

Cloud Automation Manager CE Installation

November 22, 2019

Stephen R. Poon
Ralf Schmidt-Dannert
IBM NA Power Technical Team

Table of Contents

1	DISCLAIMERS	2
1.1	NO WARRANTY	2
1.2	LIMITATION OF LIABILITY	2
1.3	COMMENTS / FEEDBACK	2
2	INTRODUCTION	3
3	ASSUMPTIONS	3
3.1	TARGET CLUSTER	4
4	PRE-INSTALLATION STEPS	4
4.1	CONFIGURE IBM Cloud PRIVATE CLI	4
4.2	CONFIGURE CLOUDCTL	4
4.3	CONFIGURE HELM CLI	4
4.4	CREATE DOCKER STORE SECRET	5
4.5	CREATE PERSISTENT VOLUMES	7
4.5.1	Configure the NFS server	7
4.5.2	Create PVs in the ICP cluster	8
4.6	GENERATE THE DEPLOYMENT SERVICEID API KEY	10
5	INSTALL IBM CLOUD AUTOMATION MANAGER	10
5.1	CAM COMMUNITY VERSION V3.2.1.2	10
5.2	UNINSTALLING CAM	16
5.3	TROUBLESHOOTING	19

1 Disclaimers

1.1 No warranty

Subject to any statutory warranties which cannot be excluded, IBM makes no warranties or conditions either express or implied, including without limitation, the warranty of non-infringement and the implied warranties of merchantability and fitness for a particular purpose, regarding the use of this document or technical support, if any.

1.2 Limitation of Liability

Neither IBM nor its suppliers will be liable for any direct or indirect damages, including without limitation, lost profits, lost savings, or any incidental, special, or other economic consequential damages, even if IBM is informed of their possibility. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above exclusion or limitation may not apply to you.

1.3 Comments / Feedback

The authors are interested in any feedback or comments on this white paper and the approach taken to implement a service on IBM Cloud Automation Manager.

Contact information:

Ralf Schmidt-Dannert, IBM
dannert@us.ibm.com

2 Introduction

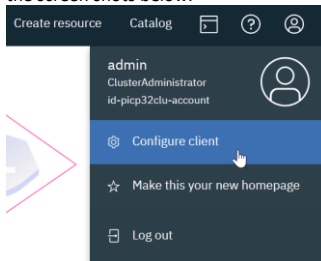
This document describes the installation approach and steps used to install Cloud Automation Manager (CAM) Community Edition (CE) 3.2.1.2 into the ICP 3.2.1 cluster for use in the demo described in the accompanying technical white paper: *Deploying Oracle Database with IBM Cloud Automation Manager and IBM PowerVC by Stephen Poon and Ralf Schmidt-Dannert, IBM.*

For further details and alternative options to install CAM 3.2.1.x please visit:

https://www.ibm.com/support/knowledgecenter/SS2L37_3.2.1.0/kc_welcome.html

3 Assumptions

- IBM Cloud Private 3.2.1 has been installed and the Helm repositories have been set up. In a standard ICP install the Helm repositories are set up during install.
- Helm and Kubernetes CLI have been installed,
 - https://www.ibm.com/support/knowledgecenter/SSBS6K_3.2.1/app_center/create_helm_cli.html
 - https://www.ibm.com/support/knowledgecenter/SSBS6K_3.2.1/manage_cluster/install_kubectl.html
- Cluster configuration details have been obtained using the IBM Cloud Private CLI or management console as illustrated in the screen shots below.



Configure client

Before you run commands in the kubectl command line interface for this cluster, you must configure the client.

Prerequisites:

Install CLI tools

To configure the CLI, paste the displayed configuration commands into your terminal window and run them:

```
kubectl config set-cluster picp32clu --server=https://129.██████████ --insecure-skip-tls-verify=true
kubectl config set-context picp32clu-context --cluster=picp32clu
kubectl config set-credentials admin --token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhdF9oYXNoIjoiNjYxNjI2ODFmNjExNjYzZjNGS
kubectl config set-context picp32clu-context --user=admin --namespace=services
kubectl config use-context picp32clu-context
```

Copy to clipboard

The security token generated expires in 12 hours. In order to continue to use the CLI, you must log in, get a new token as shown above, and reconfigure kubectl every 12 hours.

- Copy the commands and paste into a terminal window on (ICP Master node) and press Enter

3.1 Target cluster

For this install we use a cluster consisting of the following VMs and roles:

```
VMs
pIC32-mst1-38    etcd,master
pIC32-mst2-38    etcd,master
pIC32-mst3-38    etcd,master
pIC32-pxy-38     proxy
pIC32-mgm-38     management
pIC32-w-1-38     worker
pIC32-w-2-38     worker

Persistent Volumes were created on
ICP_NFSserver    ATS S922-9009-22A-
```

4 Pre-installation Steps

4.1 Configure IBM Cloud Private CLI

See section [Assumptions](#)

4.2 Configure cloudctl

ICP install provides the cloudctl binary on the boot/master node and there is no further configuration needed. In this install pIC32-mst1-38 has the role of boot/master node to run cloudctl, kubectl and helm commands.

If you want to install on local PC..., see this link:

Reference: https://www.ibm.com/support/knowledgecenter/SSBS6K_3.2.1/manage_cluster/install_cli.html

4.3 Configure Helm CLI

The helm software is already installed by the ICP installer on the master node pIC32-mst1-38, but further configuration steps need to be performed as shown in this section.

