QuickStart: Mirantis Container Cloud on AWS

version master

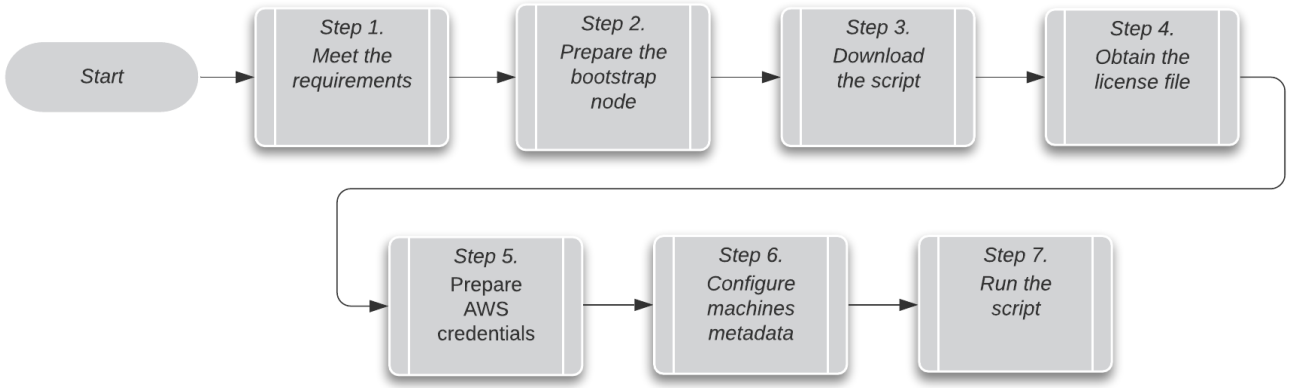
Contents

Copyright notice

Introduction

Using this QuickStart tutorial, you can deploy a Mirantis Container Cloud AWS-based management cluster containing 3 control plane nodes. This cluster will run the public API and the web UI. Using the Container Cloud web UI, you can deploy managed clusters that run Mirantis Kubernetes Engine.

The following diagram illustrates the deployment overview of a Container Cloud AWS-based management cluster:



Before you begin

Before you start the cluster deployment, verify that your system meets the following minimum hardware and software requirements for an AWS-based management cluster.

> Warning
>
> Some of the AWS features required for Container Cloud may not be included into your AWS account quota. Therefore, carefully consider the AWS fees applied to your account that may increase for the Container Cloud infrastructure.

> Note
>
> For the bootstrap node, you can use any local machine running Ubuntu 18.04 with the following resources:
>
> - 2 vCPUs
> - 4 GB of RAM
> - 5 GB of available storage

Minimum hardware requirements for a management cluster

Resource	Requirement
# of nodes	4 (3 for HA + 1 for Bastion)
# of vCPUs	25 (8 per node + 1 for Bastion)
RAM in GB	49 (16 per node + 1 for Bastion)
Storage in GB	360 (120 per node)
Instance type	c5d.2xlarge
Bastion host instance type	t2.micro
# of volumes	7 (total 110 GB)
# of Elastic load balancers	10
# of Elastic IP addresses	1

Minimum software requirements for a management cluster

Software	Version
Operating system distribution	Ubuntu 18.04

Docker	Current version available for Ubuntu 18.04

Prepare the bootstrap node

1. Log in to any personal computer or VM running Ubuntu 18.04 that you will be using as the bootstrap node.
2. If you use a newly created VM, run:

   ```
   sudo apt-get update
   ```

3. Install the current Docker version available for Ubuntu 18.04:

   ```
   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 the Container Cloud CDN. For example:

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

   The system output must not contain error records.

Download the bootstrap script

1. On the bootstrap node, 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. Change the directory to the kaas-bootstrap folder created by the script.

Obtain the Mirantis license

1. Create a user account at mirantis.com.
2. Log in to your account and download the mirantis.lic license file.
3. Save the license file as mirantis.lic under the kaas-bootstrap directory on the bootstrap node.

Prepare the AWS configuration

1. On the bootstrap node, 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 not must contain error records.

2. Generate the AWS Access Key ID with Secret Access Key for the admin user and select the AWS default region name. For details, see AWS General Reference: Programmatic access.
3. Change the directory to the kaas-bootstrap folder created by the get_container_cloud.sh script.
4. 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
   ```

5. For Container Cloud to communicate with the AWS APIs, create the AWS CloudFormation stack that contains properly configured IAM users and policies:

   1. Log in to your AWS Management Console.
   2. On the home page, expand the upper right menu with your user name and capture your Account ID.
   3. Create the CloudFormation template:

      ```
      ./kaas bootstrap aws policy --account-id <accountId> --dump > cf.yaml
      ```

   4. Send the cf.yaml template to your AWS account admin to create the CloudFormation stack from this template.

6. Using your AWS Management Console, generate the AWS Access Key ID with Secret Access Key for the bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user, created in the previous step, and select the AWS default region name.
7. Export the AWS bootstrapper.cluster-api-provider-aws.kaas.mirantis.com user credentials that were 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
   ```

8. 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
```

Finalize the bootstrap

1. Run the bootstrap script:

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

2. When the bootstrap is complete, collect and save the following management cluster details in a secure location:

   - The kubeconfig file located in the same directory as the bootstrap script. This file contains the admin credentials for the management cluster.
   - The private ssh_key for access to the management cluster nodes that is located in the same directory as the bootstrap script.
   - The URL and credentials for the Container Cloud web UI. The system outputs these details when the bootstrap completes.
   - The StackLight endpoints. For details, see Operations Guide: Access StackLight web UIs.
   - The Keycloak URL that the system outputs when the bootstrap completes. The admin password for Keycloak is located in kaas-bootstrap/passwords.yml along with other IAM passwords.

   > Note
   >
   > When the bootstrap is complete, the bootstrap cluster resources are freed up.

What’s next

Using your newly deployed management cluster, you can:

- Deploy an additional regional cluster to operate managed clusters of several cloud providers and configurations within a single Container Cloud deployment in parallel.
- Create and operate AWS-based managed clusters. Before that, verify that your planned cluster meets the requirements for managed clusters.
- Configure an external identity provider for IAM.
- Attach an existing Mirantis Kubernetes Engine cluster.

For details about all Container Cloud features, refer to the full set of Container Cloud documentation.

