Deploy Kubernetes on OpenStack with SUSE CaaS Platform

February 2019

Written by:
Simon Briggs and John Tonello, SUSE
# Table of Contents

Table of Contents ..................................................................................................................................................... 2  
Starting from Scratch .................................................................................................................................................. 3  
Kubernetes Deployment .............................................................................................................................................. 5  
Deploy the Heat Templates .......................................................................................................................................... 8  
Bootstrap the Cluster .................................................................................................................................................. 12  
Using Your New SUSE CaaS Platform Kubernetes Stack .......................................................................................... 15  
Install the Kubernetes Dashboard ............................................................................................................................... 15  
Deploy an Application ................................................................................................................................................ 17  
Resources ................................................................................................................................................................. 18
Introduction

When it comes to deploying Kubernetes to manage your containerized environment, you have a lot of choices. After all, it’s a complex system that takes planning and enough resources to not just make it work, but to make it work well. Rather than poking around in the dark, SUSE offers an integrated solution that can get you started, and allow you to easily make modifications as your needs and expertise grow.

The solution is SUSE CaaS Platform, which runs on top of SUSE OpenStack Cloud with SUSE Enterprise Storage providing the underlying storage mix. In this guide, we’ll show you how to use Heat templates to first define and then deploy your Kubernetes cluster. To keep things straightforward, we’ll keep the YAML template simple, but detailed enough to help you understand the machinations so you can customize the approach to suit your needs.

Starting from Scratch

Kubernetes does a nice job of abstracting away the underlying resources needed to deploy containerized applications, but the infrastructure matters. That is, you need to have sufficient storage, compute capacity and network resources to make everything work. As you might have heard, that means collaborating with other teams to make it a less painful exercise. In fact, if you’re looking for a DevOps project, this is a good one.

OpenStack has a lot of bits and pieces, most of which are required for Kubernetes (SUSE CaaS Platform in this case). A minimum OpenStack configuration should have the following components:

- Nova (compute)
- Neutron (network management)
- Keystone (identity)
- Glance, Cinder and Swift (Ceph storage provided by SUSE Enterprise Storage)
- Heat (orchestration)
- Horizon (dashboard)
- Glance (image registry)

Of course, with OpenStack there are many more options that you can bolt on. This base environment can be readily deployed with SUSE OpenStack Cloud using the Cloud Lifecycle Manager, a graphical tool that automates OpenStack deployments from pre-scooped templates. To learn more, visit www.suse.com/products/suse-openstack-cloud/.
Figure 1. The basic OpenStack architecture that your Kubernetes cluster will be built on.

For our OpenStack deployment, we used SUSE Enterprise Storage, which provides Ceph object and block storage for our cluster. You could also use a traditional SAN, but the flexibility of Ceph makes it easy to add and scale volumes in an OpenStack cluster. To learn more, visit www.suse.com/documentation/suse-enterprise-storage-5/book_storage_admin/data/part_dataaccess.html.
Kubernetes Deployment

With our OpenStack environment up and running, we’re ready to use Heat scripts—basic YaML files—to describe and define the SUSE CaaS Platform stack and the environment we want. This consists of three parts: caasp-stack.yaml, caasp-environment.yaml and the SUSE server image.

caasp-stack.yaml – Our example stack includes an admin node for the initial Kubernetes cluster deployment, a master node and three worker nodes. The partial YaML script below defines the resources and parameters for the Kubernetes instance, including compute (Nova), networking (Neutron), Security Group settings (including shared key pairs) and other features drawn from our OpenStack resources. The full YaML script can be downloaded from the Resources section at the end of this guide.

```yaml
parameter_groups:
  - label: general
    description: General Parameters
    parameters:
      - image
      - root_password

  - label: sizing
    description: Sizing Parameters
    parameters:
      - admin_flavor
      - master_flavor
      - worker_flavor
      - worker_count

  - label: network
    description: Network Parameters
    parameters:
      - external_net

parameters:
```