Configure HELM cli on master pIC32-mst1-38

https://www.ibm.com/support/knowledgecenter/SSBS6K_3.2.1/app_center/create_helm_cli.html

Note that the helm binary is already installed on this boot/master node - so no download needed!

In a shell on pIC32-mst1-38 run:

```
# export HELM_HOME=~/.helm
# helm init --client-only
Creating /root/.helm
Creating /root/.helm/repository
Creating /root/.helm/repository/cache
Creating /root/.helm/repository/local
Creating /root/.helm/plugins
Creating /root/.helm/starters
Creating /root/.helm/cache/archive
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.
Not installing Tiller due to 'client-only' flag having been set
Happy Helming!
```

You need to do the following before you can use the helm cli as it creates the necessary ~/.helm/cert.pem file. Adjust highlighted values to your environment.

```
# cloudctl login -a https://{ICP access IP}:8443 --skip-ssl-validation -u admin -p {admin
password}
Authenticating...
OK

Targeted account picp32clu Account

Select a namespace:
1. cert-manager
2. default
3. ibmcom
4. icp-system
5. istio-system
6. kube-public
7. kube-system
8. services
Enter a number> 7
Targeted namespace kube-system

Configuring kubectl ...
Property "clusters.picp32clu" unset.
Property "users.picp32clu-user" unset.
Property "contexts.picp32clu-context" unset.
Cluster "picp32clu" set.
User "picp32clu-user" set.
Context "picp32clu-context" created.
Switched to context "picp32clu-context".
OK

Configuring helm: /root/.helm
OK
```

And now validate that helm cli works – note the “--tls” at the end which is required for all helm commands.

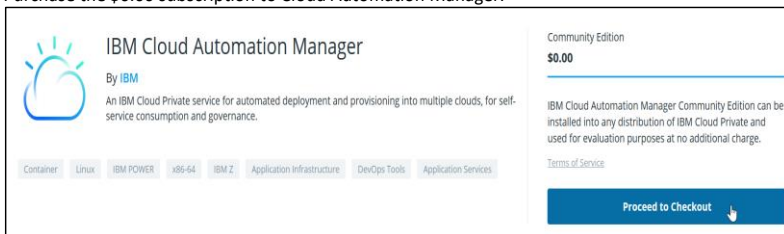
```
# helm version --tls
Client: &version.Version{SemVer:"v2.12.3",
GitCommit:"eecf22f77df5f65c823aacd2dbd30ae6c65f186e", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.12.3+icp", GitCommit:"", GitTreeState:""}
```

4.4 Create Docker store secret.

If not already existing, create a free account on hub.docker.com (<https://hub.docker.com/signup>)

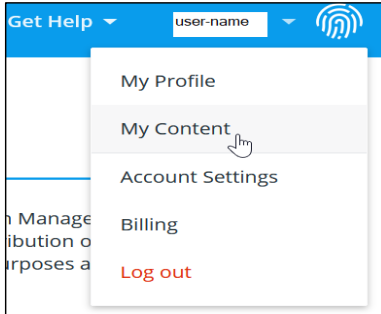
Then subscribe to <https://store.docker.com/images/ibm-cloud-automation-manager>.

- Go to above WEB site and **log in with your docker.com user ID and password**
- Purchase the \$0.00 subscription to Cloud Automation Manager.

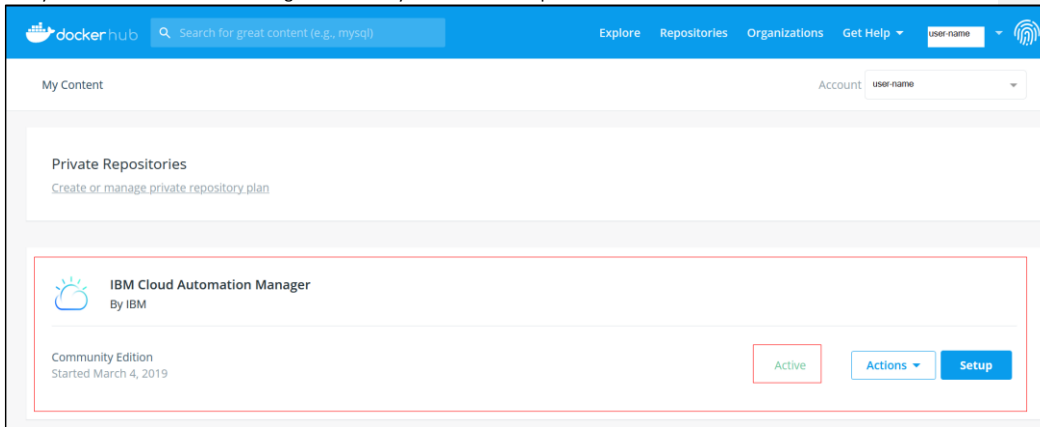


To verify the successful subscription, do the following on the hub.docker.com WEB site after logging in:

- On the top right, click your account drop-down and select My Content.



- Verify that Cloud Automation Manager is listed in your list of subscriptions and has a status of Active.



