docker Swarm

This section describes the alerts for the Docker Swarm service.
- **DockerSwarmLeadElectionLoop**
- **DockerSwarmNetworkUnhealthy**
- **DockerSwarmNodeFlapping**
- **DockerSwarmServiceReplicasDown**
- **DockerSwarmServiceReplicasFlapping**
- **DockerSwarmServiceReplicasOutage**

### DockerSwarmLeadElectionLoop

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Docker Swarm Manager leadership election loop.</td>
</tr>
<tr>
<td>Description</td>
<td>More than 2 Docker Swarm leader elections occur for the last 10 minutes.</td>
</tr>
</tbody>
</table>

### DockerSwarmNetworkUnhealthy

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Docker Swarm network is unhealthy.</td>
</tr>
<tr>
<td>Description</td>
<td>The qLen size and NetMsg showed unexpected output for the last 10 minutes. Verify the NetworkDb Stats output for the qLen size and NetMsg using journalctl -d docker.</td>
</tr>
</tbody>
</table>


**Note**
For the DockerNetworkUnhealthy alert, StackLight collects metrics from logs. Therefore, this alert is available only if logging is enabled.

### DockerSwarmNodeFlapping

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
</table>
Docker Swarm node is flapping.

**Description**
The {{ $labels.node_name }} Docker Swarm node has changed the state more than 3 times for the last 10 minutes.

---

**DockerSwarmServiceReplicasDown**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Docker Swarm replica is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.service_name }} Docker Swarm service replica is down for 2 minutes.</td>
</tr>
</tbody>
</table>

---

**DockerSwarmServiceReplicasFlapping**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Docker Swarm service replica is flapping.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.service_name }} Docker Swarm service replica is flapping for 15 minutes.</td>
</tr>
</tbody>
</table>

---

**DockerSwarmServiceReplicasOutage**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Docker Swarm service outage.</td>
</tr>
<tr>
<td>Description</td>
<td>All {{ $labels.service_name }} Docker Swarm service replicas are down for 2 minutes.</td>
</tr>
</tbody>
</table>

---

**Elasticsearch**

This section describes the alerts for the Elasticsearch service.

- ElasticHeapUsageCritical
- ElasticHeapUsageWarning
- ElasticClusterStatusCritical
- ElasticClusterStatusWarning
- NumberOfRelocationShards
- NumberOfInitializingShards
- NumberOfUnassignedShards
- NumberOfPendingTasks
- ElasticNoNewDataCluster
- ElasticNoNewDataNode

ElasticHeapUsageCritical

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Elasticsearch heap usage is too high (&gt;90%).</td>
</tr>
<tr>
<td>Description</td>
<td>Elasticsearch heap usage is over 90% for 5 minutes.</td>
</tr>
</tbody>
</table>

ElasticHeapUsageWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Elasticsearch heap usage is high (&gt;80%).</td>
</tr>
<tr>
<td>Description</td>
<td>Elasticsearch heap usage is over 80% for 5 minutes.</td>
</tr>
</tbody>
</table>

ElasticClusterStatusCritical

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Elasticsearch critical status.</td>
</tr>
</tbody>
</table>
### ElasticClusterStatusWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Elasticsearch warning status.</td>
</tr>
<tr>
<td>Description</td>
<td>The Elasticsearch cluster status has changed to YELLOW. The alert persists for the cluster in the RED status.</td>
</tr>
</tbody>
</table>

### NumberOfRelocationShards

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Shards relocation takes more than 20 minutes.</td>
</tr>
<tr>
<td>Description</td>
<td>Elasticsearch has {{ $value }} relocating shards for 20 minutes.</td>
</tr>
</tbody>
</table>

### NumberOfInitializingShards

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Shards initialization takes more than 10 minutes.</td>
</tr>
<tr>
<td>Description</td>
<td>Elasticsearch has {{ $value }} shards being initialized for 10 minutes.</td>
</tr>
</tbody>
</table>

### NumberOfUnassignedShards

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Shards have unassigned status for 5 minutes.</td>
<td>Elasticsearch has {{ $value }} unassigned shards for 5 minutes.</td>
</tr>
</tbody>
</table>

**NumberOfPendingTasks**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>Tasks have pending state for 10 minutes.</td>
</tr>
<tr>
<td></td>
<td>Elasticsearch has {{ $value }} pending tasks for 10 minutes. The cluster works slowly.</td>
</tr>
</tbody>
</table>

**ElasticNoNewDataCluster**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major</td>
<td>Elasticsearch cluster has no new data for 30 minutes.</td>
</tr>
<tr>
<td></td>
<td>No new data has arrived to the Elasticsearch cluster for 30 minutes.</td>
</tr>
</tbody>
</table>

**ElasticNoNewDataNode**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>Elasticsearch node has no new data for 30 minutes.</td>
</tr>
<tr>
<td></td>
<td>No new data has arrived to the {{ $labels.name }} Elasticsearch node for 30 minutes. The alert also indicates Elasticsearch node cordonning.</td>
</tr>
</tbody>
</table>

**etcd**

This section describes the alerts for the etcd service.

- etcdInsufficientMembers
- etcdNoLeader
- etcdHighNumberOfLeaderChanges
- etcdGRPCRequestsSlow
- etcdMemberCommunicationSlow
- etcdHighNumberOfFailedProposals
- etcdHighFsyncDurations
- etcdHighCommitDurations

### etcdInsufficientMembers

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has insufficient members.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.job }} etcd cluster has {{ $value }} insufficient members.</td>
</tr>
</tbody>
</table>

### etcdNoLeader

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has no leader.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} member of the {{ $labels.job }} etcd cluster has no leader.</td>
</tr>
</tbody>
</table>

### etcdHighNumberOfLeaderChanges

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>More than 3 leader changes occurred in the etcd cluster within the last hour.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} instance of the {{ $labels.job }} etcd cluster has {{ $value }} leader changes within the last hour.</td>
</tr>
</tbody>
</table>
### etcdGRPCRequestsSlow

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has slow gRPC requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The gRPC requests to <code>{labels.grpc_method}</code> take <code>{value}</code>s on <code>{labels.instance}</code> instance of the <code>{labels.job}</code> etcd cluster.</td>
</tr>
</tbody>
</table>

### etcdMemberCommunicationSlow

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has slow member communication.</td>
</tr>
<tr>
<td>Description</td>
<td>The member communication with <code>{labels.To}</code> on the <code>{labels.instance}</code> instance of the <code>{labels.job}</code> etcd cluster takes <code>{value}</code>s.</td>
</tr>
</tbody>
</table>

### etcdHighNumberOfFailedProposals

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has more than 5 proposal failures.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{labels.job}</code> etcd cluster has <code>{value}</code> proposal failures on the <code>{labels.instance}</code> etcd instance within the last hour.</td>
</tr>
</tbody>
</table>

### etcdHighFsyncDurations

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has high fync duration.</td>
</tr>
</tbody>
</table>
etcdHighCommitDurations

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The etcd cluster has high commit duration.</td>
</tr>
<tr>
<td>Description</td>
<td>The duration of 99% of all commit operations on the {{ $labels.instance }} of the {{ $labels.job }} etcd cluster is {{ $value }}s.</td>
</tr>
</tbody>
</table>

External endpoint

This section describes the alerts for external endpoints.

- ExternalEndpointDown
- ExternalEndpointTCPFailure

ExternalEndpointDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>External endpoint is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} external endpoint is not accessible for the last 2 minutes.</td>
</tr>
</tbody>
</table>

ExternalEndpointTCPFailure

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Failure to establish a TCP or TLS connection.</td>
</tr>
<tr>
<td>Description</td>
<td>The system cannot establish a TCP or TLS connection to {{ $labels.instance }}.</td>
</tr>
</tbody>
</table>
General alerts
This section lists the general available alerts.

- TargetDown
- TargetFlapping
- NodeDown
- Watchdog

TargetDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.job }} target is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.job }}/{{ $labels.instance }} target is down.</td>
</tr>
</tbody>
</table>

TargetFlapping

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.job }} target is flapping.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.job }}/{{ $labels.instance }} target is changing its state between UP and DOWN for 30 minutes, at least once within the 15 minutes time range.</td>
</tr>
</tbody>
</table>

NodeDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.node }} node is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.node }} node is down. Kubernetes treats the node as Not Ready and kubelet is not accessible from Prometheus.</td>
</tr>
</tbody>
</table>
Watchdog

<table>
<thead>
<tr>
<th>Severity</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Watchdog alert that is always firing.</td>
</tr>
<tr>
<td>Description</td>
<td>This alert ensures that the entire alerting pipeline is functional. This alert should always be firing in Alertmanager against a receiver. Some integrations with various notification mechanisms can send a notification when this alert is not firing. For example, the DeadMansSnitch integration in PagerDuty.</td>
</tr>
</tbody>
</table>

General node alerts

This section lists the general alerts for Kubernetes nodes.

- FileDescriptorUsageCritical
- FileDescriptorUsageMajor
- FileDescriptorUsageWarning
- SystemCpuFullWarning
- SystemLoadTooHighWarning
- SystemLoadTooHighCritical
- SystemDiskFullWarning
- SystemDiskFullMajor
- SystemMemoryFullWarning
- SystemMemoryFullMajor
- SystemDiskInodesFullWarning
- SystemDiskInodesFullMajor
- SystemDiskErrorsTooHigh

FileDescriptorUsageCritical

Available since 2.2.0

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Node uses 95% of file descriptors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.node }} node uses 95% of file descriptors.</td>
</tr>
</tbody>
</table>
### FileDescriptorUsageMajor

**Severity** Major

**Summary** Node uses 90% of file descriptors.

**Description** The {{ $labels.node }} node uses 90% of file descriptors.

### FileDescriptorUsageWarning

**Severity** Warning

**Summary** Node uses 80% of file descriptors.

**Description** The {{ $labels.node }} node uses 80% of file descriptors.

### SystemCpuFullWarning

**Severity** Warning

**Summary** High CPU consumption.

**Description** The average CPU consumption on the {{ $labels.node }} node is {{ $value }}% for 2 minutes.

### SystemLoadTooHighWarning

**Severity** Warning

**Summary**

**Description**

©2021, Mirantis Inc.
### SystemLoadTooHighCritical

<table>
<thead>
<tr>
<th>Summary</th>
<th>System load is more than 2 per CPU.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The system load per CPU on the $labels.node node is $value for 5 minutes.</td>
</tr>
</tbody>
</table>

### SystemDiskFullWarning

<table>
<thead>
<tr>
<th>Summary</th>
<th>Disk partition $labels.mountpoint is 85% full.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The $labels.device disk partition $labels.mountpoint on the $labels.node node is $value% full for 2 minutes.</td>
</tr>
</tbody>
</table>

### SystemDiskFullMajor

<table>
<thead>
<tr>
<th>Summary</th>
<th>Disk partition $labels.mountpoint is 95% full.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The $labels.device disk partition $labels.mountpoint on the $labels.node node is $value% full for 2 minutes.</td>
</tr>
</tbody>
</table>

### SystemMemoryFullWarning
<table>
<thead>
<tr>
<th>Sev</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>More than 90% of memory is used or less than 8 GB is available.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.node }} node consumes {{ $value }}% of memory for 2 minutes.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Sev</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>More than 95% of memory is used or less than 4 GB of memory is available.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.node }} node consumes {{ $value }}% of memory for 2 minutes.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Sev</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.mountpoint }} volume uses 85% of inodes.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.node }} node consumes {{ $value }}% of disk inodes in the {{ $labels.mountpoint }} volume for 2 minutes.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Sev</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.mountpoint }} volume uses 95% of inodes.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.node }} node consumes {{ $value }}% of disk inodes in the {{ $labels.mountpoint }} volume for 2 minutes.</td>
</tr>
</tbody>
</table>
**SystemDiskErrorsTooHigh**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.device }} disk is failing.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.node }} node is reporting errors for 5 minutes.</td>
</tr>
</tbody>
</table>

**Ironic**

This section describes the alerts for Ironic bare metal. The alerted events include Ironic API availability and Ironic processes availability.

- IronicBmMetricsMissing
- IronicBmApiOutage

**IronicBmMetricsMissing**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Ironic metrics missing.</td>
</tr>
<tr>
<td>Description</td>
<td>Metrics retrieved from the Ironic API are not available for 2 minutes.</td>
</tr>
</tbody>
</table>

**IronicBmApiOutage**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Ironic API outage.</td>
</tr>
<tr>
<td>Description</td>
<td>The Ironic API is not accessible.</td>
</tr>
</tbody>
</table>

**Kubernetes applications**

This section lists the alerts for Kubernetes applications.

- KubePodCrashLooping
- KubePodNotReady
- KubeDeploymentGenerationMismatch
- KubeDeploymentReplicasMismatch
- KubeStatefulSetReplicasMismatch
- KubeStatefulSetGenerationMismatch
- KubeStatefulSetUpdateNotRolledOut
- KubeDaemonSetRolloutStuck
- KubeDaemonSetNotScheduled
- KubeDaemonSetMisScheduled
- KubeCronJobRunning
- KubeJobCompletion
- KubeJobFailed

### KubePodCrashLooping

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ ${labels.pod} }</code> Pod is in a crash loop status.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{ ${labels.namespace}}/${{ $labels.pod }}</code> Pod container <code>{ ${labels.container} }</code> was restarted at least twice during the last 5 minutes.</td>
</tr>
</tbody>
</table>

### KubePodNotReady

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ ${labels.pod} }</code> Pod is in the non-ready state.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{ ${labels.namespace}}/${{ $labels.pod }}</code> Pod state is not Ready for longer than 15 minutes.</td>
</tr>
</tbody>
</table>

### KubeDeploymentGenerationMismatch

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
</table>
### KubeDeploymentReplicasMismatch

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.deployment }} Deployment has a wrong number of replicas.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.deployment }} Deployment does not match the expected number of replicas for longer than 10 minutes.</td>
</tr>
</tbody>
</table>

### KubeStatefulSetReplicasMismatch

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.statefulset }} StatefulSet has a wrong number of replicas.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.statefulset }} StatefulSet does not match the expected number of replicas for longer than 10 minutes.</td>
</tr>
</tbody>
</table>

### KubeStatefulSetGenerationMismatch

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.statefulset }} StatefulSet generation does not match the metadata.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.statefulset }} StatefulSet generation does not match the metadata, indicating that the StatefulSet failed but has not been rolled back.</td>
</tr>
</tbody>
</table>

### KubeStatefulSetUpdateNotRolledOut
<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.statefulset }} StatefulSet update has not been rolled out.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.statefulset }} StatefulSet update has not been rolled out.</td>
</tr>
</tbody>
</table>

**KubeDaemonSetRolloutStuck**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.daemonset }} DaemonSet is not ready.</td>
</tr>
<tr>
<td>Description</td>
<td>Only {{ $value }}% of the desired Pods of the {{ $labels.namespace }}/{{ $labels.daemonset }} DaemonSet are scheduled and ready.</td>
</tr>
</tbody>
</table>

**KubeDaemonSetNotScheduled**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.daemonset }} DaemonSet has not scheduled Pods.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.daemonset }} DaemonSet has {{ $value }} not scheduled Pods.</td>
</tr>
</tbody>
</table>

**KubeDaemonSetMisScheduled**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.daemonset }} DaemonSet has incorrectly scheduled Pods.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.daemonset }} DaemonSet has {{ $value }} Pods running where they are not supposed to run.</td>
</tr>
</tbody>
</table>
KubeCronJobRunning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.cronjob }} CronJob is not ready.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.cronjob }} CronJob takes more than 15 minutes to complete.</td>
</tr>
</tbody>
</table>

KubeJobCompletion

<table>
<thead>
<tr>
<th>Severity</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.job_name }} job is not completed.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.job_name }} job takes more than 15 minutes to complete.</td>
</tr>
</tbody>
</table>

KubeJobFailed

<table>
<thead>
<tr>
<th>Severity</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.job_name }} job failed.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.job_name }} job failed to complete.</td>
</tr>
</tbody>
</table>

Kubernetes resources

This section lists the alerts for Kubernetes resources.

