Deploying Oracle Database with IBM Cloud Automation Manager and IBM PowerVC

December 24, 2019

Stephen R. Poon
Ralf Schmidt-Dannert

IBM NA Power Technical Team - Oracle Solutions

This page intentionally left blank.

Table of Contents

1 Abstract

This paper builds on the Proof of Technology described in “Deploying Oracle® Database as a Service with IBM Cloud PowerVC Manager 1.3.1” (see References, page 101) where we demonstrate how IBM Cloud PowerVC Manager (PowerVC) technology enables the implementation of Database-as-A-Service (DBaaS) for an Oracle® database.

In the first section of this paper we demonstrate how to utilize IBM Cloud Automation Manager (CAM), instead of (PowerVC), to provide the control point and end-user interface for DBaaS for an Oracle database, while re-using the existing PowerVC image developed in the earlier project.

We then expand the scope and illustrate the workload orchestration capabilities of CAM by creating a deployable service which co-deploys two virtual machines (VMs) - a database server and an application server. The result of that orchestrated deployment is a running and pre-loaded Oracle database and a workload driver in the second VM ready for the end-user to connect to and drive simulated transactions against the database.

Is is assumed that readers of this paper have at least some familiarity with IBM Cloud Private v3.2, IBM Cloud Automation Manager v3.2, IBM Power servers, IBM Power Virtualization Center (PowerVC) as well as AIX and Linux system administration. The creation of the deployable image for the Oracle database server, as described in the sister white paper, requires working knowledge of Oracle Database 12c software.

2 Legal Disclaimer

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 program or technical support, if any.

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.

3 Comments / Feedback

Version: 1.2

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

4 Introduction

Application containerization has become a major trend in software development and deployment. A late 2017 survey conducted by IBM suggests that adoption is happening very quickly, revealing that 59% of adopters improved application quality and reduced defects as a result.

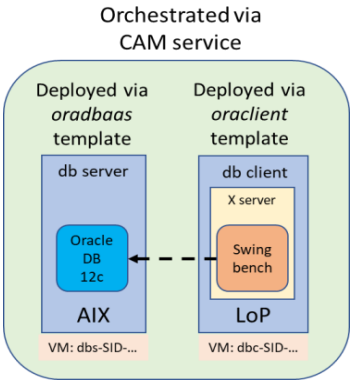
In support of this new development and deployment paradigm, IBM has created the IBM Cloud Private offering which is based on Kubernetes. Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation.

In 2019 IBM bought Red Hat®, Inc. and made the decision that Red Hat OpenShift will be the platform of choice to develop and deploy containerized applications going forward and replaces IBM Cloud Private in the future. Red Hat OpenShift is based on Kubernetes as well.

This white paper utilizes CAM 3.2.1.2 deployed into an IBM Cloud Private 3.2.1 environment on IBM Power servers, but a deployment of CAM into a Red Hat OpenShift 3.11 on Power environment would provide the same functionality and features. The latest IBM Cloud Pak for Multicloud Management version includes CAM and is available for Red Hat OpenShift 3.11 on IBM Power architecture today.

While more and more containerized applications are being developed and deployed, many critical applications today are still virtual machine (VM) based. For mixed environments, with both VM and containerized applications, IBM Cloud Automation Manager (CAM) is a Docker container application that allows you to deploy existing VM-based applications in a multi-cloud environment. Since CAM is an application that runs on IBM Cloud Private, or Red Hat OpenShift, you can integrate management of new container-based services with existing mission critical workloads and provide a single software and services catalog for both containerized and VM-based applications.

The purpose of this paper is to illustrate how CAM can be utilized to orchestrate the deployment of two VMs from within IBM Cloud Private. All relevant configuration and setup steps are described in detail so the reader can reproduce the deployment and implementation steps in his own environment.
The graphic below illustrates the end state after a successful service deployment from CAM.



The deployed environment consists of:

- A database server (db server) running Oracle Database 12c on AIX, using the image described in the IBM Whitepaper entitled “Deploying Oracle Database as a Service with IBM Cloud PowerVC Manager 1.3.1” (DBaaS whitepaper, see References, Section 11) and,
- A database client (db client) consisting of a Linux on Power (LoP) VM running the Swingbench load generator over Oracle Instant Client against the database server

After deployment, workload data is automatically loaded into the db server's database from the db client. Once the load is complete, the user/requester can access the db client to run Swingbench load generator.
The only required input from the end-user, and even that is optional, to deploy this full functionality via a “single-click” service deployment is to provide the database name to be configured.

Note: The Swingbench tool was developed by Dominic Giles, Oracle and is freely available at:
http://www.dominicgiles.com/swingbench.html

The following are pre-requisites for this proof of technology and the setup or configuration therefore are not covered in this white paper.

- IBM® Cloud Power® VC Manager (PowerVC) has been installed and configured and can successfully manage, capture, deploy, start, stop, etc. images and virtual machines onto managed IBM Power servers and storage.
- A virtual machine image with Linux on Power as OS has been created and is deployable in PowerVC.
- A virtual machine image with AIX and Oracle database software has been created and is deployable in PowerVC. See "Deploying Oracle Database as a Service with IBM Cloud PowerVC Manager 1.3.1" (see References, page 101) for details on how to create this image.
- An NFS server with Oracle client 12c software and the swingbench install zip-file accessible during deployment of the *oraclient* template.
- IBM Cloud Private (ICP) and IBM Cloud Automation Manager (CAM) have been installed. Please see "Cloud Automation Manager CE Installation" white paper as reference for the installation of CAM CE 3.2.1.2 into IBM Cloud Private 3.2.1 as utilized in this proof of technology.
- A GitHub repository to be used to store the CAM designer templates.

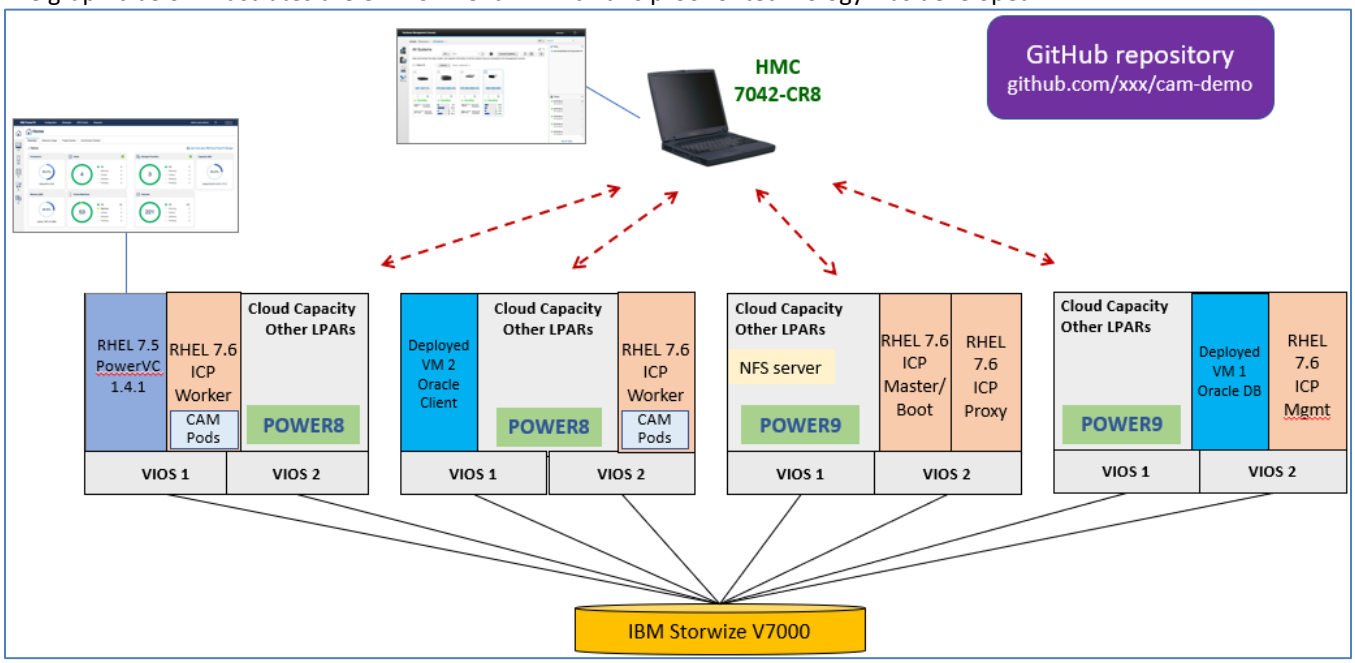
In the remainder of this white paper we provide:

- A description of the hardware / software environment in which this project was developed,
- A brief overview of IBM Cloud Private and IBM Cloud Automation Manager,
- A detailed description of how to create the Oracle DBaaS (*oradbaas*) template with the IBM Template Designer, part of CAM software package, and deploy this template via IBM Cloud Automation Manager.
- A description on how to create the Oracle client (*oraclient*) template the same way, but not in as much detail. Template file content is available in the appendix.
- A detailed description of how to create and deploy a service, *Oracle 12C on AIX with Swingbench Client on LoP*, in CAM using the two new templates *oradbaas* and *oraclient*.

Note that all related documentation and source code / scripts have been made available in a public GitHub repository. For latest versions of this document, script changes or new features please refer to that repository at: Actual examples on how to develop CAM templates and ICP orchestrated services can be found on GitHub at: https://github.com/ppc64le/devops-automation/tree/master/terraform/oracle.

5 Lab Environment

The graphic below illustrates the environment in which this proof of technology was developed:



- Two POWER9 processor-based servers and two IBM POWER8 processor-based servers, all managed by an IBM Hardware Management Console (HMC),
- An IBM Storwize V7000 providing virtualized storage,
- Two fiber-based SAN switches connecting the storage to the servers (not shown in the graphic),
- The Power servers are fully virtualized and each was configured with two Virtual I/O Servers (VIOS),
- IBM Power Virtualization Center (PowerVC) as the OpenShift endpoint for VM deployments via Terraform,
- The IBM Cloud Private environment, with IBM Cloud Automation Manager (CAM) deployed.

The software versions of ICP, CAM and PowerVC in our lab environment at the time this paper was written were:

- IBM Cloud Private Enterprise Edition Version 3.2.1
- IBM Cloud Automation Manager Version 3.2.1.2
- IBM PowerVC Version 1.4.2.1

6 About IBM Cloud Private and IBM Cloud Automation Manager

IBM® Cloud Private is an application platform for developing and managing containerized applications across hybrid cloud environments, on-premises and public clouds. It is an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image registry, a management console, and monitoring frameworks.

Many applications today are still VM-based. For these environments, IBM Cloud Automation Manager (IBM CAM) is an application that runs in a Docker container on IBM Cloud Private or Red Hat OpenShift. CAM allows you to deploy existing VM-based applications in a multi-cloud environment, add any AIX, IBM i, or Linux VM-based application to the IBM Cloud Private or Red Hat OpenShift catalog, integrate new services with existing mission critical workloads, and achieve a single catalog with coordinated orchestration.

Cloud Automation Manager uses Terraform, an open-source tool created by HashiCorp, to handle provisioning of cloud and infrastructure resources. These resources are described to Terraform using a high-level configuration language, stored in configuration files, or templates. From these templates, Terraform generates an execution plan describing what it will do, and executes it to build the described infrastructure.

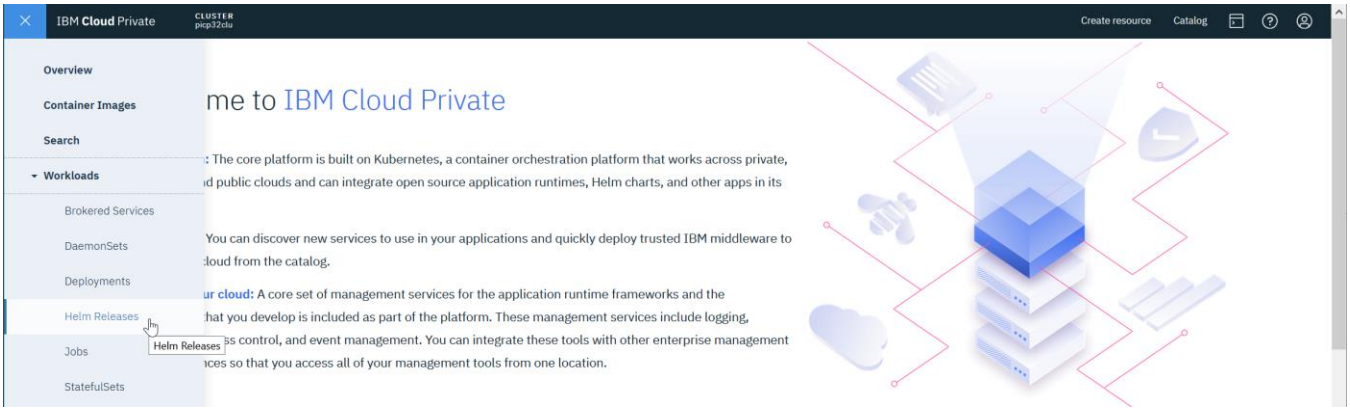
IBM CAM includes two graphical user interface (GUI) tools, accessed through a web browser, for developing templates and services:

- Template Designer is used to develop Terraform templates featuring a "drag and drop" interface. Template Designer can also be installed on a local workstation using Docker Compose. Templates created using Template Designer are "pushed" to a Git repository and published or imported into the CAM Template Library.
- Service Composer allows the creation of orchestrated services, via drag-and-drop interface, from CAM (Terraform) templates and/or Kubernetes Helm charts based on a designed flow sequence or decision tree. Service Composer is part of the CAM Service library.

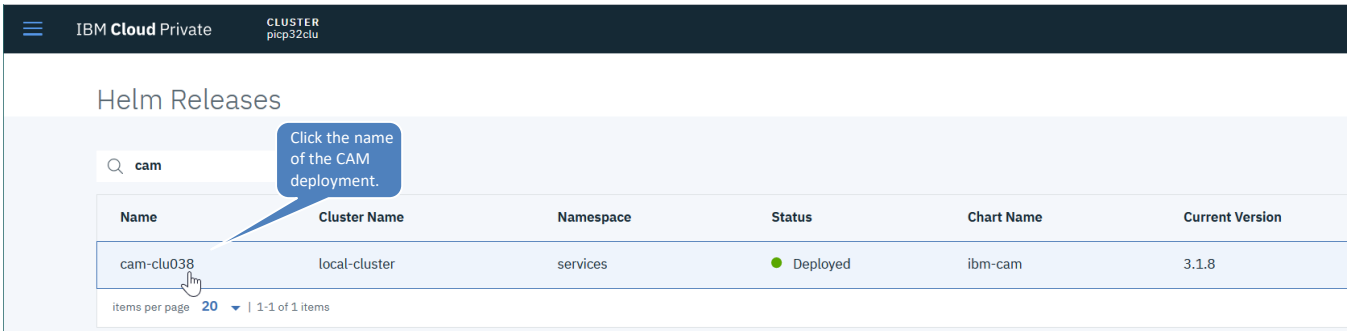
7 Working with CAM

7.1 Accessing CAM

The access URL to the CAM user interface can be found in IBM Cloud Private. Access the hamburger menu (three bars on the upper left-hand corner), select Workloads -> Helm Releases.



Search for **cam**, or the name you used when you installed the CAM Helm Release, and click the name of the CAM deployment.



Scroll all the way down to find the section about Notes and CAM UI access. In our example,

Notes

```
#
# Licensed Materials - Property of IBM
# 5737-E67
# (C) Copyright IBM Corporation 2016-2019 All Rights Reserved.
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
Note: It may take some time for Cloud Automation Manager to initialize, please be patient.
You can monitor the status of the pods with the following command:
  kubectl get -n services pods

Once all the pods are running, you can run the helm tests:
  helm test cam-clu038 --tls

To access the UI:
  export SERVICE_PORT=$(kubectl get service -n services cam-proxy -o jsonpath='{.spec.ports[0].nodePort}')
  echo https://\<ICP_PROXY_IP\>:$SERVICE_PORT/cam
```

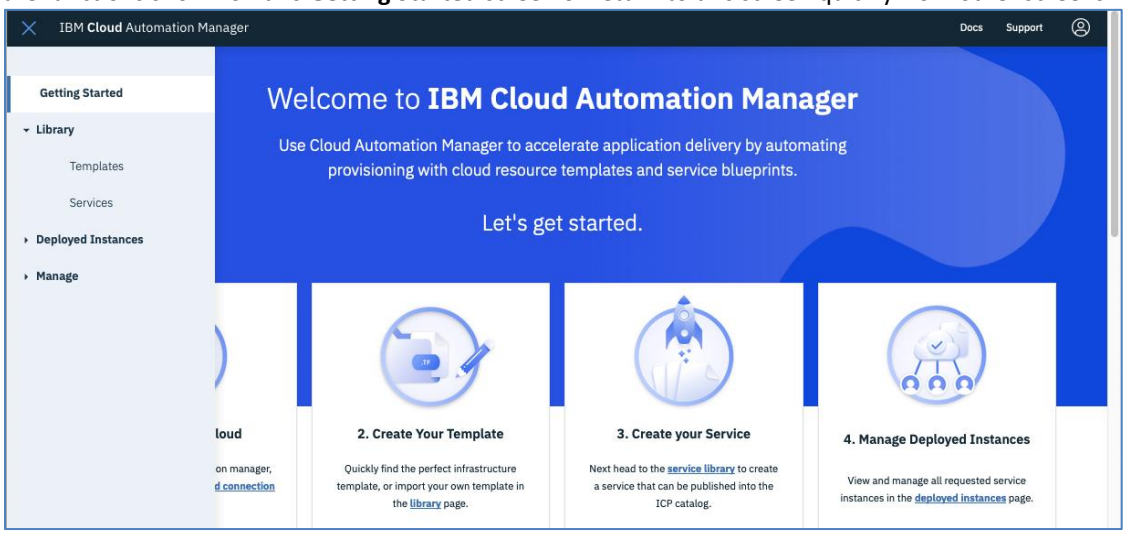
Typically,

https://<Hostname or IP address of ICP proxy node>:30000/cam

Use a browser to access CAM with the appropriate URL determined above and log in.

Once logged in, the **Welcome** screen or **Getting Started** screen is displayed as shown in the picture below.

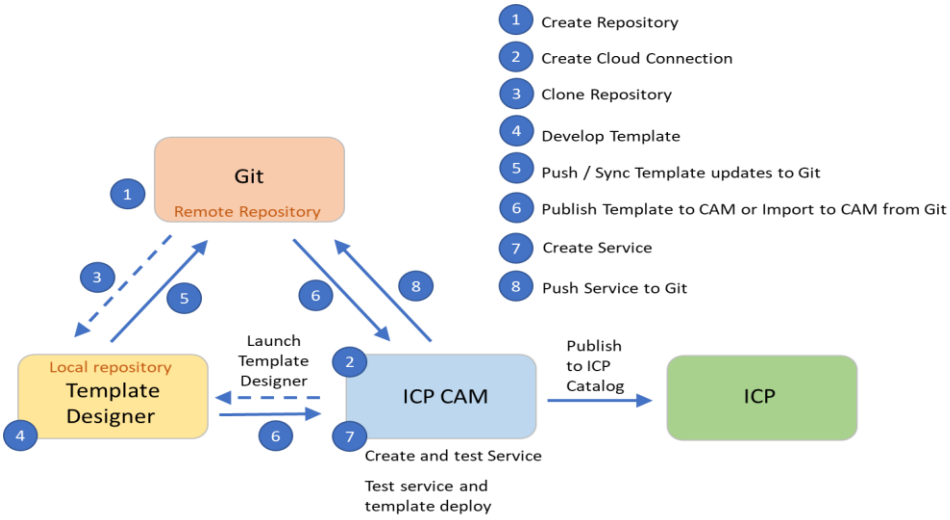
Note that the hamburger menu can be used to bring up the left-hand panel menu below. This menu can be used to access the functions shown on this **Getting Started** screen or return to this screen quickly from other screens.



7.2 Development Workflow - Overview

Templates and content libraries that are created in the Template Designer are stored and versioned in a Git repository. GitHub, GitLab and Bitbucket server are supported in CAM. For this paper, we used GitHub and have included the steps we performed to set up the repository in Appendix, Section 10.1, "Working with GitHub".

The picture below illustrates the sequence of steps we followed to prepare for, develop, test and deploy the two CAM templates and the CAM service for this proof of technology.



We started with the creation of a new GitHub repository, *cam*-demo, for this project under an existing GitHub account, then cloned that new repository (empty at this point) to the Template Designer. We then used Template Designer to create and modify the Terraform files defining the *oradbaas* and *oraclient* templates, committed our changes and pushed the updated files to the GitHub repository. Once that was done, we published the new templates (using Template Designer) to CAM, where we then tested / successfully deployed the *oradbaas* template.

Once the templates had been tested, we used CAM Service Composer to create and test the CAM Service *Oracle 12C on AIX with Swingbench Client on LoP*. This new service is an orchestration of *oradbaas* and *oraclient* templates. We then pushed the new service to the GitHub repository and published it to ICP.

Pushing templates and services to a GitHub repository serves as backup and facilitates distribution and collaboration with other interested parties. Templates and services can be easily imported into other CAM deployments, saving the effort of recreating templates and services in CAM.

7.3 Terraform Console

Git provides change control and management that may not be needed when first starting Terraform template development. One way to quickly learn, create and test early prototypes of Terraform files is to install Terraform on your Windows, Mac or Linux workstation.

This blog by Joe Cropper, IBM, describes how to install and run Terraform on your workstation.
https://developer.ibm.com/powervc/2017/06/29/infrastructure-code-terraform-meets-powervc/

One useful function of locally installed Terraform software is the Terraform console, which we used to test and verify Terraform functions, including syntax checking. For example, on our worstation, the following was used to test the timestamp function used in our *oradbaas* template.
Note that the console functionality, at least on windows, seems to be broken with current version 0.12, but works with version 0.11.14 of Terraform.

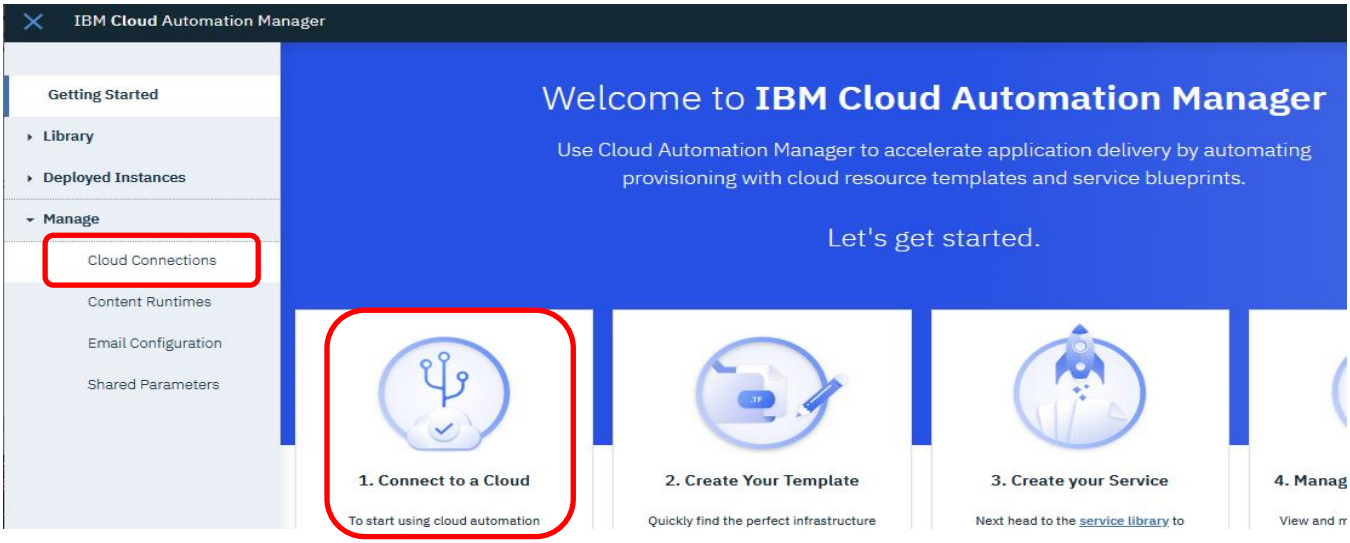
```
$ terraform console
> timestamp()
2019-05-23T18:41:34Z
> replace(replace(replace(replace(substr(timestamp(),0,19),"-",""),"T",""),"T",""),":","")
20190523184013
> replace(replace(replace(substr(timestamp(),2,17 ),"-",""),"T","_"),":","")
190911_172625
> exit
$
```

7.4 Create Cloud Connection

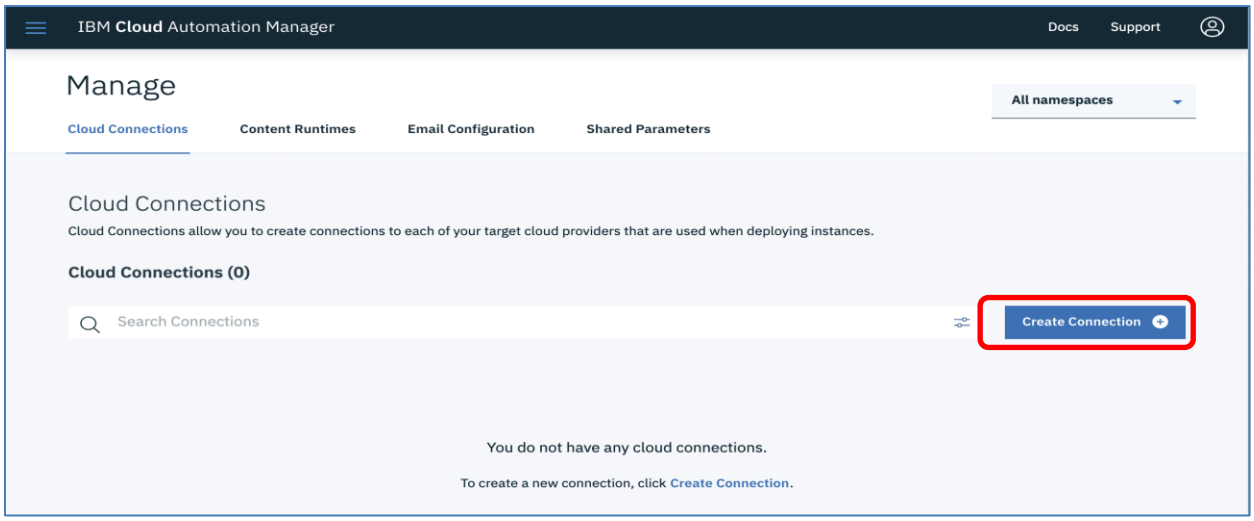
CAM is designed to work efficiently in a multi-cloud environment with multiple cloud providers. To simplify the use of a specific cloud provider CAM provides the concept of a Cloud Connection which stores the detailed connection related parameters (e.g. OpenStack user, password, auth_url, region, etc.). At deploy time – template or service - you can then simply specify which Cloud Connection to use without having to specify all the cloud provider connection details in the deployment.