To be able to access your account on hub.docker.com and download content from ICP you need to create a secret in ICP with the relevant access data. Run the following command on the master pIC32-mst1-38 to generate the Kubernetes secret *cam-clu038*.
Note that the secret needs to be created in the same namespace you plan to deploy CAM into – in this case *services*.

```
# kubectl create secret docker-registry <secretname> --docker-username=<userid> --  
docker-password=<password> --docker-email=<email> -n services  
Where:  
  <secretname> - It is your desired secret name.  
  <userid> - It is your Docker Store user name.  
  <password> - It is your Docker Store password.  
  <email> - The email address associated with your Docker Store account.
```

For further reference: https://www.ibm.com/support/knowledgecenter/SS2L37_3.2.1.0/cam_docker_secret.html

After running the kubectl create secret command, you will see the secret in ICP. Click the hamburger menu (top left), then Configuration->Secrets.

Below is how the 038 cluster secret was created and how it shows up in the GUI. Note that *{dannert password}* is a placeholder for the password and not the actually value specified in the command.

```
# kubectl create secret docker-registry cam-clu038 --docker-username=dannert --docker-  
password={dannert password} --docker-email=dannert@us.ibm.com -n services
```

IBM Cloud Private

CLUSTER
picp32clu

Create resource Catalog

Secrets

services

Search Secrets

Create Secret +

Name	Type	Created
cam-clu038	kubernetes.io/dockerconfigjson	3 minutes ago
oauth-client-secret	Opaque	16 hours ago
default-token-5nj9g	kubernetes.io/service-account-token	16 hours ago

items per page 20 | 1-3 of 3 items

4.5 Create Persistent Volumes

CAM requires persistent storage to be able to store templates, status, configuration,... CAM supports all of the persistent volume types that IBM Cloud Private supports.

In our environment, we used an NFS server. For other options, or further details, please refer to:

https://www.ibm.com/support/knowledgecenter/SS2L37_3.2.1.0/cam_create_pv.html

Ensure that the relevant NFS client software is installed on all cluster nodes if you utilize NFS for persistent storage. For example, for RedHat 7, that would be the *nfs-utils* package. You can check with "rpm -qa | grep nfs-utils".

4.5.1 Configure the NFS server

As the NFS server may be supporting multiple CAM installs it is best practice to code the ICP cluster into the directories / exports. In our case, we used CAM038_*, where 038 is from the IP address of the master node (xxx.xx.xx.38) or you can use some other suitable scheme.

1. Add exports to NFS server to be used for persistent storage in CAM.

```
# cd /export
export_dir=/export
mkdir -p ${export_dir}/CAM038_db ${export_dir}/CAM038_terraform/cam-provider-terraform \
${export_dir}/CAM038_logs/cam-provider-terraform \
${export_dir}/CAM038_BPD_appdata/mysql \
${export_dir}/CAM038_BPD_appdata/repositories \
${export_dir}/CAM038_BPD_appdata/workspace
chmod -R 2775 ${export_dir}/CAM038_db ${export_dir}/CAM038_logs \
${export_dir}/CAM038_terraform ${export_dir}/CAM038_BPD_appdata
chown -R root:1000 ${export_dir}/CAM038_logs \
${export_dir}/CAM038_BPD_appdata
chown -R root:1111 ${export_dir}/CAM038_terraform \
${export_dir}/CAM038_logs/cam-provider-terraform
chown -R 1000:1000 ${export_dir}/CAM038_BPD_appdata/mysql ${export_dir}/CAM038_db
```

2. Edit `/etc/exports` and add the following lines:

```
/export/CAM038_logs *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
/export/CAM038_db *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
/export/CAM038_terraform *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
/export/CAM038_BPD_appdata *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
```

3. Export the new directories and verify successful export:

```
# exportfs -ra
# exportfs | grep 038
/export/CAM038_logs
/export/CAM038_db
/export/CAM038_terraform
/export/CAM038_BPD_appdata
```

4.5.2 Create PVs in the ICP cluster

Log in to the master `pIC32-mst1-38` and then authenticate via `cloudctl` command.

Create the yaml files describing the persistent volumes required by CAM. Note that the “path:” entries in the yaml files below need to match what was exported from the NFS server as defined in `/etc/exports`. The entry for server: is the IP address or host name of the NFS server exporting the storage.

Create the following four yaml files as shown below and adjust the NFS server and exported path names according to your environment.

cam-bpd.yaml

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: cam-bpd-appdata-pv
  labels:
    type: cam-bpd-appdata
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: my.nfs.server
    path: /export/CAM038_BPD_appdata
```

cam-logs.yaml

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: cam-logs-pv
  labels:
    type: cam-logs
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: my.nfs.server
    path: /export/CAM038_logs
```


cam-mongo-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cam-mongo-pv
  labels:
    type: cam-mongo
spec:
  capacity:
    storage: 15Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: my.nfs.server
    path: /export/CAM038_db
```

cam-terraform.yaml

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: cam-terraform-pv
  labels:
    type: cam-terraform
spec:
  capacity:
    storage: 15Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: my.nfs.server
    path: /export/CAM038_terraform
```

Create the persistent volumes using the following commands and verify successful creation.

```
# kubectl create -f ./cam-bpd.yaml
# kubectl create -f ./cam-logs.yaml
# kubectl create -f ./cam-mongo-pv.yaml
# kubectl create -f ./cam-terraform.yaml
```

```
# kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS
cam-bpd-appdata-pv	20Gi	RWX	Retain	Available
cam-logs-pv	10Gi	RWX	Retain	Available
cam-mongo-pv	15Gi	RWX	Retain	Available
cam-terraform-pv	15Gi	RWX	Retain	Available

```
...
```

4.6 Generate the deployment ServiceID API key

Create a shell script on the master with the following commands. Assume the name of this script is create-deployment-serviceid-api-key.sh. You want to keep a copy of the created key for the future reference!

Replace the highlighted text with the values for your environment.