- KubeCPUOvercommitPods
- KubeMemOvercommitPods
- KubeCPUOvercommitNamespaces
- KubeMemOvercommitNamespaces
- KubeQuotaExceeded
- CPUMThrottlingHigh
### KubeCPUOvercommitPods

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes has overcommitted CPU requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes cluster has overcommitted CPU resource requests for Pods and cannot tolerate node failure.</td>
</tr>
</tbody>
</table>

### KubeMemOvercommitPods

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes has overcommitted memory requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes cluster has overcommitted memory resource requests for Pods and cannot tolerate node failure.</td>
</tr>
</tbody>
</table>

### KubeCPUOvercommitNamespaces

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes has overcommitted CPU requests for namespaces.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes cluster has overcommitted CPU resource requests for namespaces.</td>
</tr>
</tbody>
</table>

### KubeMemOvercommitNamespaces

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes has overcommitted memory requests for namespaces.</td>
</tr>
</tbody>
</table>
The Kubernetes cluster has overcommitted memory resource requests for namespaces.

### KubeQuotaExceeded

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{{ $labels.namespace }}</code> namespace consumes more than 90% of its <code>{{ $labels.resource }}</code> quota.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{{ $labels.namespace }}</code> namespace consumes <code>{{ printf &quot;%0.0f&quot; $value }}</code>% of its <code>{{ $labels.resource }}</code> quota.</td>
</tr>
</tbody>
</table>

### CPUThrottlingHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{{ $labels.pod_name }}</code> Pod has CPU throttling.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{{ $labels.namespace }}</code> container in the <code>{{ $labels.namespace }}/{{ $labels.pod }}</code> Pod has <code>{{ printf &quot;%0.0f&quot; $value }}</code>% of CPU throttling.</td>
</tr>
</tbody>
</table>

### Kubernetes storage

This section lists the alerts for Kubernetes storage.

**Caution!**

Due to the upstream bug in Kubernetes, metrics for the KubePersistentVolumeUsageCritical and KubePersistentVolumeFullInFourDays alerts that are collected for persistent volumes provisioned by cinder-csi-plugin are not available.

- KubePersistentVolumeUsageCritical
- KubePersistentVolumeFullInFourDays
- KubePersistentVolumeErrors
KubePersistentVolumeUsageCritical

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ { $labels.persistentvolumeclaim } }</code> PersistentVolume has less than 3% of free space.</td>
</tr>
<tr>
<td>Description</td>
<td>The PersistentVolume claimed by <code>{ { $labels.persistentvolumeclaim } }</code> in the <code>{ { $labels.namespace } }</code> namespace is only `{ { printf &quot;%0.2f&quot; $value } }% free.</td>
</tr>
</tbody>
</table>

KubePersistentVolumeFullInFourDays

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ { $labels.persistentvolumeclaim } }</code> PersistentVolume is expected to fill up in 4 days.</td>
</tr>
<tr>
<td>Description</td>
<td>Based on the recent sampling, the PersistentVolume claimed by <code>{ { $labels.persistentvolumeclaim } }</code> in the <code>{ { $labels.namespace } }</code> namespace is expected to fill up within four days. Currently, `{ { printf &quot;%0.2f&quot; $value } }% of free space is available.</td>
</tr>
</tbody>
</table>

KubePersistentVolumeErrors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The status of the <code>{ { $labels.persistentvolume } }</code> PersistentVolume is <code>{ { $labels.phase } }</code>.</td>
</tr>
<tr>
<td>Description</td>
<td>The status of the <code>{ { $labels.persistentvolume } }</code> PersistentVolume is <code>{ { $labels.phase } }</code>.</td>
</tr>
</tbody>
</table>

Kubernetes system

This section lists the alerts for the Kubernetes system.

- KubeNodeNotReady
- KubeVersionMismatch
- KubeClientErrors
- KubeletTooManyPods
- KubeAPIDown
- KubeAPIOutage
* KubeAPILatencyHighWarning
* KubeAPILatencyHighMajor
* KubeAPIErrorsHighMajor
* KubeAPIErrorsHighWarning
* KubeAPIResourceErrorsHighMajor
* KubeAPIResourceErrorsHighWarning
* KubeClientCertificateExpirationInSevenDays
* KubeClientCertificateExpirationInOneDay
* ContainerScrapeError

## KubeNodeNotReady

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.node }} node is not ready.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes {{ $labels.node }} node is not ready for more than one hour.</td>
</tr>
</tbody>
</table>

## KubeVersionMismatch

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes components have mismatching versions.</td>
</tr>
<tr>
<td>Description</td>
<td>Kubernetes has components with {{ $value }} different semantic versions running.</td>
</tr>
</tbody>
</table>

## KubeClientErrors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes API client has more than 1% of error requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{labels.job}</code>/{{ $labels.instance }} Kubernetes API server client has <code>{{ printf &quot;%0.0f&quot; $value }}%</code> errors.</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**KubeletTooManyPods**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>kubelet reached 90% of Pods limit.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{labels.instance}</code>/{{ $labels.node }} kubelet runs <code>{value}</code> Pods, nearly 90% of possible allocation.</td>
</tr>
</tbody>
</table>

**KubeAPIDown**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes API endpoint is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes API endpoint <code>{labels.instance}</code> is not accessible for the last 3 minutes.</td>
</tr>
</tbody>
</table>

**KubeAPIOutage**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Kubernetes API is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The Kubernetes API is not accessible for the last 30 seconds.</td>
</tr>
</tbody>
</table>

**KubeAPILatencyHighWarning**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The API server has a 99th percentile latency of more than 1 second.</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>The API server has a 99th percentile latency of {{ $value } } seconds for {{ $labels.verb } } {{ $labels.resource } }.</td>
</tr>
</tbody>
</table>

**KubeAPILatencyHighMajor**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The API server has a 99th percentile latency of more than 4 seconds.</td>
</tr>
<tr>
<td>Description</td>
<td>The API server has a 99th percentile latency of {{ $value } } seconds for {{ $labels.verb } } {{ $labels.resource } }.</td>
</tr>
</tbody>
</table>

**KubeAPIErrorsHighMajor**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>API server returns errors for more than 3% of requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The API server returns errors for {{ $value } }% of requests.</td>
</tr>
</tbody>
</table>

**KubeAPIErrorsHighWarning**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>API server returns errors for more than 1% of requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The API server returns errors for {{ $value } }% of requests.</td>
</tr>
</tbody>
</table>

**KubeAPIResourceErrorsHighMajor**
### KubeAPIResourceErrorsHighWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>API server returns errors for 10% of requests.</td>
</tr>
<tr>
<td>Description</td>
<td>The API server returns errors for ${{ $value }}$% of requests for ${{ $labels.verb }}$ ${{ $labels.resource }}$ ${{ $labels.subresource }}$.</td>
</tr>
</tbody>
</table>

### KubeClientCertificateExpirationInSevenDays

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>A client certificate expires in 7 days.</td>
</tr>
<tr>
<td>Description</td>
<td>A client certificate used to authenticate to the API server expires in less than 7 days.</td>
</tr>
</tbody>
</table>

### KubeClientCertificateExpirationInOneDay

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>A client certificate expires in 24 hours.</td>
</tr>
<tr>
<td>Description</td>
<td>A client certificate used to authenticate to the API server expires in less than 24 hours.</td>
</tr>
</tbody>
</table>
ContainerScrapeError

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Failure to get Kubernetes container metrics.</td>
</tr>
<tr>
<td>Description</td>
<td>Prometheus was not able to scrape metrics from the container on the <code>{{ $labels.node }}</code> Kubernetes node.</td>
</tr>
</tbody>
</table>

Netchecker

This section lists the alerts for the Netchecker service.

- NetCheckerAgentErrors
- NetCheckerReportsMissing
- NetCheckerTCPServerDelay
- NetCheckerDNSSlow

NetCheckerAgentErrors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Netchecker has a high number of errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{{ $labels.agent }}</code> Netchecker agent had <code>{{ $value }}</code> errors within the last hour.</td>
</tr>
</tbody>
</table>

NetCheckerReportsMissing

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The number of agent reports is lower than expected.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{{ $labels.agent }}</code> Netchecker agent has not reported anything for the last 5 minutes.</td>
</tr>
</tbody>
</table>

NetCheckerTCPServerDelay
<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The TCP connection to Netchecker server takes too much time.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.agent }} Netchecker agent TCP connection time to the Netchecker server has increased by {{ $value }} within the last 5 minutes.</td>
</tr>
</tbody>
</table>

**NetCheckerDNSSlow**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The DNS lookup time is too high.</td>
</tr>
<tr>
<td>Description</td>
<td>The DNS lookup time on the {{ $labels.agent }} Netchecker agent has increased by {{ $value }} within the last 5 minutes.</td>
</tr>
</tbody>
</table>

**NGINX**

This section lists the alerts for the NGINX service.

- NginxServiceDown
- NginxDroppedIncomingConnections

**NginxServiceDown**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The NGINX service is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The NGINX service on the {{ $labels.node }} node is down.</td>
</tr>
</tbody>
</table>

**NginxDroppedIncomingConnections**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Minor</th>
</tr>
</thead>
</table>
NGINX drops incoming connections.

The NGINX service on the {{ $labels.node }} node drops {{ $value }} accepted connections per second for 5 minutes.

Node network

This section lists the alerts for a Kubernetes node network.

- SystemRxPacketsErrorTooHigh
- SystemTxPacketsErrorTooHigh
- SystemRxPacketsDroppedTooHigh
- SystemTxPacketsDroppedTooHigh
- NodeNetworkInterfaceFlapping

SystemRxPacketsErrorTooHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.node }} has package receive errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} network interface has receive errors on the {{ $labels.namespace }}/{{ $labels.pod }} node exporter Pod.</td>
</tr>
</tbody>
</table>

SystemTxPacketsErrorTooHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.node }} node has package transmit errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} network interface has transmit errors on the {{ $labels.namespace }}/{{ $labels.pod }} node exporter Pod.</td>
</tr>
</tbody>
</table>

SystemRxPacketsDroppedTooHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.node }} node drops {{ $value }} accepted connections per second for 5 minutes.</td>
</tr>
<tr>
<td>Summary</td>
<td>60 or more received packets were dropped.</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>$value \text{ packets received by the } {{ $labels.device }} \text{ interface on the } {{ $labels.node }} \text{ node were dropped during the last minute.}$</td>
</tr>
</tbody>
</table>

**SystemTxPacketsDroppedTooHigh**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>100 transmitted packets were dropped.</td>
</tr>
<tr>
<td>Description</td>
<td>$value \text{ packets transmitted by the } {{ $labels.device }} \text{ interface on the } {{ $labels.node }} \text{ node were dropped during the last minute.}$</td>
</tr>
</tbody>
</table>

**NodeNetworkInterfaceFlapping**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The $labels.node } \text{ node has flapping interface.}$</td>
</tr>
<tr>
<td>Description</td>
<td>The $labels.device } \text{ network interface often changes its UP status on the } {{ $labels.namespace }/{{ $labels.pod }} \text{ node exporter.}$</td>
</tr>
</tbody>
</table>

**Node time**

This section lists the alerts for a Kubernetes node time.

**ClockSkewDetected**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The NTP offset reached the limit of 0.03 seconds.</td>
</tr>
<tr>
<td>Description</td>
<td>Clock skew was detected on the $labels.namespace }/{{ $labels.pod }} \text{ node exporter Pod. Verify that NTP is configured correctly on this host.}$</td>
</tr>
</tbody>
</table>
PostgreSQL

This section lists the alerts for the PostgreSQL and Patroni services.

- PostgresqlDataPageCorruption
- PostgresqlDeadlocksDetected
- PostgresqlInsufficientWorkingMemory
- PostgresqlPatroniClusterSplitBrain
- PostgresqlPatroniClusterUnlocked
- PostgresqlPrimaryDown
- PostgresqlReplicaDown
- PostgresqlReplicationNonStreamingReplicas
- PostgresqlReplicationPaused
- PostgresqlReplicationSlowWalApplication
- PostgresqlReplicationSlowWalDownload
- PostgresqlReplicationWalArchiveWriteFailing

---

PostgresqlDataPageCorruption

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster member is experiencing data page corruption.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.pod }} Patroni Pod in the {{ $labels.cluster }} cluster fails to calculate the data page checksum due to a possible hardware fault.</td>
</tr>
</tbody>
</table>

---

PostgresqlDeadlocksDetected

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>PostgreSQL transactions deadlocks.</td>
</tr>
<tr>
<td>Description</td>
<td>The transactions submitted to the Patroni {{ $labels.cluster }} cluster in the {{ $labels.namespace }} Namespace are experiencing deadlocks.</td>
</tr>
</tbody>
</table>
PostgresqlInsufficientWorkingMemory

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Insufficient memory for PostgreSQL queries.</td>
</tr>
<tr>
<td>Description</td>
<td>The query data does not fit into working memory on the {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace.</td>
</tr>
</tbody>
</table>

PostgresqlPatroniClusterSplitBrain

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster split-brain detected.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace has multiple primaries, split-brain detected.</td>
</tr>
</tbody>
</table>

PostgresqlPatroniClusterUnlocked

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster primary node is missing.</td>
</tr>
<tr>
<td>Description</td>
<td>The primary node of the {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace is missing.</td>
</tr>
</tbody>
</table>

PostgresqlPrimaryDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>PostgreSQL is down on the cluster primary node.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace is down due to missing primary node.</td>
</tr>
</tbody>
</table>
### PostgresqlReplicaDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Minor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster has replicas with inoperable PostgreSQL.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace has {{ $value }}% of replicas with inoperable PostgreSQL.</td>
</tr>
</tbody>
</table>

### PostgresqlReplicationNonStreamingReplicas

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster has non-streaming replicas.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.cluster }} Patroni cluster in the {{ $labels.namespace }} Namespace has replicas not streaming the segments from the primary node.</td>
</tr>
</tbody>
</table>

### PostgresqlReplicationPaused

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Replication has stopped.</td>
</tr>
<tr>
<td>Description</td>
<td>Replication has stopped on the {{ $labels.namespace }}/{{ $labels.pod }} replica Pod in the {{ $labels.cluster }} cluster.</td>
</tr>
</tbody>
</table>

### PostgresqlReplicationSlowWalApplication

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>WAL segment application is slow.</td>
</tr>
</tbody>
</table>
PostgresqlReplicationSlowWalDownload

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Streaming replication is slow.</td>
</tr>
<tr>
<td>Description</td>
<td>Slow replication while downloading WAL segments for the {{ $labels.namespace }}/{{ $labels.pod }} replica Pod in the {{ $labels.cluster }} cluster.</td>
</tr>
</tbody>
</table>

PostgresqlReplicationWalArchiveWriteFailing

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Patroni cluster WAL segment writes are failing.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.namespace }}/{{ $labels.pod }} Patroni Pod in the {{ $labels.cluster }} cluster fails to write replication segments.</td>
</tr>
</tbody>
</table>

Prometheus

This section describes the alerts for the Prometheus service.