To create a new Cloud Connection, click the left-side navigation panel, click **Manage** as shown on the screen below to expand the submenu and select **Cloud Connections**.

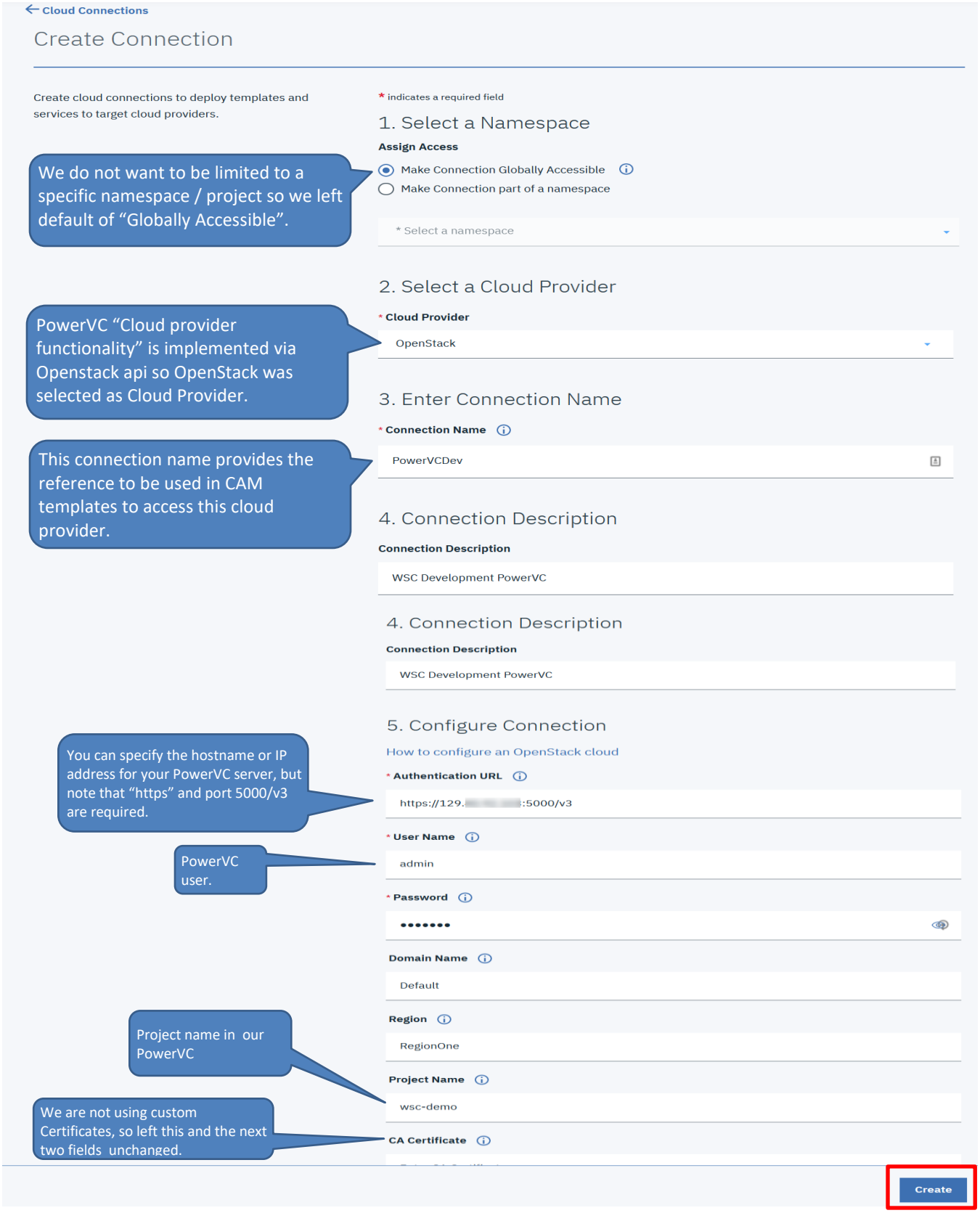
Alternatively, you can click **1. Connect to a Cloud** link on the Welcome screen



This switches the focus to the Manage (connections) screen where you click the **Create Connection** button,

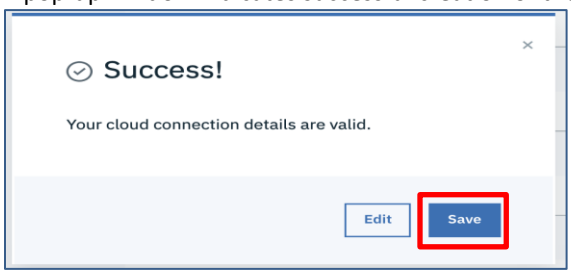


Fill in the Create Connection screen as described below. Important for a PowerVC environment is to select OpenStack as **Cloud Provider**. We also defined the new cloud connection to be available in all namespaces – Globally Accessible. Note that in this case the Select a namespace field is not editable.

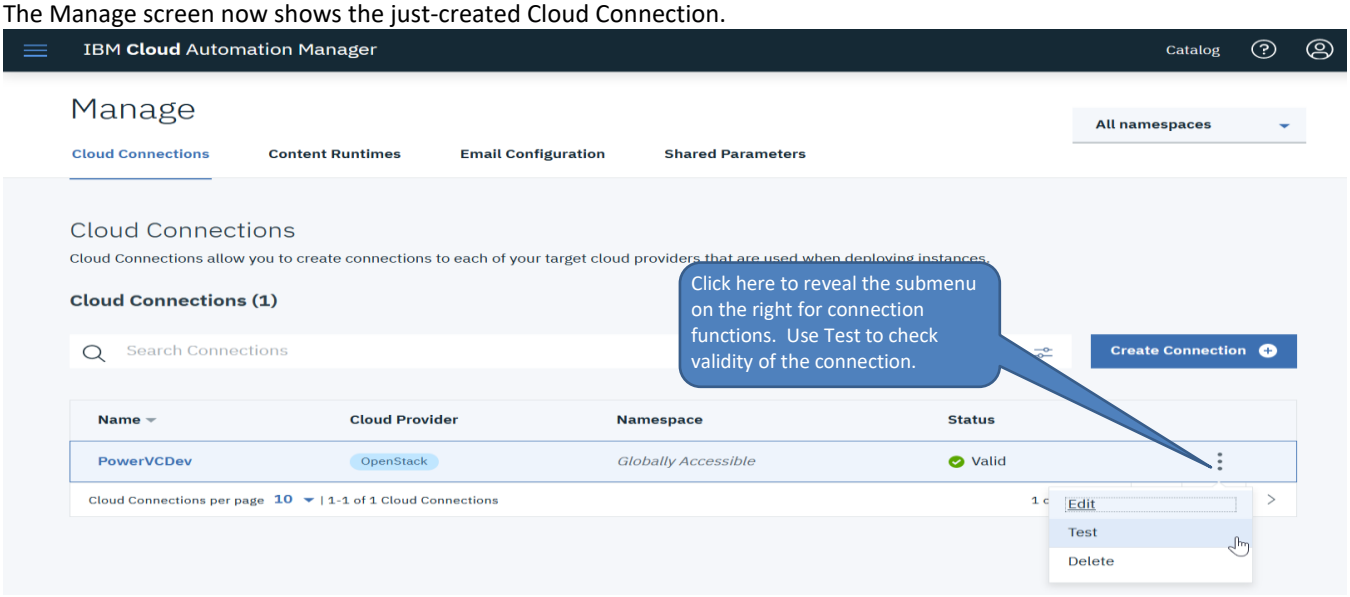


When complete, click the **Create** button.

A pop-up window indicates successful creation of the connection. Click the **Save** button.



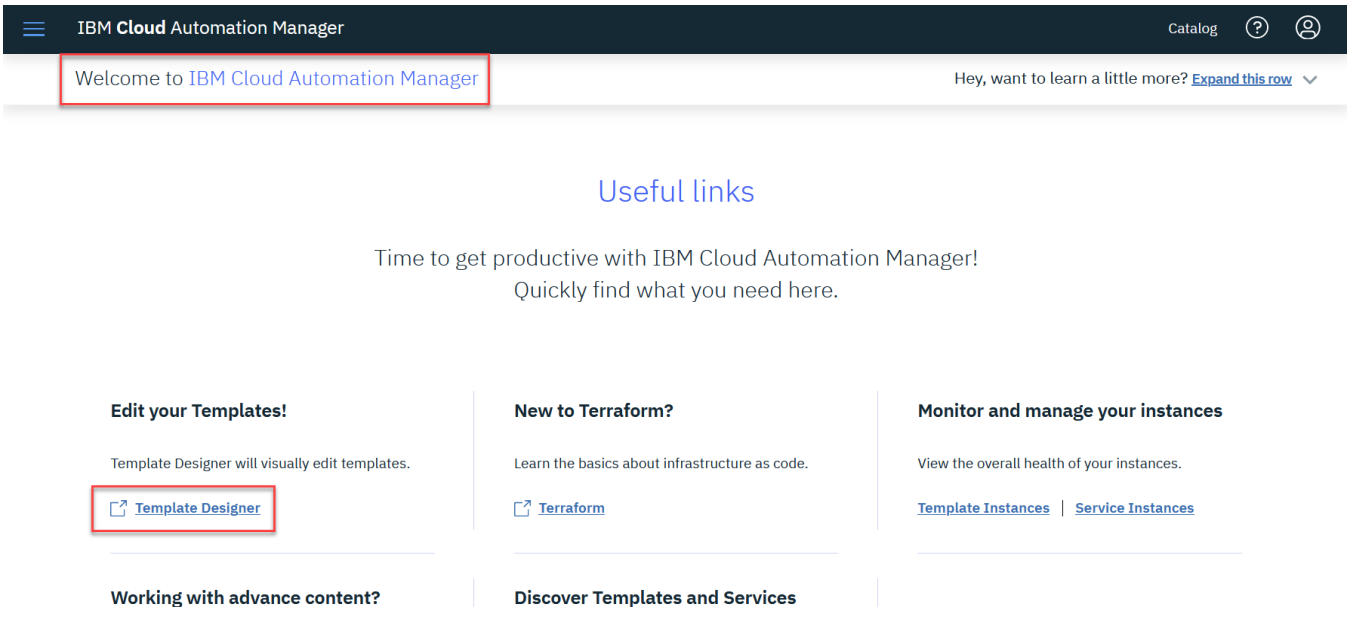
The Manage screen now shows the just-created Cloud Connection.



7.5 Prepare Template Designer

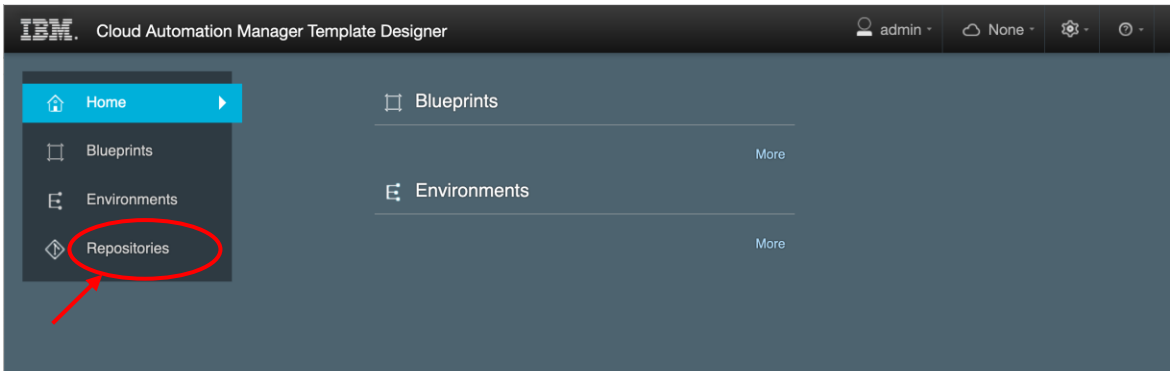
In this section we describe the steps to configure Template Designer before starting to create the first template.

From the **Getting Started** screen, scroll down and launch **Template Designer**, as shown in the graphic below. Alternatively, click the hamburger menu, select Library -> Templates. Then click the **Create Template** button.



After going through the ICP login screen again, template Designer will launch in a separate browser tab.

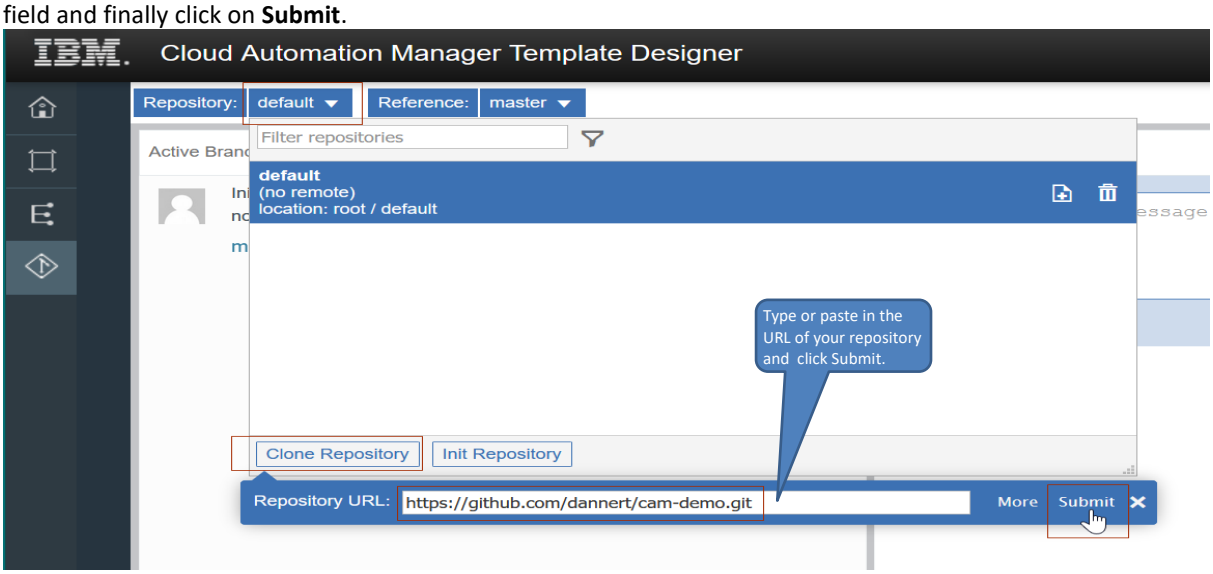
On the Template Designer browser tab, click Repositories (circled below).



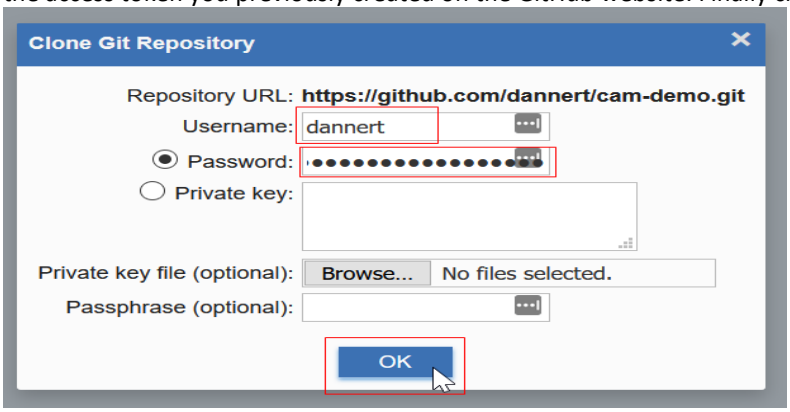
7.5.1 Create Repository

This section assumes that you have already created a Git repository for this project on github.com and have the URL and access token available. For reference see: Working with GitHub (Section: 10.1)

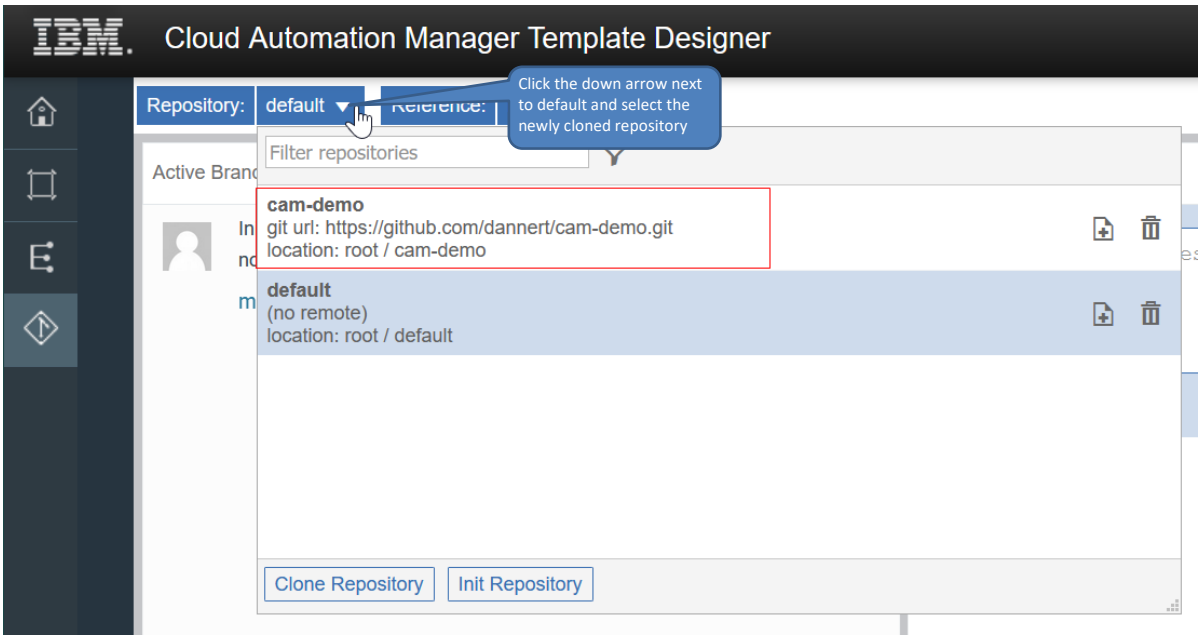
Click the down arrow next to Repository **default**, then click **Clone Repository**, fill in the GitHub repository URL into the input field and finally click on **Submit**.



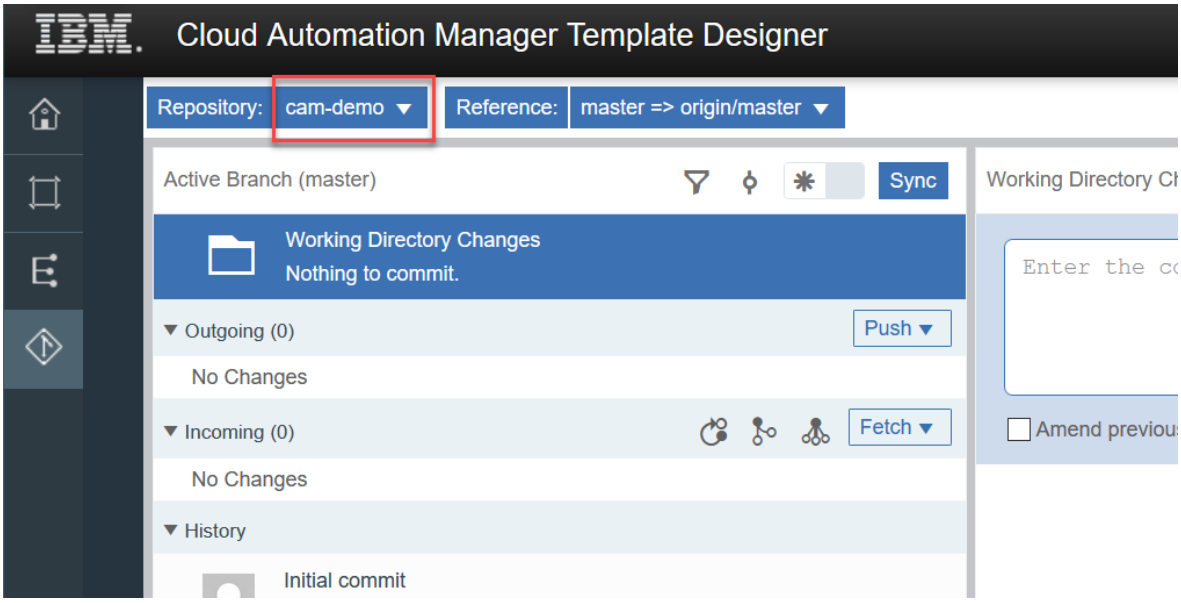
After clicking **Submit** the following window pops up. Enter your GitHub user ID into **Username** and, into the **Password** field, the access token you previously created on the GitHub website. Finally click **Ok**.



To set this new repository as the default repository click again onto the “default” drop-down and select the new project as the new default as shown below.



Now the Repository field shows **cam-demo** as selected. Note that we are referring to the master branch, but you could modify that to a different branch/fork for a project via the Reference field.



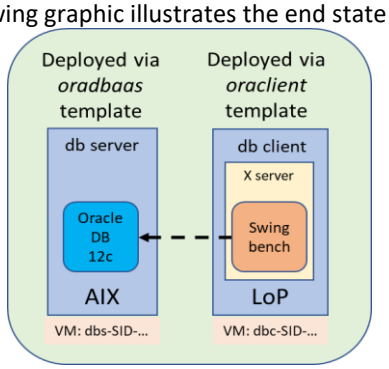
8 Developing our Templates

In this section we describe how to create the two CAM templates used in this project to achieve the following:

a) Deploy an AIX VM running Oracle Database 12c (dbserver). The Oracle database name (dbname) is entered as an input parameter and used to create the database instance in the deployed VM. This template can be deployed by itself. We will call this template *oradbaas*.

b) Deploy a Linux on Power VM, set up Oracle Instant Client and the Swingbench load generator in the deployed VM (dbclient). dbclient will wait for dbserver to complete post-deployment customization, which includes the database creation and startup, before loading the Swingbench Order Entry schema into the Oracle database. This template cannot be deployed by itself as it requires input parameters from the deployment of the oradbaas template. We will call this template *oraclient*.

The following graphic illustrates the end state after an orchestrated deployment of both templates via a service.



Note that it is possible to create only one template to deploy both VMs, but that would preclude the re-use of either template in other contexts / services. Separating the deployment into two different templates also allows us to demonstrate how required or dependent data is passed between several template deployments when deployed as an orchestrated service. An example of such required data exchange is the passing of the IP address of the dbserver and the dbname of the Oracle database to the dbclient. Those values are needed by dbclient to be able to successfully connect to the Oracle database on dbserver.

We will only describe the two-template design in this paper.

8.1 *oradbaas* Template Design

In this section we provide the step-by-step instructions on how to create the *oradbaas* template.

8.1.1 *oradbaas* template

The *oradbaas* template relies on the Oracle DBaaS image to be available in PowerVC for deployment. Details on how we created that image can be found in IBM Technical Whitepaper, *Deploying Oracle Database as a Service with IBM Cloud PowerVC Manager 1.3.1*. This image utilizes Openstack Cloud Init and accepts the Oracle database name (dbname) as an input parameter passed to the post-deployment customization scripts that are imbedded in the captured image.

The following provides a quick overview of the key configuration blocks in main.tf. Please find the actual settings used later in this section!

The *oradbaas* template has two key configuration blocks:

- openstack provider, and
- openstack compute resource (openstack_compute_instance_v2).

The **default** openstack provider configuration block in file main.tf contains the following configuration arguments:

```
provider "openstack" {
  user_name   = "<User id to log on to OpenStack server, e.g. PowerVC>"
  password    = "<User password>"
  tenant_name = "<PowerVC project, usually ibm-default>"
  domain_name = "<PowerVC domain, usually Default>"
  auth_url    = "<Identity authentication URL of OpenStack server>,
                  e.g. https://<IP address or hostname of OpenStack server>:5000/v3/>"
  insecure =    "<Trust self-signed SSL certificates>"
  version  = "~> 1.2"
}
```

All but the last two of the arguments are already specified in CAM using the Cloud Connection and therefore only the last two arguments are needed in a CAM template.

The compute resource (openstack_compute_instance_v2) requires the following arguments and they were all left as input parameters in our templates.

```
resource "openstack_compute_instance_v2" "<name of compute instance resource>" {
  name       =     "<Name of VM>"
  image_name  =    "<Name of the captured image on PowerVC>"
  flavor_name =    "<Name of the OpenStack flavor, or compute template in PowerVC>"
  user_data =      "<Special scripts processed by cloud-init. Used here for dbname>"
  network {
    uuid = "<ID of the network in PowerVC to attach to the instance>"
  }
}
```

Notes:

- Use PowerVC GUI or CLI to obtain values for image_name / image ID: **openstack image list**
- Use PowerVC GUI or CLI to obtain values for flavor_name / flavor ID (compute templates): **openstack flavor list**
- Use PowerVC CLI to obtain network name / network ID: **openstack network list**

8.1.2 Input and Output Parameters

Template developers should always have re-usability and flexibility in mind when creating a new template. To achieve maximum flexibility for our new template we decided to define all our parameters as template input parameters which could be specified / overridden at deployment time. The table below list the parameters and their meaning for the *oradbaas* template.