```
export serviceIDName='service-deploy'
export serviceApiKeyName='service-deploy-api-key'
cloudctl login -a https://{ICP cluster IP}:8443 --skip-ssl-validation -u admin -p {admin password} -n services
cloudctl iam service-id-create ${serviceIDName} -d 'Service ID for service-deploy'
cloudctl iam service-policy-create ${serviceIDName} -r Administrator,ClusterAdministrator --service-name 'idmgmt'
cloudctl iam service-policy-create ${serviceIDName} -r Administrator,ClusterAdministrator --service-name 'identity'
cloudctl iam service-api-key-create ${serviceApiKeyName} ${serviceIDName} -d 'Api key for service-deploy'
```

Execute the script,

```
# chmod +x create-deployment-serviceid-api-key.sh
# create-deployment-serviceid-api-key.sh | tee create-deployment-serviceid-api-key.sh.out
```

The last line will look like this,

```
API Key      i0VPCdZnztRhWQ1fTDP-4uA37vpNPNRJyzKHd7NcGRE
```

Save this key for later.

5 Install IBM Cloud Automation Manager

All the commands and actions described in the following sections are carried out on the ICP master node.

5.1 CAM Community Version V3.2.1.2

This section describes the installation of IBM Cloud Automation Manager Community Edition V3.2.1.2 and is based on “Documented procedure for installing CAM Community Edition Version 3.2.1 online” section in:

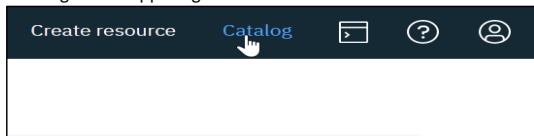
https://www.ibm.com/support/knowledgecenter/SS2L37_3.2.1.0/cam_install_CE.html

Access the ICP GUI at:

<https://{IP address of ICP cluster}:8443/oidc/login.jsp>

Log in as admin/{admin user password}

Click Catalog on the upper right-hand corner.



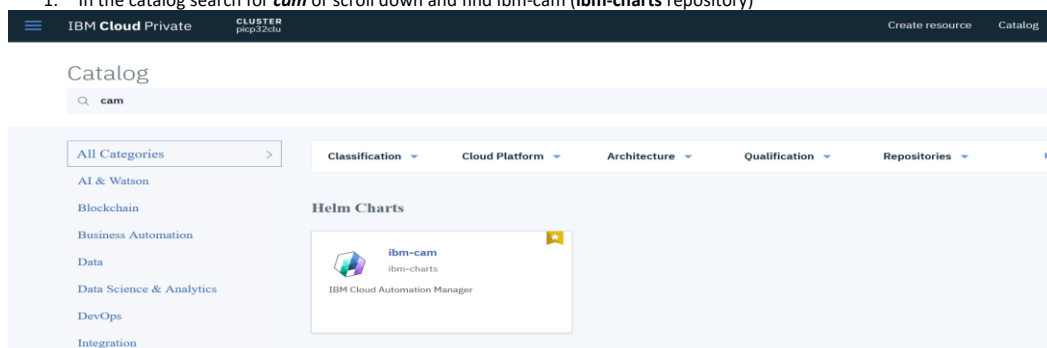
If you get a blank screen you need to sync the catalog.

- On the hamburger menu (upper left-hand corner), select Manage -> Helm Repositories
- Click Sync repositories

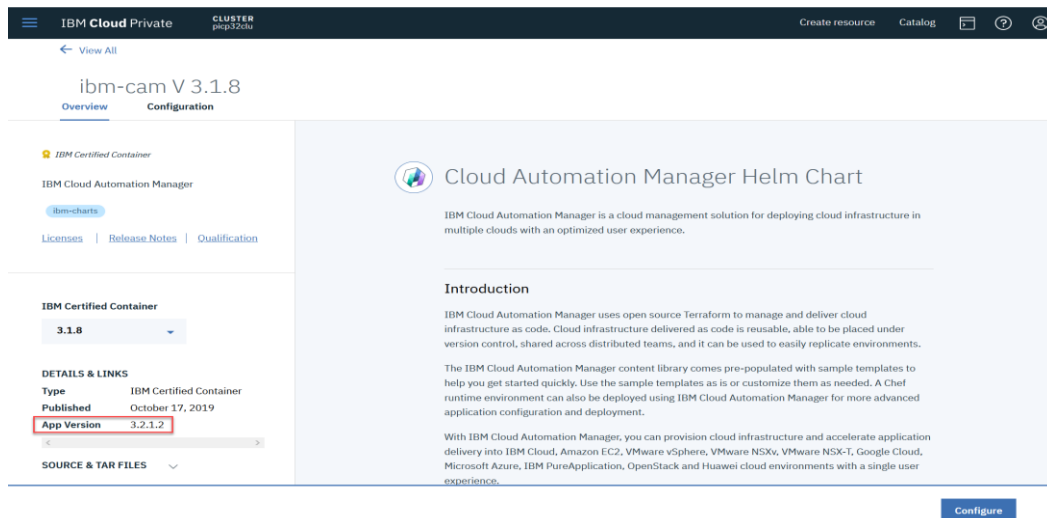


< may have to wait a minute or so >

1. In the catalog search for **cam** or scroll down and find ibm-cam (ibm-charts repository)



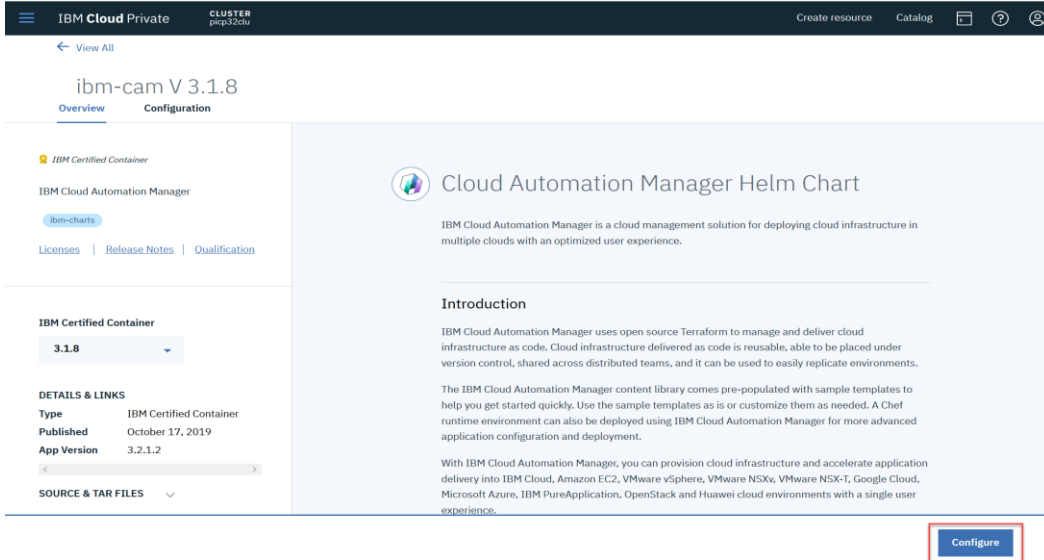
2. Click the ibm-cam Helm chart icon.



Note the CAM Application version (3.2.1.2) is not the same as the Helm Chart version (3.1.8).

Scroll down to review installation instructions, prerequisites, requirements, limitations, etc.

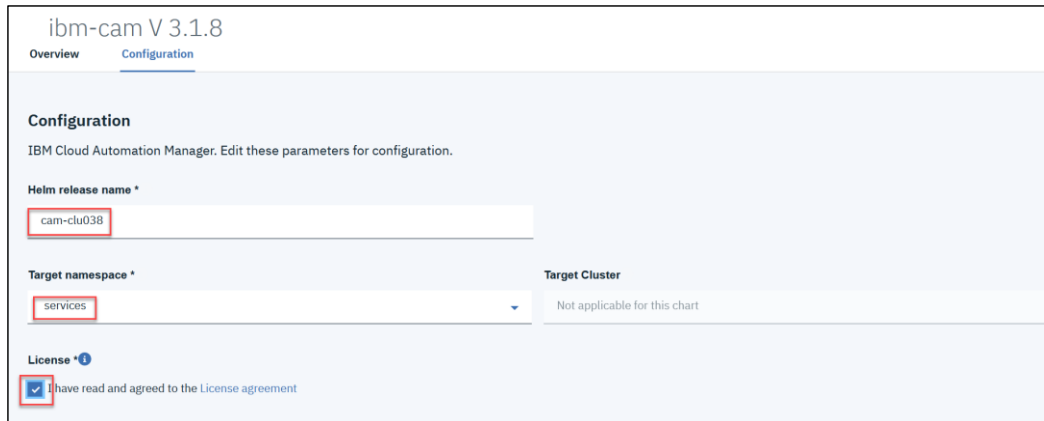