- PrometheusConfigReloadFailed
- PrometheusNotificationQueueRunningFull
- PrometheusErrorSendingAlertsWarning
- PrometheusErrorSendingAlertsMajor
- PrometheusNotConnectedToAlertmanagers
- PrometheusTSDBReloadsFailing
- PrometheusTSDBCompactionsFailing
- PrometheusTSDBWALCorruptions
- PrometheusNotIngestingSamples
- PrometheusTargetScrapesDuplicate
- PrometheusRuleEvaluationsFailed

PrometheusConfigReloadFailed

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Failure to reload the Prometheus configuration.</td>
</tr>
<tr>
<td>Description</td>
<td>Reloading of the Prometheus configuration has failed for the {{ $labels.namespace }}/{{ $labels.pod }} Pod.</td>
</tr>
</tbody>
</table>

PrometheusNotificationQueueRunningFull

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Prometheus alert notification queue is running full.</td>
</tr>
<tr>
<td>Description</td>
<td>The Prometheus alert notification queue is running full for the {{ $labels.namespace }}/{{ $labels.pod }} Pod.</td>
</tr>
</tbody>
</table>

PrometheusErrorSendingAlertsWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Errors occur while sending alerts from Prometheus.</td>
</tr>
<tr>
<td>Description</td>
<td>Errors occur while sending alerts from the {{ $labels.namespace }}/{{ $labels.pod }} Prometheus Pod to Alertmanager {{ $labels.Alertmanager }}.</td>
</tr>
</tbody>
</table>

PrometheusErrorSendingAlertsMajor

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Errors occur while sending alerts from Prometheus.

Errors occur while sending alerts from the
{{ $labels.namespace }}/{{ $labels.pod }} Prometheus Pod to Alertmanager
{{ $labels.Alertmanager }}.

Prometheus is not connected to Alertmanager.

The {{ $labels.namespace }}/{{ $labels.pod }} Prometheus Pod is not connected
to any Alertmanager instance.

Prometheus has issues reloading data blocks from disk.

Prometheus Pod had
{{ $value | humanize }} reload failures over the last 12 hours.

Prometheus has issues compacting sample blocks.

Prometheus Pod had
{{ $value | humanize }} compaction failures over the last 12 hours.
<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>Prometheus encountered WAL corruptions.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The <code>{ $labels.namespace }</code>/<code>{ $labels.pod }</code> Prometheus Pod has write-ahead log (WAL) corruptions in the time series database (TSDB) for the last 5 minutes.</td>
</tr>
</tbody>
</table>

PrometheusNotIngestingSamples

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>Prometheus does not ingest samples.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The <code>{ $labels.namespace }</code>/<code>{ $labels.pod }</code> Prometheus Pod does not ingest samples.</td>
</tr>
</tbody>
</table>

PrometheusTargetScrapesDuplicate

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>Prometheus has many rejected samples.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The <code>{ $labels.namespace }</code>/<code>{ $labels.pod }</code> Prometheus Pod has many rejected samples because of duplicate timestamps but different values.</td>
</tr>
</tbody>
</table>

PrometheusRuleEvaluationsFailed

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Summary</strong></td>
<td>Prometheus failed to evaluate recording rules.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>The <code>{ $labels.namespace }</code>/<code>{ $labels.pod }</code> Prometheus Pod has failed evaluations for recording rules. Verify the rules state in the Status/Rules section of the Prometheus Web UI.</td>
</tr>
</tbody>
</table>
Salesforce notifier
This section lists the alerts for the Salesforce notifier service.

- SfNotifierAuthFailure

---

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Failure to authenticate to Salesforce.</td>
</tr>
<tr>
<td>Description</td>
<td>The sf-notifier service fails to authenticate to Salesforce for 1 minute.</td>
</tr>
</tbody>
</table>

SMART disks
This section describes the alerts for SMART disks.

- SystemSMARTDiskUDMACrcErrorsTooHigh
- SystemSMARTDiskHealthStatus
- SystemSMARTDiskReadErrorRate
- SystemSMARTDiskSeekErrorRate
- SystemSMARTDiskTemperatureHigh
- SystemSMARTDiskReallocatedSectorsCount
- SystemSMARTDiskCurrentPendingSectors
- SystemSMARTDiskReportedUncorrectableErrors
- SystemSMARTDiskOfflineUncorrectableSectors
- SystemSMARTDiskEndToEndError

---

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.device }} disk has UDMA CRC errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.host }} node is reporting SMART UDMA CRC errors for 5 minutes.</td>
</tr>
</tbody>
</table>
### SystemSMARTDiskHealthStatus

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ $labels.device }</code> disk has bad health.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{ $labels.device }</code> disk on the <code>{ $labels.host }</code> node is reporting a bad health status for 1 minute.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskReadErrorRate

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ $labels.device }</code> disk has read errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{ $labels.device }</code> disk on the <code>{ $labels.host }</code> node is reporting an increased read error rate for 5 minutes.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskSeekErrorRate

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ $labels.device }</code> disk has seek errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The <code>{ $labels.device }</code> disk on the <code>{ $labels.host }</code> node is reporting an increased seek error rate for 5 minutes.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskTemperatureHigh

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The <code>{ $labels.device }</code> disk temperature is high.</td>
</tr>
</tbody>
</table>
### SystemSMARTDiskReallocatedSectorsCount

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.device }} disk has reallocated sectors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.host }} node has reallocated {{ $value }} sectors.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskCurrentPendingSectors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.device }} disk has current pending sectors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} current pending sectors.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskReportedUncorrectableErrors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>The {{ $labels.device }} disk has reported uncorrectable errors.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} reported uncorrectable errors.</td>
</tr>
</tbody>
</table>

### SystemSMARTDiskOfflineUncorrectableSectors

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
</table>
Summary

The {{ $labels.device }} disk has offline uncorrectable sectors

Description

The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} offline uncorrectable sectors.

SystemSMARTDiskEndToEndError

Severity

Major

Summary

The {{ $labels.device }} disk has end-to-end errors.

Description

The {{ $labels.device }} disk on the {{ $labels.host }} node has {{ $value }} end-to-end errors.

SSL certificates

This section lists the alerts for SSL certificates.

- SSLCertExpirationWarning
- SSLCertExpirationMajor
- SSLProbesFailing
- MCCSSLCertExpirationMajor
- MCCSSLCertExpirationWarning
- MCCSSLProbesFailing

SSLCertExpirationWarning

Severity

Warning

Summary

SSL certificate expires in 30 days.

Description

The SSL certificate for {{ $labels.instance }} expires in 30 days.

SSLCertExpirationMajor
<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>SSL certificate expires in 10 days.</td>
</tr>
<tr>
<td>Description</td>
<td>The SSL certificate for {$labels.instance} expires in 10 days.</td>
</tr>
</tbody>
</table>

SSLProbesFailing
Available since 2.2.0

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>SSL certificate probes are failing.</td>
</tr>
<tr>
<td>Description</td>
<td>The SSL certificate probes for the {$labels.instance} service endpoint are failing.</td>
</tr>
</tbody>
</table>

MCCSSLCertExpirationMajor

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>SSL certificate for a Container Cloud service expires in 10 days.</td>
</tr>
<tr>
<td>Description</td>
<td>The SSL certificate for the Container Cloud {$labels.service} service endpoint {$labels.instance} expires in 10 days.</td>
</tr>
</tbody>
</table>

MCCSSLCertExpirationWarning

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>SSL certificate for a Container Cloud service expires in 30 days.</td>
</tr>
<tr>
<td>Description</td>
<td>The SSL certificate for the Container Cloud {$labels.service} service endpoint {$labels.instance} expires in 30 days.</td>
</tr>
</tbody>
</table>
MCCSSSLProbesFailing

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>SSL certificate probes for a Container Cloud service are failing.</td>
</tr>
<tr>
<td>Description</td>
<td>The SSL certificate probes for the Container Cloud {{ $labels.instance }} service endpoint are failing.</td>
</tr>
</tbody>
</table>

Telemeter

This section describes the alerts for the Telemeter service.

- TelemeterClientFederationFailed

<table>
<thead>
<tr>
<th>Severity</th>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Telemeter client failed to send data to the server.</td>
</tr>
<tr>
<td>Description</td>
<td>Telemeter client has failed to send data to the Telemeter server twice for the last 30 minutes. Verify the telemeter-client container logs.</td>
</tr>
</tbody>
</table>

Mirantis Kubernetes Engine

This section describes the alerts for the Mirantis Kubernetes Engine (MKE) cluster.

- MKEAPIDown
- MKEAPIOutage
- MKEContainerUnhealthy
- MKENodeDiskFullCritical
- MKENodeDiskFullWarning
- MKENodeDown

MKEAPIDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>MKE API endpoint is down.</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Description</td>
<td>The MKE API endpoint {{ $labels.instance }} is not accessible for the last 3 minutes.</td>
</tr>
</tbody>
</table>

MKEAPIOutage

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>MKE API is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The MKE API (port 443) is not accessible for the last minute.</td>
</tr>
</tbody>
</table>

MKEContainerUnhealthy

<table>
<thead>
<tr>
<th>Severity</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>MKE container is in the Unhealthy state.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.name }} MKE container is in the Unhealthy state.</td>
</tr>
</tbody>
</table>

MKENodeDiskFullCritical

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>MKE node disk is 95% full.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} MKE node disk is 95% full.</td>
</tr>
</tbody>
</table>

MKENodeDiskFullWarning
### Severity

<table>
<thead>
<tr>
<th>Summary</th>
<th>MKE node disk is 85% full.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} MKE node disk is 85% full.</td>
</tr>
</tbody>
</table>

### MKENodeDown

<table>
<thead>
<tr>
<th>Severity</th>
<th>Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>MKE node is down.</td>
</tr>
<tr>
<td>Description</td>
<td>The {{ $labels.instance }} MKE node is down.</td>
</tr>
</tbody>
</table>

## Configure StackLight

This section describes the initial steps required for StackLight configuration. For a detailed description of StackLight configuration options, see StackLight configuration parameters.

1. Log in to the Mirantis Container Cloud web UI with the writer permissions.
2. Switch to the required project using the Switch Project action icon located on top of the main left-side navigation panel.
3. Expand the menu of the tab with your username.
4. Click Download kubeconfig to download kubeconfig of your management cluster.
5. Log in to any local machine with kubectl installed.
6. Copy the downloaded kubeconfig to this machine.
7. Run one of the following commands:
   - For a management cluster:
     ```
     kubectl --kubeconfig <KUBECONFIG_PATH> edit -n <PROJECT_NAME> cluster <MANAGEMENT_CLUSTER_NAME>
     ```
   - For a managed cluster:
     ```
     kubectl --kubeconfig <KUBECONFIG_PATH> edit -n <PROJECT_NAME> cluster <MANAGED_CLUSTER_NAME>
     ```
8. In the following section of the opened manifest, configure the required StackLight parameters as described in StackLight configuration parameters.
9. Verify StackLight after configuration.

**StackLight configuration parameters**

This section describes the StackLight configuration keys that you can specify in the values section to change StackLight settings as required. Prior to making any changes to StackLight configuration, perform the steps described in Configure StackLight. After changing StackLight configuration, verify the changes as described in Verify StackLight after configuration.

- Alerta
- Elasticsearch
- Grafana
- Logging
- High availability
- Metric collector
- Prometheus
- Cluster size
- Resource limits
- Kubernetes tolerations
- Storage class
- NodeSelector
- Salesforce reporter
- Ceph monitoring
- External endpoint monitoring
- Ironic monitoring
- SSL certificates monitoring
- Workload monitoring
- Mirantis Kubernetes Engine monitoring
- Alerts configuration
- Watchdog alert
- Alertmanager integrations
- Notifications to email
- Notifications to Salesforce
- Notifications to Slack

### Alerta

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Example values</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><code>alertEnabled</code></td>
<td>Enables or disables Alerta. Set to true by default.</td>
<td>true or false</td>
</tr>
<tr>
<td>Elasticsearch</td>
<td>Defines the Elasticsearch logstash-* index retention time in days. The logstash-* index stores all logs gathered from all nodes and containers. Set to 1 by default.</td>
<td>1, 5, 15</td>
</tr>
</tbody>
</table>
### Grafana

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>grafana.renderer.enabled (bool)</td>
<td>Disables Grafana Image Renderer. For example, for resource-limited environments. Enabled by default.</td>
<td>true or false</td>
</tr>
</tbody>
</table>
### dashboard

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>grafana.homeDASHBOARD(string)</td>
<td>Defines the home dashboard. Set to kubernetes-cluster by default. You can define any of the available dashboards.</td>
<td>kubernetes-cluster</td>
</tr>
</tbody>
</table>

### Logging

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>logging.enabled(bool)</td>
<td>Enables or disables the StackLight logging stack. For details about the logging components, see Reference Architecture: StackLight deployment architecture. Set to true by default.</td>
<td>true or false</td>
</tr>
</tbody>
</table>
## High availability

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>high availability enabled (bool)</td>
<td>Enables or disables StackLight multiserver mode. For details, see StackLight database modes in Reference Architecture: StackLight deployment architecture. Set to false by default.</td>
<td>true or false</td>
</tr>
</tbody>
</table>

## Metric collector

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Description</td>
<td>Example values</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>metric Collector enabled (bool)</td>
<td>Disables or enables the metric collector. Modify this parameter for the management cluster only. Set to false by default.</td>
<td>false or true</td>
</tr>
<tr>
<td>Prometheus Server Retention Time (string)</td>
<td>Defines the Prometheus database retention period. Passed to the <code>--storage.tsdb.retention.time</code> flag. Set to 15d by default.</td>
<td>15d, 1000h, 10d12h</td>
</tr>
<tr>
<td>Prometheus Server Retention Size (string)</td>
<td>Defines the Prometheus database retention size. Passed to the <code>--storage.tsdb.retention.size</code> flag. Set to 15GB by default.</td>
<td>15GB, 512MB</td>
</tr>
</tbody>
</table>
prometheus
Server
.alert Resend Delay (string)

Defines the minimum amount of time for Prometheus to wait before resending an alert to Alertmanager. Passed to the --rules.alert.resend-delay flag. Set to 2m by default.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2m, 90s</td>
</tr>
</tbody>
</table>

Cluster size

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster Size (string)</td>
<td>Specifies the approximate expected cluster size. Set to small by default. Other possible values include medium and large. Depending on the choice, appropriate resource limits are passed according to the resourcesPerClusterSize parameter. The values differ by the Elasticsearch and Prometheus resource limits:</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>small (default)</td>
<td>- 2 CPU, 6 Gi RAM for Elasticsearch, 1 CPU, 8 Gi RAM for Prometheus. Use small only for testing and evaluation purposes with no workloads expected.</td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>- 4 CPU, 16 Gi RAM for Elasticsearch, 3 CPU, 16 Gi RAM for Prometheus.</td>
<td></td>
</tr>
<tr>
<td>large</td>
<td>- 8 CPU, 32 Gi RAM for Elasticsearch, 6 CPU, 32 Gi RAM for Prometheus. Set to large only in case of lack of resources for Elasticsearch and Prometheus.</td>
<td></td>
</tr>
</tbody>
</table>

| small, medium, or large |

### Resource limits

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.

Page 148
resourcesPerClusterSize:
  elasticsearch:
    small:
      limits:
        cpu: "1000m"
        memory: "4Gi"
    medium:
      limits:
        cpu: "2000m"
        memory: "8Gi"
      requests:
        cpu: "1000m"
        memory: "4Gi"
    large:
      limits:
        cpu: "4000m"
        memory: "16Gi"

resources:
  alerta:
    requests:
      cpu: "50m"
      memory: "200Mi"
    limits:
      memory: "500Mi"

Using the example above, each pod in the alerta service will be requesting 50 millicores of CPU and 200 MiB of memory, while being hard-limited to 500 MiB of memory usage. Each configuration key is optional.