Input Parameter	Description
Namespace	This is the ICP / Kubernetes namespace or Red Hat project in which the instance will be deployed.
Instance name	This is the name of the instance being deployed; displayed on the CAM deployed instance screen. Choose any appropriate name.
dbserver_name	This is used to name the PowerVC VM. In our template, we concatenate this parameter to the dbname (from dbserver_user_data), and a timestamp in order to have a name that's unique and one that can be matched with the dbclient VM name. For example, **dbs**-testdb-<YYMM_hhmmss>.
openstack_flavor_name	This is the PowerVC compute template which specifies the compute characteristics of the VM to be deployed. The flavor (OpenStack) or compute template (PowerVC) specifies VM specifications, including number of CPUs, RAM, etc. Alternative option would be to use the flavor id instead.
openstack_image_name	This is the name of the deployable image in PowerVC. Alternative option would be to use image id.
openstack_network_id	This is the network id in OpenStack. Alternatively, we can choose to use network name.
dbserver_user_data	This is used to code cloud-init cloud-config. In this template, we use this to pass the desired Oracle dbname. If not provided, the PowerVC image will take the VM name, truncated to 8 characters.

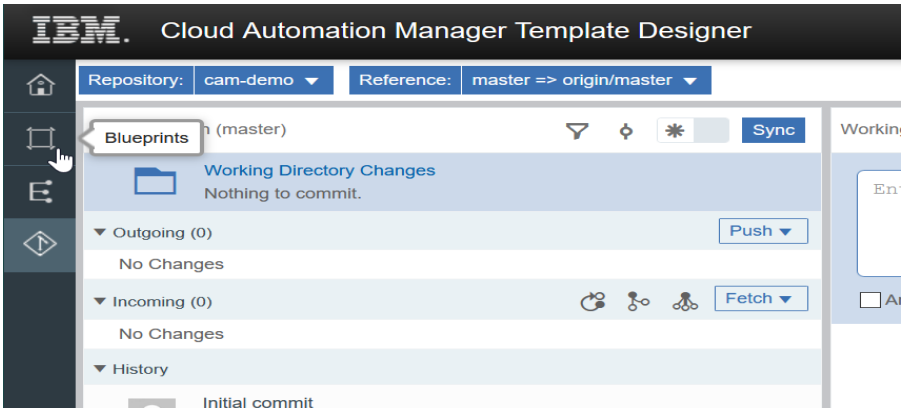
Note that image name, network name and flavor name can be changed via the GUI in PowerVC and it is therefore typically safer to use the respective ID value instead. For this project we wanted to demonstrate both options.

To be able to display deployment related information in the CAM deployment screen, or to make information available to other templates during a service deployment, we defined the following output parameters in this *oradbaas* template.

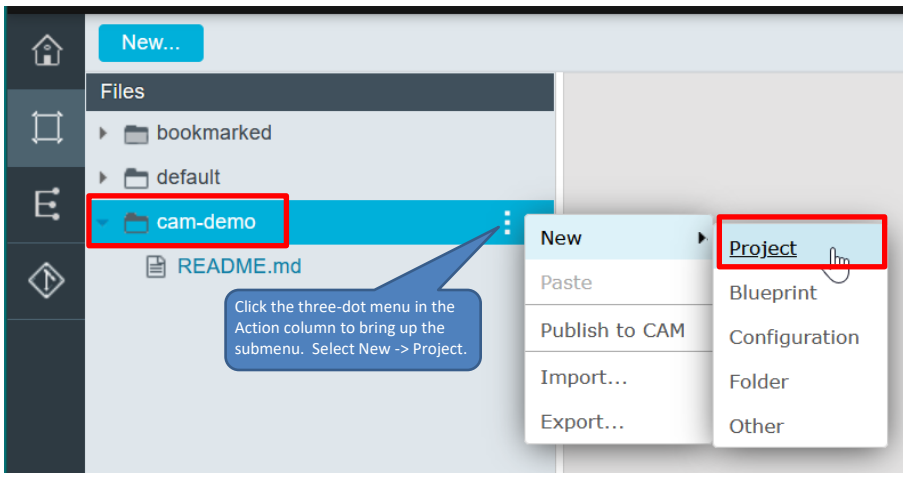
Output Parameter	Description
dbserver_ipaddr	The IP address of the db server. By not specifying a fixed ip address, the IP address is dynamically assigned by PowerVC from the pool of available IP addresses in the network specified by openstack_network_id input parameter.
dbserver_vm_name	This is the VM name in PowerVC, which is dynamically generated at deploy time based on input parameters and a local timestamp.
dbserver_ts	This output parameter is hidden and used to pass a local timestamp to another template. In our example, we pass the timestamp to oraclient for use in generating the VM name for the oraclient. This approach allows us to easily match DB server and client VMs in PowerVC to each other.

8.1.3 Create oradbaas template

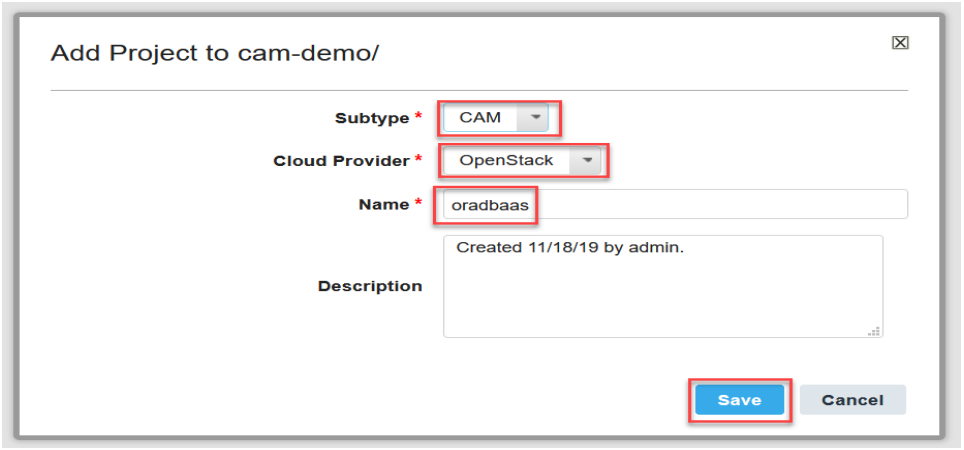
After we prepared Template Designer and defined the input and output parameters we want to use / expose in this template, it is now time to actually create the *oradbaas* template in Template Designer.
In Template Designer click the **Blueprints** icon on the left-hand panel,



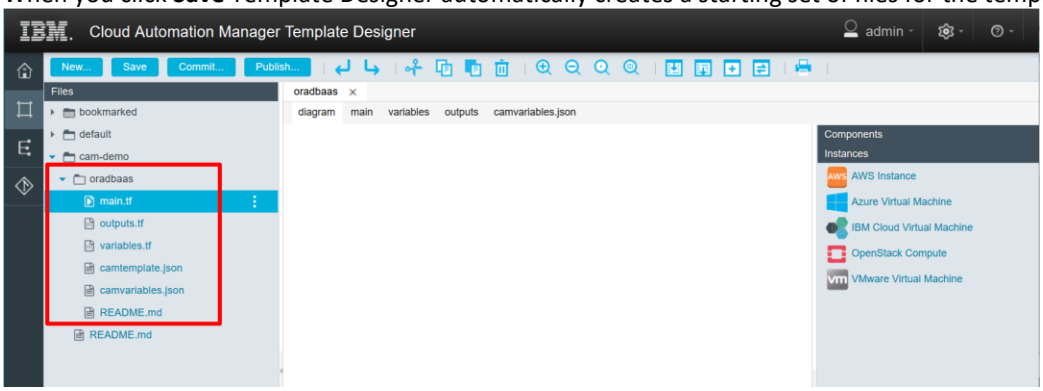
Select the cam-demo repository and click on New -> Project as shown below.



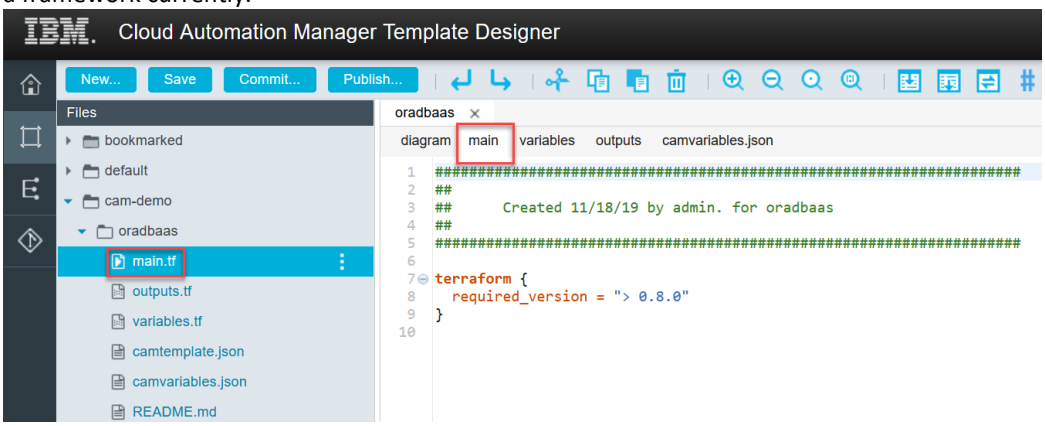
In the pop-up window below, set **Subtype**, **Cloud Provider** and **Name** exactly as shown and then click **Save**.



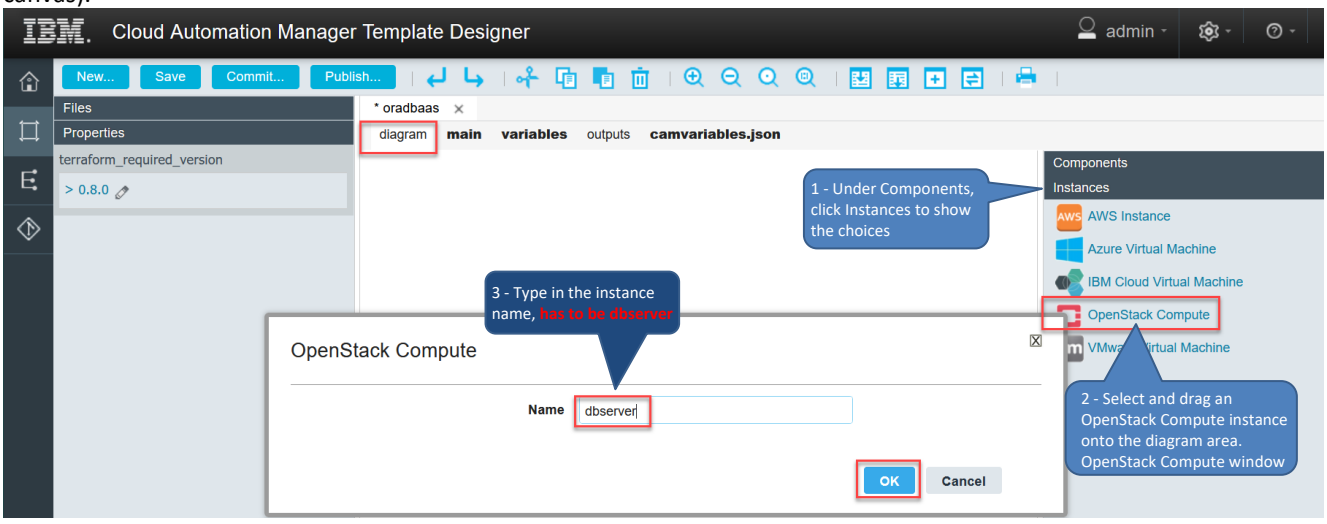
When you click **Save** Template Designer automatically creates a starting set of files for the template.



Select the main.tf entry to select the *oradbaas* template and then the main tab to examine the contents. Note that it's just a framework currently.

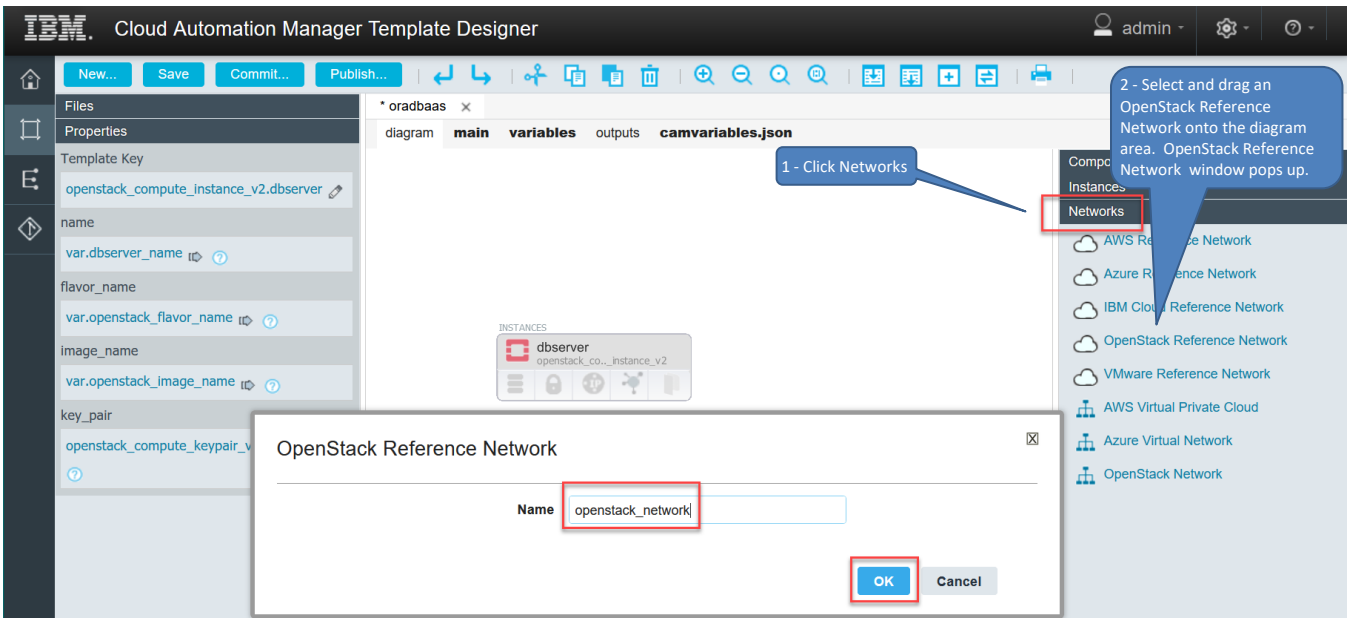


To populate the template with required resources we utilize the drag-and-drop feature provided. Select the diagram tab, select and then drag **OpenStack Compute** instance from the right-hand panel to the empty space in the middle (the canvas).

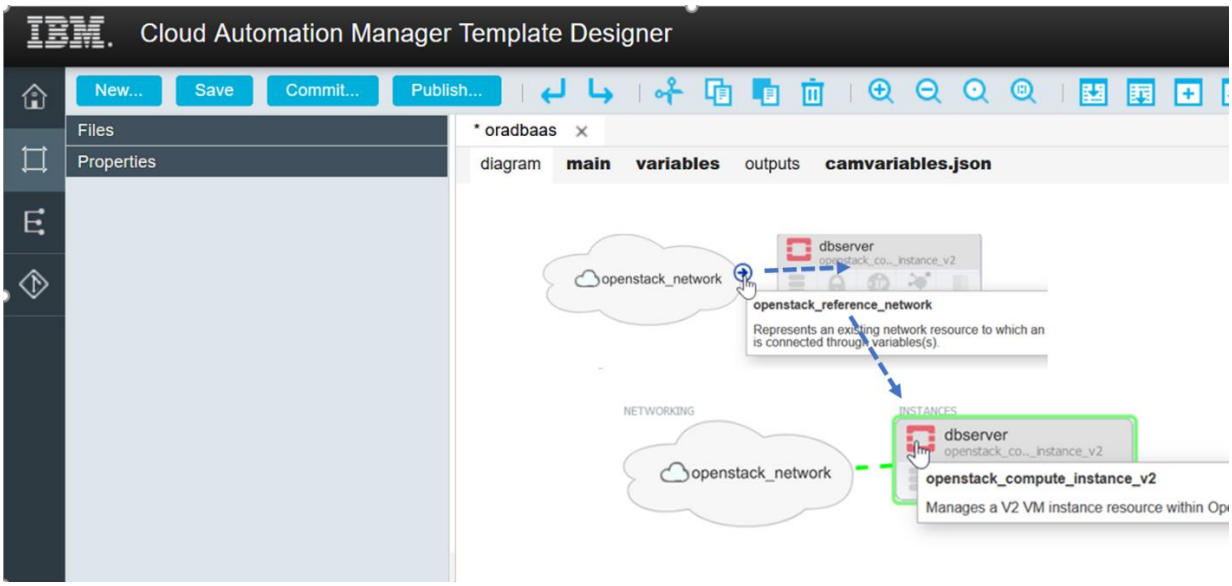


After typing in the OpenStack Compute instance name, **dbserver,** shown in the picture above, click **OK**. Please ensure that you enter the name exactly as shown as otherwise you will not be easily able to utilize the prepared template and service files as shown in the appendix. Note that this instance name becomes part of the generated template variable names, should not include spaces and should be meaningful.

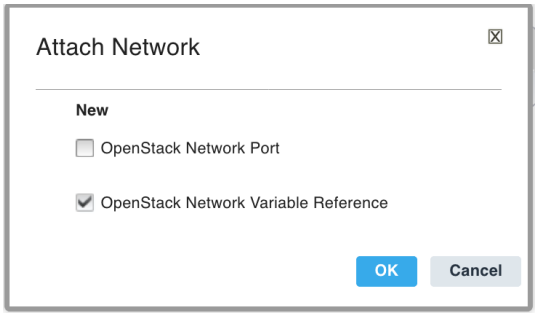
Next, select **Networks** and drag **OpenStack Reference Network,** as shown below, to the canvas. We left the Name unchanged as "openstack_network" and click **OK**.



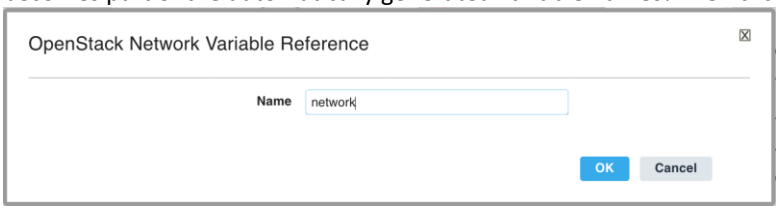
Next hover your cursor over the **openstack_network** figure and point to the right arrow when it appears (a gloved hand appears, not shown on the screen capture), hold down the left mouse button and drag to the dbserver instance. Move the cursor around until the dbserver instance box's border turns green, as shown below, and then release the mouse button.



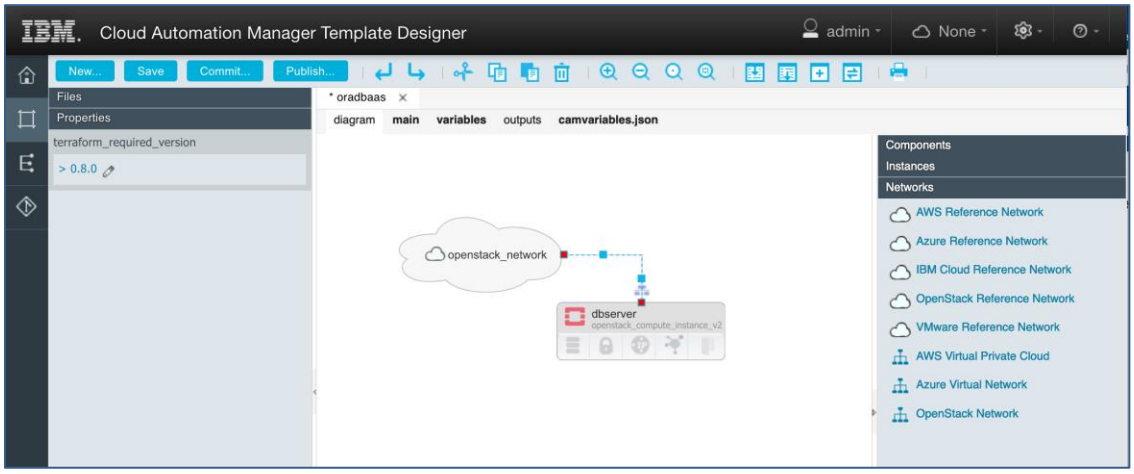
The Attach Network window pops up. **Check** OpenStack Network Variable Reference and click OK.



In the next pop-up window, OpenStack Network Variable Reference, set Name to **network** for this exercise. This value becomes part of the automatically generated variable names! Then click OK.



After completing the above steps, the diagram on the canvas looks like this:

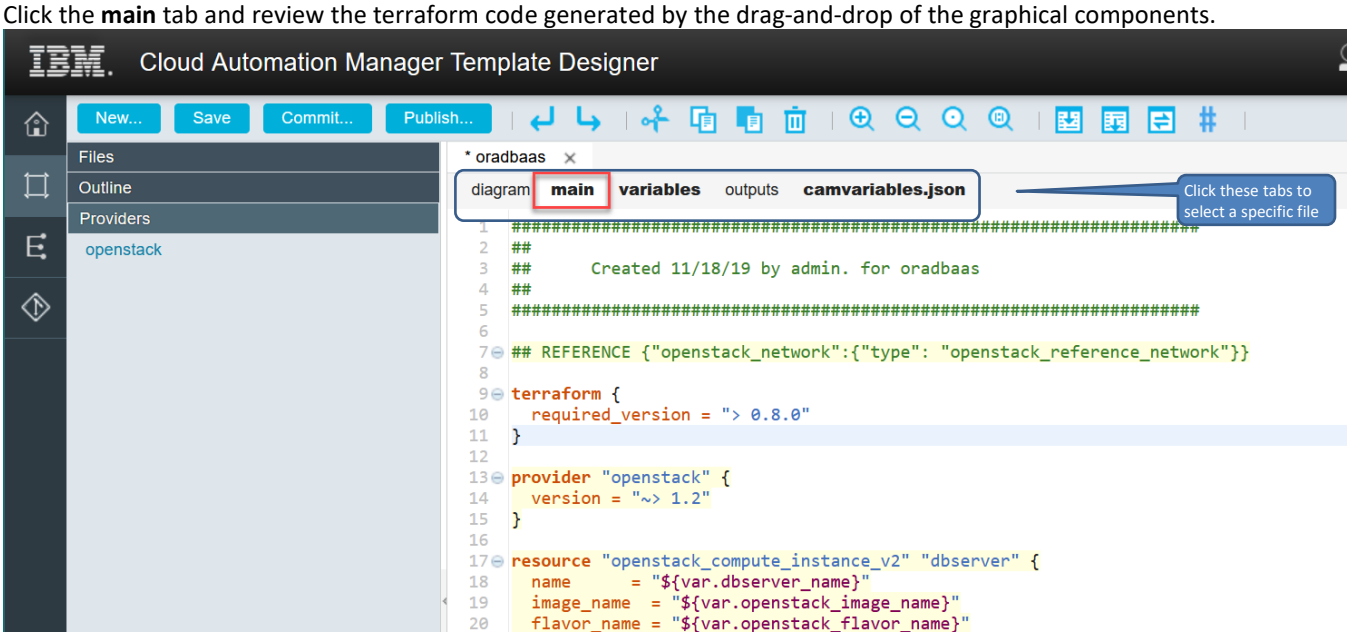


At this point Template Designer has updated the files defining this template, three terraform files (*.tf) and two metadata files (*.json).

For more information about the template structure, refer to:
https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/cam_struct_template.html

The next step is to modify these files to suit our specific requirements.

Click the **main** tab and review the terraform code generated by the drag-and-drop of the graphical components.



We will first modify the main.tf file. Click the main tab as indicated in the picture above. Note that you can find a copy of the file in the appendix.

