



Release Notes

SUSE Cloud Application Platform 1.1

Publication Date: 2018-04-26

Contents

- 1 Major Changes 2
- 2 1.1 Release, April 2018 2
- 3 1.0.1 Release, February 2018 7
- 4 1.0 Release, January 2018 8
- 5 Known Issues 8
- 6 Installing SUSE Cloud Applications Platform on SUSE Containers as a Service Platform 10
- 7 Installing SUSE Cloud Applications Platform on Microsoft Azure 18
- 8 Service Broker Setup 22
- 9 Appendix: Configuration with secrets : Section 27

This document provides guidance and an overview to high-level general features and updates for SUSE Cloud Applications Platform 1.1. It also describes capabilities and limitations of SUSE Cloud Applications Platform 1.1. For detailed information about deploying this product, see the *Deployment Guide* at https://www.suse.com/documentation/cloud-application-platform-1/book_cap_deployment/data/book_cap_deployment.html.

These release notes are updated periodically. The latest version of these release notes is always available at <https://www.suse.com/releasesnotes>. General documentation can be found at <https://www.suse.com/documentation/cloud-application-platform-1>.

1 Major Changes

2 1.1 Release, April 2018

2.1 What Is New?

- Now supported on Microsoft Azure Container Services (AKS)
- Cloud Foundry component and buildpack updates (see *Section 2.4, “Features and Fixes”*)
- PostgreSQL and MySQL service broker sidecars, configured and deployed via Helm
- cf backup + CLI plugin for saving, restoring, or migrating CF data and applications

For more information about deploying SUSE Cloud Applications Platform, see the *Deployment Guide* at https://www.suse.com/documentation/cloud-application-platform-1/book_cap_deployment/data/book_cap_deployment.html.

2.2 Configuration Changes

Changes to the format of `values.yaml` for SCF and UAA require special handling when upgrading from SUSE Cloud Applications Platform 1.0 to 1.1 if you are reusing configuration files (e.g. `scf-config-values.yaml`):

- All secrets formerly set under `env:` are now set under `secrets:`. Any `_PASSWORD`, `_SECRET`, `_CERT`, or `_KEY` value explicitly set in `values.yaml` for SUSE Cloud Applications Platform 1.0 should be moved into the `secrets:` section before running `helm upgrade` with the revised `values.yaml`. Find a sample configuration in [Section 9, "Appendix: Configuration with secrets: Section"](#).
- **These secrets must be resupplied on each upgrade** (for example, the `CLUSTER_ADMIN_PASSWORD`, `UAA_ADMIN_CLIENT_SECRET`) as they will not be carried forward automatically. We recommend always using a values file.
- To rotate secrets, increment the `kube.secret_generation_counter` (immutable generated secrets will not be reset).
- To upgrade SUSE Cloud Applications Platform 1.0.1 to 1.1, run the following commands:

```
$ helm repo update
$ helm upgrade --recreate-pods <uaa-helm-release-name> suse/uaa --values scf-config-values.yaml
$ SECRET=$(kubectl get pods --namespace uaa -o
  jsonpath='{.items[*].spec.containers[?(.name=="uaa")].env[?
  (.name=="INTERNAL_CA_CERT")].valueFrom.secretKeyRef.name}')
$ CA_CERT=$(kubectl get secret $SECRET --namespace uaa -o
  jsonpath="{.data['internal-ca-cert']}" | base64 --decode -)
$ helm upgrade --recreate-pods <scf-helm-release-name> suse/cf --values scf-config-values.yaml --set "secrets.UAA_CA_CERT=${CA_CERT}"
$ helm upgrade --recreate-pods <console-helm-release-name> suse/console --values scf-config-values.yaml
```

- The `kube.external_ip` variable has been changed to `kube.external_ips`, allowing for services to be exposed on multiple Kubernetes worker nodes (for example, behind a TCP load balancer). Before upgrading, change the setting or add a new setting specified as an array. For example:

```
kube.external_ip=10.1.1.1
kube.external_ips=["10.1.1.1"]
```

- Both variables can exist at the same time and be set to the same value for those in mixed version environments. To specify multiple addresses, use:

```
kube.external_ips=["1.1.1.1", "2.2.2.2"]
```

- Upgrading from SUSE Cloud Applications Platform 1.0.1 to 1.1

An example `scf-config-values.yaml` for SUSE Cloud Applications Platform 1.1 would look like this:

```
env:
  # Domain for SCF. DNS for *.DOMAIN must point to a kube node's (not master)
  # external ip address.
  DOMAIN: cf-dev.io

kube:
  # The IP address assigned to the kube node pointed to by the domain.
  ##### the external_ip setting changed to accept a list of IPs, and was
  ##### renamed to external_ips
  external_ips: ["192.168.77.77"]
  storage_class:
    # Make sure to change the value in here to whatever storage class you
    use
    persistent: "persistent"
    shared: "shared"

  # The registry the images will be fetched from. The values below should
  work for
  # a default installation from the suse registry.
  registry:
    hostname: "registry.suse.com"
    username: ""
    password: ""
    organization: "cap"

  auth: rbac

secrets:
  # Password for user 'admin' in the cluster
  CLUSTER_ADMIN_PASSWORD: changeme

  # Password for SCF to authenticate with UAA
  UAA_ADMIN_CLIENT_SECRET: uaa-admin-client-secret
```

To upgrade from SUSE Cloud Applications Platform 1.0.1 to 1.1, run the following commands:

```
$ helm repo update
$ helm upgrade --recreate-pods <uaa-helm-release-name> suse/uaa --values scf-
config-values.yaml
```

```
$ SECRET=$(kubectl get pods --namespace uaa -o
  jsonpath='{.items[*].spec.containers[?(.name=="uaa")].env[?
  (.name=="INTERNAL_CA_CERT")].valueFrom.secretKeyRef.name}')
$ CA_CERT="$(kubectl get secret $SECRET --namespace uaa -o
  jsonpath="{.data['internal-ca-cert']}" | base64 --decode -)"
$ helm upgrade --recreate-pods <scf-helm-release-name> suse/cf --values scf-
  config-values.yaml --set "secrets.UAA_CA_CERT=${CA_CERT}"
$ helm upgrade --recreate-pods <console-helm-release-name> suse/console --
  values scf-config-values.yaml
```

2.3 Known Issues

- Do not set the mysql-proxy, routing-api, tcp-router, blobstore or diego_access roles to more than one instance each. Doing so can cause problems with subsequent upgrades which could lead to loss of data. Scalability of these roles will be enabled in an upcoming maintenance release.
- To upgrade high availability (HA) configurations, scale down the api role count to 1. Then upon completing the upgrade, scale api up again to 2 or more.
 - The diego-api, diego-brain and routing-api roles are configured as active/passive, and passive pods can appear as *Not Ready*. This is expected behavior.
- Azure operators may not be able to connect to Azure Database for {mysql}/{postgre} databases with the current brokers.
- cf backup-restore may leave Docker apps in a stopped state. These can be started manually.
- cf backup-restore produces an unhelpful error if the file is not valid JSON.

2.4 Features and Fixes

- Ability to specify multiple external IP addresses (please see <sec.issue> > below on impact to upgrades)
- MySQL now a clustered role
- MySQL-proxy enabled for UAA

- UAA has more logging enabled, so SCF_LOG_HOST, SCF_LOG_PORT and SCF_LOG_PROTOCOL have been exposed
- TCP routing ports are configurable and can be templated
- CPU limits can be set for pods.
- Memory limits for pods now properly enforced.
- Kubernetes annotations enabled so operators can specify what nodes particular roles can be run on
- Fixed cloud controller clock so that it will wait until API is ready
- Overhauled secret rotation for upgrades
- Includes these CF component versions:
 - diego-release 1.35
 - cf-mysql-release 36.10.0
 - cflinuxfs2-release 1.187.0
 - routing-release 0.172.0
 - garden-runc-release 1.11.1
 - nats-release 22
 - capi-release 1.49.0
- Includes these Cloud Foundry buildpack versions:
 - go-buildpack-release 1.7.19-16-g37cc6b4
 - binary-buildpack-release 1.0.17
 - nodejs-buildpack-release 1.5.30-13-g584d686
 - ruby-buildpack-release 9adff61
 - php-buildpack-release ea8acd0
 - python-buildpack-release 1.5.16-14-ga2bbb4c
 - staticfile-buildpack-release 1.4.0-12-gdfc6c09