3. Click **Configure** to start deployment steps.



In the Configuration section, enter the following values and accept the license agreement:

Release name – Pick a name for your release. We used **cam-clu038** as release name.

Target namespace - From the drop-down list, select **services**.



Scroll down and expand the “All Parameters” section.

Enter/select the following values:

Worker node architecture: **ppc64le**

Platform: **icp** (we did not have MCM installed in our cluster)

Docker image pull secret – secret was created earlier: **cam-038**

Product Identifier will show: **IBMCloudAutomationManager_5737E67_3210_CE_000**.

Deploy API Key - Enter the deploy API Key you created earlier – see end of file: create-deployment-serviceid-api-key.sh.out

IBM Cloud Private

CLUSTER
pcp3z2da

Create resource

Catalog

?

> Quick start

Required and recommended parameters to view and edit.

< All parameters

Other required, optional, and read-only parameters to view and edit.

Worker node architecture

ppc64le

Platform

icp

Global

Values applicable to all components in the chart

Docker image pull secret (see help ->) *

cam-clu038

Product identifier

IBMCloudAutomationManager_5737E67_3210_CE_000

IAM service API key (see help ->) *

i0VPCdZnztkRHWQIITDP-4uA37vpNPNRjyzKHd7NcGRc

Left at default:

Optimize for offline install: (unchecked)
 Enable audit: (unchecked)
 Enable legacy audit: (unchecked)
 Enable FIPS for CAM: (unchecked)
 ICP Port: 8443
 CAM Management Console Port: 30000
 Use the internal MongoDB: (checked)
 Repository: (store/ibmcorp)
 Tag: (3.2.1.2)
 Pull Policy: (IfNotPresent)
 Docker Config: (Empty)
 Use a proxy (unchecked)
 Install CAM Controller in this cluster: (unchecked)

Scroll down to review the Persistent Volumes. The default names and values should match those created earlier with the .yaml files. None of the parameters / check boxes for PVs were changed from the default!

You can verify the configured PVs again by running the following command on the master node:

```
# kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM
cam-bpd-appdata-pv	20Gi	RWX	Retain	Available	
cam-logs-pv	10Gi	RWX	Retain	Available	
cam-mongo-pv	15Gi	RWX	Retain	Available	
cam-terraform-pv	15Gi	RWX	Retain	Available	

All other parameters after the PV section were left at default as well.

