Setting up a Proof of Concept Environment for SUSE CaaS Platform – Part 1

By Andrew Gracey, SUSE
Andrew.gracey@suse.com
# Table of Contents

Table of Contents .................................................................................................................................................................. 2

1. Environment Prep ............................................................................................................................................................. 3

2. Install Base System ........................................................................................................................................................... 4

3. Provision Cluster ............................................................................................................................................................. 11
SUSE CaaS Platform is a curated distribution of Kubernetes that makes it easy to install, maintain and secure your containerized workloads. (If you don’t know what Kubernetes is, don’t worry. We’ll go through several examples and use cases that show what it is and how to use it.)

In this three-part guide, we’ll explore what’s provided with the SUSE CaaS Platform and how to set up and use it. In the end, you will have a setup to play with and test.

**Part 1** (this document) covers the setup and installation of the platform:

1. Environment Prep
2. Install Base System(s)
3. Provision Cluster

**Part 2** covers some Kubernetes basics and looking at what you just installed:

4. A Quick Look Under the Hood
5. Using Helm
6. Building and Deploying Your First Container

**Part 3** covers other things to be aware of:

7. Storage
8. Networking
9. Security
10. Suggested Additional Reading

NOTE: In a few instances, the command specifies a backtick or minus. If you are copying and pasting, make sure that these don’t get converted into other characters (otherwise, it could cause odd behavior).

1. **Environment Prep**

   For the setup described in this guide, you will need at least two servers: one for the master node and one for the worker node. Each of these nodes should have 4GB RAM and 50GB disk space. These can be virtualized if desired; however, the virtual networking is not a part of this guide’s scope. Also be aware that having only two nodes is not a supported configuration; however, it does well enough to explore how the platform works in a test setting.

   **NOTE:**
   - For clusters in production, keep in mind that you want at least three master nodes for partition resilience.
   - You might also want another SLE system to run the skuba tool. (While this guide assumes that everything is on the master node, it’s helpful to run this tool external to the system.)
   - In an effort to keep this guide simple, we will not go into deployments with Terraform, OpenStack or any other virtualization technology.
DNS
To resolve all the nodes, it is helpful to have a private DNS available in your ecosystem, as well as having locations in the outside world. Each server will need a static IP address with a resolvable hostname. In this guide, we will use /etc/hosts. However, it's much easier to set up a quick DNS server. You can find the instructions here: setting up dns using yast.

I'll use the host names master0.local and worker0.local in this guide. If you are using DNS, replace the .local with your own domain where it's needed.

Storage
It is possible to hook into an existing storage solution. If you don’t already have one, you can also use a simple NFS shared drive. Instructions on how to set this up are in Part 3 of this guide.

2. Install Base System
Each node in your cluster will need to be loaded as a base SUSE Linux Enterprise Server 15 Service Pack.


Install the Base OS for Both Systems in the Same Way
- Boot into the installer.
- Select SUSE Linux Enterprise Server SP1.

- Agree to the Terms and Conditions.
- Set up networking.
  - Include hostnames, static IP, default gateway and DNS.

- Register the system.
  - Use either local RMT/SMT or scc.suse.com credentials.
• Add the SUSE CaaS Platform module.
  - You might need to uncheck “Hide Development Versions.”

• Don’t select an add-on product.

• Select Minimal for the System Role.
- Use Guided Setup for the file system.
  - Keep Logical Volume Management and Encryption disabled.

- Use btrfs with snapshots, but turn off the home and swap partitions.
  (NOTE: Kubernetes will not run with swap enabled.)
- Set up NTP.
- Set up an administrative user.

- Enable SSH.
- Disable KDump.
- Click “Install” and wait for the reboot.

Install the Patterns Needed
(NOTE: It’s also helpful to install useful packages such as vi, ip utils, etc.)

$ zypper in -t pattern SUSE-CaaSP-Management on the node you intend to run the install from.
$ zypper in -t pattern SUSE-CaaSP-Node on all nodes (including master nodes).

Add hostnames into /etc/hosts or set up DNS to resolve all nodes from all nodes (including themselves).

3. Provision Cluster

Now that you have two (or more) SUSE Linux Enterprise Server 15 SP1 servers running with the packages installed, you are ready to provision the cluster.

Set up SSH
Log into the master/management node as root. While this can be done from any SLES 15 SP1 environment, this guide will assume that you are running it from the first master node. The first step is to create and share an SSH key:

$ ssh-keygen -t rsa -b 4096
$ eval `ssh-agent` # note the backticks
$ ssh-add
$ ssh-copy-id master0.local
$ ssh-copy-id worker0.local
Generate Manifest

At this point, keys are shared and you are ready to install.

The next step is to generate the manifests needed:

```
$ skuba cluster init caasp4 --control-plane master0.local
```

It will create a directory with several files:

```
nuc:-/beta # cd caasp4/
nuc:-/beta/caasp4 # find .
./kubeadm-init.conf
./kubeadm-join.conf.d
./kubeadm-join.conf.d/master.conf.template
./kubeadm-join.conf.d/worker.conf.template
./addons
./addons/cni
./addons/cni/cilium.yaml
./addons/psp
./addons/psp/podsecuritypolicy-unprivileged.yaml
./addons/psp/podsecuritypolicy-privileged.yaml
./addons/kured
./addons/kured/kured.yaml
nuc:-/beta/caasp4 #
```

Switch to that directory:

```
$ cd caasp4
```

For more advanced usage, you can edit these files before bootstrapping the initial node. The only two considerations you need are regarding subnet collisions.

Inside kubeadm-init.conf, there are two lines:

```
podSubnet: 10.244.0.0/16
serviceSubnet: 10.96.0.0/12
```

Make sure these CIDR ranges don’t conflict with any others on your local network. For most users, this will not be an issue unless there are multiple clusters sharing the network.

Bootstrap First Node

You can now start the bootstrap process, which will take a few minutes:

```
$ skuba node bootstrap master0 -t master0.local
```

Once the process completes, you will see a newly created admin.conf file in the same directory. Move this to

```
~/kube/config:

$ mkdir ~/.kube
$ mv admin.conf ~/.kube/config
```

You can now use the kubectl command. Wait for the master node to fully initialize, using:

```
$ kubectl get node -w
```
Join Second Node
Once the output shows “Ready” join the second node, using:

$ skuba node join worker0 -t worker0 -r worker

To see when both nodes are ready, repeat the same kubectl command as before. (You can drop the -w to see a point in time instead of a stream.)

If you want to join more workers, repeat the same command, with worker0 replaced by the other hostnames.

Check Status of Cluster
Once both nodes are ready, you can get additional status on your cluster by running:

$ skuba cluster status

This will give you more detailed information about the nodes that are running on your cluster. It will also show whether any updates are pending and which ones might need downtime (something we try hard to avoid).

Done
To recap, you just bootstrapped a Kubernetes node and joined a worker node to it. At this point, SUSE CaaS Platform is up and running. You can begin poking around on it with the kubectl command.

Go ahead and look at what’s running, using:

$ kubectl get pods --all-namespaces

Stay tuned for Part Two of this guide, where we’ll show you how to get started exploring and using the environment that you just created.