- `dotnet-core-buildpack-release 1.0.26-14-gf951834`
- `java-buildpack-release 3.16-18-gfeab2b6`

3 1.0.1 Release, February 2018

- Using the `helm upgrade` command in SUSE Cloud Applications Platform 1.0 to 1.0.1 (scf 2.6.11 to 2.7.0) requires the use of `--force` to drop an unnecessary persistent volume. Note that `helm upgrade` only works for multi-node clusters when running with a proper HA storage class. For example, `hostpath` will not work, as required stateful data can be lost.
- Bump to Cloud Foundry Deployment (1.9.0), using Cloud Foundry Deployment not Cloud Foundry Release from now on
- Bump UAA to v53.3
- Add ability to rename immutable secrets
- Update CATS to be closer to what upstream is using
- Make RBAC the default in the `values.yaml` (no need to specify anymore)
- Increase test brain timeouts to stop randomly failing tests
- Remove unused SANs from the generated TLS certificates
- Remove the dependency on `jq` from stemcells
- Fix duplicate buildpack ids when starting Cloud Foundry
- Fix an issue in the vagrant box where compilation would fail due to old versions of docker.
- Fix an issue where diego cell could not be mounted on NFS-backed Kubernetes storage class
- Fix an issue where diego cell could not mount NFS in persi
- Fix several problems reported with the syslog forwarding implementation

4 1.0 Release, January 2018

- Initial product release

5 Known Issues

! Important

You will need Stratos UI 1.1 when running SUSE Cloud Applications Platform 1.1 and you share the `scf-values.yaml` configuration file between them. Prior versions of the Stratos UI will not work.

! Important

If you have used a configuration file from a version prior to 1.1, you will need to update it. See details below.

5.1 Upgrade Changes

- The variable `kube.external_ip` has now been renamed to `kube.external_ips`, meaning upgrades from older versions will fail unless the latter variable exists in the `scf-values.yaml` file used to deploy SUSE Cloud Applications Platform. Both variables can exist at the same time and be set to the same value for those in mixed version environments:

```
kube.external_ip=1.1.1.1
kube.external_ips=[1.1.1.1]
```

- Going forward, `kube.external_ips` is an array, hence it can be used as reproduced below:

```
kube.external_ips=["1.1.1.1", "2.2.2.2"]
```

- Also as a result of this change, the `helm` command line client must be version 2.6.0 or higher.

- All the secrets have been renamed from `env.F00` to `secrets.F00`, so all the appropriate entries in `scf-values.yaml` need to be modified to align with that change.
- You need to keep specifying **all** your secrets on each upgrade (for example, the `CLUSTER_ADMIN_PASSWORD`) as it will not be carried forward automatically.
- To rotate secrets, increment the `kube.secret_generation_counter`. Note that immutable generated secrets will not be reset.
- In HA environments, upgrades can run into an issue whereby the API pods do not all come up post-migration. The work around this issue, before the upgrade, scale down the API role to 1. After completing the upgrade, scale the API role up again to 2 or more.
 - Some roles (like `diego-api`, `diego-brain` and `routing-api`) are configured as active/passive, so passive pods can appear as `Not Ready`.
 - Other roles (`tcp-router` and `blobstore`) cannot be scaled.
- Cloud Application Platform v1.1 requires that Stratos UI use version 1.1. Older versions of the UI will not work due to the change in variable names.
- Azure operators may not be able to connect to SQL databases with the sidecar.
- Restores performed by the Backup CLI may leave docker apps in a stopped state. The workaround is to restart the affected applications.
- A proper JSON file generated by the Backup CLI needs to be provided in order to do a restore, otherwise an ugly error appears.
- Do not set the `mysql` or `diego_access` roles to more than one instance each in HA configurations. Doing so can cause problems with subsequent upgrades which could lead to loss of data. Scalability of these roles will be enabled in an upcoming maintenance release.
- A `helm upgrade` command from 1.0 to 1.0.1 (scf 2.6.11 to 2.7.0) requires the use of `--force` to drop an unnecessary persistent volume. Note that `helm upgrade` only works for multi-node clusters when running with a proper HA storage class (for example, `hostpath` will not work as required stateful data can be lost).

6 Installing SUSE Cloud Applications Platform on SUSE Containers as a Service Platform

6.1 General

- The port 2793 has to be open on the workers, for terraform in the *sechgroup_worker* resource:

```
# diego-access-public (SCF)
rule {
  from_port = 2222
  to_port   = 2222
  ip_protocol = "tcp"
  cidr       = "0.0.0.0/0"
}

# uaa-public (UAA)
rule {
  from_port = 2793
  to_port   = 2793
  ip_protocol = "tcp"
  cidr       = "0.0.0.0/0"
}

rule {
  from_port = 2793
  to_port   = 2793
  ip_protocol = "udp"
  cidr       = "0.0.0.0/0"
}

# router-public (SCF)
rule {
  from_port = 4443
  to_port   = 4443
  ip_protocol = "tcp"
  cidr       = "0.0.0.0/0"
}

# tcp-router-public (SCF)
rule {
  from_port = 2341
  to_port   = 2341
  ip_protocol = "tcp"
```

```

    cidr      = "0.0.0.0/0"
  }

# tcp-router-public (SCF)
rule {
  from_port  = 20000
  to_port    = 20008
  ip_protocol = "tcp"
  cidr       = "0.0.0.0/0"
}

# stultified-unicorn-ui-next (STRATOS)
rule {
  from_port  = 8443
  to_port    = 30820
  ip_protocol = "tcp"
  cidr       = "0.0.0.0/0"
}

```

- Designate DNS is configured and it supports wildcards, so for example:

```

lc.qa.cloud.caasp.suse.net.    A      10.84.72.127
*.lc.qa.cloud.caasp.suse.net. CNAME  lc.qa.cloud.caasp.suse.net.

```

→ 10.84.72.127 is the floating IP of the Kubernetes worker with the private IP 10.0.6.18

→ The content of scf-config-values.yaml :

```

env:
  CLUSTER_ADMIN_PASSWORD: susetesting
  DOMAIN: lc.qa.cloud.caasp.suse.net
  UAA_ADMIN_CLIENT_SECRET: uaa-admin-client-secret
  UAA_HOST: uaa.lc.qa.cloud.caasp.suse.net
  UAA_PORT: 2793
kube:
  external_ip: 10.0.6.18
  storage_class:
    persistent: persistent

```

6.2 Connecting to a Ceph cluster

- For RBD StorageClass

Only if you do not have or do not want your secrets stored in Kubernetes, /etc/ceph/ceph.conf and /etc/ceph/ceph.client.admin.keyring must be present on the Kubernetes masters so the masters can dynamically create the PVC.

For testing purposes only, I used the admin for both admin/user.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: persistent
provisioner: kubernetes.io/rbd
parameters:
  monitors: 10.0.5.245:6789,10.0.5.250:6789,10.0.5.247:6789
  adminId: admin
  adminSecretName: ceph-secret-admin
  adminSecretNamespace: default
  pool: k8s
  userId: admin
  userSecretName: ceph-secret-admin
```

- In Kubernetes, ceph secret must exist in all namespaces, so I created them before installing the charts:

```
$ kubectl create namespace uaa
$ kubectl create secret generic ceph-secret-admin
--from-file=ceph-client-key --type=kubernetes.io/rbd --namespace=uaa
$ kubectl create namespace scf
$ kubectl create secret generic ceph-secret-admin
--from-file=ceph-client-key --type=kubernetes.io/rbd --namespace=scf
$ kubectl create namespace stratos
$ kubectl create secret generic ceph-secret-admin
--from-file=ceph-client-key --type=kubernetes.io/rbd --namespace=stratos
```

6.3 Running SUSE CaaS Platform on OpenStack

There are OpenStack images of SUSE CaaS Platform which can be used to create a Kubernetes cluster running on top of OpenStack.

6.3.1 Considerations Before Deploying

- This Guide is for setting up development/test installations with one master and two workers. It is intended as a installation example and will not result in a production system.
- This guide is valid for SUSE-CaaS-Platform-2.0 Version 1.0.0-GM. It will not work for older releases and might work for newer ones.
- Provide enough disk space for images



Note

The default is to provide 40 GB of disk space to the SUSE CaaS Platform nodes. This is not sufficient for SUSE Cloud Applications Platform. So use a machine flavor with a bigger disk and let it use the additional free space for storing images. For information about resizing the root file system of the SUSE CaaS Platform node, see [Section 6.3.5.1, “Growing the Root File System”](#).

6.3.2 Initial preparations

The following steps only have to be done once before the initial SUSE CaaS Platform deployment. For following deployments of SUSE CaaS Platform this has no to be redone but already created elements can be reused. The deployment of SUSE CaaS Platform on OpenStack is done using existing Terraform rules with some additional steps required for running SUSE Cloud Applications Platform on the SUSE CaaS Platform deployment.

- Start with downloading and sourcing the openrc.sh file for OpenStack API access