Kubernetes tolerations

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.
<table>
<thead>
<tr>
<th>tolerations.default (slice)</th>
<th>Defines Kubernetes tolerations (overrides the default ones) for any StackLight component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>default:</td>
<td>- key: &quot;com.docker.ucp.manager&quot; operator: &quot;Exists&quot; effect: &quot;NoSchedule&quot;</td>
</tr>
<tr>
<td>component:</td>
<td>elasticsearch:</td>
</tr>
<tr>
<td></td>
<td>- key: &quot;com.docker.ucp.manager&quot; operator: &quot;Exists&quot; effect: &quot;NoSchedule&quot;</td>
</tr>
<tr>
<td></td>
<td>postgresql:</td>
</tr>
<tr>
<td></td>
<td>- key: &quot;node-role.kubernetes.io/master&quot; operator: &quot;Exists&quot; effect: &quot;NoSchedule&quot;</td>
</tr>
</tbody>
</table>

### Storage class

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>defaultStorageClass</strong> (string)</td>
<td>Defines the StorageClass to use for all StackLight Persistent Volume Claims (PVCs) if a component StorageClass is not defined using the componentStorageClasses. To use the cluster default storage class, leave the string empty.</td>
<td>lvp, standard</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| **componentStorageClasses** | Defines (overrides the defaultStorageClass value) the storage class for any StackLight component separately. To use the cluster default storage class, leave the string empty. | ```
componentStorageClasses:
  elasticsearch: ""
  fluentd: ""
  postgresql: ""
  prometheusAlertManager: ""
  prometheusPushGateway: ""
  prometheusServer: ""
``` |
# NodeSelector

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeSelector.default(map)</td>
<td>Defines the NodeSelector to use for the most of StackLight pods (except some pods that refer to DaemonSets) if the NodeSelector of a component is not defined.</td>
<td>default:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>role: stacklight</td>
</tr>
<tr>
<td>nodeSelector.component(map)</td>
<td>Defines the NodeSelector to use for particular StackLight component pods. Overrides nodeSelector.default.</td>
<td>component:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>alerta:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>role: stacklight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>component: alerta</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kibana:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>role: stacklight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>component: kibana</td>
</tr>
</tbody>
</table>
Salesforce reporter

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterId (string)</td>
<td>Unique cluster identifier, clusterId=&quot;&lt;namespace&gt;/&lt;name&gt;/&lt;uid&gt;&quot;, generated for each cluster from the cluster namespace, cluster name, and cluster UID, separated by a slash. Cluster &lt;namespace&gt; is a Kubernetes namespace where the cluster is deployed. Cluster &lt;name&gt; is the name of the cluster object in the namespace. Cluster &lt;uid&gt; is the annotation set for each cluster. To obtain the cluster UID:</td>
<td>default/demo-9952/bcdb27ce-34b0-11eb-805d-0242c0a85b02</td>
</tr>
<tr>
<td>sfReporter.enabled (bool)</td>
<td>Enables or disables reporting of Prometheus metrics to Salesforce. For details, see StackLight deployment architecture. Disabled by default.</td>
<td>true or false</td>
</tr>
</tbody>
</table>
**Salesforce parameters and credentials for the metrics reporting integration.**

Note
Modify this parameter if sf-notifier is not configured or if you want to use a different Salesforce user account to send reports to.

```
salesForceAuth:
  url: "<SF instance URL>
  username: "<SF account email address>"
  password: "<SF password>"
  environment_id: "<Cloud identifier>"
  organization_id: "<Organization identifier>"
  sandbox_enabled: "<Set to true or false>"
```

**Cron job for sending metrics to Salesforce. By default, reports are sent at midnight server time.**

```
cronjob:
  schedule: "0 0 * * *"
  concurrencyPolicy: "Allow"
  failedJobsHistoryLimit: ""
  successfulJobsHistoryLimit: ""
  startingDeadlineSeconds: 200
```

---

**Ceph monitoring**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Description</td>
<td>Example values</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>ceph_enabled (bool)</td>
<td>Enables or disables Ceph monitoring. Set to false by default.</td>
<td>true or false</td>
</tr>
<tr>
<td>external Endpoint Monitoring enabled (bool)</td>
<td>Enables or disables HTTP endpoints monitoring. If enabled, the monitoring tool performs the probes against the defined endpoints every 15 seconds. Set to false by default.</td>
<td>true or false</td>
</tr>
<tr>
<td><strong>external Endpoint Monitoring certificates Host path (string)</strong></td>
<td>Defines the directory path with external endpoints certificates on host.</td>
<td><code>/etc/ssl/certs/</code></td>
</tr>
<tr>
<td><strong>external Endpoint Monitoring domains (slice)</strong></td>
<td><strong>domains:</strong></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>Defines the list of HTTP endpoints to monitor.</td>
<td>- https://prometheus.io_health</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <a href="http://example.com:8080_status">http://example.com:8080_status</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- <a href="http://example.net:8080_pulse">http://example.net:8080_pulse</a></td>
<td></td>
</tr>
</tbody>
</table>

### Ironic monitoring

<table>
<thead>
<tr>
<th><strong>Key</strong></th>
<th><strong>Description</strong></th>
<th><strong>Example values</strong></th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.
Enables or disables monitoring of bare metal Ironic. To enable, specify the Ironic API URL:

```
http://ironic-api-http.kaas.svc:6385/v1
```

### SSL certificates monitoring

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>sslCertificateMonitoringEnabled</td>
<td>Enables or disables StackLight to monitor and alert on the expiration date of the TLS certificate of an HTTPS endpoint. If enabled, the monitoring tool performs the probes against the defined endpoints every hour. Set to false by default.</td>
<td>true or false</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td><strong>Description</strong></td>
<td><strong>Example values</strong></td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
|          | Defines the list of HTTPS endpoints to monitor the certificates from. | **domains:**  
- https://prometheus.io  
- http://example.com:8080 |

**Workload monitoring**

<table>
<thead>
<tr>
<th><strong>Key</strong></th>
<th><strong>Description</strong></th>
<th><strong>Example values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
On the clusters that run large-scale workloads, workload monitoring generates a big amount of resource-consuming metrics. To prevent generation of excessive metrics, you can disable workload monitoring in the StackLight metrics and monitor only the infrastructure. The metricFilter parameter enables the cAdvisor (Container Advisor) and kubeStateMetrics metric ingestion filters for Prometheus. Set to false by default. If set to true, you can define the namespaces to which the filter will apply.

<table>
<thead>
<tr>
<th>metricFilter (map)</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enabled: true</td>
<td>true or false</td>
</tr>
<tr>
<td></td>
<td>action: keep</td>
<td></td>
</tr>
<tr>
<td></td>
<td>namespaces:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- kaas</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- kube-system</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- stacklight</td>
<td></td>
</tr>
</tbody>
</table>

- enabled - enable or disable metricFilter using true or false
- action - action to take by Prometheus:
  - keep - keep only metrics from namespaces that are defined in the namespaces list
  - drop - ignore metrics from namespaces that are defined in the namespaces list
- namespaces - list of namespaces to keep or drop metrics from regardless of the boolean value for every namespace

Mirantis Kubernetes Engine monitoring

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>make enabled (bool)</td>
<td>Enables or disables Mirantis Kubernetes Engine (MKE) monitoring. Set to false by default.</td>
<td>true or false</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Example values</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>make_docker_data_root</td>
<td>Defines the dockerd data root directory of persistent Docker state. For details, see Docker documentation: Daemon CLI (dockerd).</td>
<td>/var/lib/docker</td>
</tr>
</tbody>
</table>

## Alerts configuration

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.
**prometheusServer.customAlerts**

<table>
<thead>
<tr>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>
| Defines custom alerts. Also, modifies or disables existing alert configurations. For the list of predefined alerts, see Available StackLight alerts. While adding or modifying alerts, follow the Alerting rules. | customAlerts: 
# To add a new alert:
- alert: ExampleAlert 
  annotations:
  description: Alert description
  summary: Alert summary
  expr: example_metric > 0 
  for: 5m 
  labels: 
    severity: warning
# To modify an existing alert expression:
- alert: AlertmanagerFailedReload 
  expr: alertmanager_config_last_reload_successful == 5 
# To disable an existing alert:
- alert: TargetDown 
  enabled: false 
An optional field enabled is accepted in the alert body to disable an existing alert by setting to false. All fields specified using the customAlerts definition override the default predefined definitions in the charts’ values. |

Watchdog alert

<table>
<thead>
<tr>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td></td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
<tbody>
<tr>
<td>prometheusServer.watchDogAlertEnabled (bool)</td>
<td>Enables or disables the Watchdog alert that constantly fires as long as the entire alerting pipeline is functional. You can use this alert to verify that Alertmanager notifications properly flow to the Alertmanager receivers. Set to true by default.</td>
<td>true or false</td>
</tr>
<tr>
<td>alertmanagerSimpleConfig:</td>
<td>Provides a genetic template for notifications receiver configurations. For a list of supported receivers, see Prometheus Alertmanager documentation: Receiver.</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>genericReceivers</td>
<td>For example, to enable notifications to OpsGenie:</td>
<td></td>
</tr>
</tbody>
</table>
|                           | alertmanagerSimpleConfig:
|                           |   genericReceivers:
|                           |     - name: HTTP-opsgenie
|                           |       enabled: true # optional
|                           |     opsgenie_configs:
|                           |       - api_url: "https://example.app.eu.opsgenie.com/
|                           |         api_key: "secret-key"
|                           |         send_resolved: true
|                           | |
Provides a template for notifications route configuration. For details, see Prometheus Alertmanager documentation: Route.

<table>
<thead>
<tr>
<th><code>alertmanagerSimpleConf</code></th>
<th><code>genericRoutes:</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>- receiver: HTTP-opsgenie</code></td>
</tr>
<tr>
<td></td>
<td><code>enabled: true # optional</code></td>
</tr>
<tr>
<td></td>
<td><code>match_re:</code></td>
</tr>
<tr>
<td></td>
<td>`severity: major</td>
</tr>
<tr>
<td></td>
<td><code>continue: true</code></td>
</tr>
</tbody>
</table>

Notifications to email

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.
<table>
<thead>
<tr>
<th>alertmanager</th>
<th>Enables or disables Alertmanager integration with email. Set to false by default.</th>
<th>true or false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Configemail.enabled (bool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>alertmanager</td>
<td>Email configuration map</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Defines the notification parameters for Alertmanager integration with email. For details, see Prometheus Alertmanager documentation: Email configuration. | email:
| enabled: false
| send_resolved: true
| to: "to@test.com"
| from: "from@test.com"
| smarthost: smtp.gmail.com:587
| auth_username: "from@test.com"
| auth_password: password
| auth_identity: "from@test.com"
<p>| require_tls: true |</p>
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>
| alertmanager | Defines the route for Alertmanager integration with email. For details, see Prometheus Alertmanager documentation: Route. | route: 
  match: {}
  match_re: {}
  routes: [] |

Notifications to Salesforce

| Key | Description | Example values |
Unique cluster identifier, `clusterId`="<namespace>/<name>/<uid>", generated for each cluster from the cluster namespace, cluster name, and cluster UID, separated by a slash.

Cluster `<namespace>` is a Kubernetes namespace where the cluster is deployed. Cluster `<name>` is the name of the cluster object in the namespace. Cluster `<uid>` is the annotation set for each cluster. To obtain the cluster UID:

```
kubectl get cluster -n "<namespace>" <name> -o jsonpath="\{.metadata.annotations.kaas.mirantis.com/uid\}"
```

The sf-notifier and sf-reporter services use the same `clusterId` parameter.
<table>
<thead>
<tr>
<th>alertmanager</th>
<th>Enables or disables Alertmanager integration with Salesforce using the sf-notifier service. Disabled by default.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true or false</td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.  Page 171
<table>
<thead>
<tr>
<th>alertmanager</th>
<th>Defines the Salesforce parameters and credentials for integration with Alertmanager.</th>
</tr>
</thead>
</table>