Click **Install** button to start the deployment process.

This deploy takes a few minutes. You can check status of the CAM related PODS (assuming *services* namespace is used) via the command below. The output shows a successful deploy with all PODs running.

```
# kubectl -n services get pods | grep cam
cam-bpd-cds-65b95d79f6-6k45w          1/1      Running    0          9m16s
cam-bpd-mariadb-bb56455b9-lp92m       1/1      Running    0          9m16s
cam-bpd-mds-8c44775f7-4pf74          1/1      Running    0          9m16s
cam-bpd-ui-8fb88f7d4-kgxrz           1/1      Running    0          9m15s
cam-broker-55b8f86495-hbxpk          1/1      Running    0          9m14s
cam-iaas-bf6fccd55-6xc5t             1/1      Running    0          9m13s
cam-mongo-b5b8f5788-cxn27            1/1      Running    0          9m12s
cam-orchestration-77846cff5c-khs4f    1/1      Running    0          9m12s
cam-portal-ui-864c7b77db-kjtj8       1/1      Running    0          9m11s
cam-provider-helm-689ddd4dc-fxmxh     1/1      Running    0          9m10s
cam-provider-terraform-65cc99d694-5g4hp 1/1      Running    0          9m9s
cam-proxy-6b6747cb5b-jrkb1           1/1      Running    0          9m8s
cam-service-composer-api-c747d548f-h98xt 1/1      Running    0          9m8s
cam-service-composer-ui-84d574bfc4-j5lg4 1/1      Running    0          9m7s
cam-tenant-api-54fcc9b9b5-sc5bz       1/1      Running    0          9m6s
cam-ui-basic-6d76b6595d-4gtm5        1/1      Running    0          9m5s
```

To get all details of the CAM deploy find and click on the name of the Helm release. In our case, it's **cam-clu038**.

Helm Releases					
<input type="text" value="cam"/>					
Name	Cluster Name	Namespace	Status	Chart Name	Current Version
cam-clu038	local-cluster	services	Deployed	ibm-cam	3.1.8
items per page 20 1-1 of 1 items					

When all the pods are in Running state, verify the functionality of the deploy (optional).

From the command line on the master node (replace the highlighted text with the correct values for your environment):

```
# cloudctl login -a https://{ICP access IP}:8443 --skip-ssl-validation -u admin -p {admin password}
Authenticating...
OK

Targeted account picp32clu Account

Select a namespace:
1. cert-manager
2. default
3. ibmcom
4. icp-system
5. istio-system
6. kube-public
7. kube-system
8. services
Enter a number> 8
Targeted namespace services

Configuring kubectl ...
Property "clusters.picp32clu" unset.
Property "users.picp32clu-user" unset.
Property "contexts.picp32clu-context" unset.
```

```

Cluster "picp32clu" set.
User "picp32clu-user" set.
Context "picp32clu-context" created.
Switched to context "picp32clu-context".
OK

Configuring helm: /root/.helm
OK

# helm test cam-clu038 --tls --cleanup
RUNNING: cam-clu038-cam-proxy-test
PASSED: cam-clu038-cam-proxy-test
RUNNING: cam-clu038-cam-service-composer-api-test
PASSED: cam-clu038-cam-service-composer-api-test
RUNNING: cam-clu038-cam-ui-basic-test
PASSED: cam-clu038-cam-ui-basic-test
RUNNING: cam-clu038-cam-portal-ui-test
PASSED: cam-clu038-cam-portal-ui-test
RUNNING: cam-clu038-cam-provider-terraform-test
PASSED: cam-clu038-cam-provider-terraform-test
RUNNING: cam-clu038-cam-bpd-mds-test
PASSED: cam-clu038-cam-bpd-mds-test
RUNNING: cam-clu038-cam-broker-test
PASSED: cam-clu038-cam-broker-test
RUNNING: cam-clu038-cam-orchestration-test
PASSED: cam-clu038-cam-orchestration-test
RUNNING: cam-clu038-cam-bpd-cds-test
PASSED: cam-clu038-cam-bpd-cds-test
RUNNING: cam-clu038-cam-iaas-test
PASSED: cam-clu038-cam-iaas-test
RUNNING: cam-clu038-cam-bpd-ui-test
PASSED: cam-clu038-cam-bpd-ui-test
RUNNING: cam-clu038-cam-tenant-api-test
PASSED: cam-clu038-cam-tenant-api-test
RUNNING: cam-clu038-cam-service-composer-ui-test
PASSED: cam-clu038-cam-service-composer-ui-test

```

Note: The helm test command creates pods named *-test. These are normally not automatically deleted in order to allow viewing of status, logs, or other diagnostic information. The --cleanup flag triggers the automatic deletion of the test pods after command completion.