```
$ firefox https://$OPENSTACK/project/access_and_security/api_access/openrc/  
. openrc.sh
```



Note

You need to log in to download the file. The file name might have a prefix named after the OpenStack project the file is for.

6.3.2.1 Optional Steps

These steps can be performed but are not mandatory. You can also use already existing OpenStack objects instead. For example, if you do not have the permission to create projects or networks).

1. Create an OpenStack project to run SUSE CaaS Platform in (for example, caasp), add a user as admin and export the project to be used by Terraform:

```
$ openstack project create --domain default --description "CaaS Platform Project"
caasp
$ openstack role add --project caasp --user admin admin
$ export OS_PROJECT_NAME='caasp'
```

2. Create a OpenStack network plus a subnet for caasp (for example, caasp-net) and add a router to the external (for example, floating) network:

```
$ openstack network create caasp-net
$ openstack subnet create caasp_subnet --network caasp-net --subnet-range
10.0.2.0/24
$ openstack router create caasp-net-router
$ openstack router set caasp-net-router --external-gateway floating
$ openstack router add subnet caasp-net-router caasp_subnet
```

6.3.2.2 Mandatory Steps

The following steps have to be performed at least once to be able to deploy SUSE CaaS Platform and SUSE Cloud Applications Platform on top of OpenStack.

1. Download the SUSE-CaaS-Platform-2.0-OpenStack-Cloud.x86_64-1.0.0-GM.qcow2 image from link:<https://download.suse.com/Download?buildid=tW8sXCIHrWE~> (SUSE Customer Center account required).
2. Upload the SUSE CaaS Platform 2 image to OpenStack:

```
$ openstack image create \
--file SUSE-CaaS-Platform-2.0-OpenStack-Cloud.x86_64-1.0.0-GM.qcow2 \
SUSE-CaaS-Platform-2.0-GM
```

3. Create a additional security group with rules needed for SUSE Cloud Applications Platform

```
$ openstack security group create cap --description "Allow CAP traffic"
```

```
$ openstack security group rule create cap --protocol any --dst-port any --ethertype
IPv4 --egress
$ openstack security group rule create cap --protocol any --dst-port any --ethertype
IPv6 --egress
$ openstack security group rule create cap --protocol tcp --dst-port 20000:20008 --
remote-ip 0.0.0.0/0
$ openstack security group rule create cap --protocol tcp --dst-port 443:443 --
remote-ip 0.0.0.0/0
$ openstack security group rule create cap --protocol tcp --dst-port 2793:2793 --
remote-ip 0.0.0.0/0
$ openstack security group rule create cap --protocol tcp --dst-port 4443:4443 --
remote-ip 0.0.0.0/0
$ openstack security group rule create cap --protocol tcp --dst-port 80:80 --remote-
ip 0.0.0.0/0
$ openstack security group rule create cap --protocol tcp --dst-port 2222:2222 --
remote-ip 0.0.0.0/0
```

4. Clone the Terraform script

```
$ git clone git@github.com:kubic-project/automation.git
$ cd automation/caasp-openstack-terraform
```

5. Edit `openstack.tfvars`. Use the names of the just created OpenStack objects, for exam- ple:

```
image_name = "SUSE-CaaS-Platform-2.0-GM"
internal_net = "caasp-net"
external_net = "floating"
admin_size = "m1.large"
master_size = "m1.large"
masters = 1
worker_size = "m1.xlarge"
workers = 2
```

6. Initialize Terraform:

```
$ terraform init
```

6.3.3 Deploy CaaS Platform

- Source the `openrc.sh` file, set the project and deploy

```
$ . openrc.sh
$ export OS_PROJECT_NAME='caasp'
```

```
$ ./caasp-openstack apply
```

- Wait for 5 - 10 minutes until all systems are up and running
- Get an overview of your CaaS Platform installation

```
$ openstack server list
```

- Add the initial created cap security group to all SUSE Cloud Applications Platform workers

```
$ openstack server add security group caasp-worker0 cap  
$ openstack server add security group caasp-worker1 cap
```

- Access to CaaS Platform nodes

For SUSE Cloud Applications Platform you might have to log into the CaaS Platform master and nodes. To do so, use ssh with the ssh key in the automation/caasp-openstack-terraform/ssh dir to login as root.

6.3.4 Bootstrap CaaS Platform

1. Point your browser at the IP of the SUSE CaaS Platform admin node
2. Create a new admin user
3. On *Initial CaaS Platform Configuration*
4. *Admin node*: Replace the initial value (*public/floating IP*) with internal OpenStack SUSE CaaS Platform subnet IP of the SUSE CaaS Platform admin node
5. Enable the *Install Tiller* checkbox.
6. On Bootstrap your CaaS Platform
7. Click *Next*
8. On *Select nodes and roles*
9. Click *Accept All nodes* and wait until they appear in the upper part of the page
10. Define master and nodes
11. Click *Next*

12. On *Confirm bootstrap*

13. *External Kubernetes API FQDN*: Specify the public (floating) IP from the SUSE CaaS Platform master and add the .xip.io domain suffix

14. *External Dashboard FQDN*: Specify the public (floating) IP from the SUSE CaaS Platform admin and add the .xip.io domain suffix

6.3.5 Prepare SUSE CaaS Platform for SUSE Cloud Applications Platform



Note

You can run commands on multiple nodes using Salt on the admin node.

Access it by logging in to the admin node and then enter the Salt master container:

```
$ docker exec -ti `docker ps -q --filter name=salt-master` /bin/bash
```

There you can execute commands using Salt. For executing the same command on all worker nodes use a command like:

```
$ salt -P "roles:(kube-minion)" cmd.run 'echo "hello"'
```

This gets you full access to all aspects of the nodes so be careful with what commands you run.

6.3.5.1 Growing the Root File System

Commands to run on the SUSE CaaS Platform worker nodes

Resize your root file system of the worker to match the disk provided by OpenStack:

```
$ growpart /dev/vda 3  
$ btrfs filesystem resize max /.snapshots
```

6.3.5.2 Setting Up Hostpath As the Storage Class



Warning

Setting Hostpath as the storage class is useful in demonstration environments only. It is not usable in a production environment. For example, SUSE Cloud Applications Platform cannot be updated with a Hostpath setup.

- Commands to run on the SUSE CaaS Platform master:

First edit `/etc/kubernetes/controller-manager` and add the `--enable-host-path-provisioner` option there.

Then run the following commands:

```
$ mkdir -p /tmp/hostpath_pv
$ chmod a+rwX /tmp/hostpath_pv
$ systemctl restart kube-controller-manager.service
```

- Commands to run on the SUSE CaaS Platform worker nodes

```
$ mkdir -p /tmp/hostpath_pv
$ chmod a+rwX /tmp/hostpath_pv
```

7 Installing SUSE Cloud Applications Platform on Microsoft Azure

SUSE Cloud Applications Platform 1.1 supports running on [Microsoft Azure Container Service \(AKS\)](https://azure.microsoft.com/en-us/services/container-service/) (<https://azure.microsoft.com/en-us/services/container-service/>). This document describes how to prepare the AKS environment for deployment of SUSE Cloud Applications Platform. For more detailed product information and instructions, see the SUSE Cloud Applications Platform documentation at <https://www.suse.com/documentation/cloud-application-platform-1>.

7.1 Requirements

To setup the Microsoft Azure environment, use the command line client `az`. On SUSE systems, it can be installed by executing the following commands:

```
$ sudo zypper install -y curl
$ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
$ sudo sh -c 'echo -e "[azure-cli]\nname=Azure CLI\nbaseurl=https://
packages.microsoft.com/yumrepos/azure-cli\nenabled=1\nntype=rpm-md
\npgpcheck=1\npgpkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/zypp/
repos.d/azure-cli.repo'
$ sudo zypper refresh
$ sudo zypper install -y azure-cli
```

Then log in to Microsoft Azure:

```
$ az login
```

7.2 Setting Up Kubernetes

1. Extract the subscriptions ID and write it down:

```
$ az account show --query "{ subscription_id: id }"
```

2. Create a service principal and write down the *password* and *appId*. The term *<subscription-id>* needs to be replaced by the *subscription_id* noted in the in the previous step. If the Microsoft Azure service is supposed to be located in a data center other than a Western USA data center *westus* needs to be replaced.

