KaaS Deployment Guide

version beta
Copyright notice

2020 Mirantis, Inc. All rights reserved.

This product is protected by U.S. and international copyright and intellectual property laws. No part of this publication may be reproduced in any written, electronic, recording, or photocopying form without written permission of Mirantis, Inc.

Mirantis, Inc. reserves the right to modify the content of this document at any time without prior notice. Functionality described in the document may not be available at the moment. The document contains the latest information at the time of publication.

Mirantis, Inc. and the Mirantis Logo are trademarks of Mirantis, Inc. and/or its affiliates in the United States and other countries. Third party trademarks, service marks, and names mentioned in this document are the properties of their respective owners.
Preface
This documentation provides information on how to use Mirantis products to deploy cloud environments. The information is for reference purposes and is subject to change.

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 KaaS 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 Mirantis KaaS and Kubernetes through a local machine or web UI
  • Consumes artifacts from a central repository approved by the Infrastructure Operator

Documentation history
The following table lists the released revisions of this documentation:

<table>
<thead>
<tr>
<th>Revision date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 19, 2019</td>
<td>Mirantis KaaS Beta</td>
</tr>
</tbody>
</table>
Introduction

Mirantis KaaS enables you to create, scale, and upgrade Kubernetes clusters on demand through a declarative API with a centralized identity and access management.

Mirantis KaaS is installed once to deploy the KaaS management cluster. The KaaS management cluster is deployed through the bootstrap procedure on either the OpenStack or bare metal provider. StackLight can be installed on both types of the KaaS clusters, management and child, to provide metrics for each cluster separately. The baremetal-based deployment includes Ceph as a distributed storage system.
Deploy a baremetal-based management cluster

Workflow overview

The bare metal management system enables the Infrastructure Operator to deploy Mirantis KaaS on a set of bare metal servers. It also enables Mirantis KaaS to deploy child clusters on bare metal servers without a pre-provisioned operating system.

The Infrastructure Operator performs the following steps to install KaaS in a bare metal environment:

1. Install and connect hardware servers as described in KaaS Reference Architecture: Baremetal-based KaaS cluster.

   Caution!

   The baremetal-based Mirantis KaaS does not manage the underlay networking fabric but requires specific network configuration to operate.

2. Install Ubuntu 18.04 operating system on one of the bare metal machines to create a seed node.

3. Copy the bootstrap tarball to the seed node.

4. Obtain your license file that will be required during the bootstrap.

5. Create the configuration files for the bootstrap script that contain all necessary configuration data to deploy the KaaS management cluster including the bare metal hosts metadata.

   Available since KaaS release 1.6.0 Validate the deployment templates using the preflight command before the KaaS management cluster bootstrap to ensure that the template parameters for the Baseboard Management Controller (BMC) access credentials and PXE service are configured correctly for the bootstrap to succeed.

6. Run the bootstrap script for unattended installation of the KaaS management cluster onto the selected bare metal hosts.

Using the bootstrap script, the KaaS bare metal management system prepares the seed node for the KaaS management cluster and starts the deployment of KaaS itself. The bootstrap script performs all necessary operations to perform the automated KaaS management cluster setup. The deployment diagram below illustrates the bootstrap workflow of a KaaS management cluster.
Bootstrap a management cluster

This section describes how to prepare and bootstrap a baremetal-based Mirantis KaaS management cluster. The procedure includes:

- A runbook that describes how to create a seed node that is a temporary server used to run the KaaS management cluster bootstrap scripts.
- A step-by-step instruction how to prepare metadata for the bootstrap scripts and how to run them.

Prepare the seed node

Before installing KaaS in a bare metal environment, complete the following preparation steps:

1. Verify that the hardware allocated for the installation meets the minimal requirements described in KaaS Reference Architecture: Requirements for a baremetal-based Kubernetes cluster.
2. Install basic Ubuntu 18.04 server using standard installation images of the operating system on the bare metal seed node.
3. Log in to the seed node that is running Ubuntu 18.04.
4. Create a virtual bridge to connect to your PXE network on the seed node. Use the following netplan-based configuration file as an example:

```
# cat /etc/netplan/config.yaml

---
network:
  version: 2
  renderer: networkd
ethernets:
  ens3:
    dhcp4: false
dhcp6: false
bridges:
  br0:
    addresses:
    - 10.0.0.15/24  # Please, adjust for your environment
dhcp4: false
dhcp6: false
gateway4: 10.0.0.1  # Please, adjust for your environment
interfaces:
  - ens3  # Interface name may be different in your environment
nameservers:
  addresses:
  - 172.18.208.44  # Please, adjust for your environment
parameters:
  forward-delay: 4
  stp: false
```

5. Apply the new network configuration using netplan:

```
sudo netplan apply
```

6. Verify the new network configuration:

```
sudo brctl show
```

Example of system response:

```
bridge name   bridge id          STP enabled interfaces
br0           8000.fa163e72f146  no      ens3
```

Verify that the interface connected to the PXE network belongs to the previously configured bridge.