To determine the CAM GUI access information, you can run:

```

export SERVICE_PORT=$(kubectl get service -n services cam-proxy \
-o jsonpath='{.spec.ports[0].nodePort}')
echo https://<ICP\_PROXY\_IP>:\$SERVICE\_PORT/cam
https://<ICP\_PROXY\_IP>:30000/cam

```

Replace <ICP_PROXY_IP> with the access IP of your cluster and connect to the GUI:

<https://{IP address of ICP }:30000/cam>

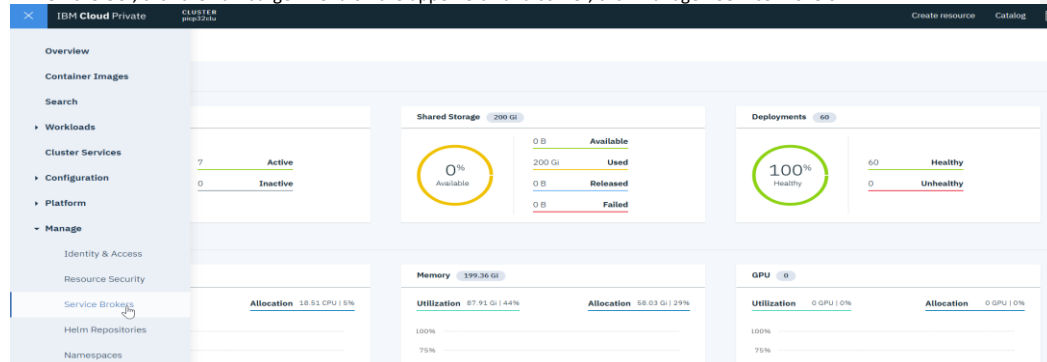


5.2 Uninstalling CAM

If the CAM install failed or you want to uninstall CAM see the steps below. For reference also look at:

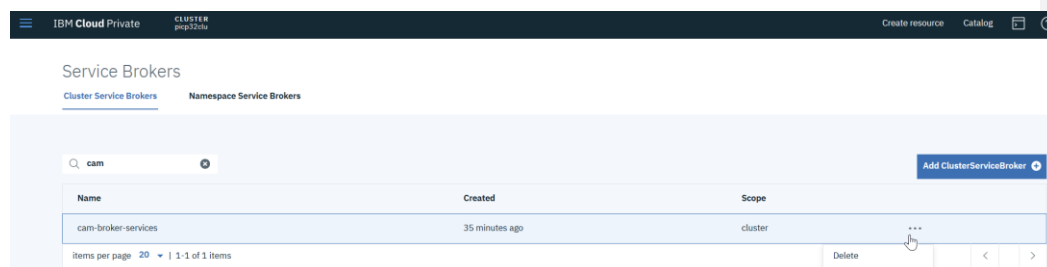
https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/cam_uninstalling.html

1. On the GUI, click the hamburger menu on the upper left-hand corner, click Manage->Service Brokers

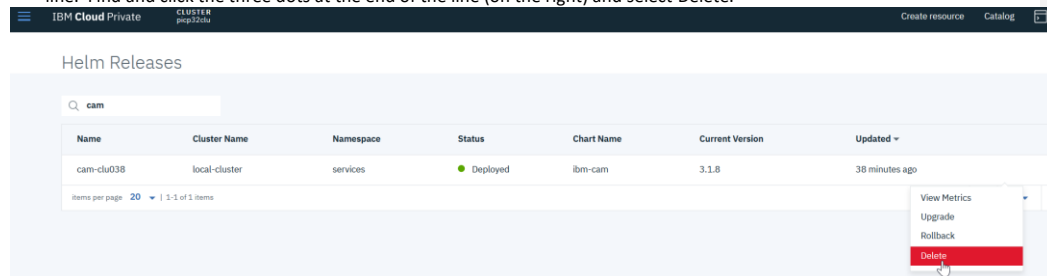


Commented [SP1]: Documentation is incorrect - cam-broker is not the correct name (incorrect on my draft). It's cam-broker-services, which made this step the same as the next GUI one.

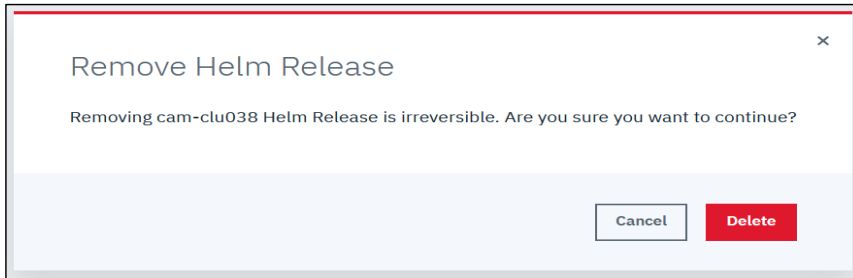
Note the service broker is named cam-broker-services. Use the overflow (3-dot) menu on the right-hand side of the cam-broker-services entry to delete.



2. From ICP menu, Workloads -> Helm Releases. Find the cam deployment under Name (e.g. cam-clu038 and highlight this line. Find and click the three dots at the end of the line (on the right) and select Delete.