```
$ export SUBSCRIPTION_ID=<subscription-id>
$ az account set --subscription $SUBSCRIPTION_ID
$ az group create --name scf-resource-group --location westus
$ az ad sp create-for-rbac --role Contributor --scopes "/subscriptions/
$SUBSCRIPTION_ID/resourceGroups/scf-resource-group"
```

3. Create a container service in Microsoft Azure and replace the terms *<password>* and *<application-id>* with the one noted from the previous step. Additionally, the phrase *<your-public-ssh-key>* should be replaced by the public SSH key to be used:

```
$ az acs create \
  --name scf-container-service \
  --resource-group scf-resource-group \
  --orchestrator-type Kubernetes \
  --dns-prefix "myscf" \
  --master-count 1 \
  --admin-username scf-admin \
  --agent-count 3 \
  --client-secret <password> \
```

```
--ssh-key-value "<your-public-ssh-key>" \
--service-principal <application-id> \
--master-vm-size Standard_D2_v2
```

4. Prepare kubernetes environment. If there is already another running Kubernetes it is advised to backup the old configuration located in `~/.kube/`.

```
$ az acs kubernetes get-credentials --resource-group="scf-resource-group" --
name="scf-container-service"
```

The above command can fail if the SSH key is password protected. Adding the `<private_key_file_name>` to the `ssh-agent` will fix the issue:

```
$ ssh-add ~/.ssh/<private_key_file_name>
```

5. The Kubernetes config can be verified by listing all the current pods. The minimal required Kubernetes version is 1.6.

```
$ kubectl get pods --all-namespaces
```

- Enable cgroup swap accounting by running the following commands. This will reboot some of the machines which takes some time. This also does require the commands `sed` and `jq`.

```
$ sudo zypper in -y jq sed
$ az vm list -g "scf-resource-group" | jq '[] | select (.tags.poolName |
contains("agent")) | .name' | \
  xargs -i{} az vm run-command invoke \
    --resource-group "scf-resource-group" \
    --command-id RunShellScript \
    --scripts "sudo sed -i 's/GRUB_CMDLINE_LINUX_DEFAULT=\"console=tty1
console=ttyS0 earlyprintk=ttyS0 rootdelay=300\"/GRUB_CMDLINE_LINUX_DEFAULT=
\"console=tty1 console=ttyS0 earlyprintk=ttyS0 rootdelay=300 swapaccount=1\"/'
g' /etc/default/grub.d/50-cloudimg-settings.cfg" --name {}

$ az vm list -g "scf-resource-group" | jq '[] | select (.tags.poolName |
contains("agent")) | .name' | \
  xargs -i{} az vm run-command invoke \
    --resource-group "scf-resource-group" \
    --command-id RunShellScript \
    --scripts "sudo update-grub" --name {}

$ az vm list -g "scf-resource-group" | jq '[] | select (.tags.poolName |
contains("agent")) | .name' | \
```

```
xargs -i{} az vm restart --no-wait \
  --resource-group "scf-resource-group" \
  --name {}
```

6. Create a new public IP address and write it down. It is needed for the later deployment.

```
$ az network public-ip create --resource-group scf-resource-group --name scf-public-
ip --allocation-method Static
```

7. Extract the name of one of the kubes agent NIC and write it down

```
$ az network nic list --resource-group scf-resource-group | grep name | grep agent |
grep 0
```

8. Attach the public IP to the master kubes node by replacing the term *<nic-name>* by the one noted down before and write down the value of *privateIpAddress*. This private IP address is needed for the later deployment.

```
$ az network nic ip-config update --resource-group scf-resource-group --nic-name
<nic-name> --name ipconfig1 --public-ip-address scf-public-ip
```

9. Extract the security group name of the master network security group and write it down:

```
$ az network nsg list --resource-group=scf-resource-group | jq -r '[] | .name' |
grep master
```

10. Create required security groups and replace *<security-group>* with the name you wrote down in the previous step:

```
$ export NSG_NAME=<security-group>
$ az network nsg rule create --resource-group scf-resource-group --priority 200 --
nsg-name $NSG_NAME --name scf-80 --direction Inbound --destination-port-ranges 80 --
access Allow
$ az network nsg rule create --resource-group scf-resource-group --priority 201 --
nsg-name $NSG_NAME --name scf-443 --direction Inbound --destination-port-ranges 443
--access Allow
$ az network nsg rule create --resource-group scf-resource-group --priority 202 --
nsg-name $NSG_NAME --name scf-4443 --direction Inbound --destination-port-ranges
4443 --access Allow
$ az network nsg rule create --resource-group scf-resource-group --priority 203 --
nsg-name $NSG_NAME --name scf-2222 --direction Inbound --destination-port-ranges
2222 --access Allow
$ az network nsg rule create --resource-group scf-resource-group --priority 204 --
nsg-name $NSG_NAME --name scf-2793 --direction Inbound --destination-port-ranges
2793 --access Allow
```

7.3 Deploying SUSE Cloud Applications Platform

For the next steps the public and private IP addresses written down before are required.

When the Kubernetes environment is prepared, proceed with deployment of SUSE Cloud Applications Platform as described in the *Deployment Guide* at https://www.suse.com/documentation/cloud-application-platform-1/book_cap_deployment/data/book_cap_deployment.html.

NOTE: When deploying to Microsoft Azure, you will need to set the Garden rootfs driver to overlay-xfs (the default would be btrfs). You can do that by setting the following key in the scf-config-values.yaml:

+

```
env:
  GARDEN_ROOTFS_DRIVER: "overlay-xfs"
```

8 Service Broker Setup

8.1 Setting up and Using a Service Broker Sidecar

We currently provide helm charts for two service brokers managing access to MySQL (see [Section 8.3, “Deploying the MySQL chart”](#)) and PostgreSQL ([Section 8.6, “Deploying the PostgreSQL chart”](#)) databases.

This document describes how to use these charts in the context of a SUSE Cloud Applications Platform cluster.

8.2 Prerequisites

- Helm must be configured. For more information, see the [Helm documentation \(https://docs.helm.sh/using_helm/#quickstart\)](https://docs.helm.sh/using_helm/#quickstart).
- A working SUSE Cloud Applications Platform deployment.
- External databases must be reachable from the running applications. For more information, see the [CloudFoundry Documentation regarding application security groups \(http://docs.cloudfoundry.org/concepts/asg.html\)](http://docs.cloudfoundry.org/concepts/asg.html).

8.3 Deploying the MySQL chart

You need an external MySQL installation, with account credentials that allow creating and deleting both databases and users.

8.4 Configuring the Deployment

Create a `values.yaml` file (the rest of the document assumes it is called `usb-config-values.yaml`) with the settings required for the install. Use the file below as a template, and modify the values to suit your installation.