File: main.tf

Before

```
## REFERENCE {"openstack_network":{"type": "openstack_reference_network"}}

terraform {
  required_version = "> 0.8.0"
}

provider "openstack" {
    version = "~> 1.2"
}


resource "openstack_compute_instance_v2" "dbserver" {
  name      = "${var.dbserver_name}"                          1
  image_name  = "${var.openstack_image_name}"
  flavor_name = "${var.openstack_flavor_name}"
  key_pair  = "${openstack_compute_keypair_v2.auth.id}"       2
  network {
    uuid = "${var.openstack_network_id}"
  }
}

resource "tls_private_key" "ssh" {
    algorithm = "RSA"
}                                                             2

resource "openstack_compute_keypair_v2" "auth" {
    name = "${var.openstack_key_pair_name}"
    public_key = "${tls_private_key.ssh.public_key_openssh}"
}
```

After

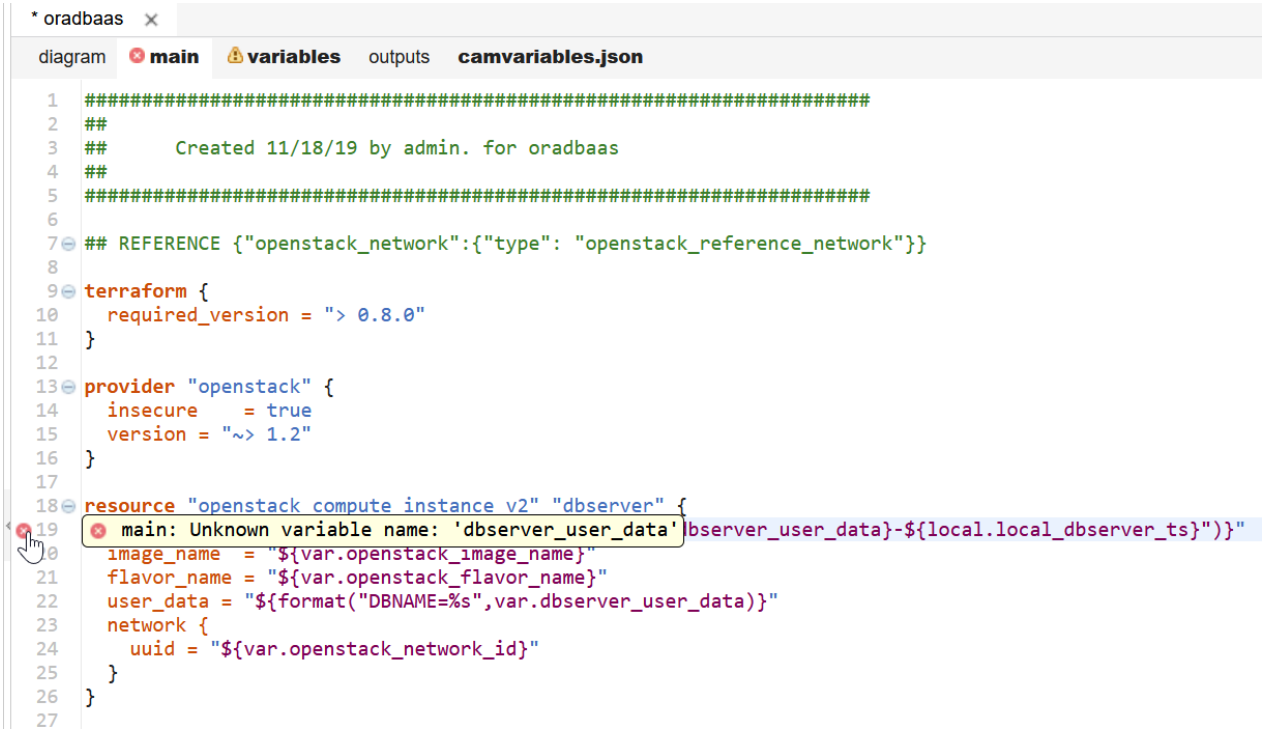
```
## REFERENCE {"openstack_network":{"type": "openstack_reference_network"}}

terraform {
  required_version = "> 0.8.0"
}

provider "openstack" {
  insecure    = true                                          3
  version = "~> 1.2"
}

resource "openstack_compute_instance_v2" "dbserver" {
  name      = "${format("${var.dbserver_name}-${var.dbserver_user_data}-      1
${local.local_dbserver_ts}")}"
  image_name  = "${var.openstack_image_name}"
  flavor_name = "${var.openstack_flavor_name}"
  user_data = "${format("DBNAME=%s",var.dbserver_user_data)}"  4
  network {
    uuid = "${var.openstack_network_id}"
  }
}
```

Notes:

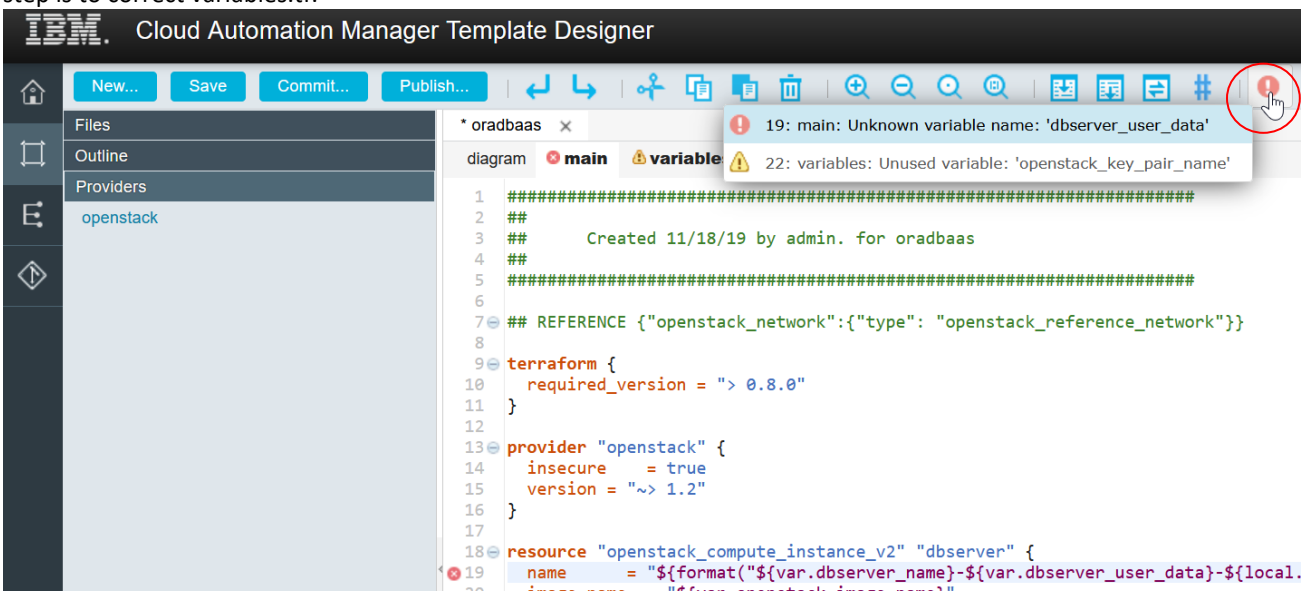
1. This is the VM name in PowerVC. We will append dbserver_user_data and a time stamp to the instance name to make the VM name unique in PowerVC.
   - dbserver_user_data is an input variable used to hold the Oracle dbname
   - the time stamp is calculated in a local variable; see variables.tf.
   - Note that this is all 1 line!
2. Delete tls_private_key and key_pair related statements. These will not be used.
3. Add line to provider stanza.
4. Add user_data to pass cloud-config directives to Openstack cloud-init. Here we pass a text string containing the Oracle Database name (dbname) to the deployed instance in the form “DBNAME=<name>”. The deployed oradbaas image will parse this string to determine the name of the database to create.

Refer to https://www.terraform.io/docs/providers/openstack/r/compute_instance_v2.html for further information on the openstack_compute-instance_v2 resource.

Note that once the above changes are made to main.tf, the Template Designer GUI will indicate errors and warnings as shown below. That is expected!



Hover your pointer over the red exclamation mark symbol as shown below to see all current errors and warnings. The next step is to correct variables.tf.



Click on the **variables** tab and modify the contents as shown below.

File: variables.tf

Before:

```
variable "dbserver_name" {
  type = "string"
  description = "Generated"
}

variable "openstack_image_name" {
  type = "string"
  description = "Generated"
}

variable "openstack_flavor_name" {
  type = "string"
  description = "Generated"
}

variable "openstack_key_pair_name" {
  type = "string"
  description = "Generated"
}

variable "openstack_network_id" {
  type = "string"
  description = "Generated"
}
```

1

After:

```
variable "dbserver_name" {
  type = "string"
  description = " db server VM name prefix. Oracle dbname and a
timestamp will be appended in the PowerVC VM name."
}

variable "openstack_image_name" {
  type = "string"
  description = "PowerVC image name"
}

variable "openstack_flavor_name" {
  type = "string"
  description = "flavor name, or PowerVC compute template)"
}

variable "openstack_network_id" {
  type = "string"
  description = "Network id in PowerVC"
}

variable "dbserver_user_data" {
  type = "string"
  description = "Oracle DB name"
}

locals {
  local_dbserver_ts =
"${replace(replace(replace(substr(timestamp(),2,17),"-
",""),"T","_"),":","")}"
}
```

2 2 2 2 3 4

Notes:

These are variable definitions used in main.tf and must be declared, removed , and made consistent with what's in main.tf.

1. Deleted. No longer needed since they were deleted in main.tf.
2. Change the descriptions as appropriate.
3. Added. Cloud-init cloud-config. We use this variable to pass the desired Oracle dbname to the deployed PowerVC VM.
4. Added. Local variable used to calculate and hold a timestamp. Our timestamp is formatted to the YYMMDD_hhmmss` format. Note that the content between '{' and '}' is all one line with no line breaks! For information on:
   - local variables: https://www.terraform.io/docs/configuration/locals.html
   - built-in functions (timestamp): https://www.terraform.io/docs/configuration/functions.html

Now switch to the tab **outputs** to modify the file: outputs.tf
The original Template Designer-generated outputs.tf is empty except for a comment block. The following will be added.

File: outputs.tf

```
output "dbserver_ipaddr" {
  value = "${format("%s",join(",",
openstack_compute_instance_v2.dbserver.*.network.0.fixed_ip_v4))}"
}

output "dbserver_vm_name" {
  value = "${format("%s",join(",", openstack_compute_instance_v2.dbserver.*.name))}"
  description = "VM name in PowerVC"
}

output "dbserver_ts" {
  value = "${format("${local.local_dbserver_ts}")}"
}

output "dbname" {
  value = "${var.dbserver_user_data}"
}
```

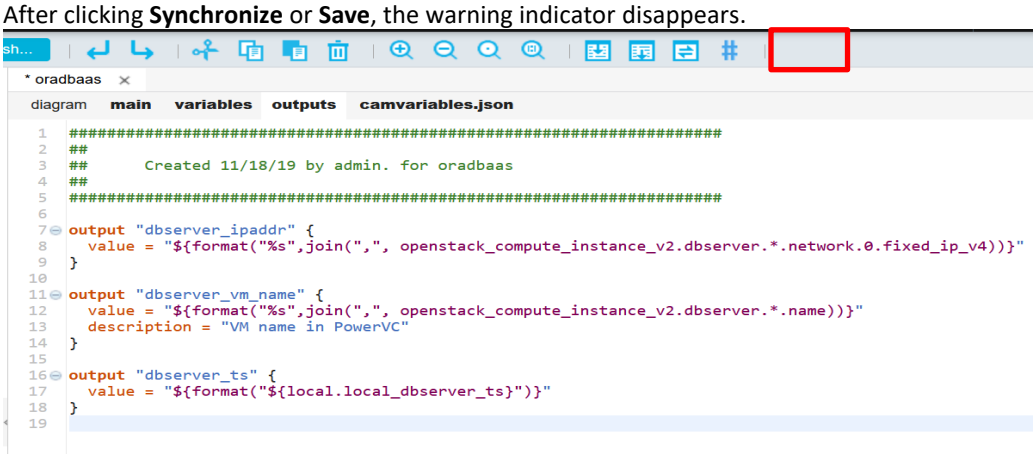
1
2
3
4

Notes:

1. The IP address is assigned by PowerVC and therefore not known until the instance is deployed. This output variable will be populated after the VM is deployed and is used to display the address on the user interface. This output variable is also referenced in the orchestrated service .
2. The VM name is dynamically created at deploy time, including a time stamp, and this variable will store that VM name for display in the user interface.
3. As a stand-alone template, the local timestamp may not be useful. However, we declare it as an output variable in order to have it available as input to the dbclient template deployment. This will be utilized when we create our service which orchestrates the co-deployment of the dbserver and the dbclient.
4. To be able to easily reference the database name in an orchestrated service deployment we make it available as an output variable as well.

Now the only warning indicator left is against camvariables.json. Click the **Synchronize** or **Save** button to synchronize the configuration. Note that the following two screen shots do not show the full content of camvariables.json.

After clicking **Synchronize** or **Save**, the warning indicator disappears.



The file Camvariables.json contains metadata that describe what and how variables appear in the CAM user interface. To suite our preferences, we modify **camvariables.json**.

1. Modify the input groups.

Template Designer defined four input_groups and no output_groups. As we don't have many input or output variables we simplified the configuration to a single input and output group each. Note that below shows only the top code section in the file.

Before

```
{
  "input_datatypes": [ ],
  "input_namespaces": [ ],
  "output_namespace": "",
  "input_groups": [
    {
      "name": "Instances-
openstack_compute_instance_v2.dbserver",
      "label": "Instances -
openstack_compute_instance_v2.dbserver"
    },
    {
      "name": "Other",
      "label": "Other"
    },
    {
      "name": "instances",
      "label": "Instances"
    },
    {
      "name":
"openstack_compute_instance_v2_networks",
      "label":
"openstack_compute_instance_v2 Networks"
    }
  ],
  "output_groups": [ ],
```

After

```
{
  "input_datatypes": [ ],
  "input_namespaces": [ ],
  "output_namespace": "",
  "input_groups": [
    {
      "name": "Input-parameters",
      "label": "Input parameters"
    }
  ],
  "output_groups": [
    {
      "name": "Output-parameters",
      "label": "Output parameters"
    }
  ],
```

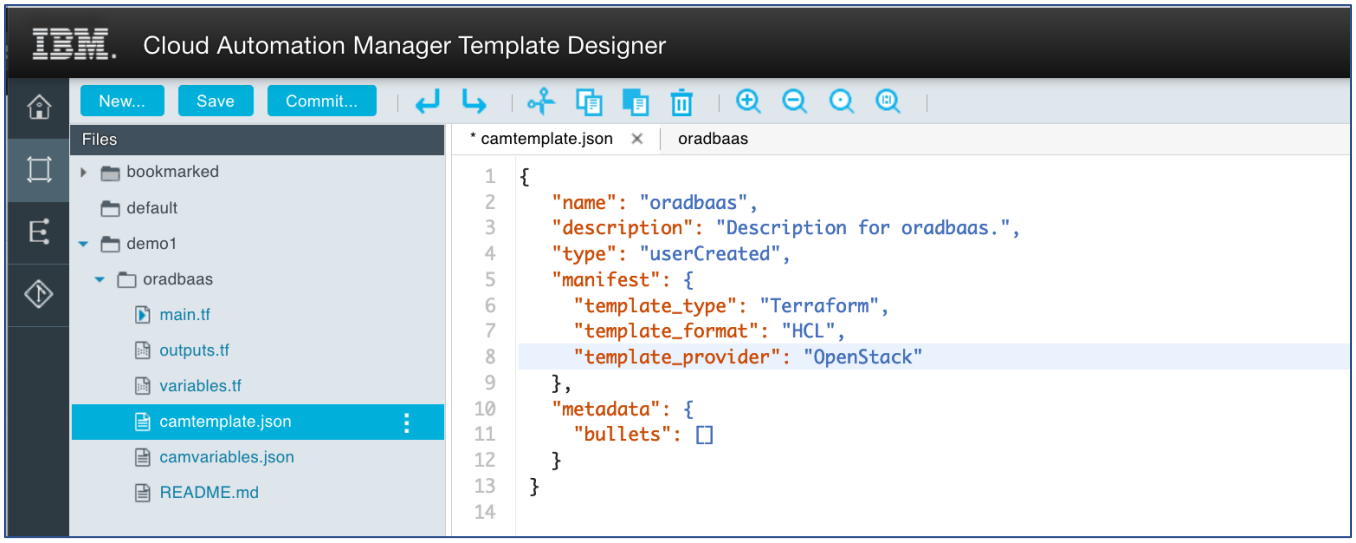
2. Next, make the changes shown below to the following variables in each stanza as highlighted:
   - group_name,
   - label,
   - immutable_after_create, and
   - for **dbserver_ts** block in the **template_output_params** group, set hidden to true.
   - To illustrate default values, I specified RSDtest as default database name in the dbserver_user_data stanza.

When camvariables.json was sync'd earlier, description labels were brought over from those specified in variables.tf. We adjusted them as well to be more descriptive.

```
 "template_input_params": [
    {
      "name": "dbserver_name",
      "label": "db server VM name prefix",
      "description": "db server VM name prefix. Oracle dbname and a timestamp will be
appended in the PowerVC VM name.",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_image_name",
      "label": "OpenStack Image Name",
      "description": "PowerVC image name",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_flavor_name",
      "label": "OpenStack Flavor Name",
      "description": "flavor name, or PowerVC compute template",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_network_id",
      "label": "OpenStack Network ID",
      "description": "Network id in PowerVC",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "dbserver_user_data",
      "label": "Oracle dbname",
      "description": "Oracle DB name",
      "type": "string",
      "default": "RSDtest",
      "validation": "",
      "group_name": "Input-parameters",
```

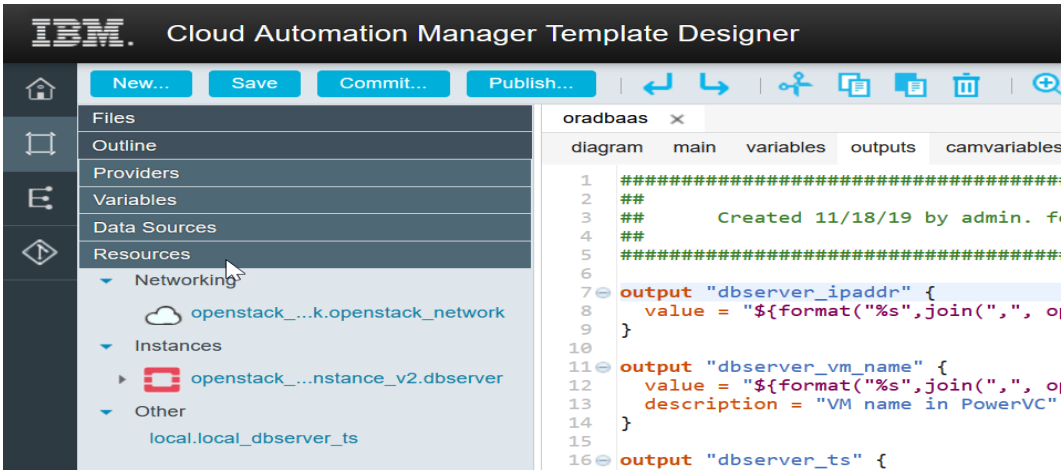
```
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    }
  ],
  "template_output_params": [
    {
      "name": "dbserver_ipaddr",
      "label": "dbserver IP address",
      "description": "dbserver IP address",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string"
    },
    {
      "name": "dbserver_vm_name",
      "label": "dbserver VM name",
      "description": "VM name in PowerVC",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string"
    },
    {
      "name": "dbserver_ts",
      "label": "dbserver ts",
      "description": "internal only - timestamp for output to dbclient",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": true,
      "shortname": "",
      "type": "string"
    }
  ]
```

Next, click camtemplate.json under Files and verify that **template_provider** is set to **"OpenStack"** as shown below.



Click the **Save** button to save your changes, making sure there are no warning or error indicators.

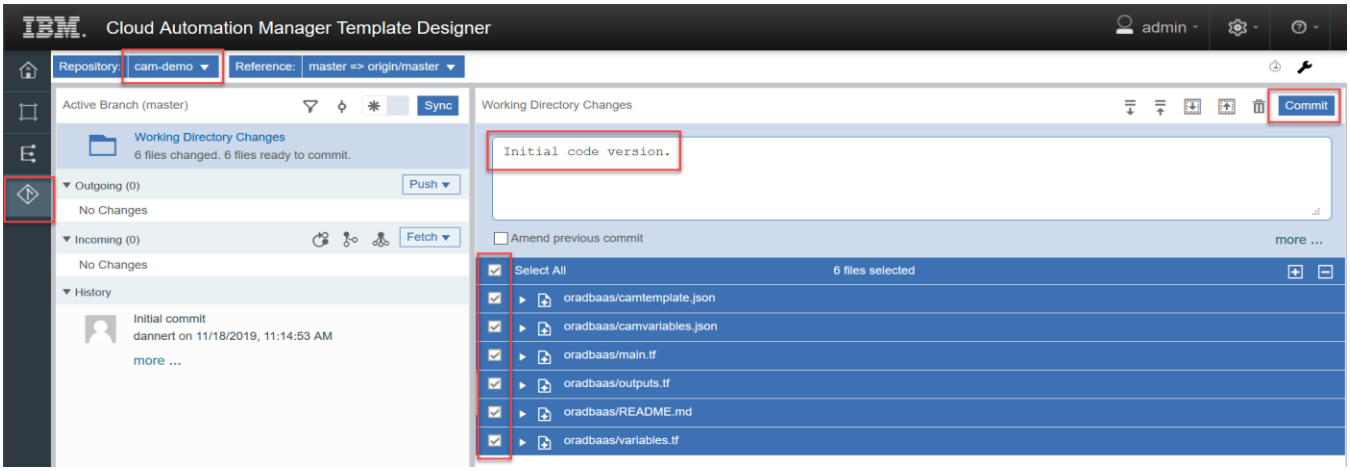
Note that you can drill down into the template details also by selection criteria like *Providers*, *Variables*, …



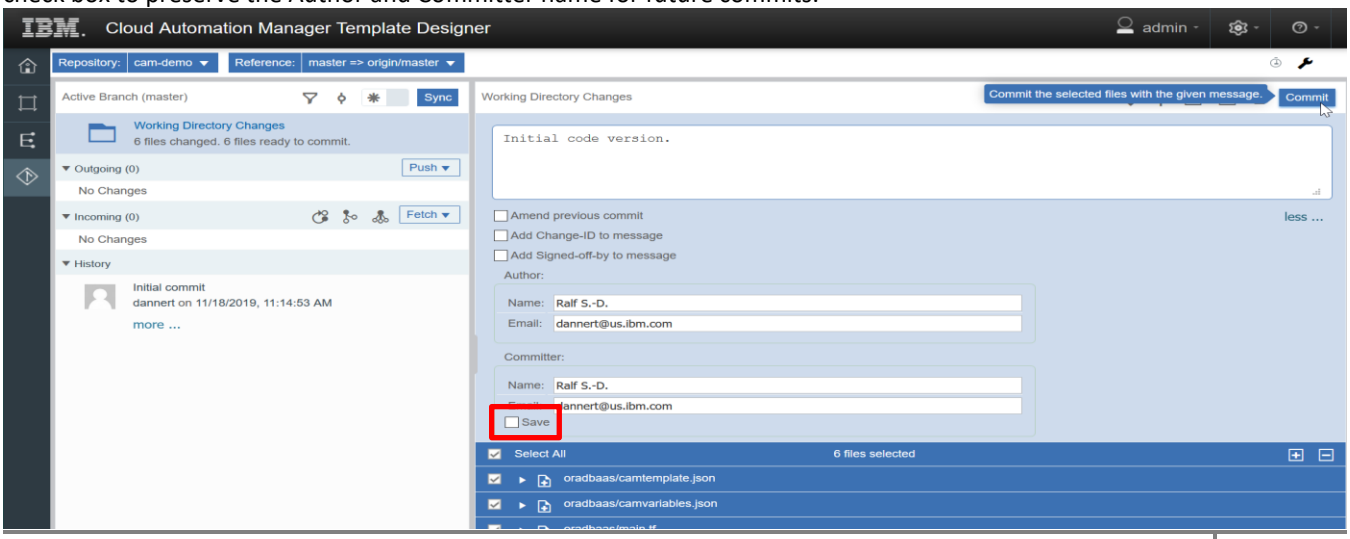
8.1.4 Commit *oradbaas* template to GitHub

After creating and modifying all the files defining the *oradbaas* template we now need to commit all those files back to the GitHub repository.

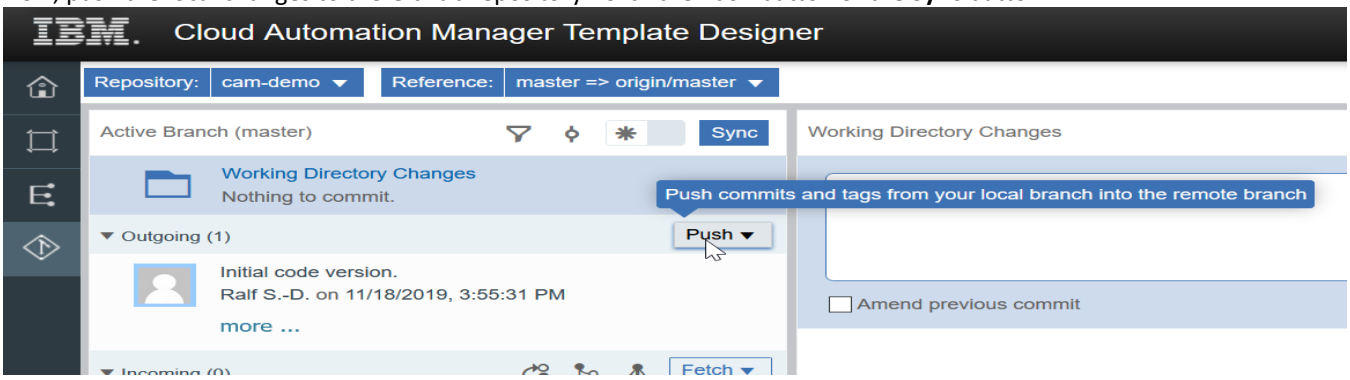
Return to the Repository menu via the lowest icon on the left-hand pane. **Ensure the correct repository is selected!** Enter a commit description, select all the files via Select All and finally click the Commit button.



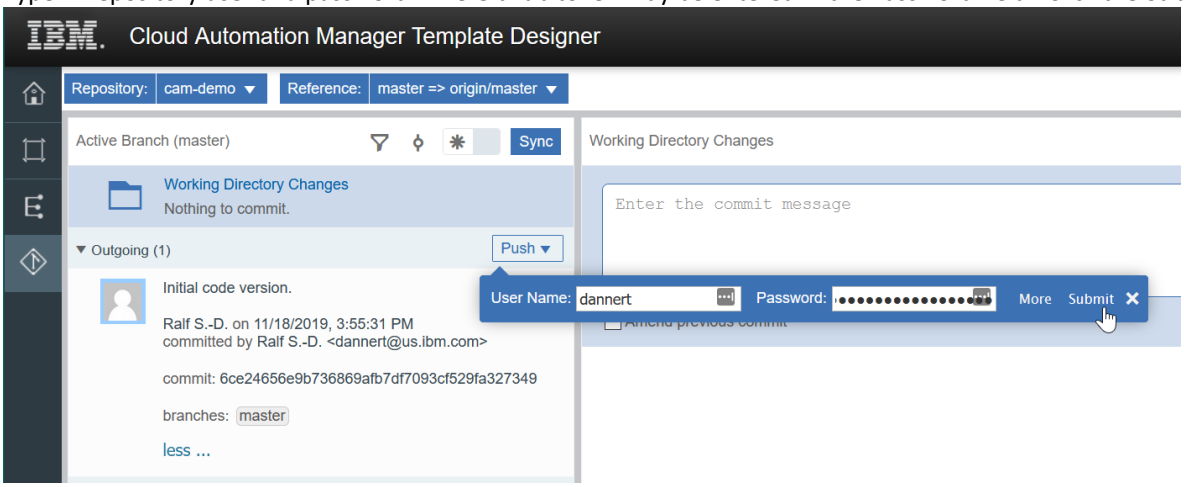
Fill in the name and email address of the author and committer, then click the Commit button. You can check the Save check box to preserve the Author and Committer name for future commits.