3. Confirm deletion.



4. Check status of the deletion request.

```
# kubectl -n services get pods
```

NAME	READY	STATUS	RESTARTS	AGE
cam-bpd-ui-7d4b4ccbd7-qh46x	0/1	Terminating	0	42m
cam-mongo-76f8964dd4-z5rkx	0/1	Terminating	0	42m
cam-provider-helm-58f96889d9-7pcdr	0/1	Terminating	0	42m
cam-provider-terraform-77bcb86954-wbt2b	1/1	Terminating	0	42m
cam-proxy-9d867754-2mdfk	1/1	Terminating	0	42m
cam-service-composer-api-6d7cd568cc-5clt7	0/1	Terminating	0	42m
cam-service-composer-ui-54f497df6-pfq6k	1/1	Terminating	0	42m
cam-tenant-api-6cf7b7755b-dlk9z	1/1	Terminating	0	42m
cam-ui-basic-6f85768d96-9mnwk	0/1	Terminating	0	42m
cam-ui-connections-56f8849bfb-krmjc	0/1	Terminating	0	42m
cam-ui-templates-647676d9df-9dqqr	0/1	Terminating	0	42m

5. From the ICP menu, Platform -> Storage. Click the PersistentVolumeClaim tab (left hand corner). Find the cam persistent volumes (e.g. cam-logs-pv, etc.). Highlight each one, go to the end of the line (right hand side), click on the three dots and Remove.

Name	Namespace	Status	PersistentVolume	Requests	Access mode	Created
cam-bpd-appdata-pv	services	Bound	cam-bpd-appdata-pv	15Gi	RWX	2 hours ago
cam-logs-pv	services	Bound	cam-logs-pv	10Gi	RWX	2
cam-mongo-pv	services	Bound	cam-mongo-pv	15Gi	RWX	2

6. Click the persistentVolume tab and remove the cam persistent volumes in the same manner.
7. If you are not reinstalling CAM, delete the docker secret created in Section 4.4, "Create Docker store secret." on page 5. From the ICP menu, Configuration->Secrets. Find the secret, cam-clu038, go to the end of the line (right hand side), click on the three dots and Delete.

Name	Namespace	Created
cam-clu038	kubernetes.io/dockerconfigjson	7 hours ago
sa-kube-system	kubernetes.io/dockerconfigjson	
oauth-client-secret	Opaque	

Alternative option, much simpler, is to remove the CAM install from the command line using the following script:

Note: Original script found in (CAM 3.1.0),

<https://raw.githubusercontent.com/IBM/charts/master/repo/stable/ibm-cam-3.1.0.tgz>

```
#!/bin/bash
#
#Licensed Materials - Property of IBM
#5737-E67
#(C) Copyright IBM Corporation 2016, 2017 All Rights Reserved.
#US Government Users Restricted Rights - Use, duplication or
#disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#
# change to your deployment name
CAMDEPLOY=cam-clu038
#
export tiller_version=$(docker images | grep icp | grep tiller | head -1 | awk '{print $2}' | sed
's|v||g')
export tiller_icp=$(echo $tiller_version | awk -F'[' '{print $3}' | awk -F'[' '{print $2}')

# Added per CAM 3.2 documentation
kubectl delete clusterservicebrokers.servicecatalog.k8s.io cam-broker-services

if [ ! -z $tiller_icp ]; then
    echo helm del $CAMDEPLOY --purge --tls
    helm del $CAMDEPLOY --purge --tls
else
    echo helm del $CAMDEPLOY --purge
    helm del $CAMDEPLOY --purge
fi

echo kubectl -n services delete -l release=$CAMDEPLOY job
kubectl -n services delete -l release=$CAMDEPLOY job

echo "Waiting for Pods to terminate"
kubectl -n services get -l release=$CAMDEPLOY pod
pods=$(kubectl -n services get -l release=$CAMDEPLOY pods | grep -)
while [ "${pods}" ]; do
    sleep 2
    kubectl -n services get -l release=$CAMDEPLOY pod
    pods=$(kubectl -n services get -l release=$CAMDEPLOY pods | grep -)
done
echo "All pods terminated"

echo kubectl delete -n services pvc cam-logs-pv
kubectl delete -n services pvc cam-logs-pv

echo kubectl delete -n services pvc cam-mongo-pv
kubectl delete -n services pvc cam-mongo-pv

echo kubectl delete -n services pvc cam-terraform-pv
kubectl delete -n services pvc cam-terraform-pv

echo kubectl delete -n services pvc cam-bpd-appdata-pv
kubectl delete -n services pvc cam-bpd-appdata-pv

echo kubectl delete pv cam-logs-pv
kubectl delete pv cam-logs-pv

echo kubectl delete pv cam-mongo-pv
kubectl delete pv cam-mongo-pv

echo kubectl delete pv cam-terraform-pv
kubectl delete pv cam-terraform-pv

echo kubectl delete pv cam-bpd-appdata-pv
kubectl delete pv cam-bpd-appdata-pv

echo sleep 15, waiting for things to settle down
sleep 15
```

Monitor progress,

```
# kubectl -n services get pod | grep cam
```

If any pods show up, wait a bit for them to terminate. If not, delete using:

```
kubectl delete pod --grace-period=0 --force --namespace services -l release=cam-clu038
```

If you are not reinstalling CAM, delete the docker secret,

```
kubectl delete secret cam-clu038 -n services
```

5.3 Troubleshooting

Useful for troubleshooting:

Display cam pods that do not have Status of **Running** or **Completed**. This assumes you have deployed CAM into the *services* namespace.

```
# kubectl get pod -n services | grep cam | grep -v Running | grep -v Completed
```

Display details and logs of a pod:

```
# kubectl -n services describe pod <name of pod>
# kubectl -n services logs <name of pod>
```