```
env:
  # Database access credentials; the given user must have privileges to create
  # and delete both databases and users
  SERVICE_MYSQL_HOST: mysql.example.com
  SERVICE_MYSQL_PORT: 3306
  SERVICE_MYSQL_USER: AzureDiamond
  SERVICE_MYSQL_PASS: hunter2

  # CAP access credentials
  CF_ADMIN_USER: admin
  CF_ADMIN_PASSWORD: changeme
  CF_DOMAIN: example.com

  # CAP internal certificate authorities
  # CF_CA_CERT can be obtained via the command line:
  #   kubectl get secret -n $NAMESPACE secrets-$REVISION -o jsonpath='{.data.internal-ca-
cert}' | base64 -d
  # Where $NAMESPACE is the namespace CAP was deployed in, and $REVISION is the helm
revision number
  CF_CA_CERT: |
    -----BEGIN CERTIFICATE-----
    MIIESGVsbG8gdGhlcmUgdGhlcmUgaXMgbm8gc2VjcmlV0IG1lc3NhZ2UsIHNvcnJ5Cg==
    -----END CERTIFICATE-----

  # UAA_CA_CERT can be obtained with the command line:
  #   kubectl get secret -n $NAMESPACE secret-$REVISION -o jsonpath='{.data.internal-ca-
cert}' | base64 -d
  UAA_CA_CERT: |
    -----BEGIN CERTIFICATE-----
    MIIETm8gcmlVhbGx5IEkgc2FpZCB0aGVyZSBpcyBubyBzZWNyZXQgbWVzc2FnZSEhCg==
    -----END CERTIFICATE-----

  SERVICE_TYPE: mysql # Optional
```

```
# The whole "kube" section is optional
kube:
  organization: library # Docker registry organization
  registry:          # Docker registry access configuration
    hostname: registry.example.com
    username: AzureDiamond
    password: hunter2
```

8.5 Deploy the Chart

When deploying the chart, a Kubernetes namespace to install the sidecar to is required. It may optionally be the same namespace as SUSE Cloud Applications Platform is installed to, though only one MySQL service may be deployed into a namespace at a time.

1. Ensure that you have the SUSE helm chart repository available:

```
$ helm repo add suse https://kubernetes-charts.suse.com/
```

1. Install the helm chart:

```
$ SIDECAR_NAMESPACE=my_sidecar
$ helm install suse/cf-usb-sidecar-mysql \
  --devel \
  --namespace ${SIDECAR_NAMESPACE} \
  --set "env.SERVICE_LOCATION=http://cf-usb-sidecar-mysql.
${SIDECAR_NAMESPACE}:8081" \
  --values usb-config-values.yaml \
```



```
--wait
```

2. Wait for all the pods to be ready (Press Ctrl+C once all the pods are shown as fully ready):

```
$ watch kubectl get pods --namespace=${SIDECAR_NAMESPACE}
```

3. Confirm that the service has been added to your SUSE Cloud Applications Platform installation:

```
$ cf marketplace
```

8.5.1 Using the Service

To create a new service instance, use the Cloud Foundry command line client:

```
$ cf create-service mysql default service_instance_name
```

The last argument, service_instance_name, is the desired name of the service instance.

To bind the service instance to an application, use the bind-service subcommand:

```
$ cf bind-service my_application my_service_instance_name
```

8.6 Deploying the PostgreSQL chart

The PostgreSQL configuration is slightly different from the MySQL configuration; the database-specific keys are named differently, and an additional key is introduced:

```
env:
  # Database access credentials; the given user must have privileges to create
  # delete both databases and users
  SERVICE_POSTGRESQL_HOST: postgres.example.com
  SERVICE_POSTGRESQL_PORT: 5432
  SERVICE_POSTGRESQL_USER: AzureDiamond
  SERVICE_POSTGRESQL_PASS: hunter2
  # The SSL connection mode when connecting to the database. For a list of
  # valid values, please see https://godoc.org/github.com/lib/pq
  SERVICE_POSTGRESQL_SSLMODE: disable
```

```

# CAP access credentials
CF_ADMIN_USER: admin
CF_ADMIN_PASSWORD: changeme
CF_DOMAIN: example.com

# CAP internal certificate authorities
# CF_CA_CERT can be obtained via the command line:
#   kubectl get secret -n $NAMESPACE secret-$REVISION \
#     -o jsonpath='{.data.internal-ca-cert}' | base64 -d
# Where $NAMESPACE is the namespace CAP was deployed in, and $REVISION is
# the helm revision number
CF_CA_CERT: |
  -----BEGIN CERTIFICATE-----
  MIIESGVsbG8gdGhlcmUgdGhlcmUgaXMgbm8gc2VjcmV0IG1lc3NhZ2UsIHNvcnJ5Cg==
  -----END CERTIFICATE-----

# UAA_CA_CERT can be obtained with the command line:
#   kubectl get secret -n $NAMESPACE secret-$REVISION \
#     -o jsonpath='{.data.internal-ca-cert}' | base64 -d
UAA_CA_CERT: |
  -----BEGIN CERTIFICATE-----
  MIIETm8gcVhbGx5IEkgc2FpZCB0aGVyZSBpcyBubyBzZWNyZXQgbWVzc2FnZSEhCg==
  -----END CERTIFICATE-----

SERVICE_TYPE: postgres # Optional

# The whole "kube" section is optional
kube:
  organization: library # Docker registry organization
  registry:          # Docker registry access configuration
  hostname: registry.example.com
  username: AzureDiamond
  password: hunter2

```

The command to install the Helm chart is also different in that it needs a different host name for the service location:

```

SIDECAR_NAMESPACE=psql_sidecar
$ helm install suse/cf-usb-sidecar-postgres \
  --devel \
  --namespace ${SIDECAR_NAMESPACE} \
  --set "env.SERVICE_LOCATION=http://cf-usb-sidecar-postgres.${SIDECAR_NAMESPACE}:8081" \
  --values usb-config-values.yaml \
  --wait

```

8.7 Removing Service Broker Sidecar Deployments

To correctly remove sidecar deployments, take the following actions in order:

1. Unbind any applications using instances of the service, and delete those instances:

```
$ cf unbind-service my_app my_service_instance
$ cf delete-service my_service_instance
```

1. Install the CF-USB CLI plugin (<https://github.com/SUSE/cf-usb-plugin/>)  for the Cloud Foundry CLI (<https://github.com/cloudfoundry/cli/>) :

```
$cf install-plugin \
  https://github.com/SUSE/cf-usb-plugin/releases/download/1.0.0/cf-usb-
  plugin-1.0.0.0.g47b49cd-linux-amd64
```

2. Configure the Cloud Foundry USB CLI plugin:

```
$ cf usb-target https://usb.${DOMAIN}
```

3. Remove the services:

```
# See `env.SERVICE_LOCATION` configuration value when deploying the helm chart.
$ cf usb delete-driver-endpoint "http://cf-usb-sidecar-mysql.
${SIDECAR_NAMESPACE}:8081"
```

4. Delete Helm release from Kubernetes:

```
$ helm list # Find the name of the helm deployment
$ helm delete --purge ...
```

9 Appendix: Configuration with secrets : Section

```
---
env:
  # List of domains (including scheme) from which Cross-Origin requests will be
  # accepted, a * can be used as a wildcard for any part of a domain.
  ALLOWED_CORS_DOMAINS: "[]"

  # Allow users to change the value of the app-level allow_ssh attribute.
  ALLOW_APP_SSH_ACCESS: "true"

  # Extra token expiry time while uploading big apps, in seconds.
```

```

APP_TOKEN_UPLOAD_GRACE_PERIOD: "1200"

# List of allow / deny rules for the blobstore internal server. Will be
# followed by 'deny all'. Each entry must be follow by a semicolon.
BLOBSTORE_ACCESS_RULES: "allow 10.0.0.0/8; allow 172.16.0.0/12; allow 192.168.0.0/16;"

# Maximal allowed file size for upload to blobstore, in megabytes.
BLOBSTORE_MAX_UPLOAD_SIZE: "5000"

# The set of CAT test suites to run. If not specified it falls back to a
# hardwired set of suites.
CATS_SUITES: ~

# URI for a CDN to use for buildpack downloads.
CDN_URI: ""

# The OAuth2 authorities available to the cluster administrator.
CLUSTER_ADMIN_AUTHORITIES:
"scim.write,scim.read,openid,cloud_controller.admin,clients.read,clients.write,doppler.firehose,routin

# 'build' attribute in the /v2/info endpoint
CLUSTER_BUILD: "2.0.2"

# 'description' attribute in the /v2/info endpoint
CLUSTER_DESCRIPTION: "SUSE Cloud Foundry"

# 'name' attribute in the /v2/info endpoint
CLUSTER_NAME: "SCF"

# 'version' attribute in the /v2/info endpoint
CLUSTER_VERSION: "2"

# The standard amount of disk (in MB) given to an application when not
# overridden by the user via manifest, command line, etc.
DEFAULT_APP_DISK_IN_MB: "1024"

# The standard amount of memory (in MB) given to an application when not
# overridden by the user via manifest, command line, etc.
DEFAULT_APP_MEMORY: "1024"