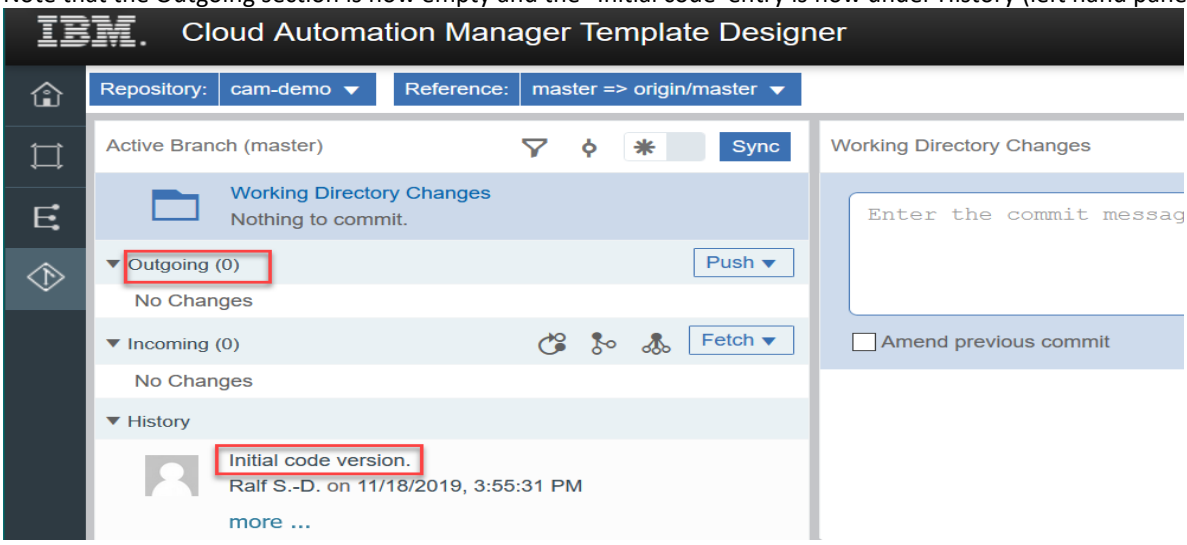
Now, push the local changes to the GitHub repository. Click the **Push** button or the **Sync** button.



Type in repository user and password. The GitHub token may be entered in the Password field. Click the **Submit** button.

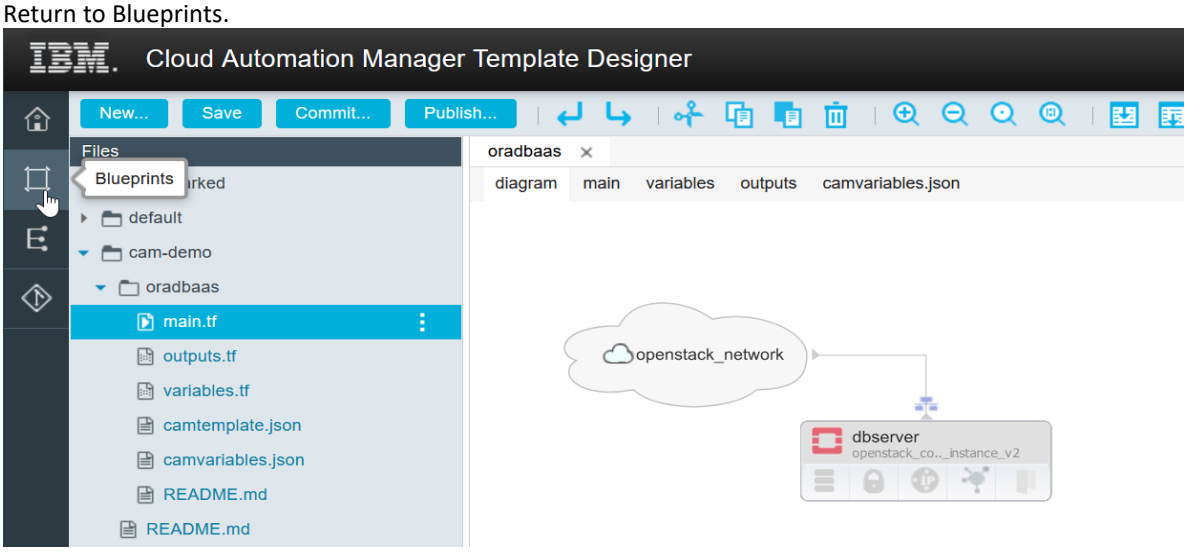


Note that the Outgoing section is now empty and the "Initial code' entry is now under History (left hand panel).

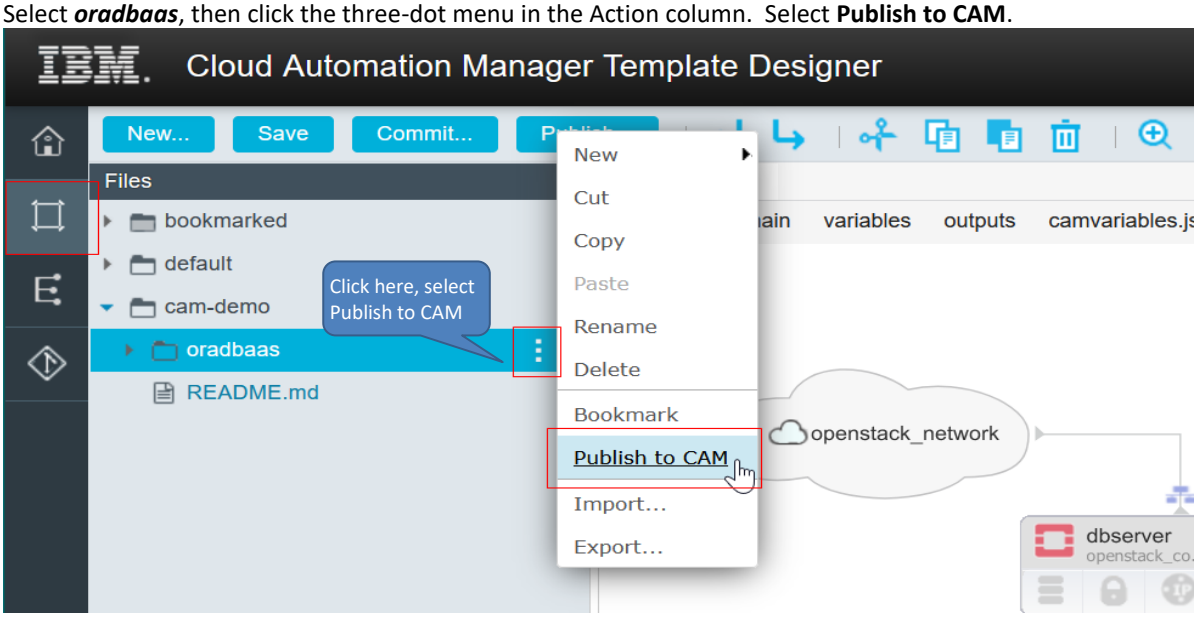


8.1.5 Publish the *oradbaas* template to CAM

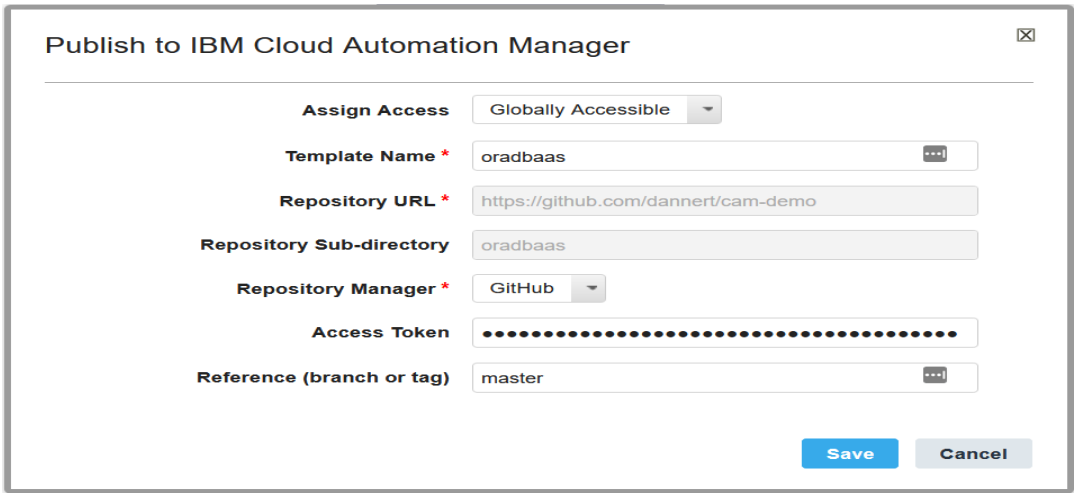
Return to Blueprints.



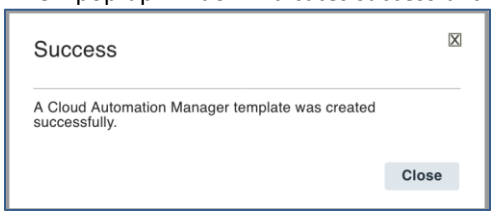
Select ***oradbaas***, then click the three-dot menu in the Action column. Select **Publish to CAM**.



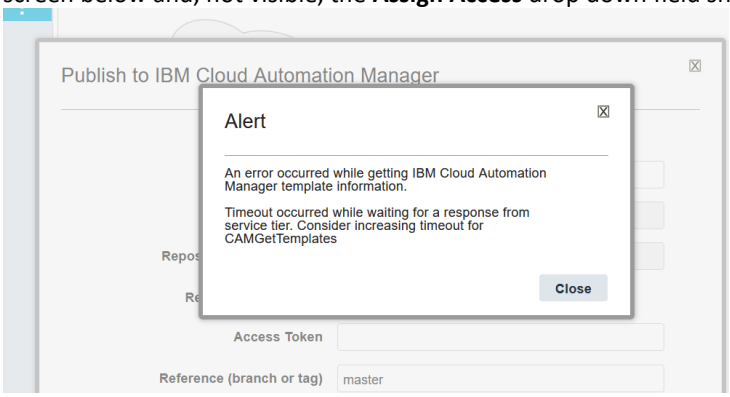
A pop-up window appears. Enter a template name *oradbaas* and the GitHub access token. Click **Save** button.



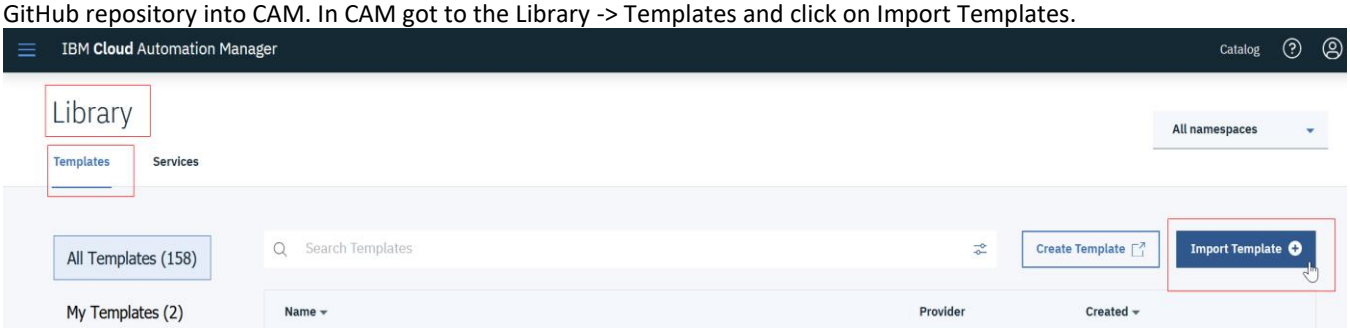
A new pop-up window indicates successful creation of the CAM template.



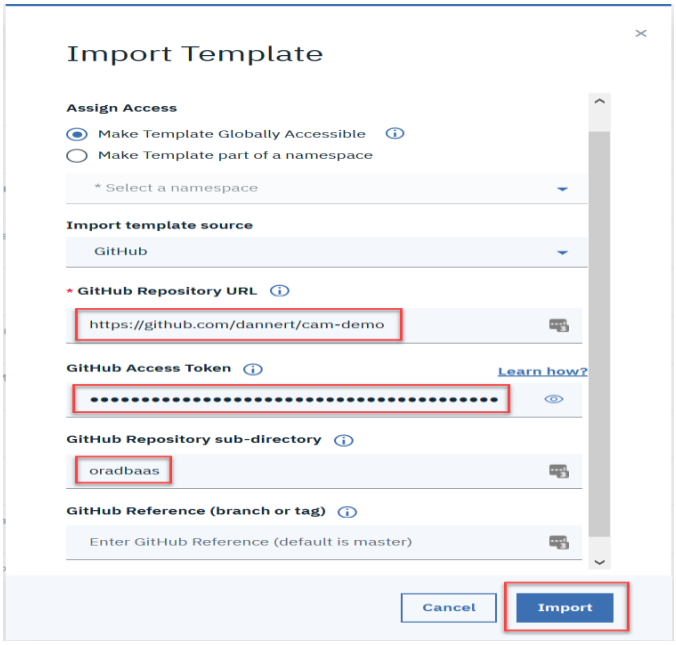
Note: In our testing we ran sporadically into challenges with the "Publish to CAM" functionality and it failed with the error screen below and, not visible, the **Assign Access** drop down field showed ERROR text.



As workaround we utilized instead the **Import Template** functionality in CAM to import the template definition from the GitHub repository into CAM. In CAM got to the Library -> Templates and click on Import Templates.

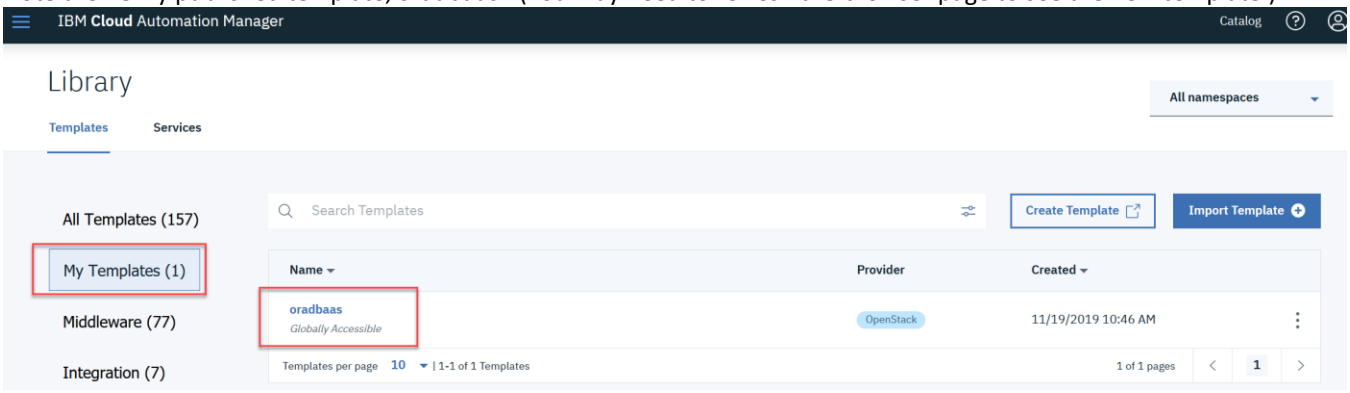


In the new pop-up window provide the relevant information to access GitHub and select the correct template, *oradbaas*, as indicated by the sub-directory we choose.



After the Publish to CAM or Import Template in CAM go the Template Library.

Note the newly published template, *oradbaas*. (You may need to refresh the browser page to see the new template.)

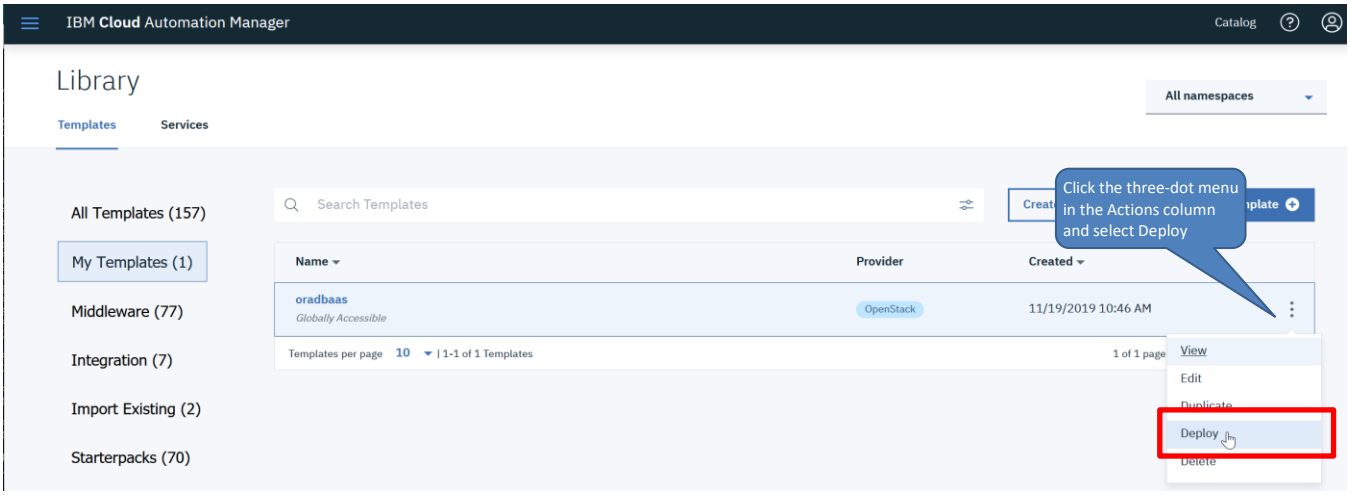


The new template can now be deployed in CAM or be used in a service which can also be published to ICP.

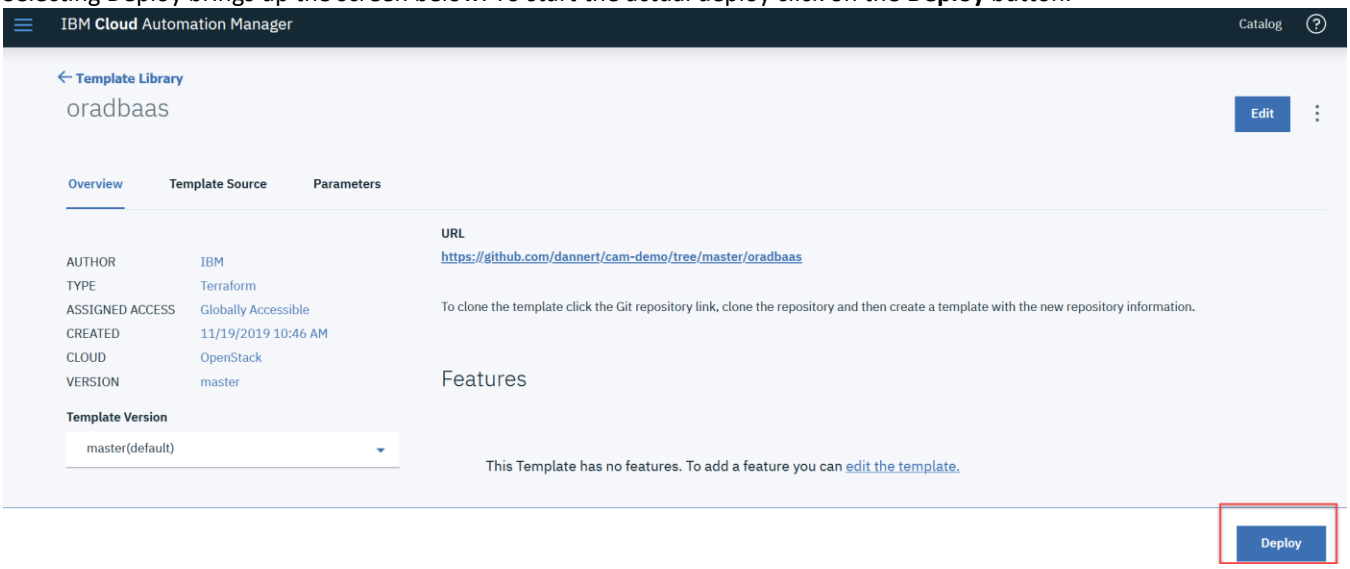
Note: The template Provider shown in the CAM screen above shows **OpenStack**. This value is taken from **template_provider** in **camtemplate.json**. If this does not show Openstack then this can be corrected in CAM by clicking the three-button menu on the right, selecting Edit and changing **Cloud Provider** to OpenStack.

8.2 Deploying the *oradbaas* template

Select the *oradbaas* template, click the menu and select **Deploy** to test the template.



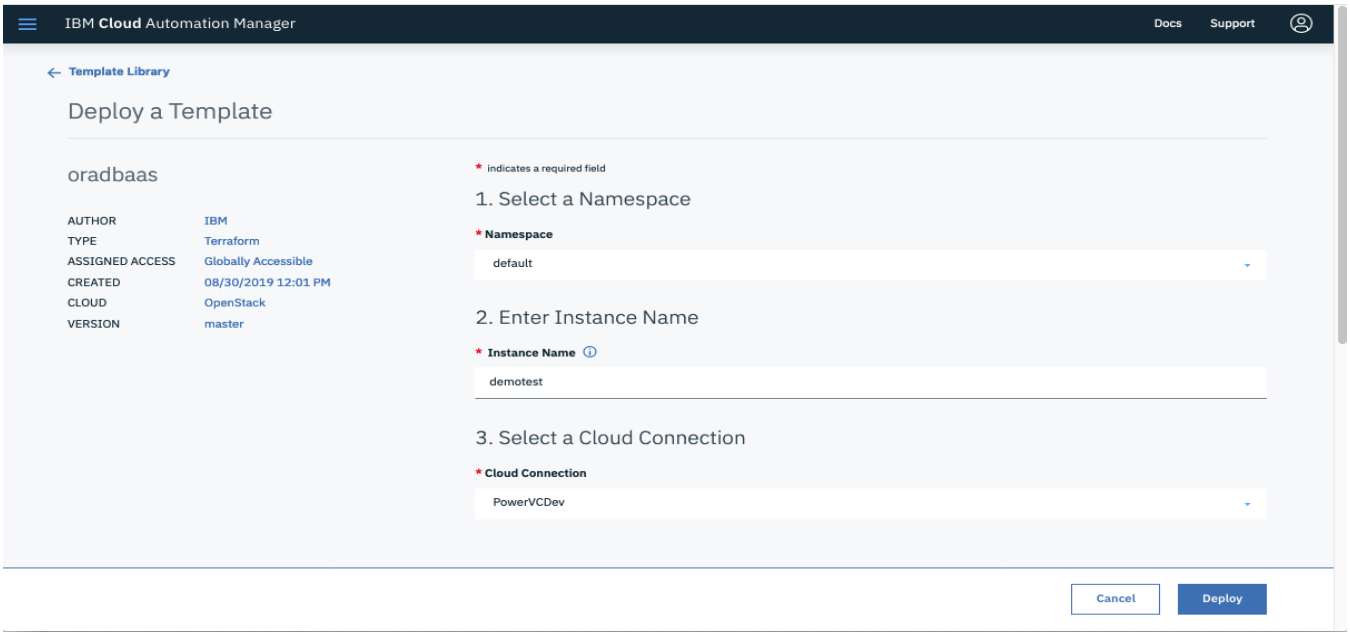
Selecting Deploy brings up the screen below. To start the actual deploy click on the **Deploy** button.



On the Deploy a Template screen, select the Namespace, e.g. default.

Once Namespace is selected, other input parameters appear as shown on the screen below. Type in an **Instance Name**, e.g. demotest. This name is what will be listed under Deployed Instances in CAM.

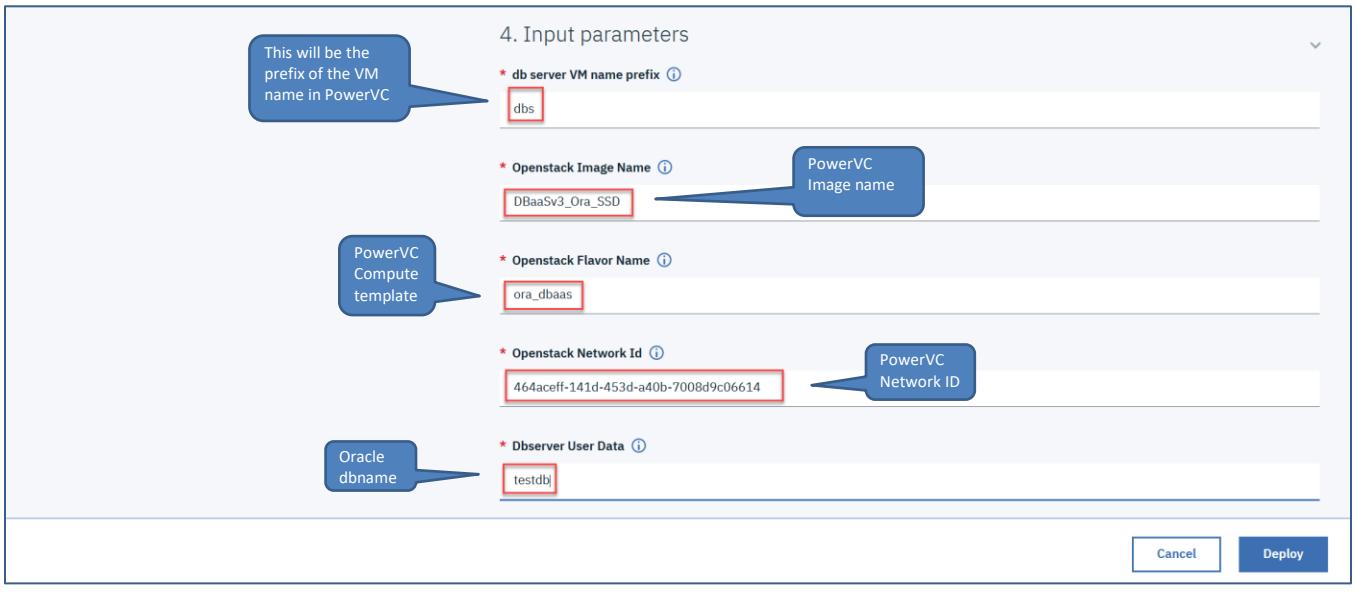
Note that our Cloud Connection, PowerVCDev has been filled in already (it's the only OpenStack connection we've defined).