7. Install Docker version 18.09:

```
sudo apt install docker.io
```
8. Verify that your logged USER has access to the Docker daemon:

```
sudo usermod -aG docker $USER
```

9. Log out and log in again to the seed node to apply the changes.

10. Verify that Docker is configured correctly and has access to Mirantis KaaS CDN. For example:

```
docker run --rm alpine sh -c "apk add --no-cache curl; \
curl https://binary.mirantis.com"
```

The system output must contain a json file with no error messages. In case of errors, follow the steps provided in Troubleshooting.

Proceed with Verify the seed node.

### Verify the seed node

Before you proceed to bootstrapping the KaaS management cluster on bare metal, perform the following steps:

1. Verify that the seed node has direct access to the Baseboard Management Controller (BMC) of each bare metal host. For example, using the IPMI tool:

```
ipmitool -I lanplus -H 'IPMI IP' -U 'IPMI Login' -P 'IPMI password'
```

Example of system response:

```
Chassis Power is off
```

2. Verify that you configured each bare metal host as follows:

   - Enable the boot NIC support for UEFI load. Usually, at least the built-in network interfaces support it.
   - Enable the UEFI-LAN-OPROM support in BIOS -> Advanced -> PCIPCIe.
   - Enable the IPv4-PXE stack.
   - Set the UEFI-DISK => UEFI-PXE boot order.
   - If your PXE network is not configured to use the first network interface, fix the UEFI-PXE boot order to make nodes discovering faster by selecting only one required network interface.
   - Power off all bare metal hosts.

Proceed with Prepare bare metal prerequisites data.

### Prepare bare metal prerequisites data
Prepare a table with the summary of addresses and credentials for your KaaS management cluster using the example table below.