# If set apps pushed to spaces that allow SSH access will have SSH enabled by
# default.
DEFAULT_APP_SSH_ACCESS: "true"

# The default stack to use if no custom stack is specified by an app.
DEFAULT_STACK: "sle12"

```

```

# The container disk capacity the cell should manage. If this capacity is
# larger than the actual disk quota of the cell component, over-provisioning
# will occur.
DIEGO_CELL_DISK_CAPACITY_MB: "auto"

# The memory capacity the cell should manage. If this capacity is larger than
# the actual memory of the cell component, over-provisioning will occur.
DIEGO_CELL_MEMORY_CAPACITY_MB: "auto"

# Maximum network transmission unit length in bytes for application
# containers.
DIEGO_CELL_NETWORK_MTU: "1400"

# A CIDR subnet mask specifying the range of subnets available to be assigned
# to containers.
DIEGO_CELL_SUBNET: "10.38.0.0/16"

# Disable external buildpacks. Only admin buildpacks and system buildpacks
# will be available to users.
DISABLE_CUSTOM_BUILDPACKS: "false"

# The host to ping for confirmation of DNS resolution.
DNS_HEALTH_CHECK_HOST: "127.0.0.1"

# Base domain of the SCF cluster.
# Example: my-scf-cluster.com
DOMAIN: ~

# The number of versions of an application to keep. You will be able to
# rollback to this amount of versions.
DROPLET_MAX_STAGED_STORED: "5"

# Enables setting the X-Forwarded-Proto header if SSL termination happened
# upstream and the header value was set incorrectly. When this property is set
# to true, the gorouter sets the header X-Forwarded-Proto to https. When this
# value set to false, the gorouter sets the header X-Forwarded-Proto to the
# protocol of the incoming request.
FORCE_FORWARDED_PROTO_AS_HTTPS: "false"

# URL pointing to the Docker registry used for fetching Docker images. If not
# set, the Docker service default is used.
GARDEN_DOCKER_REGISTRY: "registry-1.docker.io"

# Whitelist of IP:PORT tuples and CIDR subnet masks. Pulling from docker
# registries with self signed certificates will not be permitted if the
# registry's address is not listed here.
GARDEN_INSECURE_DOCKER_REGISTRIES: ""

```

```

# Override DNS servers to be used in containers; defaults to the same as the
# host.
GARDEN_LINUX_DNS_SERVER: ""

# The filesystem driver to use (btrfs or overlay-xfs).
GARDEN_ROOTFS_DRIVER: "btrfs"

# Location of the proxy to use for secure web access.
HTTPS_PROXY: ~

# Location of the proxy to use for regular web access.
HTTP_PROXY: ~

KUBE_SERVICE_DOMAIN_SUFFIX: ~

# The cluster's log level: off, fatal, error, warn, info, debug, debug1,
# debug2.
LOG_LEVEL: "info"

# The maximum amount of disk a user can request for an application via
# manifest, command line, etc., in MB. See also DEFAULT_APP_DISK_IN_MB for the
# standard amount.
MAX_APP_DISK_IN_MB: "2048"

# Maximum health check timeout that can be set for an app, in seconds.
MAX_HEALTH_CHECK_TIMEOUT: "180"

# Sets the maximum allowed size of the client request body, specified in the
# "Content-Length" request header field, in megabytes. If the size in a
# request exceeds the configured value, the 413 (Request Entity Too Large)
# error is returned to the client. Please be aware that browsers cannot
# correctly display this error. Setting size to 0 disables checking of client
# request body size. This limits application uploads, buildpack uploads, etc.
NGINX_MAX_REQUEST_BODY_SIZE: "2048"

# Comma separated list of IP addresses and domains which should not be
# directed through a proxy, if any.
NO_PROXY: ~

# Comma separated list of white-listed options that may be set during create
# or bind operations.
# Example:
# uid,gid,allow_root,allow_other,nfs_uid,nfs_gid,auto_cache,fsname,username,password
PERSI_NFS_ALLOWED_OPTIONS: "uid,gid,auto_cache,username,password"

# Comma separated list of default values for nfs mount options. If a default

```

```

# is specified with an option not included in PERSI_NFS_ALLOWED_OPTIONS, then
# this default value will be set and it won't be overridable.
PERSI_NFS_DEFAULT_OPTIONS: ~

# Comma separated list of white-listed options that may be accepted in the
# mount_config options. Note a specific 'sloppy_mount:true' volume option
# tells the driver to ignore non-white-listed options, while a
# 'sloppy_mount:false' tells the driver to fail fast instead when receiving a
# non-white-listed option."
#
# Example:
# allow_root,allow_other,nfs_uid,nfs_gid,auto_cache,sloppy_mount,fsname
PERSI_NFS_DRIVER_ALLOWED_IN_MOUNT: "auto_cache"

# Comma separated list of white-listed options that may be configured in
# supported in the mount_config.source URL query params
#
# Example: uid,gid,auto-traverse-mounts,dircache
PERSI_NFS_DRIVER_ALLOWED_IN_SOURCE: "uid,gid"

# Comma separated list default values for options that may be configured in
# the mount_config options, formatted as 'option:default'. If an option is not
# specified in the volume mount, or the option is not white-listed, then the
# specified default value will be used instead.
#
# Example:
# allow_root:false,nfs_uid:2000,nfs_gid:2000,auto_cache:true,sloppy_mount:true
PERSI_NFS_DRIVER_DEFAULT_IN_MOUNT: "auto_cache:true"

# Comma separated list of default values for options in the source URL query
# params, formatted as 'option:default'. If an option is not specified in the
# volume mount, or the option is not white-listed, then the specified default
# value will be applied.
PERSI_NFS_DRIVER_DEFAULT_IN_SOURCE: ~

# Disable Persi NFS driver
PERSI_NFS_DRIVER_DISABLE: "false"

# LDAP server host name or ip address (required for LDAP integration only)
PERSI_NFS_DRIVER_LDAP_HOST: ""

# LDAP server port (required for LDAP integration only)
PERSI_NFS_DRIVER_LDAP_PORT: "389"

# LDAP server protocol (required for LDAP integration only)
PERSI_NFS_DRIVER_LDAP_PROTOCOL: "tcp"

```

```

# LDAP service account user name (required for LDAP integration only)
PERSI_NFS_DRIVER_LDAP_USER: ""

# LDAP fqdn for user records we will search against when looking up user uids
# (required for LDAP integration only)
# Example: cn=Users,dc=corp,dc=test,dc=com
PERSI_NFS_DRIVER_LDAP_USER_FQDN: ""

# Certificates to add to the rootfs trust store. Multiple certs are possible by
# concatenating their definitions into one big block of text.
ROOTFS_TRUSTED_CERTS: ""

# The algorithm used by the router to distribute requests for a route across
# backends. Supported values are round-robin and least-connection.
ROUTER_BALANCING_ALGORITHM: "round-robin"

# The log destination to talk to. This has to point to a syslog server.
SCF_LOG_HOST: ~

# The port used by rsyslog to talk to the log destination. If not set it
# defaults to 514, the standard port of syslog.
SCF_LOG_PORT: ~

# The protocol used by rsyslog to talk to the log destination. The allowed
# values are tcp, and udp. The default is tcp.
SCF_LOG_PROTOCOL: "tcp"

# A comma-separated list of insecure Docker registries in the form of
# '<HOSTNAME|IP>:PORT'. Each registry must be quoted separately.
#
# Example: "docker-registry.example.com:80", "hello.example.org:443"
STAGER_INSECURE_DOCKER_REGISTRIES: ""

# Timeout for staging an app, in seconds.
STAGING_TIMEOUT: "900"

# Support contact information for the cluster
SUPPORT_ADDRESS: "support@example.com"

# TCP routing domain of the SCF cluster; only used for testing;
# Example: tcp.my-scf-cluster.com
TCP_DOMAIN: ~

# Concatenation of trusted CA certificates to be made available on the cell.
TRUSTED_CERTS: ~

# The host name of the UAA server (root zone)

```



```

UAA_HOST: ~

# The tcp port the UAA server (root zone) listens on for requests.
UAA_PORT: "2793"

# Whether or not to use privileged containers for buildpack based
# applications. Containers with a docker-image-based rootfs will continue to
# always be unprivileged.
USE_DIEGO_PRIVILEGED_CONTAINERS: "false"