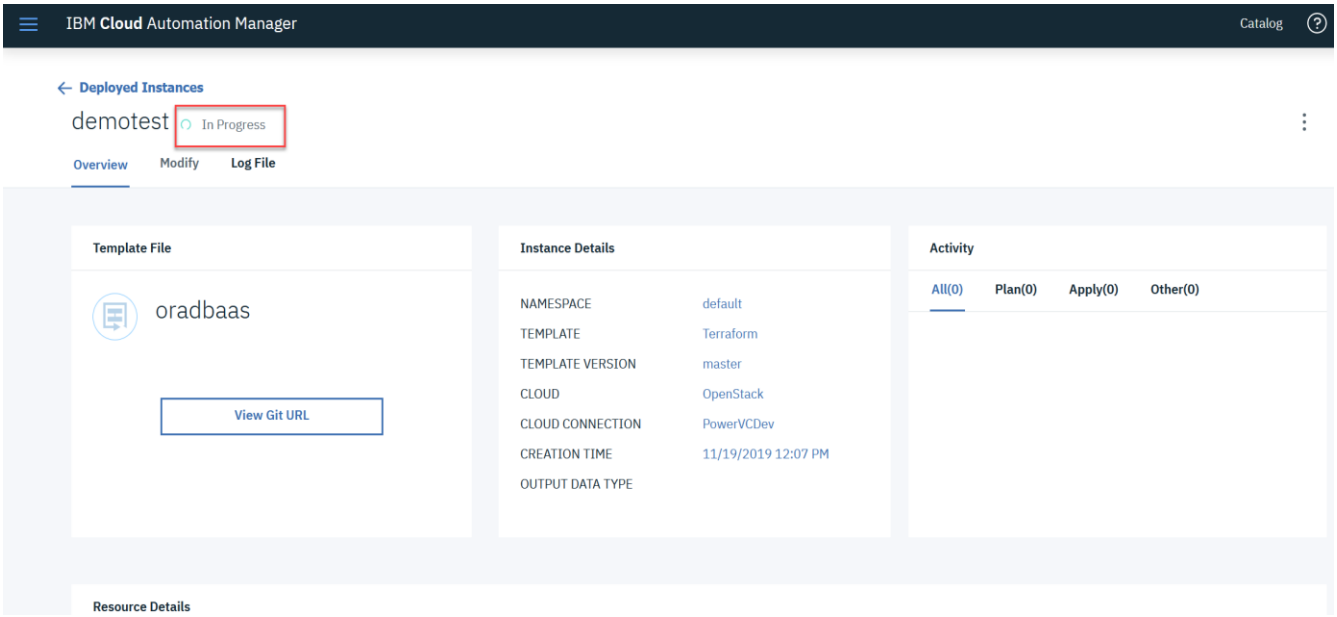
Type in the Input Parameters as shown below.

Note:

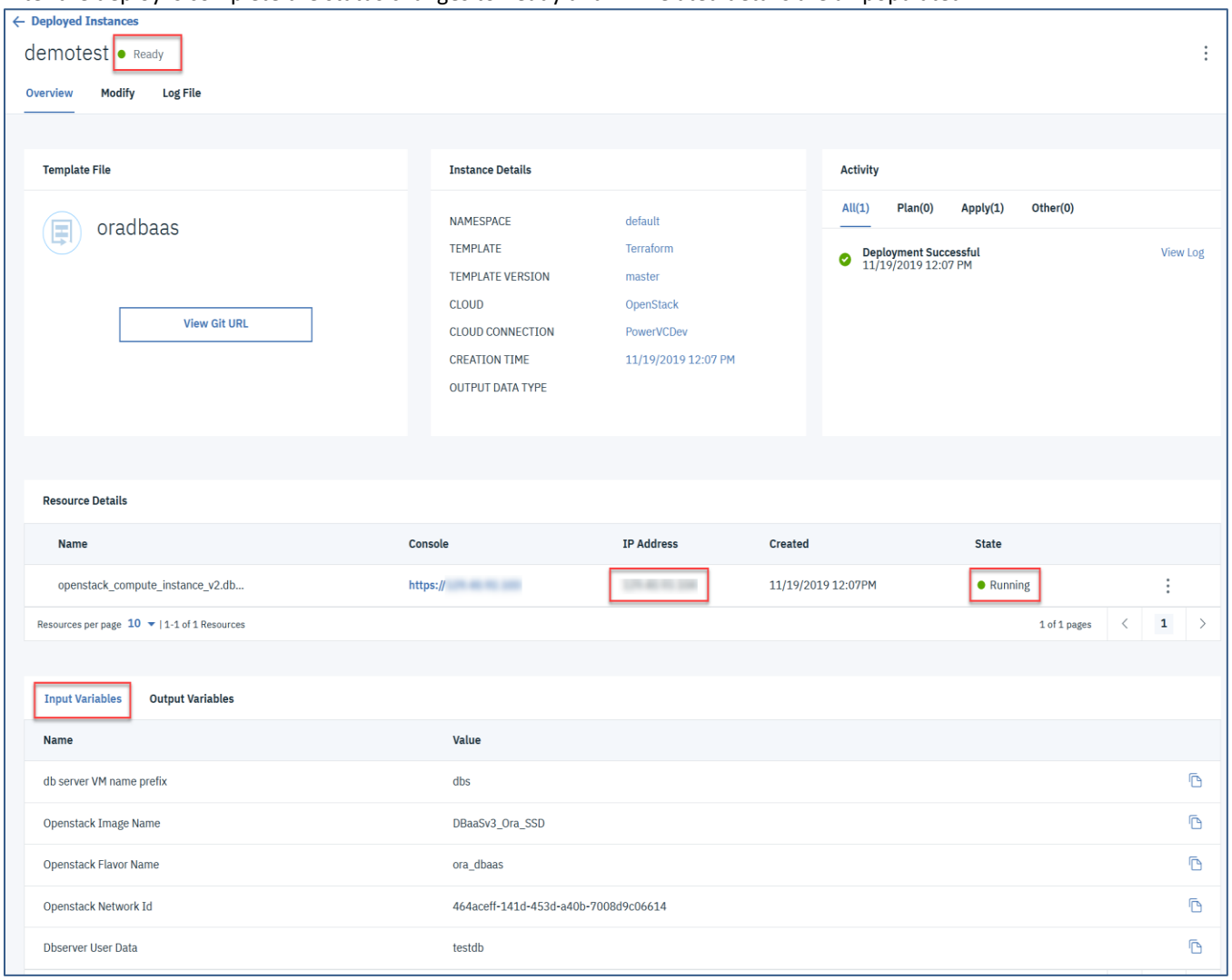
- All our template inputs are grouped under **Input Parameters**, which is our one input group defined in camvariables.json.
- In this example, only the Oracle dbname field had a default value set – RSDtest – and therefore all other input parameters show as **“”**. Note that you can overwrite the specified default value as you can see below. For a real production environment, it probably would make sense to set default values for the other input fields as well. If desired, this can be done in the camvariables.json file.
- Image name, flavor name and network ID can be determined via PowerVC gui or CLI command - openstack {image|flavor|network} list.



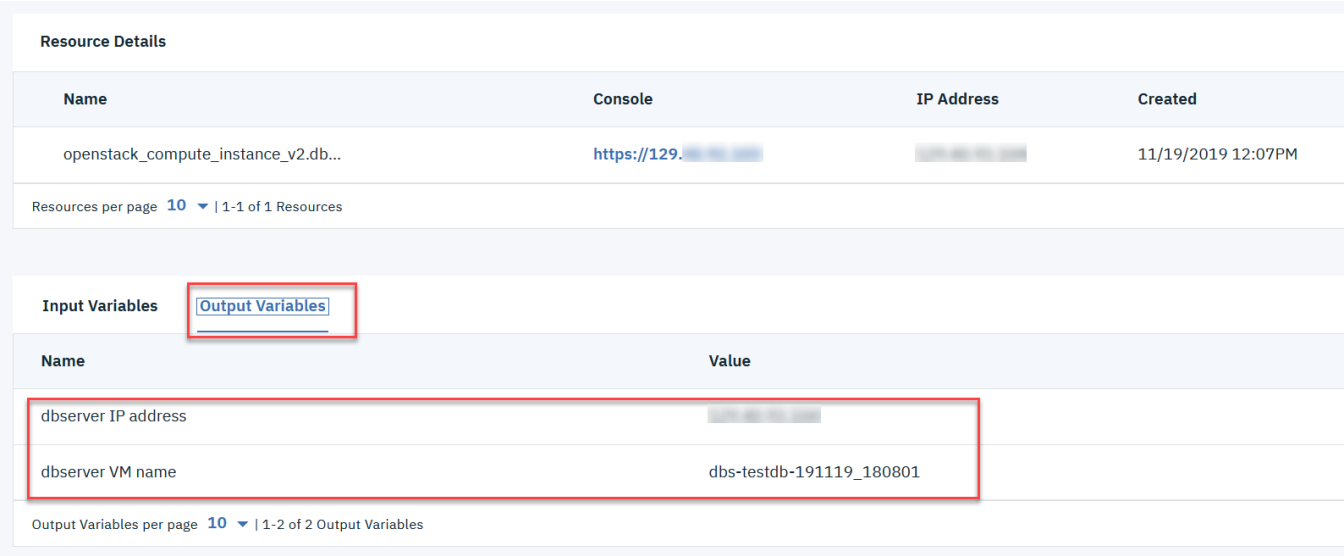
Click the **Deploy** button to deploy the template.
After clicking Deploy the following screen is displayed where you can observe the status of the deploy. Note the “In Progress” in the top left.



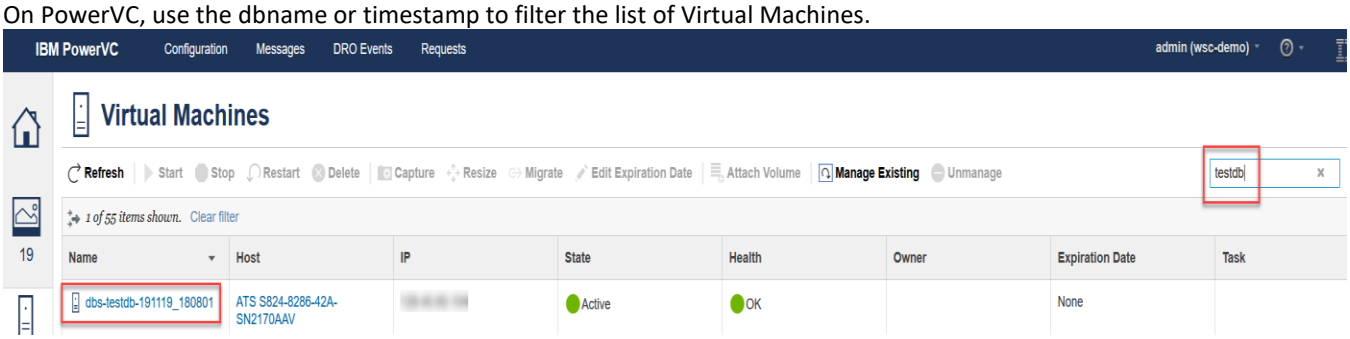
After the deploy is complete the status changes to Ready and VM related details are all populated.



To see the populated output variables, we specified in the “output parameters” section of the camvariables.json file of the *oradbaas* template, click **Output Variables**.
Note that the “hidden” output parameter, as expected, is not displayed. The VM image name was correctly constructed as specified.



On PowerVC, use the dbname or timestamp to filter the list of Virtual Machines.



8.3 *oraclient* Template Design

The *oraclient* template deploys a VM running Linux on Power. The PowerVC deployment image contains Red Hat Linux, configured with graphical user interface, VNC client and NFS client software, but there are no post-deployment scripts, or oracle / swingbench related software included in that image.
Important: This template is written to be co-deployed with the *oradbaas* template as it expects values like DB server IP and database name as parameter which are the result of a successful deploy of the *oradbaas* template.

This template illustrates how you can inject scripts, via cloud-init cloud-config, into a VM at deploy time and the use of a known NFS export to access required additional software to be installed via the injected post-processing scripts. This approach is very flexible as it does not need the creation of a new image in PowerVC every time a post-deployment script needs to be changed. The availability of the NFS server to stage the software is, of course, critical at deploy time.

In contrast, the PowerVC image used in the *oradbaas* template deployment included the oracle database software and several post-processing scripts to configure, create and start the oracle database at VM deployment. Cloud-init user data was only used to provide simple parameters like the Oracle database name to the post-deployment scripts.

Note: All template files for the *oraclient* template are available in the appendix of this document.

8.3.1 *oraclient* template

As mentioned above, the PowerVC deployment image does not include any client-specific post-deployment customization processing. Post-deployment performed by *oraclient* is done using cloud-init cloud-config which injects the following files into the deployed VM and executes several additional shell commands at deployment.

- Inject four script files using the write_files configuration statement. These four files are:
  - dbserver.cfg - contains export statements to set environment variables
  - sbsetup.sh - shell script to check availability of the database, then load the Swingbench Order Entry schema using Swingbench's oewizard in silent mode (non-GUI).
  - dropsoe.sh - shell script to drop the Swingbench Order Entry. Invokes the oewizard Java program in silent mode.
  - startsb.sh - shell script to start the Swingbench GUI load generator. This is intended to be invoked in the VNC viewer.
- run shell commands to do the following:
  - restart (reset) vncserver
  - mount an NFS drive and copy Oracle Instant Client and Swingbench software to local filesystem
  - add Oracle Instant Client binary path to root user's PATH in .bash_profile
  - wait for *oradbaas* deploy to complete post-deployment and load the Swingbench Order Entry schema by running sbsetup.sh

8.3.2 Input and Output parameters

oraclient template input parameters:

Input Parameter	Description
Namespace	This is the ICP namespace or Red Hat OpenShift project name in which the instance will be deployed.
Instance name	This is the name of the instance being deployed; displayed on the CAM deployed instance screen. Choose any appropriate name.
dbclient_name	This is used to name the PowerVC VM. In our template, we concatenate this parameter to the dbname (dbserver_user_data), and a timestamp in order to have a unique name that can be matched up to the dbclient VM. For example, **dbc**-testdb-<YYMM_hhmmss>.
openstack_image_name	This is the PowerVC Compute template which specifies the compute characteristics of the VM to be deployed. The flavor (OpenStack) or compute template (PowerVC) specifies VM specifications, including number of CPUs, RAM, etc. Alternative option would be to use the flavor id instead.
openstack_flavor_name	This is the name of the deployable image in PowerVC. Alternative option would be to use image id.
openstack_network_name	This is the network name in OpenStack. Note that we'd typically use the network ID but also wanted to show the use of network name in this example.
dbserver_user_data	This parameter will be linked to oradbaas' input parameter of the same name, which is used for the Oracle dbname. The dbname is needed for the connect string to the Oracle database, and it is also needed to build the VM name. See dbclient_name above.
dbserver_ipaddr	dbserver_ipaddr is assigned by PowerVC, therefore is not known at deployment request time. dbserver_ipaddr is obtained as input from the *oradbaa*s output variable.
dbserver_ts	This is used to accept *oradbaas*' output parameter of the same name. See *oradbaas* output parameter above.

oraclient template output parameters are for display on the CAM deployed instance screen:

Output Parameter	Description
dbclient_ipaddr	IP address of the dbclient VM, needed for user/requester access.
dbclient_vm_name	dbclient VM name in PowerVC, useful for verification and troubleshooting.
dbserver_connect	Oracle jdbc connect string. Used by the Swingbench load generator to connect to the database.
dbclient_vnc	Host and display information for access using VNC viewer.

8.3.3 Create *oraclient* template

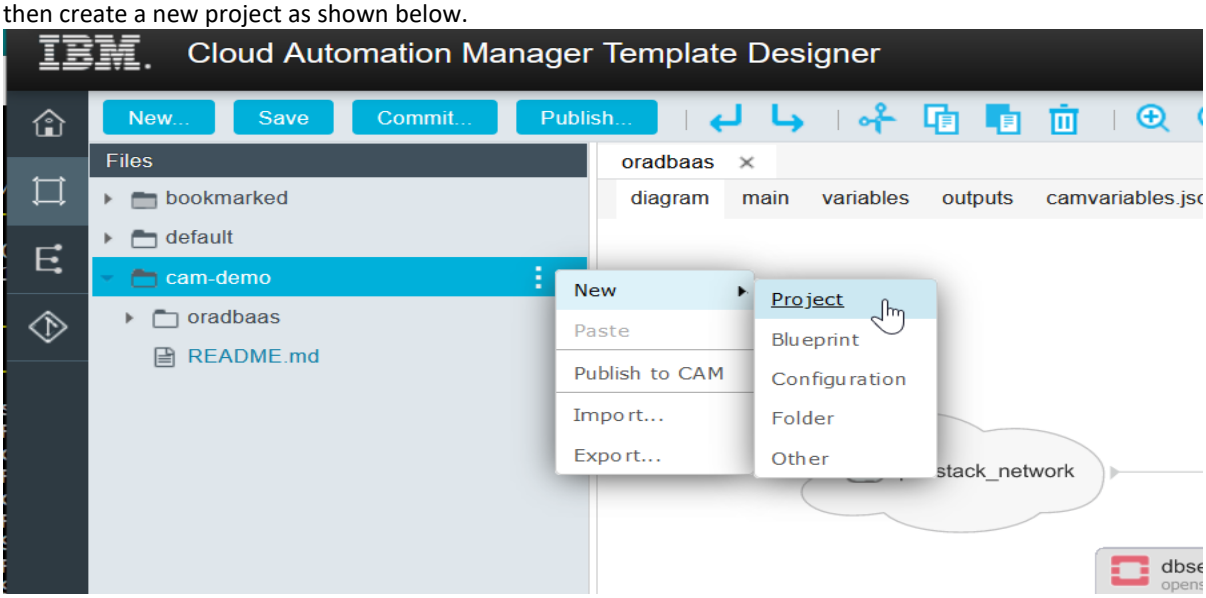
In this section we provide only the critical steps to create the oraclient template. The contents of the template files can be found in section 10.3, *oraclient* Template files, and can be copied / pasted into the correct locations in Template Designer.

It is important to closely look at the main.tf file and understand how the file injection is specified in the openstack compute instance resource definition.
Here a sample snippet which, at deploy time, creates the file “dbserver.cfg” in the directory /root within the VM.

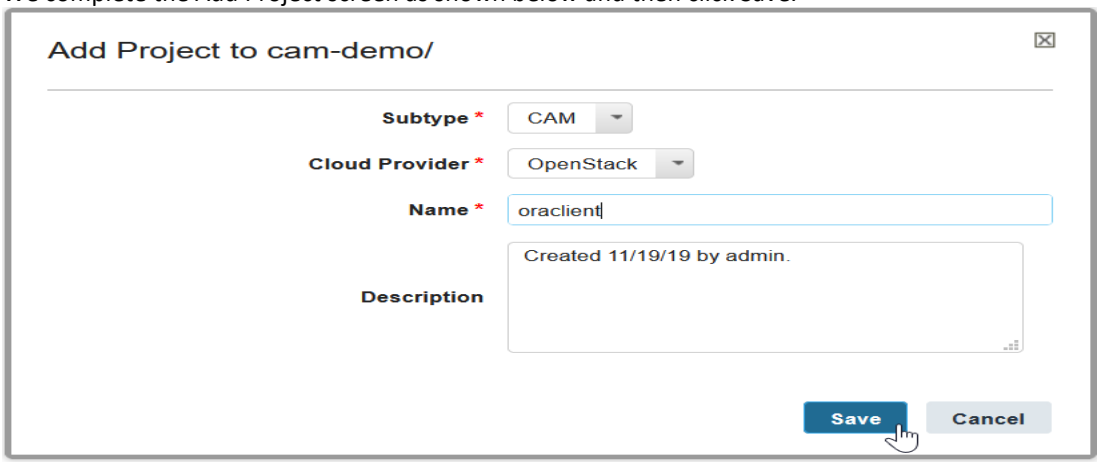
```
#cloud-config
write_files:
  - content: |
      # dbserver parameters
      ${format("export DBNAME=%s",var.dbserver_user_data)}
      ${format("export DBSERVERIPADDR=%s",var.dbserver_ipaddr)}
      export ORASYSPW=oracle
      export SWINGBENCHBINDIR=/opt/swingbench/bin
    path: /root/dbserver.cfg
```

Note how the content of the file is dynamically generated at deploy time based on the values of input parameters derived from the output of the *oradbaas* template deploy step. The parameter transfer between the *oradbaas* and *oraclient* templates is discussed in more detail in section 9, Developing our Service.

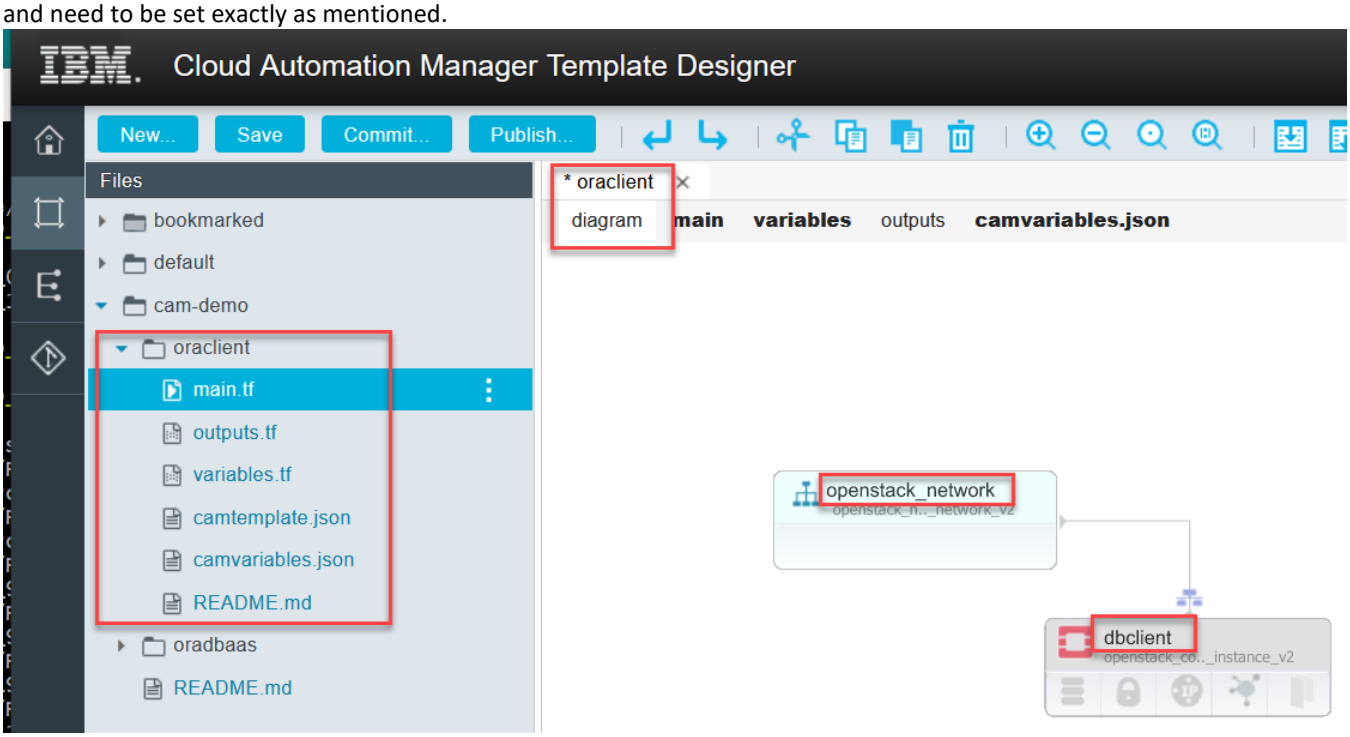
To create the *oraclient* template, we go back to the Template Designer, select Blueprints, our cam-demo repository and then create a new project as shown below.



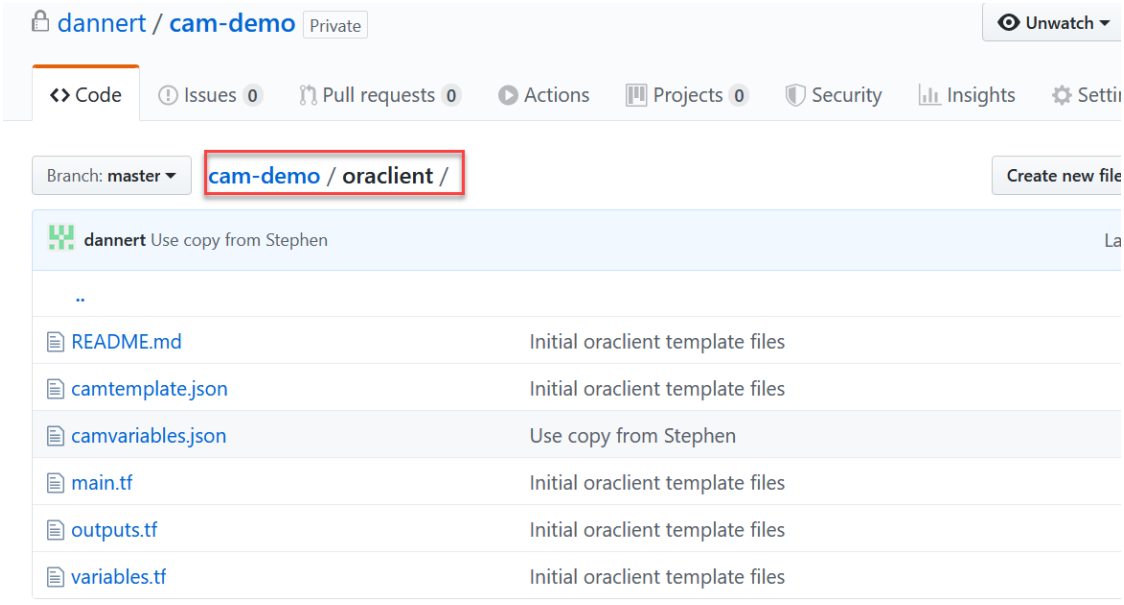
We complete the Add Project screen as shown below and then click Save.