```yaml
auth:
  url: "<SF instance URL>"
  username: "<SF account email address>"
  password: "<SF password>"
  environment_id: "<Cloud identifier>"
  organization_id: "<Organization identifier>"
  sandbox_enabled: "<Set to true or false>"
```
<table>
<thead>
<tr>
<th>alertermanagerSimpleConfigSalesforce.route (map)</th>
<th>Defines the notifications route for Alertmanager integration with Salesforce. For details, see Prometheus Alertmanager documentation: Route.</th>
</tr>
</thead>
<tbody>
<tr>
<td>route:</td>
<td><code>{}</code></td>
</tr>
<tr>
<td>match:</td>
<td><code>{}</code></td>
</tr>
<tr>
<td>match_re:</td>
<td><code>{}</code></td>
</tr>
<tr>
<td>routes:</td>
<td><code>[]</code></td>
</tr>
</tbody>
</table>

Notifications to Slack

<table>
<thead>
<tr>
<th>K e y</th>
<th>Description</th>
<th>Example values</th>
</tr>
</thead>
</table>

©2021, Mirantis Inc.
| alertmanager | Enables or disables Alertmanager integration with Slack. For details, see Prometheus Alertmanager documentation: Slack configuration. Set to false by default. | true or false |

Enables or disables Alertmanager integration with Slack. For details, see Prometheus Alertmanager documentation: Slack configuration. Set to false by default.
| alert_manager | Defines the Slack webhook URL. | http://localhost:8888 |
| alert manager Simple Config.slack.channel(string) | Defines the Slack channel or user to send notifications to. | monitoring |
### Verify StackLight after configuration

This section describes how to verify StackLight after configuring its parameters as described in Configure StackLight and StackLight configuration parameters. Perform the verification procedure described for a particular modified StackLight key.

To verify StackLight after configuration:

<table>
<thead>
<tr>
<th>Key</th>
<th>Verification procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>alerta.enabled</td>
<td>Verify that Alerta is present in the list of StackLight resources. An empty output indicates that Alerta is disabled.</td>
</tr>
<tr>
<td></td>
<td><code>kubectl get all -n stacklight -l app=alerta</code></td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Elasticsearch Log Stash Retention Time</td>
<td>Verify that the unit_count parameter contains the desired number of days:</td>
</tr>
<tr>
<td>Grafana Image Renderer Enabled</td>
<td>Verify the Grafana Image Renderer. If set to true, the output should include HTTP Server started, listening at <a href="http://localhost:8081">http://localhost:8081</a>.</td>
</tr>
<tr>
<td>Grafana Home Dashboard</td>
<td>In the Grafana web UI, verify that the desired dashboard is set as a home dashboard.</td>
</tr>
<tr>
<td>Logging Enabled</td>
<td>Verify that Elasticsearch, Fluentd, and Kibana are present in the list of StackLight resources. An empty output indicates that the StackLight logging stack is disabled.</td>
</tr>
<tr>
<td>High Availability Enabled</td>
<td>Run <code>kubectl get sts -n stacklight</code>. The output includes the number of services replicas for the HA or non-HA StackLight modes. For details, see StackLight deployment architecture.</td>
</tr>
<tr>
<td>Metric Collector Enabled</td>
<td>Verify that metric collector is present in the list of StackLight resources. An empty output indicates that metric collector is disabled.</td>
</tr>
<tr>
<td>Prometheus Server Retention Time</td>
<td>In the Prometheus web UI, navigate to Status &gt; Command-Line Flags. Verify the values for the following flags:</td>
</tr>
<tr>
<td></td>
<td>• <code>storage.tsdb.retention.size</code></td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.
1. Obtain the list of pods:
   kubectl get po -n stacklight

2. Verify that the desired resource limits or requests are set in the resources section of every container in the pod:
   kubectl get po <pod_name> -n stacklight -o yaml

Verify that the appropriate components pods are located on the intended nodes:

   kubectl get pod -o=custom-columns=NAME:.metadata.name,\ 
   STATUS:.status.phase,NODE:.spec.nodeName -n stacklight

Verify that the appropriate components PVCs have been created according to the configured StorageClass:

   kubectl get pvc -n stacklight

Verify that Salesforce reporter is enabled. The SUSPEND field in the output must be False.

   kubectl get cronjob -n stacklight

2. Verify that the Salesforce reporter configuration includes all expected queries:

   kubectl get configmap -n stacklight \ 
   sf-reporter-config -o yaml

3. After cron job execution (by default, at midnight server time), obtain the Salesforce reporter pod name. The output should include the Salesforce reporter pod name and STATUS must be Completed.

   kubectl get pods -n stacklight

4. Verify that Salesforce reporter successfully authenticates to Salesforce and creates records. The output must include the Salesforce authentication successful, Created record or Duplicate record and Updated record lines.

   kubectl logs -n stacklight <sf-reporter-pod-name>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the Grafana web UI, verify that Ceph dashboards are present in the list of dashboards and are populated with data. 2. In the Prometheus web UI, click Alerts and verify that the list of alerts contains Ceph alerts.</td>
<td></td>
</tr>
</tbody>
</table>
| • externalEndpointMonitoring.enabled  
• externalEndpointMonitoring.domains | 1. In the Prometheus web UI, navigate to Status -> Targets. 2. Verify that the blackbox-external-endpoint target contains the configured domains (URLs). |
| In the Grafana web UI, verify that the Ironic BM dashboard displays valuable data (no false-positive or empty panels). | 1. In the Prometheus web UI, navigate to Status -> Targets. 2. Verify that the blackbox target contains the configured domains (URLs). |
| 1. In the Prometheus web UI, navigate to Status > Configuration. 2. Verify that the following fields in the metric_relabel_configs section for the kubernetes-nodes-cadvisor and prometheus-kube-state-metrics scrape jobs have the required configuration:  
• action is set to keep or drop  
• regex contains a regular expression with configured namespaces delimited by |  
• source_labels is set to [namespace] | 1. In the Prometheus web UI, navigate to Status -> Targets. 2. Verify that the blackbox target contains the configured domains (URLs). |
<p>| 1. In the Grafana web UI, verify that the MKE Cluster and MKE Containers dashboards are present and not empty. 2. In the Prometheus web UI, navigate to Alerts and verify that the MKE* alerts are present in the list of alerts. | In the Prometheus web UI, navigate to Alerts and verify that the MKEAPIDown is not false-positively firing due to the certificate absence. |
| In the Prometheus web UI, navigate to Alerts and verify that the list of alerts has changed according to your customization. |</p>
<table>
<thead>
<tr>
<th>prometheusServer.watchDogAlertsEnabled</th>
<th>In the Prometheus web UI, navigate to Alerts and verify that the list of alerts contains the Watchdog alert.</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertmanagerSimpleConfig.email</td>
<td>In the Alertmanager web UI, navigate to Status and verify that the Config section contains the intended receiver(s).</td>
</tr>
<tr>
<td>alertmanagerSimpleConfig.email.route</td>
<td>In the Alertmanager web UI, navigate to Status and verify that the Config section contains the intended route(s).</td>
</tr>
<tr>
<td></td>
<td>In the Alertmanager web UI, navigate to Status and verify that the Config section contains the Email receiver and route.</td>
</tr>
<tr>
<td></td>
<td>In the Alertmanager web UI, navigate to Status and verify that the Config section contains the HTTP-salesforce receiver and route.</td>
</tr>
</tbody>
</table>

1. Verify that sf-notifier is enabled. The output must include the sf-notifier pod name, 1/1 in the READY field and Running in the STATUS field.

   ```
   kubectl get pods -n stacklight
   ```

2. Verify that sf-notifier successfully authenticates to Salesforce. The output must include the Salesforce authentication successful line.

   ```
   kubectl logs -f -n stacklight <sf-notifier-pod-name>
   ```

3. In the Alertmanager web UI, navigate to Status and verify that the Config section contains the HTTP-salesforce receiver and route.
In the Alertmanager web UI, navigate to Status and verify that the Config section contains the HTTP Slack receiver and route.

- alertmanagerSimpleConfig.slack.enabled
- alertmanagerSimpleConfig.slack.api_url
- alertmanagerSimpleConfig.slack.channel
- alertmanagerSimpleConfig.slack.route

Enable generic metric scraping

StackLight can scrape metrics from any service that exposes Prometheus metrics and is running on the Kubernetes cluster. Such metrics appear in Prometheus under the `{job="stacklight-generic",service="<service_name>",namespace="<service_namespace>"}` set of labels. If the Kubernetes service is backed by Kubernetes pods, the set of labels also includes `{pod="<pod_name>"}`.

To enable the functionality, define at least one of the following annotations in the service metadata:

- "generic.stacklight.mirantis.com/scrape-port" - the HTTP endpoint port. By default, the port number found through Kubernetes service discovery, usually `__meta_kubernetes_pod_container_port_number`. If none discovered, use the default port for the chosen scheme.
- "generic.stacklight.mirantis.com/scrape-path" - the HTTP endpoint path, related to the Prometheus scrape_config.metrics_path option. By default, /metrics.
- "generic.stacklight.mirantis.com/scrape-scheme" - the HTTP endpoint scheme between HTTP and HTTPS, related to the Prometheus scrape_config.scheme option. By default, http.

For example:

```
metadata:
  annotations:
    "generic.stacklight.mirantis.com/scrape-path": "/metrics"
```

```
metadata:
  annotations:
    "generic.stacklight.mirantis.com/scrape-port": "8080"
```

See also

Prometheus scrape job configuration reference
Manage Ceph

This section outlines Ceph LCM operations such as adding Ceph Monitor, Ceph nodes, and RADOS Gateway nodes to an existing Ceph cluster or removing them, as well as removing or replacing Ceph OSDs or updating your Ceph cluster.

Enable automated Ceph LCM

Ceph controller can automatically redeploy Ceph OSDs in case of significant configuration changes such as changing the block.db device or replacing Ceph OSDs. Ceph controller can also clean disks and configuration during a Ceph OSD removal.

To remove a single Ceph OSD or the entire Ceph node, manually remove its definition from the kaasCephCluster CR.

To enable automated management of Ceph OSDs:

1. Log in to a local machine running Ubuntu 18.04 where kubectl is installed.
2. Obtain and export kubeconfig of the management cluster as described in Connect to a Mirantis Container Cloud cluster.
3. Open the KaasCephCluster CR for editing. Choose from the following options:
   - For a management cluster:
     ```
     kubectl edit kaascephcluster
     ```
   - For a managed cluster:
     ```
     kubectl edit kaascephcluster -n <managedClusterProjectName>
     ```
     Substitute `<managedClusterProjectName>` with the corresponding value.
4. Set the manageOsds parameter to true:
   ```
   spec:
     cephClusterSpec:
       manageOsds: true
   ```

Once done, all Ceph OSDs with a modified configuration will be redeployed. Mirantis recommends modifying only one Ceph node at a time. For details about supported configuration parameters, see OSD Configuration Settings.

Add, remove, or reconfigure Ceph nodes

Mirantis Ceph controller simplifies a Ceph cluster management by automating LCM operations. To modify Ceph components, only the MiraCeph custom resource (CR) update is required. Once you update the MiraCeph CR, the Ceph controller automatically adds, removes, or reconfigures Ceph nodes as required.
To add, remove, or reconfigure Ceph nodes on a management or managed cluster:

1. To modify Ceph OSDs, verify that the manageOsds parameter is set to true in the KaasCephCluster CR as described in Enable automated Ceph LCM.
2. Log in to a local machine running Ubuntu 18.04 where kubectl is installed.
3. Obtain and export kubeconfig of the management cluster as described in Connect to a Mirantis Container Cloud cluster.
4. Open the KaasCephCluster CR for editing. Choose from the following options:
   - For a management cluster:
     
     ```bash
     kubectl edit kaascephcluster
     ```
   - For a managed cluster:
     
     ```bash
     kubectl edit kaascephcluster -n <managedClusterProjectName>
     ```

     Substitute `<managedClusterProjectName>` with the corresponding value.
5. In the nodes section, specify or remove the parameters for a Ceph OSD as required. For the parameters description, see OSD Configuration Settings.

   For example:

   ```yaml
   nodes:
   kaas-mgmt-node-5bgk6:
     roles:
     - mon
     - mgr
     storageDevices:
     - config:
       storeType: bluestore
       name: sdb
   ```

   Note
   To use a new Ceph node for a Ceph Monitor or Ceph Manager deployment, also specify the roles parameter.

Note
When adding a Ceph node with the Ceph Monitor role, if any issues occur with the Ceph Monitor, rook-ceph removes it and adds a new Ceph Monitor instead, named using the next alphabetic character in order. Therefore, the Ceph Monitor names may not follow the alphabetical order. For example, a, b, d, instead of a, b, c.
6. If you are making changes for your managed cluster, obtain and export kubeconfig of the managed cluster as described in Connect to a Mirantis Container Cloud cluster. Otherwise, skip this step.

7. Monitor the status of your Ceph cluster deployment. For example:

   kubectl -n rook-ceph get pods
   kubectl -n ceph-lcm-mirantis logs ceph-controller-78c95fb75c-dtbxk
   kubectl -n rook-ceph logs rook-ceph-operator-56d6b49967-5swxr

8. Connect to the terminal of the ceph-tools pod:

   kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod
   -l "app=rook-ceph-tools" -o jsonpath='{.items[0].metadata.name}') bash

9. Verify that the Ceph node has been successfully added, removed, or reconfigured:

   1. Verify that the Ceph cluster status is healthy:

      ceph status

      Example of a positive system response:

      cluster:
      id: 0868d89f-0e3a-456b-afc4-59f06ed9fbf7
      health: HEALTH_OK

      services:
      mon: 3 daemons, quorum a,b,c (age 20h)
      mgr: a(active, since 20h)
      osd: 9 osds: 9 up (since 20h), 9 in (since 2d)

      data:
      pools: 1 pools, 32 pgs
      objects: 0 objects, 0 B
      usage: 9.1 GiB used, 231 GiB / 240 GiB avail
      pgs: 32 active+clean

   2. Verify that the status of the Ceph OSDs is up:

      ceph osd tree

      Example of a positive system response:
<table>
<thead>
<tr>
<th>ID</th>
<th>CLASS</th>
<th>WEIGHT</th>
<th>TYPE</th>
<th>NAME</th>
<th>STATUS</th>
<th>REWEIGHT</th>
<th>PRI-AFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td></td>
<td>0.23424</td>
<td>root</td>
<td>default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td></td>
<td>0.07808</td>
<td></td>
<td>host osd1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.1</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>hdd</td>
<td>0.01949</td>
<td>osd.3</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.6</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>-15</td>
<td></td>
<td>0.07808</td>
<td></td>
<td>host osd2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.2</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>hdd</td>
<td>0.01949</td>
<td>osd.5</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.8</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td></td>
<td>0.07808</td>
<td></td>
<td>host osd3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.0</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>hdd</td>
<td>0.01949</td>
<td>osd.4</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>hdd</td>
<td>0.02930</td>
<td>osd.7</td>
<td>up</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
</tbody>
</table>

See also

**Storage known issues: Ceph OSD node removal fails**

**Replace a failed Ceph OSD**

After a physical disk replacement, you can use Rook to redeploy a failed Ceph OSD by restarting rook-operator that triggers the reconfiguration of the management or managed cluster.

To redeploy a failed Ceph OSD:

1. Log in to a local machine running Ubuntu 18.04 where kubectl is installed.
2. Obtain and export kubeconfig of the required management or managed cluster as described in Connect to a Mirantis Container Cloud cluster.
3. Identify the failed Ceph OSD ID:
   
   ```
   ceph osd tree
   ```

4. Remove the Ceph OSD deployment from the management or managed cluster:
   
   ```
   kubectl delete deployment -n rook-ceph rook-ceph-osd-<ID>
   ```

5. Connect to the terminal of the ceph-tools pod:
   
   ```
   kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod \ 
   -l "app=rook-ceph-tools" -o jsonpath='{.items[0].metadata.name}') bash
   ```

6. Remove the failed Ceph OSD from the Ceph cluster:
   
   ```
   ceph osd purge osd.<ID>
   ```
7. Replace the failed disk.
8. Restart the Rook operator:

```
kubectl delete pod $(kubectl -n rook-ceph get pod -l "app=rook-ceph-operator" -o jsonpath='{.items[0].metadata.name}') -n rook-ceph
```

**Update Ceph cluster**

You can update Ceph cluster to the latest minor version of Ceph Nautilus by triggering the existing Ceph cluster update.

To update Ceph cluster:

1. Verify that your management cluster is automatically upgraded to the latest Mirantis Container Cloud release:
   1. Log in to the Container Cloud web UI with the writer permissions.
   2. On the bottom of the page, verify the Container Cloud version number.
2. Verify that your managed clusters are updated to the latest Cluster release. For details, see Update a managed cluster.
3. Log in to a local machine running Ubuntu 18.04 where kubectl is installed.
4. Obtain and export kubeconfig of the management cluster as described in Connect to a Mirantis Container Cloud cluster.
5. Open the KaasCephCluster CR for editing:

```
kubectl edit kaascephcluster
```

6. Update the version parameter. For example:

```
version: 14.2.9
```

7. Obtain and export kubeconfig of the managed clusters as described in Connect to a Mirantis Container Cloud cluster.
8. Repeat the steps 5-7 to update Ceph on every managed cluster.

**Enable Ceph RGW Object Storage**

**Caution!**

This feature is available starting from the Container Cloud release 2.3.0.
Ceph controller enables you to deploy RADOS Gateway (RGW) Object Storage instances and automatically manages its resources such as users and buckets.

To enable the RGW Object Storage:

1. Select from the following options:
   - If you do not have a management cluster yet, open kaascephcluster.yaml.template for editing.
   - If the management cluster is already deployed, open the KaasCephCluster CR for editing. Select from the following options:
     - If the Ceph cluster is placed in the management cluster:
       
       kubectl edit kaascephcluster

     - If the Ceph cluster is placed in a managed cluster:
       
       kubectl edit kaascephcluster -n <managedClusterProjectName>

       Substitute <managedClusterProjectName> with the corresponding value.

2. Update the rgw section specification as required. For example:

```
rgw:
  name: rgw-store
  dataPool:
    erasureCoded:
      codingChunks: 1
      dataChunks: 2
      failureDomain: host
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
  gateway:
    allNodes: false
    instances: 1
    port: 80
    securePort: 8443
  preservePoolsOnDelete: false
  ssl:
    -----BEGIN RSA PRIVATE KEY-----
    -----END RSA PRIVATE KEY-----
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
```

The following tables describe the specification keys.

<table>
<thead>
<tr>
<th>Specification Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGW section specification</td>
</tr>
</tbody>
</table>

©2021, Mirantis Inc.

Page 188
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of Ceph RGW Object Storage.</td>
<td>true</td>
</tr>
</tbody>
</table>
| dataPool         | The specification to create the object storage data pool. May also contain the failureDomain key (set to osd or host) defining the failure domain across which the data will be spread. Can use replicated or erasure coded settings. For example:  

```json
rgw:
dataPool:
  replicated:
    size: 3
metadataPool:
  replicated:
    size: 3
```

where replicated.size is the number of full copies of the data on multiple nodes.  