# Whether or not to use privileged containers for staging tasks.
USE_STAGER_PRIVILEGED_CONTAINERS: "false"

sizing:
  # Flag to activate high-availability mode
  HA: false

  # The api role contains the following jobs:
  #
  # - global-properties: Dummy BOSH job used to host global parameters that are
  #   required to configure SCF
  #
  # - authorize-internal-ca: Install both internal and UAA CA certificates
  #
  # - patch-properties: Dummy BOSH job used to host parameters that are used in
  #   SCF patches for upstream bugs
  #
  # - cloud_controller_ng: The Cloud Controller provides primary Cloud Foundry
  #   API that is by the CF CLI. The Cloud Controller uses a database to keep
  #   tables for organizations, spaces, apps, services, service instances, user
  #   roles, and more. Typically multiple instances of Cloud Controller are load
  #   balanced.
  #
  # - route_registrar: Used for registering routes
  #
  # Also: metron_agent, statsd_injector, go-buildpack, binary-buildpack,
  # nodejs-buildpack, ruby-buildpack, php-buildpack, python-buildpack,
  # staticfile-buildpack, java-buildpack, dotnet-core-buildpack
  api:
    # Node affinity rules can be specified here
    affinity: {}

    # The api role can scale between 1 and 65535 instances.
    # For high availability it needs at least 2 instances.
    count: 1

    # Unit [millicore]

```

```

cpu:
  request: 4000
  limit: ~

# Unit [MiB]
memory:
  request: 2421
  limit: ~

# The blobstore role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - route_registrar: Used for registering routes
#
# Also: blobstore, metron_agent
blobstore:
  # Node affinity rules can be specified here
  affinity: {}

# The blobstore role cannot be scaled.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

disk_sizes:
  blobstore_data: 50

# Unit [MiB]
memory:
  request: 420
  limit: ~

# The cc-clock role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - cloud_controller_clock: The Cloud Controller clock periodically schedules
#   Cloud Controller clean up tasks for app usage events, audit events, failed
#   jobs, and more. Only single instance of this job is necessary.

```

```

#
# Also: metron_agent, statsd_injector
cc_clock:
  # Node affinity rules can be specified here
  affinity: {}

  # The cc-clock role cannot be scaled.
  count: 1

  # Unit [millicore]
  cpu:
    request: 2000
    limit: ~

  # Unit [MiB]
  memory:
    request: 789
    limit: ~

# The cc-uploader role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: tps, cc_uploader, metron_agent
cc_uploader:
  # Node affinity rules can be specified here
  affinity: {}

  # The cc-uploader role can scale between 1 and 3 instances.
  # For high availability it needs at least 2 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 4000
    limit: ~

  # Unit [MiB]
  memory:
    request: 129
    limit: ~

# The cc-worker role contains the following jobs:
#

```

```

# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - cloud_controller_worker: Cloud Controller worker processes background
#   tasks submitted via the.
#
# Also: metron_agent
cc_worker:
  # Node affinity rules can be specified here
  affinity: {}

  # The cc-worker role can scale between 1 and 65535 instances.
  # For high availability it needs at least 2 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 2000
    limit: ~

  # Unit [MiB]
  memory:
    request: 753
    limit: ~

# The cf-usb role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - route_registrar: Used for registering routes
#
# Also: cf-usb
cf_usb:
  # Node affinity rules can be specified here
  affinity: {}

  # The cf-usb role can scale between 1 and 3 instances.
  # For high availability it needs at least 2 instances.
  count: 1

  # Unit [millicore]
  cpu:

```

```

    request: 2000
    limit: ~

# Unit [MiB]
memory:
    request: 117
    limit: ~

# Global CPU configuration
cpu:
    # Flag to activate cpu requests
    requests: false

    # Flag to activate cpu limits
    limits: false

# The diego-access role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: ssh_proxy, metron_agent, file_server
diego_access:
    # Node affinity rules can be specified here
    affinity: {}

# The diego-access role can scale between 1 and 3 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
    request: 2000
    limit: ~

# Unit [MiB]
memory:
    request: 123
    limit: ~

# The diego-api role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#

```

```

# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: bbs, metron_agent
diego_api:
  # Node affinity rules can be specified here
  affinity: {}

  # The diego-api role can scale between 1 and 3 instances.
  # The instance count must be an odd number (not divisible by 2).
  # For high availability it needs at least 3 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 2000
    limit: ~

  # Unit [MiB]
  memory:
    request: 138
    limit: ~

# The diego-brain role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: auctioneer, metron_agent
diego_brain:
  # Node affinity rules can be specified here
  affinity: {}

  # The diego-brain role can scale between 1 and 3 instances.
  # For high availability it needs at least 2 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 4000
    limit: ~

  # Unit [MiB]
  memory:
    request: 99
    limit: ~

```

```

# The diego-cell role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - wait-for-uaa: Wait for UAA to be ready before starting any jobs
#
# Also: rep, route_emitter, garden, cflinuxfs2-rootfs-setup,
# opensuse42-rootfs-setup, cf-sle12-setup, metron_agent, nfsv3driver
diego_cell:
  # Node affinity rules can be specified here
  affinity: {}

  # The diego-cell role can scale between 1 and 254 instances.
  # For high availability it needs at least 3 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 4000
    limit: ~

  disk_sizes:
    grootfs_data: 50

  # Unit [MiB]
  memory:
    request: 4677
    limit: ~

# The diego-locket role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: locket, metron_agent
diego_locket:
  # Node affinity rules can be specified here
  affinity: {}

  # The diego-locket role can scale between 1 and 3 instances.
  # For high availability it needs at least 3 instances.

```

```

count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 90
  limit: ~

# The doppler role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# Also: doppler, metron_agent
doppler:
  # Node affinity rules can be specified here
  affinity: {}

# The doppler role can scale between 1 and 65535 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 390
  limit: ~

# The loggregator role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - route_registrar: Used for registering routes
#
# Also: loggregator_trafficcontroller, metron_agent
loggregator:

```



```

# Node affinity rules can be specified here
affinity: {}

# The loggregator role can scale between 1 and 65535 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 153
  limit: ~

# Global memory configuration
memory:
  # Flag to activate memory requests
  requests: false

  # Flag to activate memory limits
  limits: false

# The mysql role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - patch-properties: Dummy BOSH job used to host parameters that are used in
#   SCF patches for upstream bugs
#
# Also: mysql
mysql:
  # Node affinity rules can be specified here
  affinity: {}

  # The mysql role can scale between 1 and 3 instances.
  # For high availability it needs at least 2 instances.
  count: 1

  # Unit [millicore]
  cpu:
    request: 2000
    limit: ~

```

```

disk_sizes:
  mysql_data: 20

# Unit [MiB]
memory:
  request: 2841
  limit: ~

# The mysql-proxy role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - patch-properties: Dummy BOSH job used to host parameters that are used in
#   SCF patches for upstream bugs
#
# Also: proxy
mysql_proxy:
  # Node affinity rules can be specified here
  affinity: {}

# The mysql-proxy role can scale between 1 and 3 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 63
  limit: ~

# The nats role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - nats: The NATS server provides publish-subscribe messaging system for the
#   Cloud Controller, the DEA , HM9000, and other Cloud Foundry components.
#
# Also: metron_agent
nats:
  # Node affinity rules can be specified here
  affinity: {}

```

```

# The nats role can scale between 1 and 3 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 60
  limit: ~

# The nfs-broker role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: metron_agent, nfsbroker
nfs_broker:
  # Node affinity rules can be specified here
  affinity: {}

# The nfs-broker role can scale between 1 and 3 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 63
  limit: ~

# The post-deployment-setup role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#

```

```

# - uaa-create-user: Create the initial user in UAA
#
# - configure-scf: Uses the cf CLI to configure SCF once it's online (things
#   like proxy settings, service brokers, etc.)
post_deployment_setup:
  # Node affinity rules can be specified here
  affinity: {}

# The post-deployment-setup role cannot be scaled.
count: 1

# Unit [millicore]
cpu:
  request: 1000
  limit: ~

# Unit [MiB]
memory:
  request: 256
  limit: ~

# The router role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - gorouter: Gorouter maintains a dynamic routing table based on updates
#   received from NATS and (when enabled) the Routing API. This routing table
#   maps URLs to backends. The router finds the URL in the routing table that
#   most closely matches the host header of the request and load balances
#   across the associated backends.
#
# Also: metron_agent
router:
  # Node affinity rules can be specified here
  affinity: {}