Similar to the steps performed when creating the *oradbaas template*, we then switch to the diagram view and create an openstack instance, openstack network and connect the two. Note the names we used for the openstack_network "**network**" and the openstack_instance **"dbclient**". Those names are referenced in the variable names in the template files and need to be set exactly as mentioned.



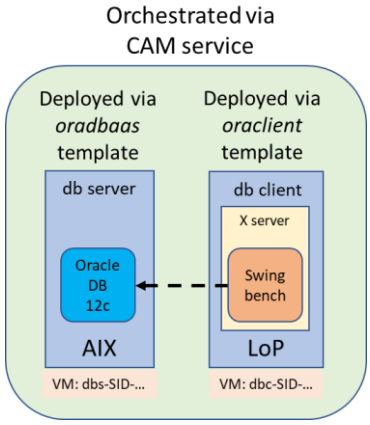
We then copied the respective code from section 10.3 in the appendix into the respective files and finally committed the new project to the GitHub repository. GitHub now shows a new sub-directory for *oraclient* with the respective template files:



As the last step we published the *oraclient* template to CAM. For the required steps refer to the section *oradbaas* template.

9 Developing our Service

In this section we describe how to create a service to deploy the *oradbaas* and *oraclient* templates in an orchestrated fashion. In the end state, after the service has been deployed by the end user, the requester will be able to access the dbclient VM using VNC viewer and run Swingbench's Order Entry load generator against the database server.



9.1 Service Design

Use CAM Service Composer to create the service to deploy the *oradbaas* and *oraclient* templates together. To minimize the number of parameters an end-user must fill in, we specify most of the input parameters required for the *oradbaas* and *oraclient* templates in the service definition. For example, this includes image and flavor names as well as network id and leaves only the Oracle database name as input parameter for the deployer of the service.

Input Parameter	Description
dbserver_user_data	Oracle dbname. This is linked to the oradbaas input parameter.

Create service output parameters for display on the deployed service screen.

Output Parameter	Description
dbserver_ipaddr	dbserver IP address
dbserver_vm_name	dbserver VM (in PowerVC) name. Expected value <prefix>-<dbname>-<timestamp>
dbclient_ipaddr	dbclient IP address
dbclient_vm_name	dbclient VM (in PowerVC) name. See dbserver_vm_name.
ora_dbname	Oracle database dbname.
oradb_connect	Oracle database connect string (jdbc). Expected value - //<ipaddress>/<dbname>
dbclient_vnc_addr	VNC host and display information. Expected value - <ipaddress>:<display#>.

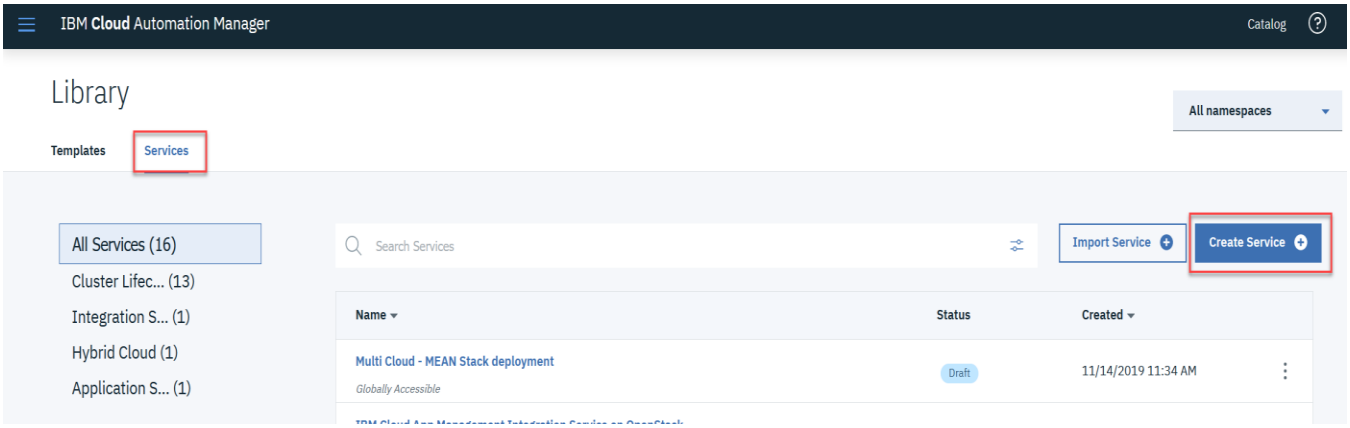
9.2 Creating the Service

The Service Composer, part of the CAM Service library, enables the authoring of Cloud Automation Manager services. After Cloud Automation Manager templates are created, administrators can use the Service Composer to create a service, consisting of CAM templates or Kubernetes Helm charts from IBM Cloud Private, based on a designed sequence or decision.

9.2.1 Create Service

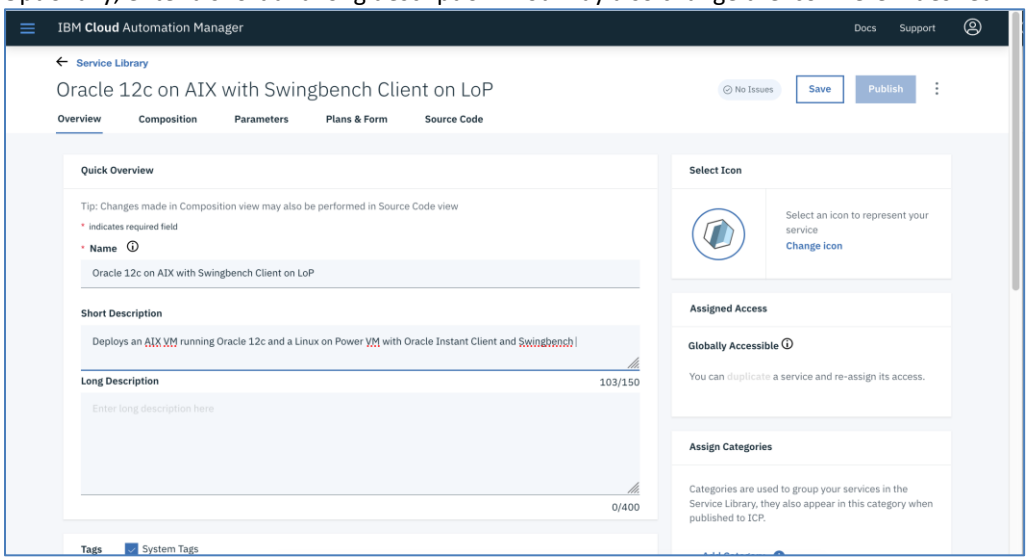
Invoke the Service Composer from the Service Library. Access the Service Library by clicking the hamburger menu and select Library->Services.

On the Service Library screen, click the **Create Service** button.



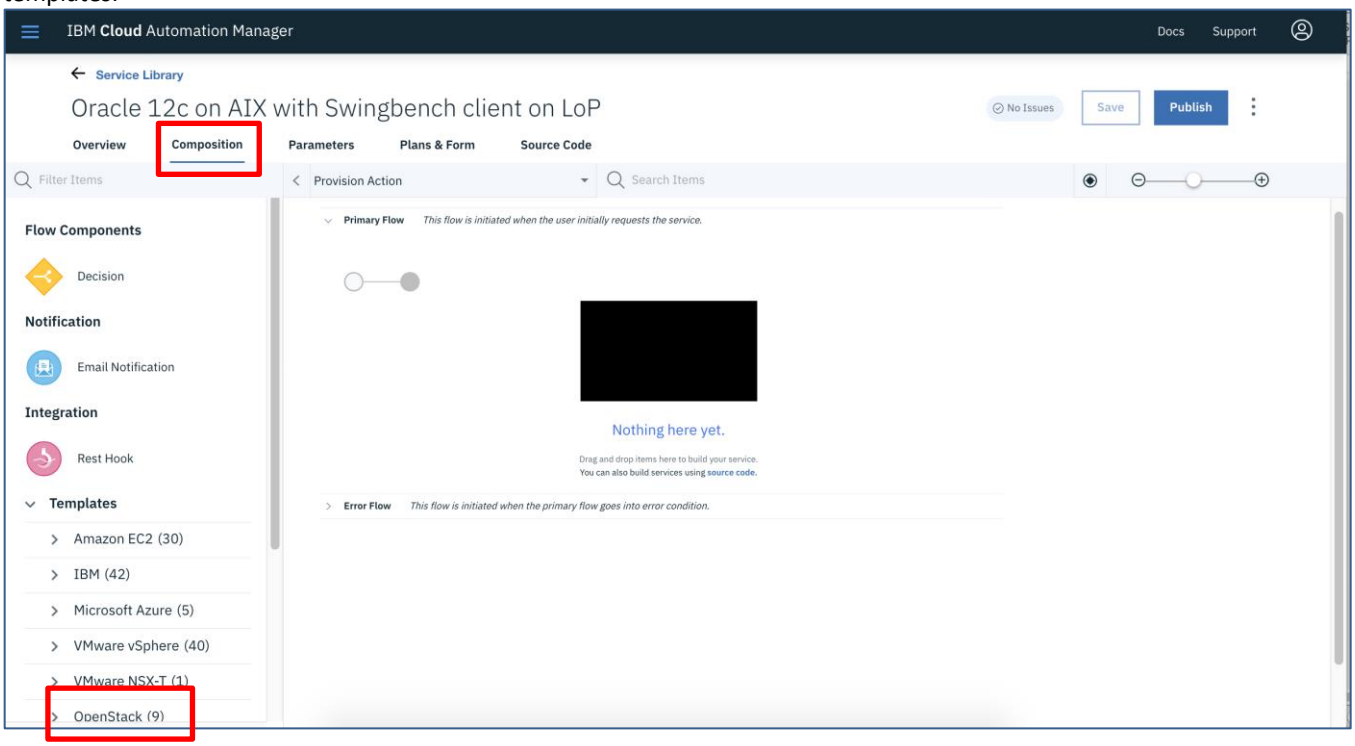
On the Create New Service window, enter the Service name and add a category or pick one from the drop-down list. We created a new category in our example.

Optionally, enter a short and long description. You may also change the icon here if desired.

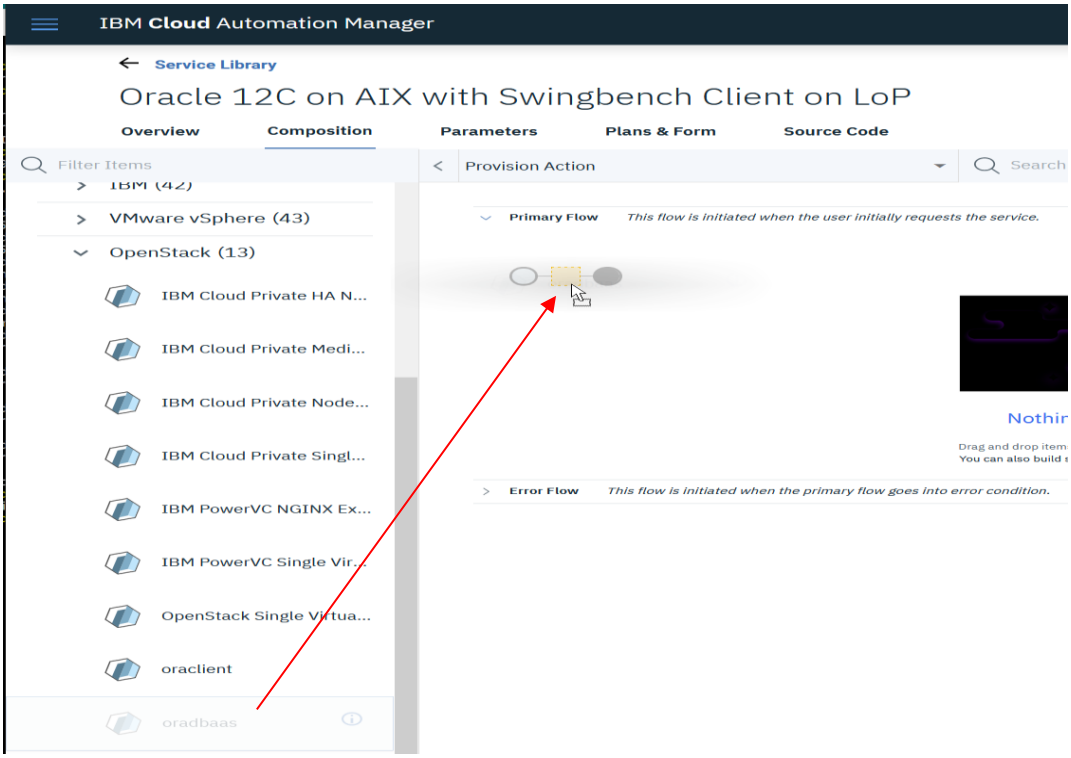


Click **Composition**.

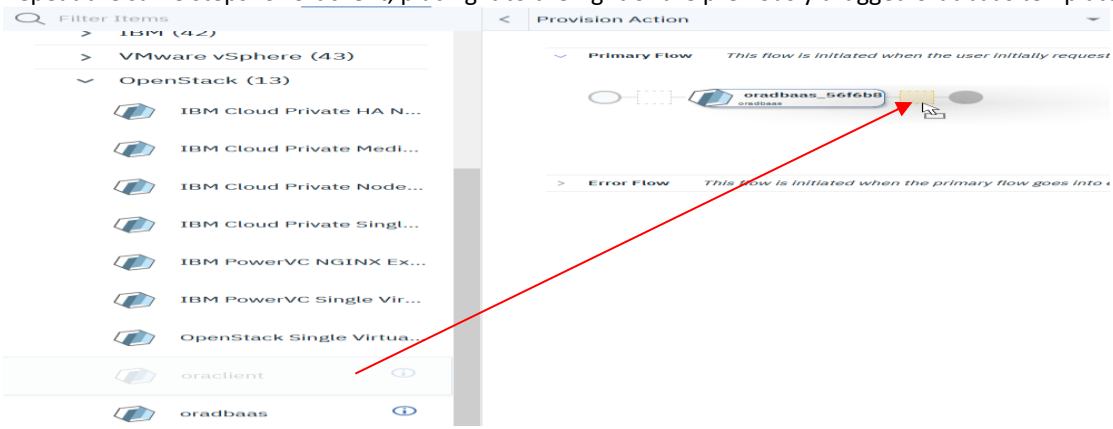
On the left-hand panel, expand the **OpenStack** group under Templates and scroll down to find the *oradbaas* and *oraclient* templates.



Point to the *oradbaas* template, left click and hold on the *oradbaas* template, and drag it to the space between the two dots, shown by the arrow above. A rectangular box will appear between the two dots when the dragged object approaches the line. Release the mouse button to drop the template into this rectangular box.

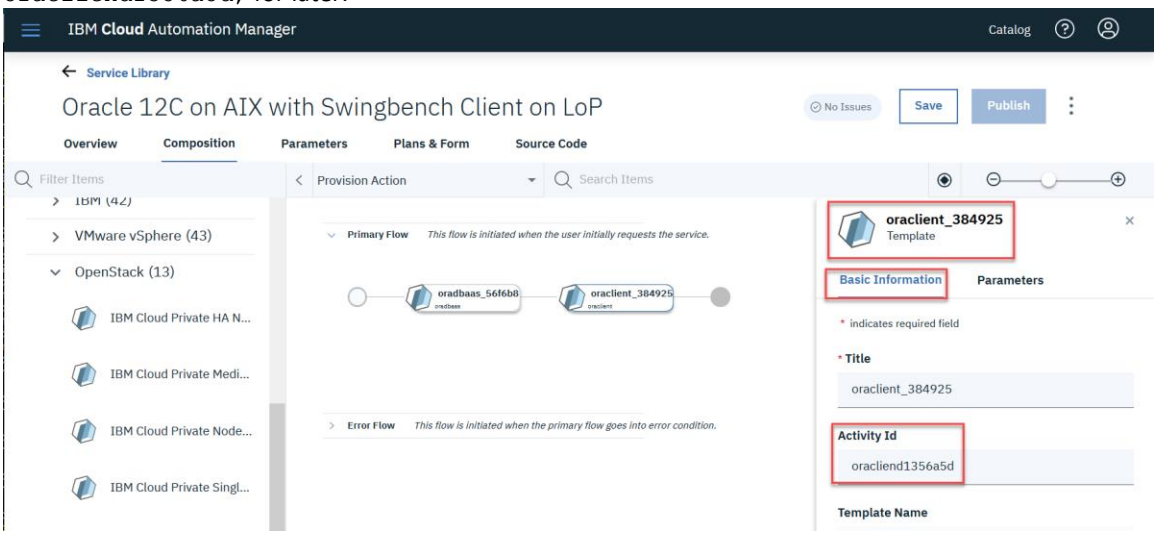


Repeat the same steps for oraclient, placing it to the right of the previously dragged oradbaas template.

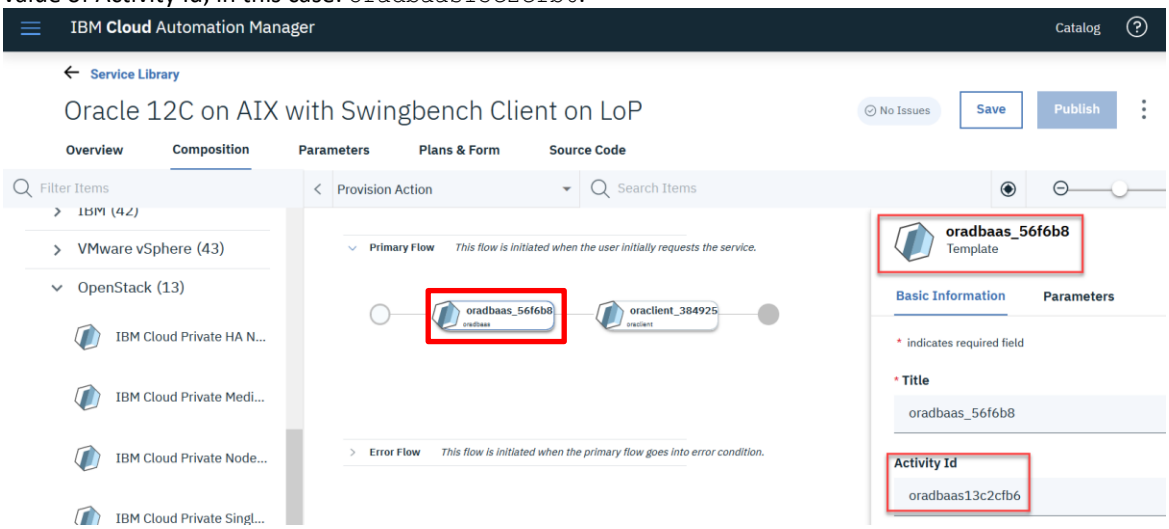


When done, the two templates will be lined up between the two dots as shown below.

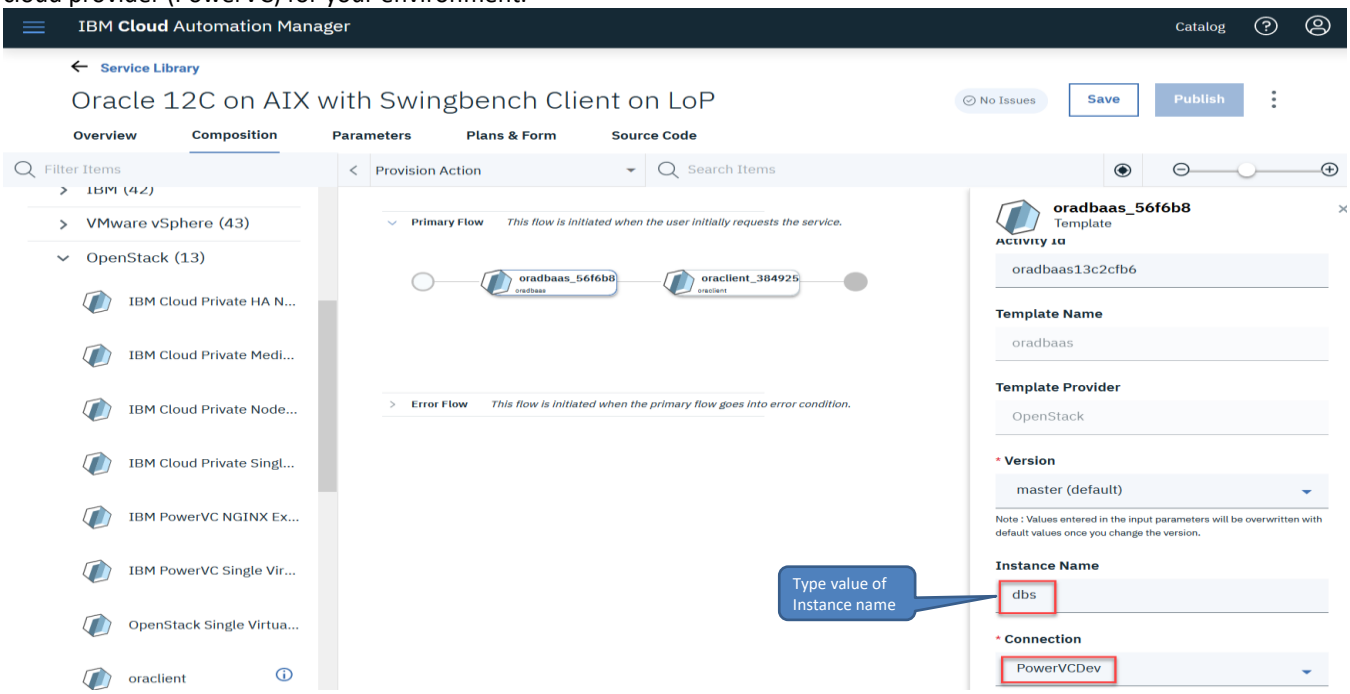
Since it's the most recently dragged/placed object, *oraclient* template's **Basic Information** tab will be shown on the right-hand panel. (If not, select Basic Information.) Save the value for **Activity Id,** which in this example, is `oracliend1356a5d`, for later.



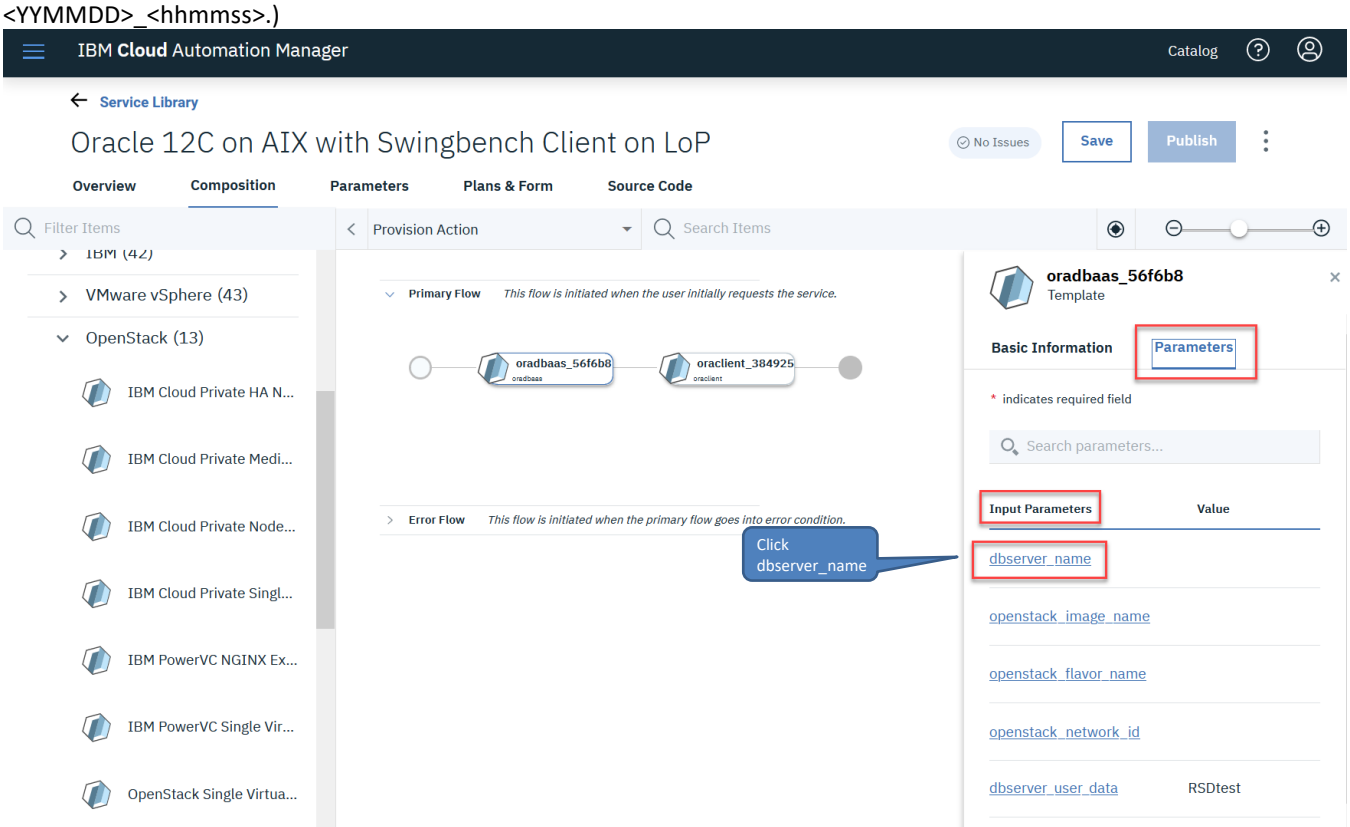
Click the *oradbaas* template icon to show *oradbaas*' **Basic Information** tab on the right-hand panel. As before, save the value of Activity Id, in this case: `oradbaas13c2cfb6`.