### Bare metal prerequisites data

<table>
<thead>
<tr>
<th>Entity</th>
<th>Data</th>
<th>Parameter</th>
<th>Example value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PXE network</td>
<td>The PXE network CIDR. It is not used directly in the bootstrap script parameters but it is required that all IP addresses below belong to this network, unless stated otherwise.</td>
<td>n/a</td>
<td>10.0.0.0/26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PXE_NW_GW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The default gateway in the PXE network. Since this is the only network KaaS will use, this gateway must provide access to:</td>
<td>SET_PXE_NW_GW</td>
<td>10.0.0.1</td>
</tr>
<tr>
<td></td>
<td>• The Internet to download the Mirantis artifacts</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The OOB network of the KaaS cluster</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KAAS_BM_PXE_MASK</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>The CIDR prefix for the PXE network. It will be used with all of the addresses below when assigning them to interfaces.</td>
<td>KAAS_BM_PXE_IP</td>
<td>10.0.0.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PXE_NW_DNS</td>
<td>172.18.176.6</td>
</tr>
<tr>
<td></td>
<td>The provisioning IP address. This address will be assigned to the interface of the seed node defined by the KAAS_BM_PXE_BRIDGE parameter (see below). The PXE service of the bootstrap cluster will use this address to network boot the bare metal hosts for the KaaS Management cluster.</td>
<td>SET_PXE_NW_DNS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>An external (non-Kubernetes) DNS server available from the PXE network. This server will be used by the bare metal hosts in all KaaS clusters.</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The pool of IP addresses to be assigned to bare metal hosts.</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td>10.0.050-10.0 .0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The CIDsr definition for the SET_IPAM_POOL_RANGE pool.</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td>10.0.0.26</td>
</tr>
<tr>
<td></td>
<td>The PXE-DHCP range parameters:</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• start_ip and end_ip must be within the PXE network</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• netmask is the mask of the PXE network in the x.x.x.x format</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• broadcast is the actual broadcast address of the PXE network</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• lease time</td>
<td>SET_DHCP_RAANGE_PARAMS</td>
<td></td>
</tr>
<tr>
<td>Seed node</td>
<td>The default IP address of the seed node in the PXE network. It is not used in the bootstrap process but it must NOT duplicate any other address from the PXE network assigned to other nodes and services.</td>
<td>n/a</td>
<td>10.0.0.15</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>node0</td>
<td>The default gateway for the seed node. Typically, it will be the same as the default gateway in the PXE network, as defined by the SET_PXE_NW_GW parameter.</td>
<td>n/a</td>
<td>10.0.0.1</td>
</tr>
<tr>
<td>node0</td>
<td>The PXE network bridge name. The name must match the name of the bridge created on the seed node during the Prepare the seed node stage.</td>
<td>KAAS_BM_PXE_BRIDGE</td>
<td>br0</td>
</tr>
<tr>
<td>node0</td>
<td>The IP address of the first KaaS management master node in the PXE network. Replace it in the templates and export as an environment variable.</td>
<td>SET_MASTER0_ADDR,KAAS_BM_MGMT_IP</td>
<td>10.0.0.22</td>
</tr>
<tr>
<td>node0</td>
<td>The MAC address of the first KaaS management master node in the PXE network.</td>
<td>SET_MACHINE_0_MAC</td>
<td>ac:1f:6b:02:84:71</td>
</tr>
<tr>
<td>node0</td>
<td>The IP address of the BMC endpoint for the first master node in the KaaS management cluster. Must be an address from OOB network that is accessible via PXE network default gateway.</td>
<td>SET_MACHINE_0_BMC_ADDRESS</td>
<td>192.168.100.1</td>
</tr>
<tr>
<td>node0</td>
<td>The IPMI user name to access the BMC, in base64 encoding.</td>
<td>SET_MACHINE_0_IPMI_USER_NAME</td>
<td>user</td>
</tr>
<tr>
<td>node0</td>
<td>The IPMI password to access the BMC, in base64 encoding.</td>
<td>SET_MACHINE_0_IPMI_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>node1</td>
<td>The IP address of the second KaaS management master node in the PXE network.</td>
<td>SET_MASTER1_ADDR</td>
<td>10.0.0.23</td>
</tr>
<tr>
<td>node1</td>
<td>The MAC address of the second KaaS management master node in the PXE network.</td>
<td>SET_MACHINE_1_MAC</td>
<td>ac:1f:6b:02:84:72</td>
</tr>
<tr>
<td>node1</td>
<td>The IP address of the BMC endpoint for the second master node in the KaaS management cluster. Must be an address from the OOB network that is accessible through the PXE network default gateway.</td>
<td>SET_MACHINE_1_BMC_ADDRESS</td>
<td>192.168.100.12</td>
</tr>
<tr>
<td>node1</td>
<td>The IPMI user name to access the BMC, in the base64 encoding.</td>
<td>SET_MACHINE_1_IPMI_USER_NAME</td>
<td>user</td>
</tr>
<tr>
<td>node1</td>
<td>The IPMI password to access the BMC, in the base64 encoding.</td>
<td>SET_MACHINE_1_IPMI_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>node2</td>
<td>The IP address of the third KaaS management master node in the PXE network.</td>
<td>SET_MASTER2_ADDR</td>
<td>10.0.0.23</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------------------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>The MAC address of the third KaaS management master node in the PXE network.</td>
<td>SET_MACHIN_2_MAC</td>
<td>ac:1f:6b:02:84:73</td>
</tr>
<tr>
<td></td>
<td>The IP address of the BMC endpoint for the third master node in the KaaS management cluster. Must be an address from the OOB network that is accessible through the PXE network default gateway.</td>
<td>SET_MACHIN_2_BMC_ADDRESS</td>
<td>192.168.100.13</td>
</tr>
<tr>
<td></td>
<td>The IPMI user name to access the BMC, in the base64 encoding.</td>
<td>SET_MACHIN_2_IPMI_USERNAME</td>
<td>user</td>
</tr>
<tr>
<td></td>
<td>The IPMI password to access the BMC, in the base64 encoding.</td>
<td>SET_MACHIN_2_IPMI_PASSWORD</td>
<td>password</td>
</tr>
<tr>
<td>Other</td>
<td>The IP range to be used as external load balancers for the Kubernetes services with the LoadBalancer type. This range must be within the PXE network.</td>
<td>SET_METALLB_ADDR_POOL</td>
<td>10.0.0.61-10.0.0.80</td>
</tr>
<tr>
<td></td>
<td>The spec.loadBalancerIP address for the Keycloak service. This address must be within the METALLB_ADDR_POOL range.</td>
<td>KEYCLOAK_FLOATING_IP</td>
<td>10.0.0.70</td>
</tr>
<tr>
<td></td>
<td>The spec.loadBalancerIP address for the IAM service. This address must be within the METALLB_ADDR_POOL range.</td>
<td>IAM_FLOATING_IP</td>
<td>10.0.0.71</td>
</tr>
<tr>
<td></td>
<td>The spec.loadBalancerIP address for the Squid service. This address must be within the METALLB_ADDR_POOL range.</td>
<td>PROXY_FLOATING_IP</td>
<td>10.0.0.72</td>
</tr>
<tr>
<td></td>
<td>The IP address of the externally accessible API endpoint of the KaaS management cluster. This address must NOT be within the METALLB_ADDR_POOL range but must be from the PXE network. External load balancers are not supported.</td>
<td>SET_LB_HOST</td>
<td>10.0.0.90</td>
</tr>
</tbody>
</table>

Proceed with Prepare metadata and deploy the KaaS management cluster.