```json
rgw:
dataPool:
  erasureCoded:
    codingChunks: 1
dataChunks: 2
```

For details, see Rook documentation: Erasure coding.  

| metadataPool     | The specification to create all object storage metadata pools. May also contain the failureDomain key (set to osd or host) defining the failure domain across which the data will be spread. Can use only replicated settings. | true     |
| gateway          | The specification of the Ceph RGW daemon settings.                          | true     |
ssl: | 
<add-certs-here>

Note
An empty ssl field of the rgw section causes automatic generation of required certificates. Mirantis strongly recommends that you do not use generated certificates for production deployment.

<p>| RGW Gateway specification |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>The port on which the Ceph RGW service will be listening on HTTP.</td>
<td>true</td>
</tr>
<tr>
<td>securePort</td>
<td>The port on which the RGW service will be listening on HTTPS.</td>
<td>true</td>
</tr>
<tr>
<td>instances</td>
<td>The number of pods in the Ceph RGW ReplicaSet. If allNodes is specified, a DaemonSet is created.</td>
<td>true</td>
</tr>
<tr>
<td>allNodes</td>
<td>Defines whether the Ceph RGW pods should be started as a DaemonSet on all nodes.</td>
<td>true</td>
</tr>
</tbody>
</table>

3. Optional. To create new Ceph RGW resources, such as buckets or users, update the rgw section with the following keys. Once done, Ceph controller automatically creates the specified object storage users and buckets in the Ceph cluster.

rgw:
  ...
  buckets:
Verify Ceph components

The starting point for Ceph troubleshooting is the ceph-controller and rook-operator logs. Once you locate the component that causes issues, verify the logs of the related pod.

This section describes how to verify the components of a Ceph cluster.

Verify the Ceph core services

To confirm that all Ceph components including mon, mgr, osd, and rgw have joined your cluster properly, analyze the logs for each pod and verify the Ceph status:

```
kubectl exec -it rook-ceph-tools-5748bc69c6-cpzf8 -n rook-ceph bash ceph -s
```

Example of a positive system response:

```
cluster:
  id:     4336ab3b-2025-4c7b-b9a9-3999944853c8
  health: HEALTH_OK

services:
  mon: 3 daemons, quorum a,b,c (age 20m)
  mgr: a(active, since 19m)
  osd: 6 osds: 6 up (since 16m), 6 in (since 16m)
  rgw: 1 daemon active (miraobjstore.a)

data:
  pools:   12 pools, 216 pgs
  objects: 201 objects, 3.9 KiB
  usage:   6.1 GiB used, 174 GiB / 180 GiB avail
  pgs:     216 active+clean
```

Verify rook-discover

To ensure that rook-discover is running properly, verify if the local-device configmap has been created for each Ceph node specified in the cluster configuration:

1. Obtain the list of local devices:
kubectl get configmap -n rook-ceph | grep local-device

Example of a system response:

<table>
<thead>
<tr>
<th>Name</th>
<th>Count</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>local-device-01</td>
<td>1</td>
<td>30m</td>
</tr>
<tr>
<td>local-device-02</td>
<td>1</td>
<td>29m</td>
</tr>
<tr>
<td>local-device-03</td>
<td>1</td>
<td>30m</td>
</tr>
</tbody>
</table>

2. Verify that each device from the list contains information about available devices for the Ceph node deployment:

kubectl describe configmap local-device-01 -n rook-ceph

Example of a positive system response:

Verify Ceph cluster state

To verify the state of a Ceph cluster, Ceph controller provides a Kubernetes API that includes a custom MiraCephLog resource. The resource contains information about the state of different components of your Ceph cluster.

To verify the Ceph cluster state:

1. Obtain kubeconfig of the management or managed cluster and provide it as an environment variable:

   export KUBECONFIG=<path-to-kubeconfig>

2. Obtain MiraCephLog:

   kubectl get miracephlog rook-ceph -o yaml

3. Verify the state of the required component using the MiraCephLog specification description below.

   Specification fields of the MiraCephLog object

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lastLogs</td>
<td>The tail of the ceph-operator pod logs. Use the logs for investigation and troubleshooting.</td>
</tr>
<tr>
<td>osdStatus</td>
<td>The string result of the current state of Ceph OSDs. If all OSDs operate properly, the value is ALL OK.</td>
</tr>
<tr>
<td>pools</td>
<td>The list of Ceph block pools. Use this list to verify whether all defined pools have been created properly.</td>
</tr>
</tbody>
</table>
Troubleshooting

This section provides solutions to the issues that may occur while operating a Mirantis Container Cloud management, regional, or child cluster.

Collect cluster logs

Caution!

This feature is available starting from the Container Cloud release 2.2.0.

While operating your management, regional, or managed cluster, you may require collecting and inspecting the cluster logs to analyze cluster events or troubleshoot issues. For the logs structure, see Deployment Guide: Collect the bootstrap logs.

To collect cluster logs:

1. Choose from the following options:
   - If you did not delete the kaas-bootstrap folder from the bootstrap node, log in to the bootstrap node.
   - If you deleted the kaas-bootstrap folder:
     1. Log in to a local machine running Ubuntu 18.04 where kubectl is installed.
     2. Download and run the Container Cloud bootstrap script:

```
wget https://binary.mirantis.com/releases/get_container_cloud.sh
chmod 0755 get_container_cloud.sh
./get_container_cloud.sh
```

2. Obtain kubeconfig of the required cluster. The management or regional cluster kubeconfig files are created during the last stage of the management or regional cluster bootstrap. To obtain a managed cluster kubeconfig, see Connect to a Mirantis Container Cloud cluster.

3. Obtain the private SSH key of the required cluster. For a management or regional cluster, this key is created during bootstrap of a management cluster in ~/.ssh/openstack_tmp. For a managed cluster, this is an SSH key added in the Container Cloud web UI before the managed cluster creation.

4. Depending on the cluster type that you require logs from, run the corresponding command:
   - For a management cluster:
kaas collect logs --management-kubeconfig <pathToMgmtClusterKubeconfig> \
--key-file <pathToMgmtClusterPrivateSshKey> \
--cluster-name <clusterName> --cluster-namespace <clusterProject>

• For a regional cluster:

kaas collect logs --management-kubeconfig <pathToMgmtClusterKubeconfig> \
--key-file <pathToRegionalClusterSshKey> --kubeconfig <pathToRegionalClusterKubeconfig> \
--cluster-name <clusterName> --cluster-namespace <clusterProject>

• For a managed cluster:

kaas collect logs --management-kubeconfig <pathToMgmtClusterKubeconfig> \
--key-file <pathToManagedClusterSshKey> --kubeconfig <pathToManagedClusterKubeconfig> \
--cluster-name <clusterName> --cluster-namespace <clusterProject>

Substitute the parameters enclosed in angle brackets with the corresponding values of your cluster.

Optionally, add --output-dir that is a directory path to collect logs. The default value is logs/. For example, logs/<clusterName>/events.log.
Mirantis Container Cloud API

Warning
This section is intended only for advanced Infrastructure Operators who are familiar with Kubernetes Cluster API.

Mirantis currently supports only those Mirantis Container Cloud API features that are implemented in the Container Cloud web UI. Use other Container Cloud API features for testing and evaluation purposes only.

The Container Cloud APIs are implemented using the Kubernetes CustomResourceDefinitions (CRDs) that enable you to expand the Kubernetes API. Different types of resources are grouped in the dedicated files, such as cluster.yaml or machines.yaml.

This section contains descriptions and examples of the Container Cloud API resources for the bare metal cloud provider.

Note
The API documentation for the OpenStack, AWS, and VMWare vSphere resources will be added in the upcoming Container Cloud releases.

Public key resources
This section describes the PublicKey resource used in Mirantis Container Cloud API for all supported providers: OpenStack, AWS, and bare metal. This resource is used to provide SSH access to every machine of a Container Cloud cluster.

The Container Cloud PublicKey CR contains the following fields:

- apiVersion
  API version of the object that is kaas.mirantis.com/v1alpha1
- kind
  Object type that is PublicKey
- metadata
  The metadata object field of the PublicKey resource contains the following fields:
  - name
    Name of the public key
  - namespace
    Project where the public key is created
The spec object field of the PublicKey resource contains the publicKey field that is an SSH public key value.

The PublicKey resource example:

```
apiVersion: kaas.mirantis.com/v1alpha1
g kind: PublicKey
metadata:
  name: demokey
  namespace: test
spec:
  publicKey: |
    ssh-rsa AAAAB3NzaC1yc2EAAAA...
```

Bare metal resources

This section contains descriptions and examples of the baremetal-based Kubernetes resources for Mirantis Container Cloud.

Cluster

This section describes the Cluster resource used the in Mirantis Container Cloud API that describes the cluster-level parameters.

For demonstration purposes, the Container Cloud Cluster custom resource (CR) is split into the following major sections:

- metadata
- spec:providerSpec
- spec:providerSpec common
- spec:providerSpec configuration
- status:providerStatus common
- status:providerStatus for cluster readiness
- status:providerStatus for Open ID Connect
- status:providerStatus for cluster releases

**Warning**

The fields of the Cluster resource that are located under the status section including providerStatus are available for viewing only. They are automatically generated by the bare metal cloud provider and must not be modified using Container Cloud API.
metadata

The Container Cloud Cluster CR contains the following fields:

- **apiVersion**
  
  API version of the object that is ipam.mirantis.com/v1alpha1.

- **kind**
  
  Object type that is Cluster.

The metadata object field of the Cluster resource contains the following fields:

- **name**
  
  Name of a cluster. A managed cluster name is specified under the Cluster Name field in the Create Cluster wizard of the Container Cloud web UI. A management and regional cluster names are configurable in the bootstrap script.

- **namespace**
  
  Project in which the cluster object was created. The management and regional clusters are created in the default project. The managed cluster project equals to the selected project name.

- **labels**
  
  Key-value pairs attached to the object:

  - **kaas.mirantis.com/provider**
    
    Provider type that is baremetal for the baremetal-based clusters.

  - **kaas.mirantis.com/region**
    
    Region name. The default region name for the management cluster is region-one. For the regional cluster, it is configurable using the REGION parameter in the bootstrap script.

Configuration example:

```
apiVersion: cluster.k8s.io/v1alpha1
gkind: Cluster
metadata:
  name: demo
  namespace: test
labels:
  kaas.mirantis.com/provider: baremetal
  kaas.mirantis.com/region: region-one
```

**spec:providerSpec**

The spec object field of the Cluster object represents the BaremetalClusterProviderSpec subresource that contains a complete description of the desired bare metal cluster state and all details to create the cluster-level resources. It also contains the fields required for LCM deployment and integration of the Container Cloud components.

The providerSpec object field is custom for each cloud provider and contains the following generic fields for the bare metal provider:
- **apiVersion**
  API version of the object that is baremetal.k8s.io/v1alpha1
- **kind**
  Object type that is BaremetalClusterProviderSpec

Configuration example:

```yaml
spec:
  ...
  providerSpec:
    value:
      apiVersion: baremetal.k8s.io/v1alpha1
      kind: BaremetalClusterProviderSpec
```

**spec:providerSpec common**

The `providerSpec` object field of the Cluster resource contains the following common fields for all Container Cloud providers:

- **publicKeys**
  List of the SSH public key references
- **release**
  Name of the ClusterRelease object to install on a cluster
- **helmReleases**
  List of the enabled Helm releases from the Release object that run on a Container Cloud cluster

Configuration example:

```yaml
spec:
  ...
  providerSpec:
    value:
      publicKeys:
        - name: bootstrap-key
      release: ucp-5-7-0-3-3-3-tp11
      helmReleases:
        - name: metallb
          values:
            configInline:
              address-pools:
                addresses:
                  - 10.0.0.101-10.0.0.120
                name: default
                protocol: layer2
            ...
        - name: stacklight
```

©2021, Mirantis Inc.
spec:providerSpec configuration
This section represents the Container Cloud components that are enabled on a cluster. It contains the following fields:

- **management**
  Configuration for the management cluster components:
  - **enabled**
    Management cluster enabled (true) or disabled (false).
  - **helmReleases**
    List of the management cluster Helm releases that will be installed on the cluster. A Helm release includes the name and values fields. The specified values will be merged with relevant Helm release values of the management cluster in the Release object.

- **regional**
  List of regional clusters components on the Container Cloud cluster for each configured provider available for a specific region:
  - **provider**
    Provider type that is baremetal.
  - **helmReleases**
    List of the regional Helm releases that will be installed on the cluster. A Helm release includes the name and values fields. The specified values will be merged with relevant regional Helm release values in the Release object.

- **release**
  Name of the Container Cloud Release object.

Configuration example:

```yaml
spec:
  ...
providerSpec:
  value:
    kaas:
      management:
        enabled: true
        helmReleases:
          - name: kaas-ui
            values:
              serviceConfig:
                server: https://10.0.0.117
      regional:
        helmReleases:
          - name: baremetal-provider
            values: {}
            provider: baremetal
          - helmReleases:
```

©2021, Mirantis Inc.
status:providerStatus common

Must not be modified using API

The common providerStatus object field of the Cluster resource contains the following fields:

- **apiVersion**
  API version of the object that is baremetal.k8s.io/v1alpha1

- **kind**
  Object type that is BaremetalClusterProviderStatus

- **loadBalancerHost**
  Load balancer IP or host name of the Container Cloud cluster

- **apiServerCertificate**
  Server certificate of Kubernetes API

- **ucpDashboard**
  URL of the Mirantis Kubernetes Engine (MKE) Dashboard

Configuration example:

```yaml
status:
  providerStatus:
    apiVersion: baremetal.k8s.io/v1alpha1
    kind: BaremetalClusterProviderStatus
    loadBalancerHost: 10.0.0.100
    apiServerCertificate: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS...
    ucpDashboard: https://10.0.0.100:6443
```

status:providerStatus for cluster readiness

Must not be modified using API

The providerStatus object field of the Cluster resource that reflects the cluster readiness contains the following fields:

- **persistentVolumesProviderProvisioned**
  Status of the persistent volumes provisioning. Prevents the Helm releases that require persistent volumes from being installed until some default StorageClass is added to the Cluster object.

- **helm**
  Details about the deployed Helm releases:

    - **ready**
Status of the deployed Helm releases. The true value indicates that all Helm releases are deployed successfully.

• releases
  List of the enabled Helm releases that run on the Container Cloud cluster:
    • releaseStatuses
      List of the deployed Helm releases. The success: true field indicates that the release is deployed successfully.
    • stacklight
      Status of the StackLight deployment. Contains URLs of all StackLight components. The success: true field indicates that StackLight is deployed successfully.

• nodes
  Details about the cluster nodes:
    • ready
      Number of nodes that completed the deployment or update.
    • requested
      Total number of nodes. If the number of ready nodes does not match the number of requested nodes, it means that a cluster is being currently deployed or updated.
    • notReadyObjects
      The list of the services, deployments, and statefulsets Kubernetes objects that are not in the Ready state yet. A service is not ready if its external address has not been provisioned yet. A deployment or statefulset is not ready if the number of ready replicas is not equal to the number of desired replicas. Both objects contain the name and namespace of the object and the number of ready and desired replicas (for controllers). If all objects are ready, the notReadyObjects list is empty.

Configuration example:

```json
status:
  providerStatus:
    persistentVolumesProviderProvisioned: true
  helm:
    ready: true
  releases:
    releaseStatuses:
      iam:
        success: true
      ...
    stacklight:
      alerta:
        url: http://10.0.0.106
      alertmanager:
        url: http://10.0.0.107
```
grafana:
  url: http://10.0.0.108
kibana:
  url: http://10.0.0.109
prometheus:
  url: http://10.0.0.110

success: true

nodes:
  ready: 3
  requested: 3

notReadyObjects:
  services:
    - name: testservice
      namespace: default
  deployments:
    - name: baremetal-provider
      namespace: kaas
      replicas: 3
      readyReplicas: 2

statefulsets: {}

status:providerStatus for Open ID Connect

Must not be modified using API

The o:idc section of the providerStatus object field in the Cluster resource reflects the Open ID Connect configuration details. It contains the required details to obtain a token for a Container Cloud cluster and consists of the following fields:

- certificate
  Base64-encoded OIDC certificate.
- clientId
  Client ID for OIDC requests.
- groupsClaim
  Name of an OIDC groups claim.
- issuerUrl
  Issuer URL to obtain the representation of the realm.
- ready
  OIDC status relevance. If true, the status corresponds to the LCMCluster OIDC configuration.

Configuration example:

```yaml
status:
  providerStatus:
    oidc:
      certificate: LS0tLS1CRUdJTiBDRVjUSUZJQ0FURS0tLS0tCk1JSUREekNDQWZ...
```
status:providerStatus for cluster releases

Must not be modified using API

The releaseRefs section of the providerStatus object field in the Cluster resource provides the current Cluster release version as well as the one available for upgrade. It contains the following fields:

- current
  Details of the currently installed Cluster release:

  - lcmType
    Type of the Cluster release (ucp).
  - name
    Name of the Cluster release resource.
  - version
    Version of the Cluster release.
  - unsupportedSinceKaaSVersion
    Indicates that a Container Cloud release newer than the current one exists and that it does not support the current Cluster release.

- available
  List of the releases available for upgrade. Contains the name and version fields.

Configuration example:

```
status:
  providerStatus:
    releaseRefs:
      available:
      - name: ucp-5-5-0-3-4-0-dev
        version: 5.5.0+3.4.0-dev
      current:
        lcmType: ucp
        name: ucp-5-4-0-3-3-0-beta1
        version: 5.4.0+3.3.0-beta1
```

Machine

This section describes the Machine resource used in Mirantis Container Cloud API for bare metal provider. The Machine resource describes the machine-level parameters.

For demonstration purposes, the Container Cloud Machine custom resource (CR) is split into the following major sections:
metadata

The Container Cloud Machine CR contains the following fields:

- **apiVersion**
  API version of the object that is cluster.k8s.io/v1alpha1.

- **kind**
  Object type that is Machine.

The metadata object field of the Machine resource contains the following fields:

- **name**
  Name of the Machine object.

- **namespace**
  Project in which the Machine object is created.

- **annotations**
  Key-value pair to attach arbitrary metadata to the object:
  - **metal3.io/BareMetalHost**
    Annotation attached to the Machine object to reference the corresponding BareMetalHost object in the <BareMetalHostProjectName/BareMetalHostName> format.

- **labels**
  Key-value pairs that are attached to the object:
  - **kaas.mirantis.com/provider**
    Provider type that matches the provider type in the Cluster object and must be baremetal.
  - **kaas.mirantis.com/region**
    Region name that matches the region name in the Cluster object.
  - **cluster.sigs.k8s.io/cluster-name**
    Cluster name that the Machine object is linked to.
  - **cluster.sigs.k8s.io/control-plane**
    For the control plane role of a machine, this label contains any value, for example, "true". For the worker role, this label is absent or does not contain any value.

Configuration example:

```yaml
apiVersion: cluster.k8s.io/v1alpha1
kind: Machine
metadata:
```
spec:providerSpec for instance configuration

The spec object field of the Machine object represents the BareMetalMachineProviderSpec subresource with all required details to create a bare metal instance. It contains the following fields:

- **apiVersion**
  API version of the object that is baremetal.k8s.io/v1alpha1.

- **kind**
  Object type that is BareMetalMachineProviderSpec.

- **bareMetalHostProfile**
  Configuration profile of a bare metal host:
  - **name**
    Name of a bare metal host profile
  - **namespace**
    Project in which the bare metal host profile is created.

- **l2TemplateIfMappingOverride**
  If specified, overrides the interface mapping value for the corresponding L2Template object.

- **l2TemplateSelector**
  If specified, contains the name (first priority) or label of the L2 template that will be applied during a machine creation. The l2TemplateSelector field is copied from the Machine providerSpec object to the IpamHost object only once, during a machine creation. To modify l2TemplateSelector after creation of a Machine CR, edit the IpamHost object.

- **hostSelector**
  Specifies the matching criteria for labels on the bare metal hosts. Limits the set of the BareMetalHost objects considered for claiming for the Machine object. The following selector labels can be added when creating a machine using the Container Cloud web UI:
  - **hostlabel.bm.kaas.mirantis.com/controlplane**
  - **hostlabel.bm.kaas.mirantis.com/worker**
Any custom label that is assigned to one or more bare metal hosts using API can be used as a host selector. If the BareMetalHost objects with the specified label are missing, the Machine object will not be deployed until at least one bare metal host with the specified label is available.

• **nodeLabels**

  List of node labels to be attached to the corresponding node. Enables running of certain components on separate cluster nodes. The list of allowed node labels is defined in the providerStatus.releaseRef.current.allowedNodeLabels cluster status. Addition of any unsupported node label not from this list is restricted.

Configuration example:

```yaml
spec:
  ...  
  providerSpec:
    value:
      apiVersion: baremetal.k8s.io/v1alpha1
      kind: BareMetalMachineProviderSpec
      bareMetalHostProfile:
        name: default
        namespace: default
      l2TemplateIfMappingOverride:
        - eno1
        - enp0s0
      l2TemplateSelector:
        label: l2-template1-label-1
      hostSelector:
        matchLabels:
          baremetal: hw-master-0
          kind: BareMetalMachineProviderSpec
      nodeLabels:
        key: stacklight
        value: enabled
```

**Machine status**

The status object field of the Machine object represents the BareMetalMachineProviderStatus subresource that describes the current bare metal instance state and contains the following fields:

• **apiVersion**
  
  API version of the object that is cluster.k8s.io/v1alpha1.

• **kind**
  
  Object type that is BareMetalMachineProviderStatus.

• **hardware**
Provides a machine hardware information:

- **cpu**
  - Number of CPUs.

- **ram**
  - RAM capacity in GB.

- **storage**
  - List of hard drives mounted on the machine. Contains the disk name and size in GB.

- **status**
  - Represents the current status of a machine:
    - **Provision**
      - Machine is yet to obtain a status.
    - **Uninitialized**
      - Machine is yet to obtain a node IP address and hostname.
    - **Pending**
      - Machine is yet to receive the deployment instructions. It is either not booted yet or waits for the LCM controller to be deployed.
    - **Prepare**
      - Machine is running the Prepare phase when mostly Docker images and packages are being predownloaded.
    - **Deploy**
      - Machine is processing the LCM controller instructions.
    - **Reconfigure**
      - Some configurations are being updated on a machine.
    - **Ready**
      - Machine is deployed and the supported Mirantis Kubernetes Engine (MKE) version is set.

Configuration example:

```
status:
  providerStatus:
    apiVersion: baremetal.k8s.io/v1alpha1
    kind: BareMetalMachineProviderStatus
  hardware:
    cpu: 11
    ram: 16
  storage:
    - name: /dev/vda
      size: 61
    - name: /dev/vdb
```
BareMetalHostProfile

This section describes the BareMetalHostProfile resource used in Mirantis Container Cloud API to define how the storage devices and operating system are provisioned and configured.

For demonstration purposes, the Container Cloud BareMetalHostProfile custom resource (CR) is split into the following major sections:

- metadata
- spec

metadata

The Container Cloud BareMetalHostProfile CR contains the following fields:

- apiVersion
  API version of the object that is metal3.io/v1alpha1.
- kind
  Object type that is BareMetalHostProfile.
- metadata
  The metadata field contains the following subfields:
  - name
    Name of the bare metal host profile.
  - namespace
    Project in which the bare metal host profile was created.

Configuration example:

```yaml
apiVersion: metal3.io/v1alpha1
kind: BareMetalHostProfile
metadata:
  name: default
  namespace: default
```

spec

The spec field of BareMetalHostProfile object contains the fields to customize your hardware configuration:

- devices
  List of definitions of the physical storage devices. To configure more than three storage devices per host, add additional devices to this list. Each device in the list may have one or more partitions defined by the list in the partitions field.
• **fileSystems**
  List of file systems. Each file system can be created on top of either device, partition, or logical volume. If more file systems are required for additional devices, define them in this field.

• **logicalVolumes**
  List of LVM logical volumes. Every logical volume belongs to a volume group from the volumeGroups list and has the `sizeGiB` attribute for size in gigabytes.

• **volumeGroups**
  List of definitions of LVM volume groups. Each volume group contains one or more devices or partitions from the devices list.

• **preDeployScript**
  Shell script that is executed on a host before provisioning the target operating system inside the ramfs system.

• **postDeployScript**
  Shell script that is executed on a host after deploying the operating system inside the ramfs system that is chrooted to the target operating system.

• **grubConfig**
  List of options passed to the Linux GRUB bootloader. Each string in the list defines one parameter.

• **kernelParameters:sysctl** *Available since 2.2.0*
  List of options passed to `/etc/sysctl.d/999-baremetal.conf` during bmh provisioning.

Configuration example:

```yaml
spec:
  devices:
    - device:
        wipe: true
      partitions:
        - dev: ""
          name: bios_grub
          partflags:
            - bios_grub
          sizeGiB: 0.00390625
        ...
    - device:
        wipe: true
      partitions:
        - dev: ""
          name: lvm_lvp_part
      fileSystems:
        - fileSystem: vfat
          partition: config-2
        - fileSystem: vfat
          mountPoint: /boot/efi
```
BareMetalHost

This section describes the BareMetalHost resource used in the Mirantis Container Cloud API. BareMetalHost object is being created for each Machine and contains all information about machine hardware configuration. It is needed for further selecting which machine to choose for the deploy. When machine is created the provider assigns a BareMetalHost to that machine based on labels and BareMetalHostProfile configuration.
For demonstration purposes, the Container Cloud BareMetalHost custom resource (CR) can be split into the following major sections:

- BareMetalHost metadata
- BareMetalHost configuration
- BareMetalHost status

BareMetalHost metadata

The Container Cloud BareMetalHost CR contains the following fields:

- apiVersion
  API version of the object that is metal3.io/v1alpha1.
- kind
  Object type that is BareMetalHost.
- metadata
  The metadata field contains the following subfields:
  - name
    Name of the BareMetalHost object.
  - namespace
    Project in which the BareMetalHost object was created.
  - labels
    Labels used by the bare metal provider to find a matching BareMetalHost object to deploy a machine:
    - hostlabel.bm.kaas.mirantis.com/controlplane
    - hostlabel.bm.kaas.mirantis.com/worker
    - hostlabel.bm.kaas.mirantis.com/storage

Each BareMetalHost object added using the Container Cloud web UI will be assigned one of these labels. If the BareMetalHost and Machine objects are created using API, any label may be used to match these objects for a bare metal host to deploy a machine.

Configuration example:

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: master-0
  namespace: default
  labels:
    baremetal: hw-master-0
```
BareMetalHost configuration

The spec section for the BareMetalHost object defines the desired state of BareMetalHost. It contains the following fields:

- **bmc**
  
  Details for communication with the Baseboard Management Controller (bmc) module on a host:
  
  - **address**
    
    URL for accessing bmc in the network.
  
  - **credentialsName**
    
    Name of the secret containing the bmc credentials. The secret requires the username and password keys in the Base64 encoding.
  
  - **bootMACAddress**
    
    MAC address for booting.
  
  - **bootUEFI**
    
    UEFI boot mode enabled (true) or disabled (false).
  
  - **online**
    
    Defines whether the server must be online after inspection.

Configuration example:

```json
spec:
  bmc:
    address: 5.43.227.106:623
    credentialsName: master-0-bmc-secret
    bootMACAddress: 0c:c4:7a:a8:d3:44
    bootUEFI: true
  consumerRef:
    apiVersion: cluster.k8s.io/v1alpha1
    kind: Machine
    name: master-0
    namespace: default
  online: true
```

BareMetalHost status

The status field of the BareMetalHost object defines the current state of BareMetalHost. It contains the following fields:

- **errorMessage**
  
  Last error message reported by the provisioning subsystem.

- **goodCredentials**
  
  Last credentials that were validated.

- **hardware**
Hardware discovered on the host. Contains information about the storage, CPU, host name, firmware, and so on.

• operationalStatus
  Status of the host:
  • OK
    Host is configured correctly and is manageable.
  • discovered
    Host is only partially configured. For example, the bmc address is discovered but not the login credentials.
  • error
    Host has any sort of error.
• poweredOn
  Host availability status: powered on (true) or powered off (false).
• provisioning
  State information tracked by the provisioner:
  • state
    Current action being done with the host by the provisioner.
  • id
    UUID of a machine.
• triedCredentials
  Details of the last credentials sent to the provisioning back end.

Configuration example:

```json
status:
  errorMessage: ""
goodCredentials:
  credentials:
    name: master-0-bmc-secret
    namespace: default
    credentialsVersion: "13404"
hardware:
  cpu:
    arch: x86_64
    clockMegahertz: 3000
    count: 32
  flags:
    - 3dnowprefetch
    - abm
    ...
  model: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
firmware:
```
IpamHost

This section describes the IpamHost resource used in Mirantis Container Cloud API. The kaas-ipam controller monitors the current state of the bare metal Machine, verifies if BareMetalHost is successfully created and inspection is completed. Then the kaas-ipam controller fetches the information about the network card, creates the IpamHost object, and requests the IP address.
The IpamHost object is created for each Machine and contains all configuration of the host network interfaces and IP address. It also contains the information about associated BareMetalHost, Machine, and MAC addresses.

For demonstration purposes, the Container Cloud IpamHost custom resource (CR) is split into the following major sections:

• IpamHost metadata
• IpamHost configuration
• IpamHost status

IpamHost metadata

The Container Cloud IpamHost CR contains the following fields:

• apiVersion
  API version of the object that is ipam.mirantis.com/v1alpha1
• kind
  Object type that is IpamHost
• metadata
  The metadata field contains the following subfields:
  • name
    Name of the IpamHost object
  • namespace
    Project in which the IpamHost object has been created
  • labels
    Key-value pairs that are attached to the object:
    • cluster.sigs.k8s.io/cluster-name
      References the Cluster object name that IpamHost is assigned to
    • ipam/BMHostID
      Unique ID of the associated BareMetalHost object
    • ipam/MAC-XX-XX-XX-XX-XX-XX: "1"
      Number of NICs of the host that the corresponding MAC address is assigned to
    • ipam/MachineID
      Unique ID of the associated Machine object
    • ipam/UID
      Unique ID of the IpamHost object

Configuration example:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: IpamHost
```
The spec field of the IpamHost resource describes the desired state of the object. It contains the following fields:

- **nicMACmap**
  
  Represents an unordered list of all NICs of the host. Each NIC entry contains such fields as name, mac, ip, and so on. The primary field defines that the current NIC is primary. Only one NIC can be primary.

- **l2TemplateSelector**
  
  If specified, contains the name (first priority) or label of the L2 template that will be applied during a machine creation. The l2TemplateSelector field is copied from the Machine providerSpec object to the IpamHost object only once, during a machine creation. To modify l2TemplateSelector after creation of a Machine CR, edit the IpamHost object.

Configuration example:

```yaml
spec:
  nicMACmap:
    - mac: 0c:c4:7a:1e:a9:5c
      name: ens11f0
      ip: 172.16.48.157
      mac: 0c:c4:7a:1e:a9:5d
      name: ens11f1
      primary: true
  l2TemplateSelector:
    label: xxx
```

IpamHost status

The status field of the IpamHost resource describes the observed state of the object. It contains the following fields:

- **ipAllocationResult**
  
  Status of IP allocation for the primary NIC (PXE boot). Possible values are OK or ERR if no IP address was allocated.
- **l2RenderResult**
  Result of the L2 template rendering, if applicable. Possible values are OK or an error message.

- **lastUpdated**
  Date and time of the last IpamHost status update.

- **nicMACmap**
  Unordered list of all NICs of host with a detailed description. Each nicMACmap entry contains additional fields such as ipRef, nameservers, online, and so on.

- **osMetadataNetwork**
  Configuration of the host OS metadata network. This configuration is used in the cloud-init tool and is applicable to the primary NIC only. It is added when the IP address is allocated and the ipAllocationResult status is OK.

- **versionIpam**
  IPAM version used during the last update of the object.

Configuration example:

```json
status:
ipAllocationResult: OK
l2RenderResult: There are no available L2Templates
lastUpdated: "2020-09-16T11:02:39Z"
nicMACmap:
- mac: 0C:C4:7A:1E:A9:5C
  name: ens11f0
- gateway: 172.16.48.1
  ip: 172.16.48.200/24
  ipRef: default/auto-0c-c4-7a-a8-d3-44
  mac: 0C:C4:7A:1E:A9:5D
  name: ens11f1
  nameservers:
  - 172.18.176.6
  online: true
  primary: true
osMetadataNetwork:
  links:
  - ethernet_mac_address: 0C:C4:7A:A8:D3:44
    id: enp8s0f0
    type: phy
  networks:
  - ip_address: 172.16.48.200
    link: enp8s0f0
    netmask: 255.255.255.0
  routes:
  - gateway: 172.16.48.1
    netmask: 0.0.0.0
    network: 0.0.0.0
```
Subnet metadata
The Container Cloud Subnet CR contains the following fields:

- apiVersion
  API version of the object that is ipam.mirantis.com/v1alpha1.
- kind
  Object type that is Subnet
- metadata
This field contains the following subfields:

- **name**
  Name of the Subnet object.

- **namespace**
  Project in which the Subnet object was created.

- **labels**
  Key-value pairs that are attached to the object:
  - `ipam/DefaultSubnet: "1"`
    Indicates that the subnet was automatically created for the PXE network. The subnet with this label is unique for a specific region and global for all clusters and projects in the region.
  - `ipam/UID`
    Unique ID of a subnet.
  - `kaas.mirantis.com/provider`
    Provider type.
  - `kaas.mirantis.com/region`
    Region type.

Configuration example:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: Subnet
metadata:
  name: kaas-mgmt
  namespace: default
labels:
  ipam/DefaultSubnet: "1"
  ipam/UID: 1bae269c-c507-4404-b534-2c135edaebf5
  kaas.mirantis.com/provider: baremetal
  kaas.mirantis.com/region: region-one
```

Subnet spec

The spec field of the Subnet resource describes the desired state of a subnet. It contains the following fields:

- **cidr**
  A valid IPv4 CIDR, for example, 10.11.0.0/24.