Figure 2. OpenStack takes advantage of Ceph, which provides block and object volumes.
image:  
  type: string  
  description: Name of image to use for servers  
  constraints:  
    - custom_constraint: glance.image  

external_net:  
  type: String  
  description: >  
    Name or ID of public network for which floating IP addresses will be allocated  
  default: floating  

admin_flavor:  
  type: string  
  description: Admin Flavor  
  default: m1.medium  
  constraints:  
    - custom_constraint: nova.flavor  

master_flavor:  
  type: string  
  description: Master Flavor  
  default: m1.medium  
  constraints:  
    - custom_constraint: nova.flavor  

worker_flavor:  
  type: string  
  description: Worker Flavor  
  default: m1.medium  
  constraints:  
    - custom_constraint: nova.flavor  

worker_count:  
  type: number  
  description: Number of Worker nodes to boot  
  default: 3  

root_password:  
  type: string  
  description: Root Password for the VMs

...

secgroup_base:  
  type: OS::Neutron::SecurityGroup  
  properties:  
    rules:  
      - protocol: icmp  
      - protocol: tcp  
        port_range_min: 22  
        port_range_max: 22  
      - protocol: tcp  
        port_range_min: 2379  
        port_range_max: 2379

...

worker:  
  type: OS::Heat::AutoScalingGroup  
  depends_on: [ external_router_int, admin ]  
  # - external_router_int  
  # - admin
As you can see, this Heat script defines the OpenStack resources that will be used to build the Kubernetes cluster. By using YAML to define the infrastructure, we can easily make adjustments and redeploy our SUSE CaaS Platform on this or any other OpenStack cluster.

**caasp-environment.yaml.** This short YAML script defines m1.medium instances for each of the node types (admin, master and worker), sets a root password and sets the external network as floating IP. The complete script is just a few lines:

```yaml
---
parameters:
  root_password: Your-Password-Here
  admin_flavor: m1.medium
  master_flavor: m1.medium
  worker_flavor: m1.medium
  external_net: floating
```

**SUSE server image.** Each instance used for the Kubernetes deployment is built using a SUSE CaaS Platform OS image, which in this case is based on SUSE Linux Enterprise Server 12. A standard image or a lighter-weight custom image can
be deployed to suit your needs. The only requirement is that the image you want to use is available in your OpenStack image library (provided by Glance).

**Figure 3. Some instances available in our SUSE OpenStack Cloud library.**

**Deploy the Heat Templates**

In your OpenStack dashboard, navigate to Orchestration → Stacks and click the “+Launch Stack” button to start creating a stack with the example YaML scripts. The Template Source is `caasp-stack.yaml` and the Environment Source is `caasp-environment.yaml`.

**Figure 4. Under Orchestration → Stacks, we import the Template and Environment source files.**
After clicking “Next,” manually select a variety of parameters for your SUSE CaaS Platform cluster, including the instance image you previously loaded (SUSE-CaaS-3.0 in this example), a password for the admin interface and the flavors for each node:

![Launch Stack interface]

*Figure 5: Provide basic information, including a password, for your stack.*

The Heat scripts will now trigger the automated build of the SUSE CaaS Platform cluster. Progress can be seen graphically in the Orchestration → Stacks view. As each node and service comes up, its icon turns green.
Although this seems straightforward, which it is, these simple steps are automating the deployment of the Kubernetes cluster and will make it available in a matter of minutes, not hours or days.

![Figure 6. As the components of the Heat template files are deployed, the resources appear in the OpenStack Topology.](image)

When the deployment is complete (everything in the Topology Map is green), navigate to the OpenStack → Instance listing to get the external IP address of the admin node:

![Table](image)

Paste that IP address into your browser to call up the SUSE CaaS Platform dashboard, where you’ll create an account with any email address and password. After you log in, you’ll configure the cluster itself:
The first thing you need to do is set the "Internal Dashboard Location" address to the internal IP address for the SUSE CaaS Platform admin node. Initially, this is pre-populated with the external IP address, which we now replace with 10.3.0.75:

![Figure 7. The SUSE CaaS Platform dashboard, available at the public IP address of the admin node.](image)

![Figure 8. Set the Internal Dashboard Location to the internal IP address of the admin node.](image)
In addition to adding 10.3.0.75 for the Internal Dashboard Location, tick the “Install Tiller” box and make adjustments to the Overlay network settings so that the addresses do not conflict with other resources in your OpenStack cluster. For this example, the defaults for the Cluster CIDR are reset to 172.16.0.0/13, 172.16.0.0 (lower bound) and 172.23.255.255 (upper bound). These addresses shouldn’t overlap:

![Figure 9. Set network addresses appropriate to your environment.](image)

**Bootstrap the Cluster**

The SUSE CaaS Platform is now ready to be bootstrapped:

![Figure 10. Bootstrap the cluster.](image)

The nodes are initially in a pending state. Select them all, wait a moment for them to be discovered and then assign their roles. Note that the master node is manually set as the master role and all others are set to the worker role.
Each node is automatically discovered. Accept each before moving on to assign them roles.