# The router role can scale between 1 and 65535 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 4000
  limit: ~

```

```

# Unit [MiB]
memory:
  request: 135
  limit: ~

# The routing-api role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# Also: metron_agent, routing-api
routing_api:
  # Node affinity rules can be specified here
  affinity: {}

# The routing-api role can scale between 1 and 3 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 4000
  limit: ~

# Unit [MiB]
memory:
  request: 114
  limit: ~

# The secret-generation role contains the following jobs:
#
# - generate-secrets: This job will generate the secrets for the cluster
secret_generation:
  # Node affinity rules can be specified here
  affinity: {}

# The secret-generation role cannot be scaled.
count: 1

# Unit [millicore]
cpu:
  request: 1000
  limit: ~

```

```

# Unit [MiB]
memory:
  request: 256
  limit: ~

# The syslog-adapter role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# Also: adapter, metron_agent
syslog_adapter:
  # Node affinity rules can be specified here
  affinity: {}

# The syslog-adapter role can scale between 1 and 65535 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 78
  limit: ~

# The syslog-rlp role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# Also: metron_agent, reverse_log_proxy
syslog_rlp:
  # Node affinity rules can be specified here
  affinity: {}

# The syslog-rlp role can scale between 1 and 65535 instances.
# For high availability it needs at least 2 instances.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

```

```

# Unit [MiB]
memory:
  request: 93
  limit: ~

# The syslog-scheduler role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# Also: scheduler, metron_agent
syslog_scheduler:
  # Node affinity rules can be specified here
  affinity: {}

# The syslog-scheduler role cannot be scaled.
count: 1

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 69
  limit: ~

# The tcp-router role contains the following jobs:
#
# - global-properties: Dummy BOSH job used to host global parameters that are
#   required to configure SCF
#
# - authorize-internal-ca: Install both internal and UAA CA certificates
#
# - wait-for-uaa: Wait for UAA to be ready before starting any jobs
#
# Also: tcp_router, metron_agent
tcp_router:
  # Node affinity rules can be specified here
  affinity: {}

# The tcp-router role can scale between 1 and 3 instances.
# For high availability it needs at least 2 instances.
count: 1

```

```

# Unit [millicore]
cpu:
  request: 2000
  limit: ~

# Unit [MiB]
memory:
  request: 99
  limit: ~

ports:
  tcp_route:
    count: 9

secrets:
  # The password for the cluster administrator.
  # This value is immutable and must not be changed once set.
  CLUSTER_ADMIN_PASSWORD: ~

  # LDAP service account password (required for LDAP integration only)
  # This value is immutable and must not be changed once set.
  PERSI_NFS_DRIVER_LDAP_PASSWORD: "-"

  # The password of the admin client - a client named admin with uaa.admin as an
  # authority.
  # This value is immutable and must not be changed once set.
  UAA_ADMIN_CLIENT_SECRET: ~

  # The CA certificate for UAA
  UAA_CA_CERT: ~

  # PEM encoded RSA private key used to identify host.
  # This value uses a generated default.
  APP_SSH_KEY: ~

  # MD5 fingerprint of the host key of the SSH proxy that brokers connections to
  # application instances.
  # This value uses a generated default.
  APP_SSH_KEY_FINGERPRINT: ~

  # PEM-encoded certificate
  # This value uses a generated default.
  AUCTIONEER_REP_CERT: ~

  # PEM-encoded key
  # This value uses a generated default.
  AUCTIONEER_REP_KEY: ~

```



```
# PEM-encoded server certificate
# This value uses a generated default.
AUCTIONEER_SERVER_CERT: ~

# PEM-encoded server key
# This value uses a generated default.
AUCTIONEER_SERVER_KEY: ~

# PEM-encoded certificate
# This value uses a generated default.
BBS_AUCTIONEER_CERT: ~

# PEM-encoded key
# This value uses a generated default.
BBS_AUCTIONEER_KEY: ~

# PEM-encoded client certificate.
# This value uses a generated default.
BBS_CLIENT_CERT: ~

# PEM-encoded client key.
# This value uses a generated default.
BBS_CLIENT_KEY: ~

# PEM-encoded certificate
# This value uses a generated default.
BBS_REP_CERT: ~

# PEM-encoded key
# This value uses a generated default.
BBS_REP_KEY: ~

# PEM-encoded client certificate.
# This value uses a generated default.
BBS_SERVER_CERT: ~

# PEM-encoded client key.
# This value uses a generated default.
BBS_SERVER_KEY: ~

# The PEM-encoded certificate (optionally as a certificate chain) for serving
# blobs over TLS/SSL.
# This value uses a generated default.
BLOBSTORE_TLS_CERT: ~

# The PEM-encoded private key for signing TLS/SSL traffic.
```

```
# This value uses a generated default.
BLOBSTORE_TLS_KEY: ~

# The PEM-encoded certificate for internal cloud controller traffic.
# This value uses a generated default.
CC_SERVER_CERT: ~

# The PEM-encoded private key for internal cloud controller traffic.
# This value uses a generated default.
CC_SERVER_KEY: ~

# The PEM-encoded certificate for internal cloud controller uploader traffic.
# This value uses a generated default.
CC_UPLOADER_CERT: ~

# The PEM-encoded private key for internal cloud controller uploader traffic.
# This value uses a generated default.
CC_UPLOADER_KEY: ~

# PEM-encoded broker server certificate.
# This value uses a generated default.
CF_USB_BROKER_SERVER_CERT: ~

# PEM-encoded broker server key.
# This value uses a generated default.
CF_USB_BROKER_SERVER_KEY: ~

# PEM-encoded client certificate
# This value uses a generated default.
DIEGO_CLIENT_CERT: ~

# PEM-encoded client key
# This value uses a generated default.
DIEGO_CLIENT_KEY: ~

# PEM-encoded certificate.
# This value uses a generated default.
DOPPLER_CERT: ~

# PEM-encoded key.
# This value uses a generated default.
DOPPLER_KEY: ~

# PEM-encoded CA certificate used to sign the TLS certificate used by all
# components to secure their communications.
# This value uses a generated default.
INTERNAL_CA_CERT: ~
```

```
# PEM-encoded CA key.
# This value uses a generated default.
INTERNAL_CA_KEY: ~

# PEM-encoded certificate.
# This value uses a generated default.
METRON_CERT: ~

# PEM-encoded key.
# This value uses a generated default.
METRON_KEY: ~

# PEM-encoded server certificate
# This value uses a generated default.
REP_SERVER_CERT: ~

# PEM-encoded server key
# This value uses a generated default.
REP_SERVER_KEY: ~

# The public ssl cert for ssl termination.
# This value uses a generated default.
ROUTER_SSL_CERT: ~

# The private ssl key for ssl termination.
# This value uses a generated default.
ROUTER_SSL_KEY: ~

# PEM-encoded certificate
# This value uses a generated default.
SYSLOG_ADAPT_CERT: ~

# PEM-encoded key.
# This value uses a generated default.
SYSLOG_ADAPT_KEY: ~

# PEM-encoded certificate
# This value uses a generated default.
SYSLOG_RLP_CERT: ~

# PEM-encoded key.
# This value uses a generated default.
SYSLOG_RLP_KEY: ~

# PEM-encoded certificate
# This value uses a generated default.
```

```

SYSLOG_SCHED_CERT: ~

# PEM-encoded key.
# This value uses a generated default.
SYSLOG_SCHED_KEY: ~

# PEM-encoded client certificate for internal communication between the cloud
# controller and TPS.
# This value uses a generated default.
TPS_CC_CLIENT_CERT: ~

# PEM-encoded client key for internal communication between the cloud
# controller and TPS.
# This value uses a generated default.
TPS_CC_CLIENT_KEY: ~

# PEM-encoded certificate for communication with the traffic controller of the
# log infra structure.
# This value uses a generated default.
TRAFFICCONTROLLER_CERT: ~

# PEM-encoded key for communication with the traffic controller of the log
# infra structure.
# This value uses a generated default.
TRAFFICCONTROLLER_KEY: ~

services:
  loadbalanced: false
kube:
  external_ips: []
  # Increment this counter to rotate all generated secrets
  secrets_generation_counter: 1
  storage_class:
    persistent: "persistent"
    shared: "shared"
  registry:
    hostname: "registry.suse.com"
    username: ""
    password: ""
  organization: "cap"
  auth: "rbac"

```