Prepare metadata and deploy the KaaS management cluster

Using the example procedure below, replace the addresses and credentials in the configuration YAML files with the data from your configuration table prepared in Prepare bare metal prerequisites data. Keep everything else as is, including the file names and YAML structure.

1. Log in to the seed node that you configured as described in Prepare the seed node.
2. Create and change to the directory that will contain the bootstrap metadata files:
3. Download and run the Mirantis KaaS bootstrap script to this directory:

```bash
mkdir -p /home/ubuntu/bootstrap/
cd /home/ubuntu/bootstrap/
wget https://binary.mirantis.com/releases/get_kaas.sh
chmod 0755 get_kaas.sh
./get_kaas.sh
```

4. Change the directory to the kaas-bootstrap folder created by the get_kaas.sh script.

5. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with KaaS.

6. Save the license file as mirantis-kaas.lic under the Mirantis KaaS kaas-bootstrap directory.

7. Copy the template files to the bootstrap/templates folder for further modifications:

```bash
mkdir -p /home/ubuntu/bootstrap/templates
cp -r kaas-bootstrap/templates/bm/* /home/ubuntu/bootstrap/templates/
```

8. Update the bare metal hosts definition template in

```yaml
/home/ubuntu/templates/baremetalhosts.yaml.template
```
according to the bare metal prerequisite data prepared as described in Prepare bare metal prerequisites data. Manually set all parameters that start with SET_. For example, SET_MACHINE_0_BMC_ADDRESS.

---

1(1, 2, 3, 4, 5, You can obtain the base64-encoded user name and password using the following command in your Linux console:

```bash
$ echo -n <username|password> | base64
```
9. Update the machines hosts definition template in /home/ubuntu/templates/machines.yaml.template according to the bare metal prerequisite data prepared as described in Prepare bare metal prerequisites data. Manually set all parameters that start with SET_. For example, SET_MASTER0_ADDR.

Caution!

Since the bootstrap procedure relies on the NodeSelector constraint for a successful deployment, do not change the machineList order. master-0 must always be the first in the list.

10. Update the cluster definition template in /home/ubuntu/templates/cluster.yaml.template according to the bare metal prerequisite data prepared as described in Prepare bare metal prerequisites data. Manually set all parameters that start with SET_. For example, SET_METALLB_ADDR_POOL.

11. Configure the IAM parameters.

12. Configure the Ceph cluster:

   1. Set up the disk configuration according to your hardware node specification in /home/ubuntu/templates/kaascephcluster.yaml.template. Also, verify that the storageDevices section has a valid list of the HDD device names. For example:

   ```yaml
   storageDevices:
   - name: sdb
     role: hdd
   # Each storageDevices dicts can have several devices
   storageDevices:
   - name: sdb
     role: hdd
   - name: sdc
     role: hdd
   # Do not to include first devices here (like vda or sda)
   # because they will be allocated for operating system
   
   2. Verify that the machine names in the spec:nodes structure are relevant to the metadata:name structure data in machines.yaml.template.

13. Update the bootstrap overrides definition template in /home/ubuntu/templates/bootstrapoverrides.yaml.template according to the bare metal prerequisite data prepared as described in Prepare bare metal prerequisites data. Manually set all parameters that start with SET_. For example, SET_MASTER0_ADDR.
14 Update the IP address management definition template in 
./home/ubuntu/templates/ipam-objects.yaml.template according to the bare metal 
prerequisite data prepared as described in Prepare bare metal prerequisites data. Manually 
set all parameters that start with SET_. For example, SET_IPAM_POOL_RANGE.

15 Verify that the /bootstrap/ directory contains the following files:

```bash
# tree /home/ubuntu/bootstrap/
|-- kaas-bootstrap
|   |-- bootstrap.sh
|   |    ...
|   `-- templates
|       |-- baremetalhosts.yaml.template
|       |-- bootstrap-overrides.yaml.template
|       |-- cluster.yaml.template
|       |-- kaascephcluster.yaml.template
|       |-- ipam-objects.yaml.template
|       `-- machines.yaml.template
```

16 Export all required parameters:

```bash
cd /home/ubuntu/bootstrap/kaas-bootstrap/
export DEBUG="true"
export KAAS_BM_ENABLED="true"

# export KAAS_BM_PXE_IP="10.0.0.20"
export KAAS_BM_PXE_MASK="24"
export KAAS_BM_PXE_BRIDGE="br0"
export KAAS_BM_MGMT_IP='10.0.0.22'

# export KEYCLOAK_FLOATING_IP="10.0.0.70"
export IAM_FLOATING_IP="10.0.0.71"
export PROXY_FLOATING_IP="10.0.0.72"
export HOME="/home/ubuntu/bootstrap/kaas-bootstrap/"
export TEMPLATES_DIR="/home/ubuntu/bootstrap/templates/"
```

17 Applicable since KaaS release 1.6.0  Run the verification preflight script to validate the deployment 
templates configuration:

```bash
./bootstrap.sh preflight
```
The command outputs a human-readable report that includes the list of verified bare metal nodes and their Chassis Power status. This status is based on the deployment templates configuration used during the verification.

**Caution!**

If the report contains information about missing dependencies or incorrect configuration, fix the issues before proceeding to the next step.

18 Run the bootstrap script:

```
./bootstrap.sh all
```

**Warning**

During the bootstrap process, do not manually restart or power off any of the bare metal hosts.
Deploy an OpenStack-based management cluster

This section describes how to bootstrap an OpenStack-based KaaS management cluster.

Prerequisites

Before you start with bootstrapping the OpenStack-based KaaS management cluster, complete the following prerequisite steps:

1. Verify that your planned cloud meets the reference hardware bill of material and software requirements as described in KaaS Reference Architecture: Requirements for an OpenStack-based Kubernetes cluster.
2. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
3. Install Docker version 18.09:

   ```
   sudo apt install docker.io
   ```

4. Grant your USER access to the Docker daemon:

   ```
   sudo usermod -aG docker $USER
   ```

5. Log off and log in again to the bootstrap node to apply the changes.
6. Verify that Docker is configured correctly and has access to Mirantis KaaS CDN. For example:

   ```
   docker run --rm alpine sh -c "apk add --no-cache curl; \curl https://binary.mirantis.com"
   ```

   The system output must contain a json file with no error records. In case of issues, follow the steps provided in Troubleshooting.
7. Proceed to Bootstrap a management cluster.

Bootstrap a management cluster

After you complete the prerequisite steps described in Prerequisites, proceed with bootstrapping your OpenStack-based KaaS management cluster.

To bootstrap an OpenStack-based KaaS management cluster:

1. Log in to the bootstrap node running Ubuntu 18.04 that is configured as described in Prerequisites.
2. Download and run the Mirantis KaaS bootstrap script:
3. Change the directory to the kaas-bootstrap folder created by the get_kaas.sh script.

4. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with KaaS.

5. Save the license file as mirantis-kaas.lic under the Mirantis KaaS bootstrap directory.

6. Log in to the OpenStack Horizon.

7. In the Project section, select API Access.

8. In the right-side drop-down menu Download OpenStack RC File, select OpenStack clouds.yaml File.

9. Add the downloaded clouds.yaml file to the directory with the bootstrap.sh script.

10. In clouds.yaml, add the password field with your OpenStack password under the clouds/openstack/auth section.

   Example:

   ```yaml
   clouds:
       openstack:
           auth:
               auth_url: https://auth.openstack.example.com:5000/v3
               username: your_username
               password: your_secret_password
               project_id: your_project_id
               user_domain_name: your_ldap_password
               region_name: RegionOne
               interface: public
               identity_api_version: 3
   ```

11. Verify access to the target cloud endpoint from Docker. For example:

   ```bash
   docker run --rm alpine sh -c "apk add --no-cache curl; \curl https://auth.openstack.example.com:5000/v3"
   ```

   The system output must contain a json file with no error records. In case of issues, follow the steps provided in Troubleshooting.

12. In templates/machines.yaml.template, modify the spec:providerSpec:value sections for set: master and set: node by substituting the flavor and image parameters with the corresponding values of your OpenStack cluster. For example:

   ```yaml
   spec:
       providerSpec:
           value:
   ```
```yaml
apiVersion: "openstackproviderconfig.k8s.io/v1alpha1"
kind: "OpenstackMachineProviderSpec"
flavor: kaas.minimal
image: bionic-server-cloudimg-amd64-20190612
```

Also, modify other parameters as required.

13 Modify the templates/cluster.yaml.template parameters as required.

14 Configure the IAM parameters.

15 Run the bootstrap script:

```
./bootstrap.sh all
```

When the bootstrap is complete:

- The script creates the kubeconfig file in the directory where the script is located. This file contains the admin credentials for the KaaS management cluster.
- The system outputs the URL and credentials for the KaaS web UI.
- The system outputs the URL for Keycloak.
- The KaaS bootstrap cluster resources are freed up.

16 In case of deployment issues, collect and inspect the bootstrap and management cluster logs as described in Troubleshooting.

Now, you can proceed with managing the KaaS management cluster using the KaaS web UI and deploying the KaaS child clusters as described in Create an OpenStack-based KaaS child cluster.
Deploy an AWS-based management cluster

This section describes how to bootstrap a KaaS management cluster that is based on the Amazon Web Services (AWS) cloud provider.

Prerequisites

Before you start with bootstrapping the AWS-based KaaS management cluster, complete the following prerequisite steps:

1. Inspect the [Requirements for an AWS-based Kubernetes cluster](#) to understand the potential impact of the KaaS deployment on your AWS cloud usage.
2. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
3. Install Docker version 18.09:

   ```bash
   sudo apt install docker.io
   ```

4. Grant your USER access to the Docker daemon:

   ```bash
   sudo usermod -aG docker $USER
   ```

5. Log out and log in again to the bootstrap node to apply the changes.
6. Verify that Docker is configured and works correctly. For example:

   ```bash
   docker run --rm alpine sh -c "apk add --no-cache curl; \curl https://binary.mirantis.com"
   ```

   The system output must contain a json file with no error records. In case of issues, follow the steps provided in Troubleshooting.

7. Proceed to Bootstrap a management cluster.

Bootstrap a management cluster

After you complete the prerequisite steps described in Prerequisites, proceed with bootstrapping your AWS-based KaaS management cluster.

To bootstrap an AWS-based KaaS management cluster:

1. Log in to the bootstrap node running Ubuntu 18.04 that is configured as described in Prerequisites.
2. Download and run the Mirantis KaaS bootstrap script:

   ```bash
   wget https://binary.mirantis.com/releases/get_kaas.sh
   chmod 0755 get_kaas.sh
   ```
3. Change the directory to the kaas-bootstrap folder created by the get_kaas.sh script.
4. Obtain your license file that will be required during the bootstrap. See step 3 in Getting Started with KaaS.
5. Save the license file as mirantis-kaas.lic under the Mirantis KaaS bootstrap directory.
6. Verify access to the target cloud endpoint from Docker. For example:

```
docker run --rm alpine sh -c "apk add --no-cache curl; curl https://ec2.amazonaws.com"
```

The system output must contain a json file with no error records. In case of issues, follow the steps provided in Troubleshooting.
7. In templates/aws/machines.yaml.template, modify the spec:providerSpec:value section by substituting the ami:id parameter with the corresponding value for Ubuntu 18.04 from the required AWS region. For example:

```
spec:
  providerSpec:
    value:
      apiVersion: aws.kaas.mirantis.com/v1alpha1
      kind: AWSMachineProviderSpec
      instanceType: c5d.2xlarge
      ami:
        id: ami-033a0960d9d83ead0
```

Also, modify other parameters as required.
8. Modify the templates/aws/cluster.yaml.template parameters as required.
9. Configure the IAM parameters.
10. Generate the AWS Access Key ID with Secret Access Key for the admin user and select the AWS default region name. For details, see Official AWS documentation.
11. Export the following parameters by adding the corresponding values for the AWS admin credentials created in the previous step:

```
export KAAS.AWS.ENABLED=true
export AWS_SECRET_ACCESS_KEY=XXXXXXX
export AWS_ACCESS_KEY_ID=XXXXXXX
export AWS_DEFAULT_REGION=us-east-2
```

12. Create the AWS CloudFormation template for IAM policy:
13 Generate the AWS Access Key ID with Secret Access Key for the bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user, that was created in the previous step, and select the AWS default region name.

14 Export the AWS bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user credentials that were created in the previous step:

```bash
export KAAS_AWS_ENABLED=true
export AWS_SECRET_ACCESS_KEY=XXXXXXX
export AWS_ACCESS_KEY_ID=XXXXXXX
export AWS_DEFAULT_REGION=us-east-2
```

15 Run the bootstrap script:

```bash
./bootstrap.sh all
```

When the bootstrap is complete:

- The script creates the kubeconfig file in the directory where the script is located. This file contains the admin credentials for the KaaS management cluster.
- The system outputs the URL and credentials for the KaaS web UI.
- The system outputs the URL for Keycloak.
- The KaaS bootstrap cluster resources are freed up.

16 In case of deployment issues, collect and inspect the bootstrap and management cluster logs as described in Troubleshooting.

Now, you can proceed with managing the KaaS management cluster using the KaaS web UI and deploying the KaaS child clusters as described in Create an AWS-based KaaS child cluster.
Troubleshooting

This section provides solutions to the issues that may occur while deploying a Mirantis KaaS management cluster.

Collect the bootstrap logs

If the bootstrap script fails during the deployment process, collect and inspect the bootstrap and management cluster logs.

To collect the bootstrap logs:

1. Log in to your local machine where the bootstrap script was executed.
2. Run the following command:
   
   ```
   ./bootstrap.sh collect_logs
   ```

   The logs are collected in the directory where the bootstrap script is located.

   Depending on the type of issue found in logs, apply the corresponding fixes. For example, if you detect the LoadBalancer ERROR state errors during the bootstrap of an OpenStack-based KaaS management cluster, contact your system administrator to fix the issue. To troubleshoot other issues, refer to the corresponding section in Troubleshooting.

DNS settings

If you have issues related to the DNS settings, the following error message may occur:

```
curl: (6) Could not resolve host
```

The issue may occur if a VPN is used to connect to the cloud or a local DNS forwarder is set up. The workaround is to change the default DNS settings for Docker:

1. Log in to your local machine.
2. Identify your internal or corporate DNS server address:

   ```
   systemd-resolve --status
   ```

3. Create or edit `/etc/docker/daemon.json` by specifying your DNS address:

   ```
   {
   "dns": ["<YOUR_DNS_ADDRESS>"]
   }
   ```

4. Restart the Docker daemon:

   ```
   sudo systemctl restart docker
   ```
Default network address

If you have issues related to the default network address configuration, cURL either hangs or the following error occurs:

curl: (7) Failed to connect to xxx.xxx.xxx.xxx port xxxx: Host is unreachable

The issue may occur because the default Docker network address 172.17.0.0/16 overlaps with your cloud address or other addresses of the network configuration.

Workaround:

1. Log in to your local machine.
2. Verify routing to the IP addresses of the target cloud endpoints:
   1. Obtain the IP address of your target cloud. For example:

   nslookup auth.openstack.example.com

   Example of system response:

   Name: auth.openstack.example.com
   Address: 172.17.246.119

   2. Verify that this IP address is not routed through docker0 but through any other interface, for example, ens3:

   ip r get 172.17.246.119

   Example of the system response if the routing is configured correctly:

   172.17.246.119 via 172.18.194.1 dev ens3 src 172.18.1.1 uid 1000 cache

   Example of the system response if the routing is configured incorrectly:

   172.17.246.119 via 172.18.194.1 dev docker0 src 172.18.1.1 uid 1000 cache

3. If the routing is incorrect, change the IP address of the default Docker bridge:

   1. Create or edit /etc/docker/daemon.json by adding the "bip" option:

   ```json
   {
     "bip": "192.168.91.1/24"
   }
   ```
2. Restart the Docker daemon:

```
sudo systemctl restart docker
```

**TLS handshake timeout**

If you execute the bootstrap.sh script from an OpenStack VM that is running on the OpenStack environment used for bootstrapping the management cluster, the following error messages may occur that can be related to the MTU settings discrepancy:

```
curl: (35)  OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to server:port
Failed to check if machine "<machine_name>" exists:
failed to create provider client ... TLS handshake timeout
```

To identify whether the issue is MTU-related:

1. Log in to the OpenStack VM in question.
2. Compare the MTU outputs for the docker0 and ens3 interfaces:

```
ip addr
```

Example of system response:

```
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500...
... 2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450...
```

If the MTU output values differ for docker0 and ens3, proceed with the workaround below. Otherwise, inspect the logs further to identify the root cause of the error messages.

**Workaround:**

1. In your OpenStack environment used for KaaS, log in to any machine with CLI access to OpenStack. For example, you can create a new Ubuntu VM (separate from the bootstrap VM) and install the python-openstackclient package on it.
2. Change the vXLAN MTU size for the VM to the required value depending on your network infrastructure and considering your physical network configuration, such as Jumbo frames, and so on.