- **gateway**
  A valid gateway address, for example, 10.11.0.9.

- **includeRanges**
A list of IP address ranges within the given CIDR that should be used in the allocation of IPs for nodes. The gateway, network, broadcast, and DNS addresses will be excluded (protected) automatically if they intersect with one of the range. The IPs outside the given ranges will not be used in the allocation. Each element of the list can be either an interval 10.11.0.5-10.11.0.70 or a single address 10.11.0.77. The includeRanges parameter is mutually exclusive with excludeRanges.

- **excludeRanges**
  A list of IP address ranges within the given CIDR that should not be used in the allocation of IPs for nodes. The IPs within the given CIDR but outside the given ranges will be used in the allocation. The gateway, network, broadcast, and DNS addresses will be excluded (protected) automatically if they are included in the CIDR. Each element of the list can be either an interval 10.11.0.5-10.11.0.70 or a single address 10.11.0.77. The excludeRanges parameter is mutually exclusive with includeRanges.

- **useWholeCidr**
  If set to false (by default), the subnet address and broadcast address will be excluded from the address allocation. If set to true, the subnet address and the broadcast address are included into the address allocation for nodes.

- **nameservers**
  The list of IP addresses of name servers. Each element of the list is a single address, for example, 172.18.176.6.

Configuration example:

```
spec:
cidr: 172.16.48.0/24
excludeRanges:
- 172.16.48.99
- 172.16.48.101-172.16.48.145

gateway: 172.16.48.1
nameservers:
- 172.18.176.6
```

Subnet status

The status field of the Subnet resource describes the actual state of a subnet. It contains the following fields:

- **allocatable**
  The number of IP addresses that are available for allocation.

- **allocatedIPs**
  The list of allocated IP addresses in the IP:<IPAddr object UID> format.

- **capacity**
  The total number of IP addresses to be allocated, including the sum of allocatable and already allocated IP addresses.

- **cidr**
The IPv4 CIDR for a subnet.

- **gateway**
  The gateway address for a subnet.

- **nameservers**
  The list of IP addresses of name servers.

- **ranges**
  The list of IP address ranges within the given CIDR that are used in the allocation of IPs for nodes.

Configuration example:

```
status:
  allocatable: 51
  allocatedIPs:
  - 172.16.48.200:24e94698-f726-11ea-a717-0242c0a85b02
  - 172.16.48.201:2bb62373-f726-11ea-a717-0242c0a85b02
  - 172.16.48.202:37806659-f726-11ea-a717-0242c0a85b02
  capacity: 54
  cidr: 172.16.48.0/24
  gateway: 172.16.48.1
  lastUpdate: "2020-09-15T12:27:58Z"
  nameservers:
  - 172.18.176.6
  ranges:
  - 172.16.48.200-172.16.48.253
  statusMessage: OK
```

**SubnetPool**

This section describes the SubnetPool resource used in Mirantis Container Cloud API to manage a pool of addresses from which subnets can be allocated.

For demonstration purposes, the Container Cloud SubnetPool custom resource (CR) is split into the following major sections:

- SubnetPool metadata
- SubnetPool spec
- SubnetPool status

**SubnetPool metadata**

The Container Cloud SubnetPool CR contains the following fields:

- **apiVersion**
  API version of the object that is ipam.mirantis.com/v1alpha1.

- **kind**
  Object type that is SubnetPool.
• metadata
  The metadata field contains the following subfields:
  • name
    Name of the SubnetPool object.
  • namespace
    Project in which the SubnetPool object was created.
  • labels
    Key-value pairs that are attached to the object:
    • kaas.mirantis.com/provider
      Provider type that is baremetal.
    • kaas.mirantis.com/region
      Region name.

Configuration example:

```yaml
apiVersion: ipam.mirantis.com/v1alpha1
kind: SubnetPool
metadata:
  name: kaas-mgmt
  namespace: default
  labels:
    kaas.mirantis.com/provider: baremetal
    kaas.mirantis.com/region: region-one
```

SubnetPool spec
The spec field of the SubnetPool resource describes the desired state of a subnet pool. It contains the following fields:

• cidr
  Valid IPv4 CIDR. For example, 10.10.0.0/16.

• blockSize
  IP address block size to use when assigning an IP address block to every new child Subnet object. For example, if you set /25, every new child Subnet will have 128 IPs to allocate. Possible values are from /29 to the cidr size. Immutable.

• nameservers
  Optional. List of IP addresses of name servers to use for every new child Subnet object. Each element of the list is a single address, for example, 172.18.176.6. Default: empty.

• gatewayPolicy
  Optional. Method of assigning a gateway address to new child Subnet objects. Default: none. Possible values are:
  • first - first IP of the IP address block assigned to a child Subnet, for example, 10.11.10.1.
- last - last IP of the IP address block assigned to a child Subnet, for example, 10.11.10.254.
- none - no gateway address.

Configuration example:

```
spec:
  cidr: 10.10.0.0/16
  blockSize: /25
  nameservers:
    - 172.18.176.6
  gatewayPolicy: first
```

SubnetPool status

The status field of the SubnetPool resource describes the actual state of a subnet pool. It contains the following fields:

- statusMessage
  Message that reflects the current status of the SubnetPool resource. Possible values are:
  - OK - a subnet pool is active.
  - ERR: <error message> - a subnet pool is in the Failure state.
  - TERM - a subnet pool is terminating.
- allocatedSubnets
  List of allocated subnets. Each subnet has the <CIDR>:<SUBNET_UID> format.
- blockSize
  Block size to use for IP address assignments from the defined pool.
- capacity
  Total number of IP addresses to be allocated. Includes the number of allocatable and already allocated IP addresses.
- allocatable
  Number of subnets with the blockSize size that are available for allocation.
- lastUpdate
  Date and time of the last SubnetPool status update.
- versionIpam
  IPAM version used during the last object update.

Example:

```
status:
  allocatedSubnets:
    - 10.10.0.0/24:0272bfa9-19de-11eb-b591-0242ac110002
```
This section describes the IPaddr resource used in Mirantis Container Cloud API. The IPAddr object describes an IP address and contains all information about the associated MAC address.

For demonstration purposes, the Container Cloud IPaddr custom resource (CR) is split into the following major sections:

- IPaddr metadata
- IPAddr spec
- IPAddr status

IPaddr metadata

The Container Cloud IPaddr CR contains the following fields:

- apiVersion
  API version of the object that is ipam.mirantis.com/v1alpha1
- kind
  Object type that is IPaddr
- metadata
  The metadata field contains the following subfields:
  - name
    Name of the IPaddr object in the auto-XX-XX-XX-XX-XX-XX format where XX-XX-XX-XX-XX-XX is the associated MAC address
  - namespace
    Project in which the IPaddr object was created
  - labels
    Key-value pairs that are attached to the object:
    - ipam/IP
      IPv4 address
    - ipam/IpmHostID
      Unique ID of the associated IpamHost object
    - ipam/MAC
      MAC address
• ipam/SubnetID
  Unique ID of the Subnet object
• ipam/UID
  Unique ID of the IPAddr object

Configuration example:

```
apiVersion: ipam.mirantis.com/v1alpha1
type: IPaddr
metadata:
  name: auto-0c-c4-7a-a8-b8-18
  namespace: default
  labels:
    ipam/IP: 172.16.48.201
    ipam/IpamHostID: 848b59cf-f804-11ea-88c8-0242c0a85b02
    ipam/MAC: 0C-C4-7A-A8-B8-18
    ipam/SubnetID: 572b38de-f803-11ea-88c8-0242c0a85b02
    ipam/UID: 84925cac-f804-11ea-88c8-0242c0a85b02
```

IPAddr spec

The spec object field of the IPAddr resource contains the associated MAC address and the reference to the Subnet object:

- mac
  MAC address in the XX:XX:XX:XX:XX:XX format
- subnetRef
  Reference to the Subnet resource in the <subnetProjectName>/</subnetName> format

Configuration example:

```
spec:
  mac: 0C:C4:7A:A8:B8:18
  subnetRef: default/kaas-mgmt
```

IPAddr status

The status object field of the IPAddr resource reflects the actual state of the IPAddr object. It contains the following fields:

- address
  IP address.
- cidr
  IPv4 CIDR for the Subnet.
- gateway
  Gateway address for the Subnet.
• lastUpdate
  Date and time of the last IPAddr status update.

• mac
  MAC address in the XX:XX:XX:XX:XX:XX format.

• nameservers
  List of the IP addresses of name servers of the Subnet. Each element of the list is a single address, for example, 172.18.176.6.

• phase
  Current phase of the IP address. Possible values: Active, Failed, or Terminating.

• versionIpam
  IPAM version used during the last update of the object.

Configuration example:

```
status:
  address: 172.16.48.201
  cidr: 172.16.48.201/24
  gateway: 172.16.48.1
  lastUpdate: "2020-09-16T10:08:07Z"
  mac: 0C:C4:7A:A8:B8:18
  nameservers:
    - 172.18.176.6
  phase: Active
  versionIpam: v3.0.999-20200807-130909-44151f8
```

L2Template

This section describes the L2Template resource used in Mirantis Container Cloud API.

By default, Container Cloud configures a single interface on cluster nodes, leaving all other physical interfaces intact. With L2Template, you can create advanced host networking configurations for your clusters. For example, you can create bond interfaces on top of physical interfaces on the host.

For demonstration purposes, the Container Cloud L2Template custom resource (CR) is split into the following major sections:

• L2Template metadata
• L2Template configuration
• L2Template status

L2Template metadata

The Container Cloud L2Template CR contains the following fields:

• apiVersion
  API version of the object that is ipam.mirantis.com/v1alpha1.
• kind
  Object type that is L2Template.

• metadata
  The metadata field contains the following subfields:
  
  • name
    Name of the L2Template object.
  
  • namespace
    Project in which the L2Template object was created.

  • labels
    Key-value pairs that are attached to the object:
    
    • ipam/Cluster
      References the Cluster object name that this template is applied to. The process of selecting the L2Template object for a specific cluster is as follows:
      
      1. The kaas-ipam controller monitors the L2Template objects with the ipam/Cluster:<clusterName> label.
      2. The L2Template object with the ipam/Cluster: <clusterName> label is assigned to a cluster with Name: <clusterName>, if available. Otherwise, the default L2Template object with the ipam/Cluster: default label is assigned to a cluster.
    
    • ipam/PreInstalledL2Template: "1"
      Indicates that the current L2Template object was preinstalled. Represents L2 templates that are automatically copied to a project once it is created. Once the L2 templates are copied, the ipam/PreInstalledL2Template label is removed. This label is set automatically and cannot be configured manually.

    • ipam/DefaultForCluster
      This label is unique per cluster. When you use several L2 templates per cluster, only the first template is automatically labeled as the default one. All consequent templates must be referenced in the machines configuration files using L2templateSelector.

    • ipam/UID
      Unique ID of an object.

    • kaas.mirantis.com/provider
      Provider type.

    • kaas.mirantis.com/region
      Region type.

Configuration example:
apiVersion: ipam.mirantis.com/v1alpha1
kind: L2Template
metadata:
  name: l2template-test
  namespace: default
labels:
  ipam/Cluster: test
  ipam/DefaultForCluster: "1"
  kaas.mirantis.com/provider: baremetal
  kaas.mirantis.com/region: region-one

L2Template configuration

The spec field of the L2Template resource describes the desired state of the object. It contains the following fields:

- **clusterRef**
  The Cluster object that this template is applied to. The default value is used to apply the given template to all clusters unless an L2 template that references a specific cluster name exists.

  **Caution!**
  - A cluster can be associated with only one template.
  - An L2 template must have the same namespace as the referenced cluster.
  - A project can have only one default L2 template.

- **ifMapping**
  The list of interface names for the template. The interface mapping is defined globally for all bare metal hosts in the cluster but can be overridden at the host level, if required, by editing the IpamHost object for a particular host. The ifMapping parameter is mutually exclusive with autoIfMappingPrio.

- **autoIfMappingPrio**
  The list of prefixes, such as eno, ens, and so on, to match the interfaces to automatically create a list for the template. The result of generation may be overridden at the host level using ifMappingOverride in the corresponded IpamHost spec. The autoIfMappingPrio parameter is mutually exclusive with ifMapping.

- **npTemplate**
  A netplan-compatible configuration with special lookup functions that defines the networking settings for the cluster hosts, where physical NIC names and details are parameterized. This configuration will be processed using Go templates. Instead of specifying IP and MAC addresses, interface names, and other network details specific to a particular host, the template supports use of special lookup functions. These
lookup functions, such as nic, mac, ip, and so on, return host-specific network information when the template is rendered for a particular host.

Caution!

All rules and restrictions of the netplan configuration also apply to L2 templates. For details, see the official netplan documentation.

Configuration example:

```yaml
spec:
  autoIfMappingPrio:
    - provision
    - eno
    - ens
    - enp
  l3Layout: null
  npTemplate: |
    version: 2
    ethernets:
      {{nic 0}}:
        dhcp4: false
        dhcp6: false
        addresses:
          - {{ip "0:kaas-mgmt"}}
        gateway4: {{gateway_from_subnet "kaas-mgmt"}}
        nameservers:
          addresses: {{nameservers_from_subnet "kaas-mgmt"}}
        match:
          macaddress: {{mac 0}}
          set-name: {{nic 0}}
```

L2Template status

The status field of the L2Template resource reflects the actual state of the L2Template object and contains the following fields:

- phase
  Current phase of the L2Template object. Possible values: Ready, Failed, or Terminating.
- reason
  Detailed error message in case L2Template has the Failed status.
- lastUpdate
  Date and time of the last L2Template status update.
- versionIpam

©2021, Mirantis Inc.
IPAM version used during the last update of the object.

Configuration example:

```yaml
status:
  lastUpdate: "2020-09-15T08:08Z"
  phase: Failed
  reason: The kaas-mgmt subnet in the terminating state.
  versionIpam: v3.0.999-20200807-130909-44151f8
```