Assign roles to each node.
Before confirming the bootstrap, enter the External Kubernetes API, FQDN (in this case, the IP address of the master instance) and the Velum External Dashboard FQDN (in this case the same IP address you originally used to access the admin dashboard):

![Table showing IP addresses and status of cluster nodes](image1)

*Figure 13. Assign external IP addresses. In production, these would typically be domain names.*

When the deployment completes, log in to the “Admin” console, wait for the nodes to finish bootstrapping and click the “kubeconfig” button. That will download the credentials file you’ll need in order to access your Kubernetes cluster. When prompted, save the kubeconfig file to your local machine and copy it to your home directory. For example, on a Linux system:

```
$ mkdir ~/.kube
$ cp /home/user/Downloads/kubeconfig ~/.kube/config
```
Using Your New SUSE CaaS Platform Kubernetes Stack

You’re now ready to start using the Kubernetes cluster. To do that, you’ll need `kubectl` installed on your workstation. For example, on a SUSE workstation:

```
$ sudo zypper in kubectl
```

You’ll use `kubectl` for a number of actions, including installing and launching the proxy in order to access the Kubernetes dashboard. If you’re on a lab PC or don’t have access to `kubectl` on your workstation, you can use Google’s Cloud Shell at https://console.cloud.google.com/cloudshell. It has `kubectl` built in.

Test to make sure you have connectivity:

```
$ kubectl get nodes
```

```
NAME STATUS ROLES AGE VERSION
master Ready master 18m v1.8.2
node1 Ready master 14m v1.8.2
node2 Ready master 11m v1.8.2
```

If all goes well, you’ll see a listing of your running Kubernetes nodes, such as the above.

**Install the Kubernetes Dashboard**

Kubernetes makes the dashboard available from a public GitHub repository, so you can apply the YaML configuration file directly from there:

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml
```
When the installation completes, launch the dashboard using `kubectl proxy`. The dashboard will be made available through a `localhost` or `127.0.0.1` address, not the address of the SUSE CaaS Platform node hosting the master. If using the Google Cloud Shell, be sure to set the port to `8001`.

```
$ kubectl proxy
```

Now, open a browser and go to this URL:

```
http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/
```

If using Google Cloud Shell, replace “`localhost`” with the name of your cloud host, for example “`6801-dot-4837320-dot-devshell.appspot.com`”.

When prompted to log in to the dashboard, select “Token” from the login screen. Grab the token from your `kubeconfig` file (now located in `~/.kube/config`):

```
$ cat ~/.kube/config | grep id-token
```

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Ijc3MmMzYjAwMjcyZjJkMGNiNWQ5NjFjNzQ5Zjc4MTMxMDMxMTBkMTci
```

Copy and paste the token into the “Enter token” field:

![Kubernetes Dashboard](image)

**Figure 15. Enter the token.**
Deploy an Application

In the main Kubernetes dashboard, you can now create a new application deployment by clicking on the “+ CREATE” button at the top right. There are two other options available for creating new applications, but we’ll use the ad hoc tool that enables us to enter parameters in a form. Here, we deploy GitLab CE, but you can deploy anything you like.

We start by giving the new app a name, entering the public gitlab/gitlab-ce Docker repository and adding an External service to make port 80 available.

![Application creation form]

*Figure 16. Use the Kubernetes dashboard to create an app from text input, an existing file or by defining the app parameters.*

This will create both the application container and a service for the external endpoint, which you’ll use to access the newly created GitLab server. You might need to wait a few minutes for the container and external address service to complete, but once they do, copy the external IP address to your browser. You should see something like this:

![GitLab login screen]

*Figure 17. The GitLab dashboard initial login.*
Resources

- SUSE OpenStack Cloud documentation
- SUSE CaaS Platform documentation
- SUSE Enterprise Storage documentation
- The caasp-stack.yaml and caasp-environment.yaml files used in this example