```
openstack network set --mtu <YOUR_MTU_SIZE> <network-name>
```
3. Stop and start the VM in Nova.
4. Log in to the bootstrap VM dedicated for the KaaS management cluster.
5. Re-execute the bootstrap.sh script.
Configure authentication for KaaS

This section describes how to configure authentication for KaaS depending on the external Identity Provider type integrated to your deployment.

Configure the IAM parameters

Before bootstrapping a KaaS management cluster, configure the IAM parameters for an Identity Provider and for the IAM roles with passwords.

After you complete the prerequisite steps depending on the provider type of your choice and download the bootstrap script, configure cluster.yaml.template for IAM as described below.

To configure cluster.yaml.template for IAM:

1. Choose from the following options:
   - For a baremetal-based KaaS cluster, open the templates/bm/cluster.yaml.template file for editing.
   - For an OpenStack or AWS-based KaaS cluster, open the templates/cluster.yaml.template file for editing.

2. Configure the external Identity Provider parameters:
   - For LDAP, configure the keycloak:userFederation:providers: and keycloak:userFederation:mappers: sections. For details, see Configure LDAP for IAM.
   - For Google OAuth on the DNS-based management cluster, configure the keycloak:externalIdP: section by adding the following snippet with your credentials:

   ```yaml
   keycloak:
   externalIdP:
   google:
   enabled: true
   config:
   clientId: <Google_OAuth_client_ID>
   clientSecret: <Google_OAuth_client_secret>
   ```

   To create the credentials, see Configure Google OAuth IdP for IAM.

3. Create hashed passwords for every IAM role: operator, writer, and reader. These passwords will be used to access the KaaS web UI.

   ```shell
   ./bin/hash-generate
   ```

   The hash-generate utility requests you to enter a password and outputs the parameters required for the next step. Use the tool to generate three hashed passwords for every IAM role.

4. In the initUsers section, add the following parameters for each IAM role that you generated in the previous step:
• passwordSalt - base64-encoded randomly generated sequence of bytes.
• passwordHash - base64-encoded password hash generated using passwordHashAlgorithm with passwordHashIterations. Supported algorithms include pbkdf2-sha256 and pbkdf-sha512.

For example:

```
passwordSalt: 6ibPZdUfQK8PsOpSmyVJnA==
passwordHash: 23W1l65FBdI3NL7LMiUQG9Cu62bWLtqIsOgdW8xNsqw=
passwordHashAlgorithm: pbkdf2-sha256
passwordHashIterations: 300000
```

Now, return to the deployment instruction depending on the provider type of your KaaS management cluster:

• For a baremetal-based KaaS cluster, see step 12 of the Prepare metadata and deploy the KaaS management cluster section in Bootstrap a baremetal-based management cluster.
• For an OpenStack-based KaaS cluster, see step 15 in Bootstrap an OpenStack-based management cluster.
• For an AWS-based KaaS cluster, see step 10 in Bootstrap an AWS-based management cluster.

### Configure LDAP for IAM

If you integrate LDAP for IAM to KaaS, add the required LDAP configuration to templates/cluster.yaml.template during the bootstrap of the KaaS management cluster. For details, see Bootstrap a management cluster.

LDAP configuration example:

```
keycloak:
  userFederation:
    providers:
      - displayName: "<LDAP_NAME>"
        providerName: "ldap"
        priority: 1
        fullSyncPeriod: -1
        changedSyncPeriod: -1
        config:
          pagination: "true"
          debug: "false"
          searchScope: "1"
          connectionPooling: "true"
          usersDn: "<DN>" # "ou=People, o=<ORGANIZATION>, dc=<DOMAIN_COMPONENT>"
          userObjectClasses: "inetOrgPerson, organizationalPerson"
          usernameLDAPAttribute: "uid"
          rdnLDAPAttribute: "uid"
          vendor: "ad"
         EditMode: "READ_ONLY"
```
Configure Google OAuth IdP for IAM

If you integrate Google OAuth external Identity Provider for IAM to KaaS, create the authorization credentials for IAM in your Google OAuth account before deploying a KaaS management cluster.

To create Google OAuth credentials for IAM:

2. Navigate to Credentials.
3. In the APIs Credentials menu, select OAuth client ID.

4. In the window that opens:

   1. In the Application type menu, select Web application.
   2. In the Authorized redirect URIs field, type in
      \(<keycloak-url>/auth/realms/iam/broker/google/endpoint\), where \(<keycloak-url>\) is the
      corresponding DNS address.
   3. Press Enter to add the URI.
   4. Click Create.

      A page with your client ID and client secret opens. Save these credentials for further
      usage.

5. Proceed to the Bootstrap a management cluster procedure. The Google OAuth credentials
   are required in the step 12 of that procedure.