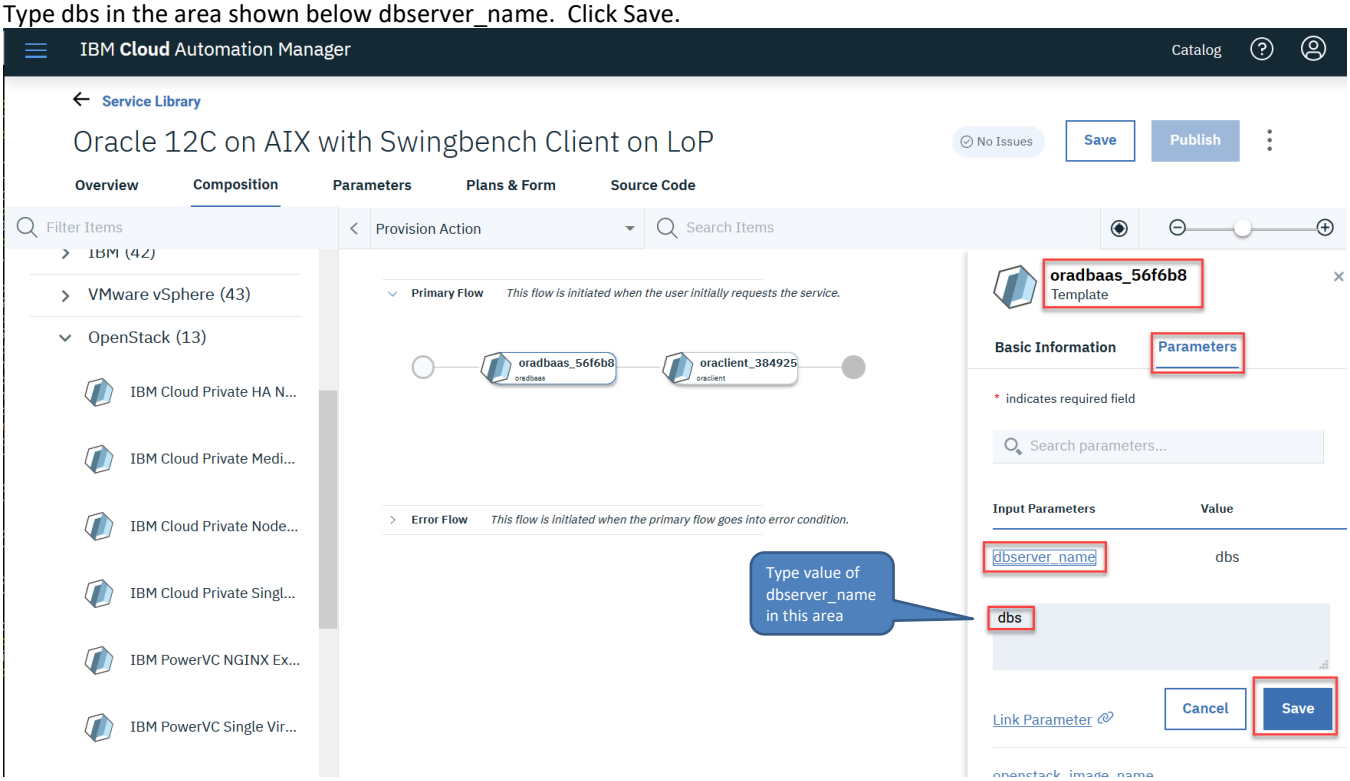
With oradbaas still selected, scroll down the right-hand panel to find **Instance Name**. This will specify the deployment instance name in CAM. Type in a suitable name, e.g. **dbs**. Also verify that the value under **Connection** reflects the correct cloud provider (PowerVC) for your environment.



Select the Parameters tab and click dbserver_name under Input Parameters. This dbserver_name will be the prefix of the VM name in PowerVC. (Recall from earlier that we are building the dbserver VM name in the format <prefix>-<dbname>-<YYMMDD>_<hhmmss>.)



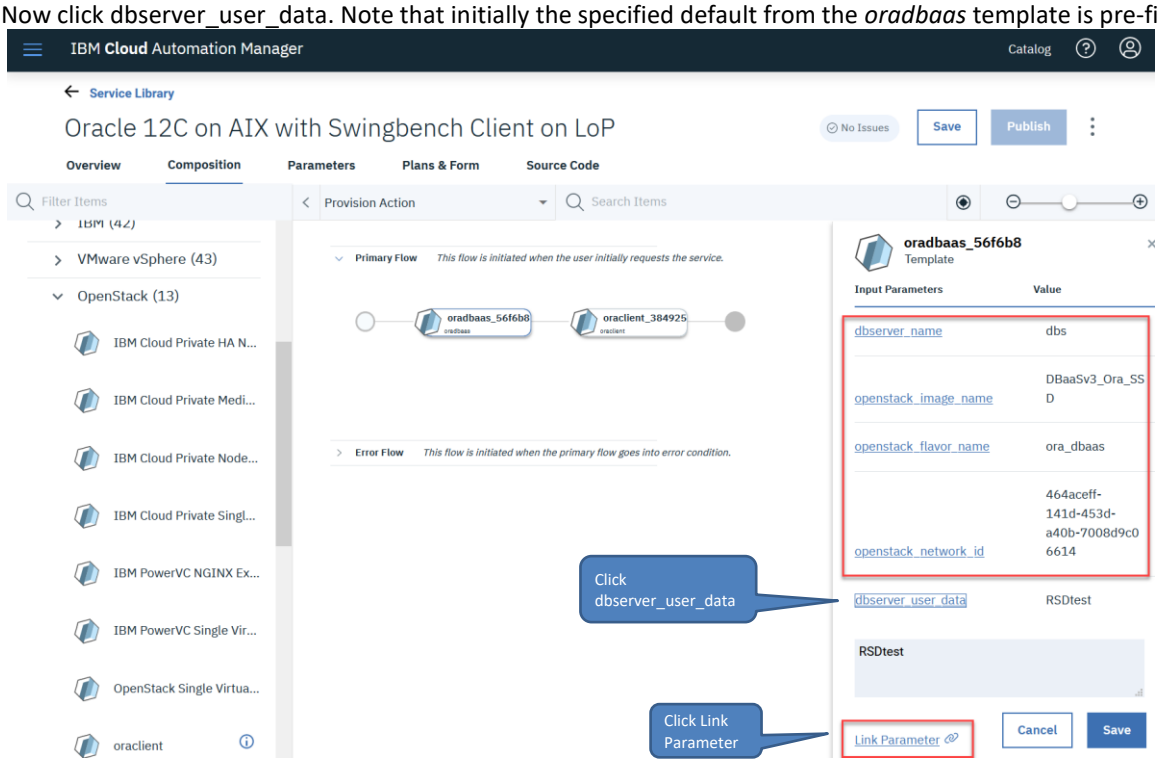
Type dbs in the area shown below dbserver_name. Click Save.



Repeat the previous step to specify the other input parameters we want to pre-set for this service. You need to determine the respective values from your specific PowerVC environment!

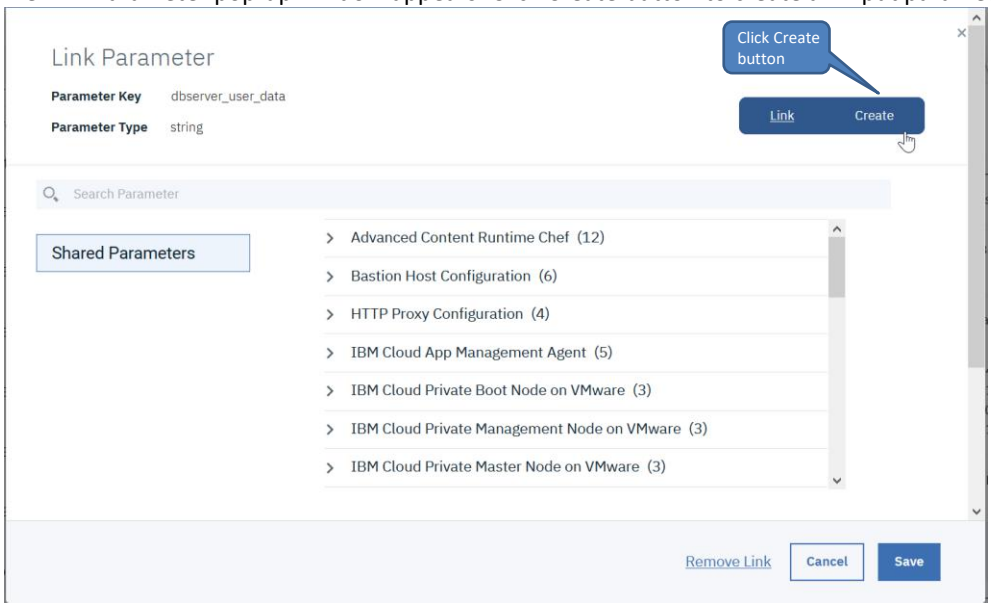
Input Parameter	Values from PowerVC
openstack_image_name,	DBaaSv3_Ora_SSD
openstack_flavor_name	ora_dbaas
openstack_network_id	464aceff-141d-453d-a40b-7008d9c06614

When completed, the right-hand panel of the screen will look like on the screen below.
Now click dbserver_user_data. Note that initially the specified default from the *oradbaas* template is pre-filled here.



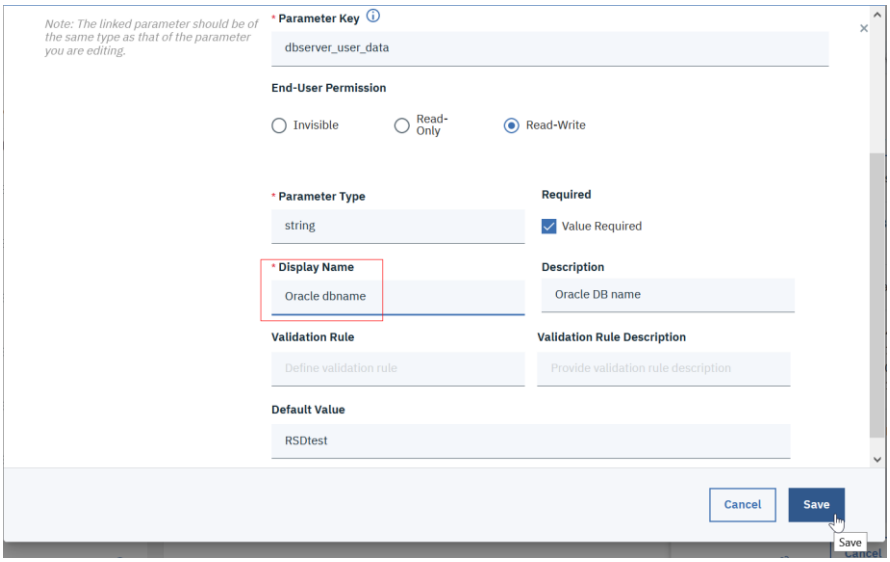
Under dbserver_user_data (may need to scroll down to find), click **Link Parameter.**

The Link Parameter pop-up window appears. Click Create button to create an input parameter for the service.

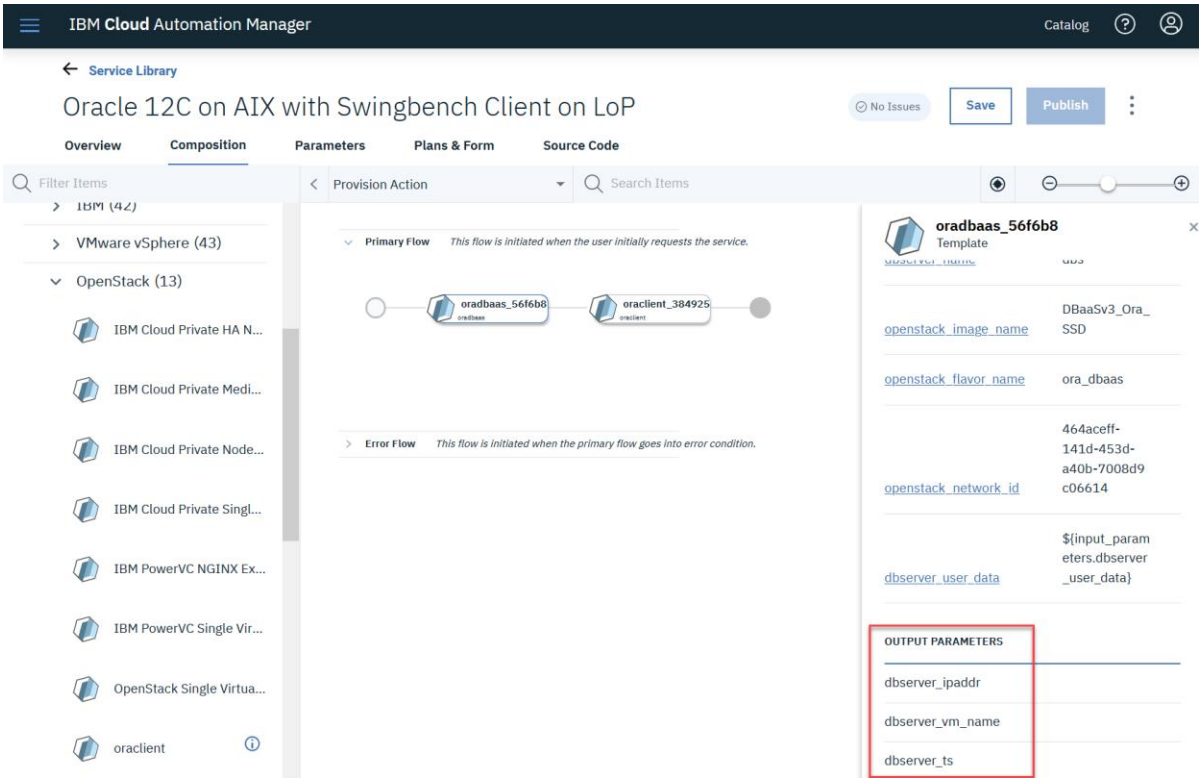


Scroll down to **Display Name** in the pop-up window and adjust to read “Oracle dbname”.

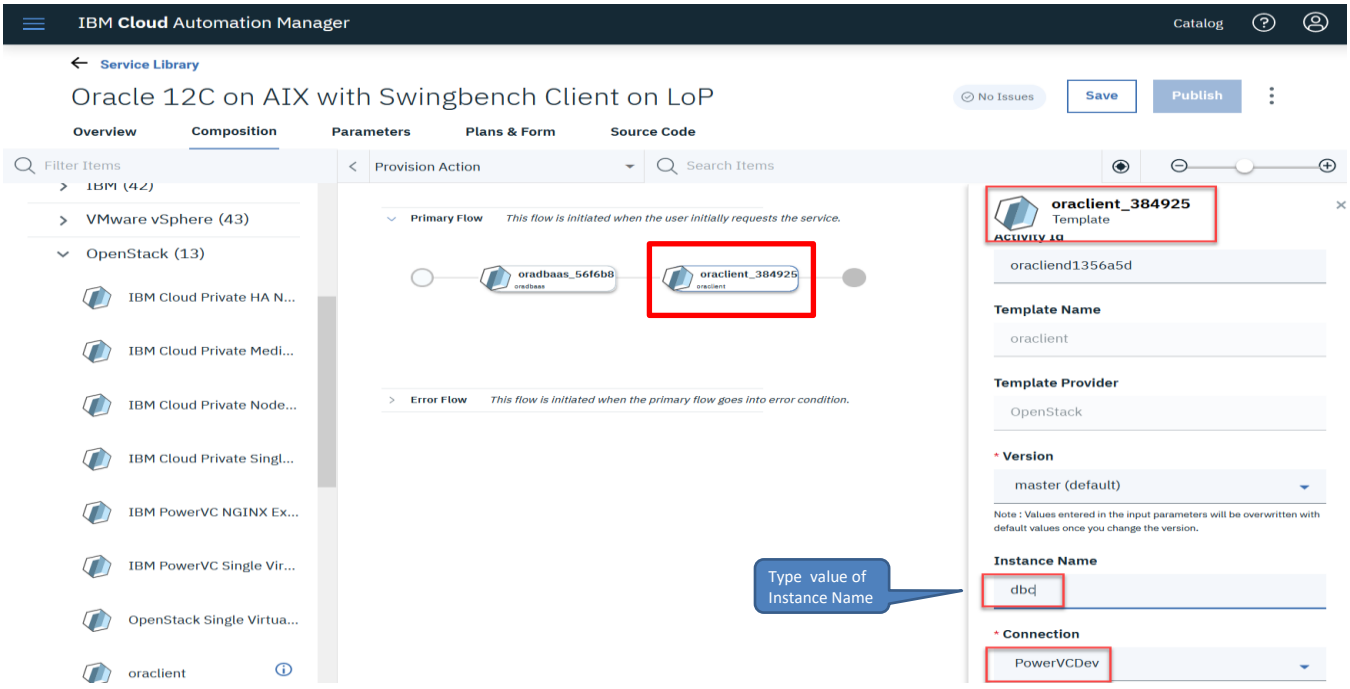
Review and click the **Save** button.



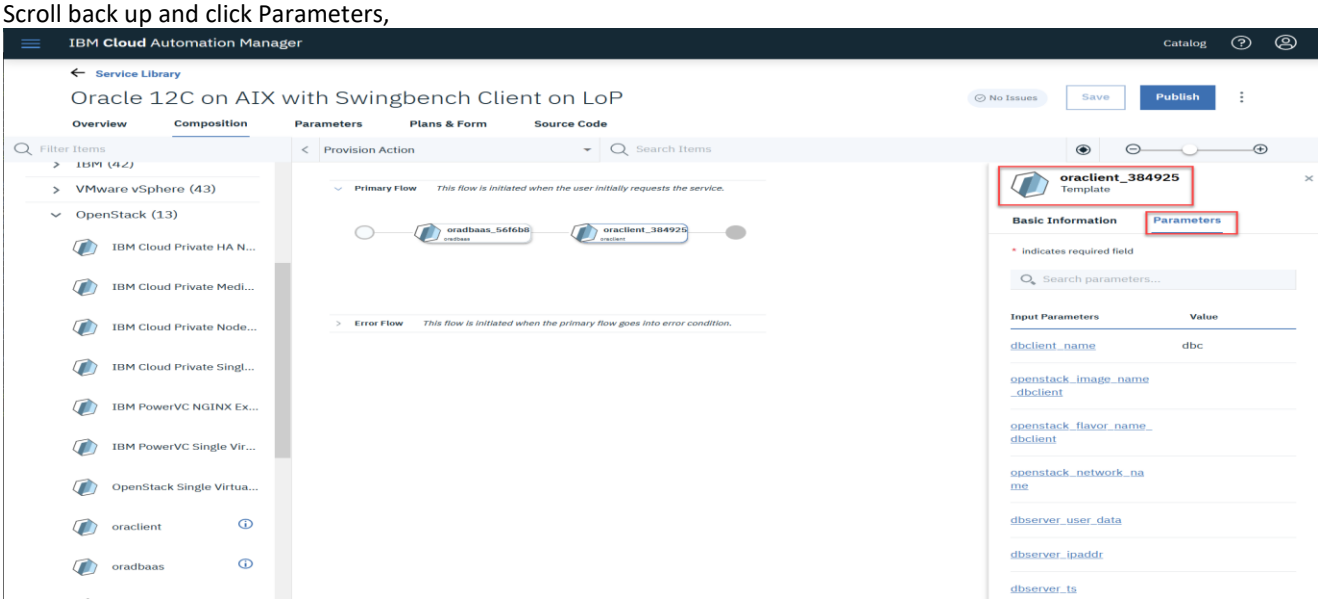
Note the Output Parameters (right-hand panel on next picture). These were defined in the *oradbaas* template (outputs.tf). Later, in section 9.2.2, we will create Service Output Parameters for dbserver_ipaddr and dbserver_vm_name, so that they will be displayed on the Services detailed information page.



Next, we will set up the input parameters of the *oraclient* template.
Select the *oraclient* template by clicking the oraclient_384925 icon in the middle panel of the screen. Scroll down on the right-hand panel to find **Instance Name** and type in **dbc** as shown below. Also verify that the correct cloud provider is selected in the **Connection** field. If you decide to use a different cloud provider (still PowerVC), that is possible as long as there is network connectivity between the two VMs to be created.



Scroll back up and click Parameters,



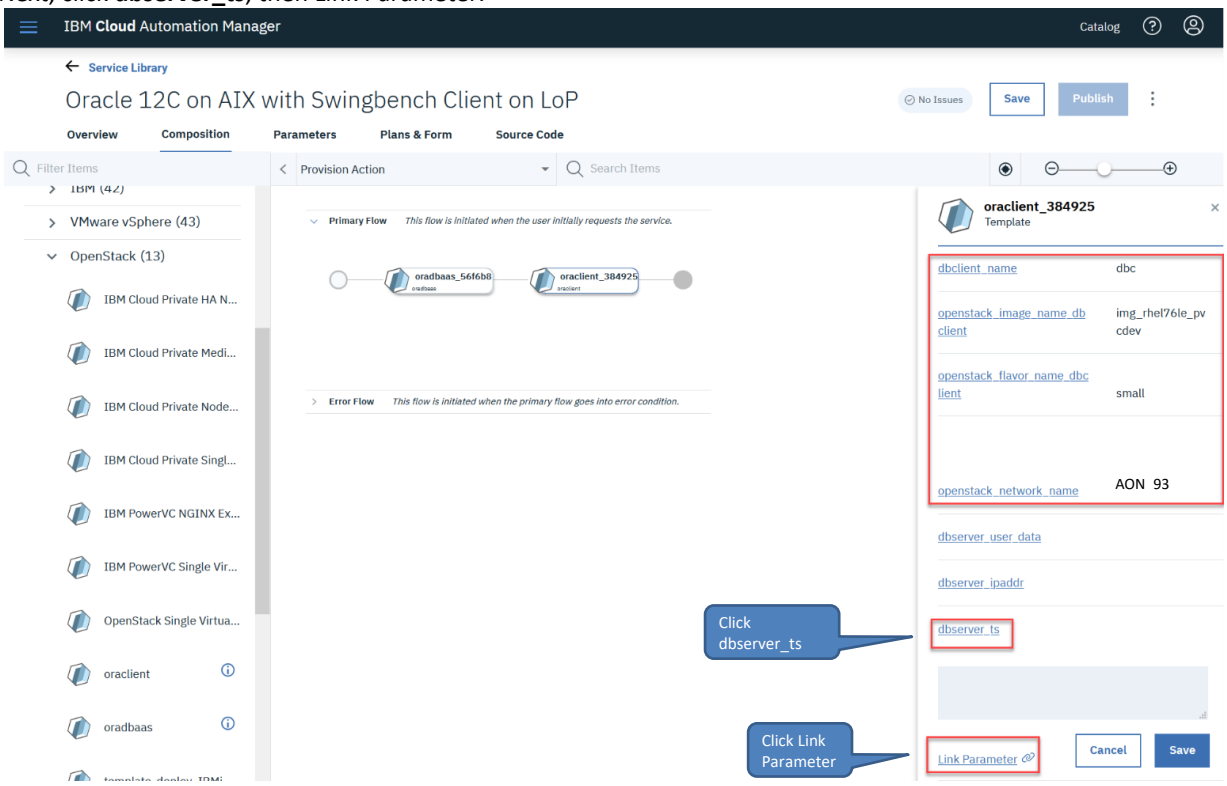
As we did before with the *oradbaas* template, click and fill in the values for your environment for the *oraclient* template. Note that at most the openstack_network_id will be identical to the *oradbaas* template values. Again, adjust the PowerVC specific values to your environment – rows 3-5 in the table below.

Input Parameter	Values
dbclient_name	dbc
openstack_image_name	img_rhel76le_pvcdev
dbclient_flavor_name	small
openstack_network_name	AON_93

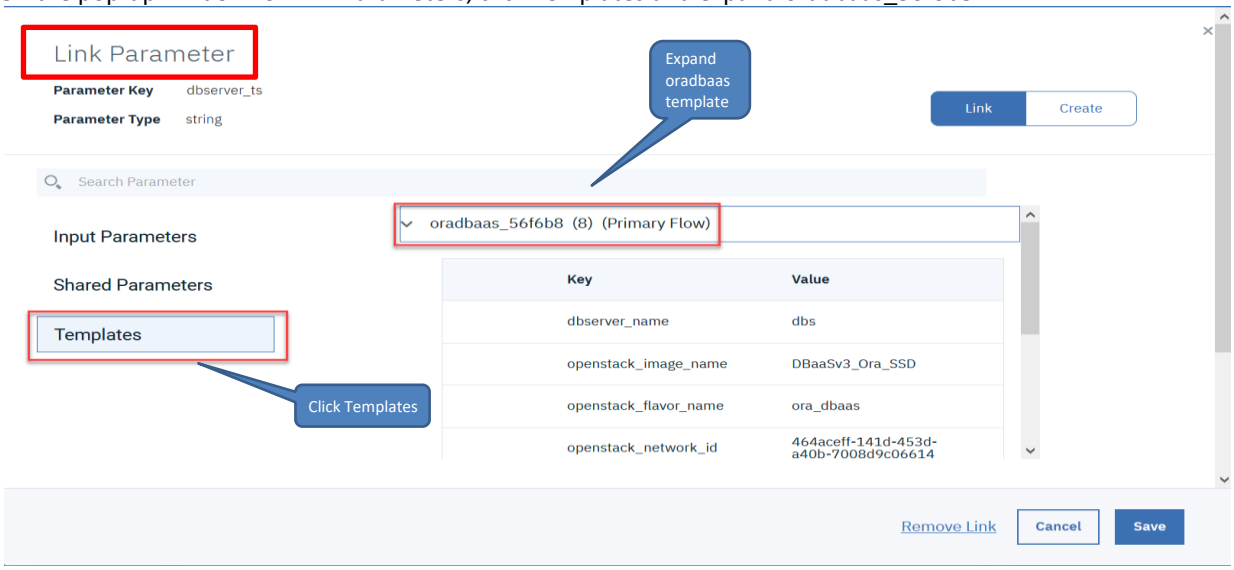
Note that the *oraclient* template uses the network name instead of network ID (UUID). AON_93 in this case is the name for the same network we specified for the *oradbaas* template as UUID.

Be sure to click the Save button for each parameter.

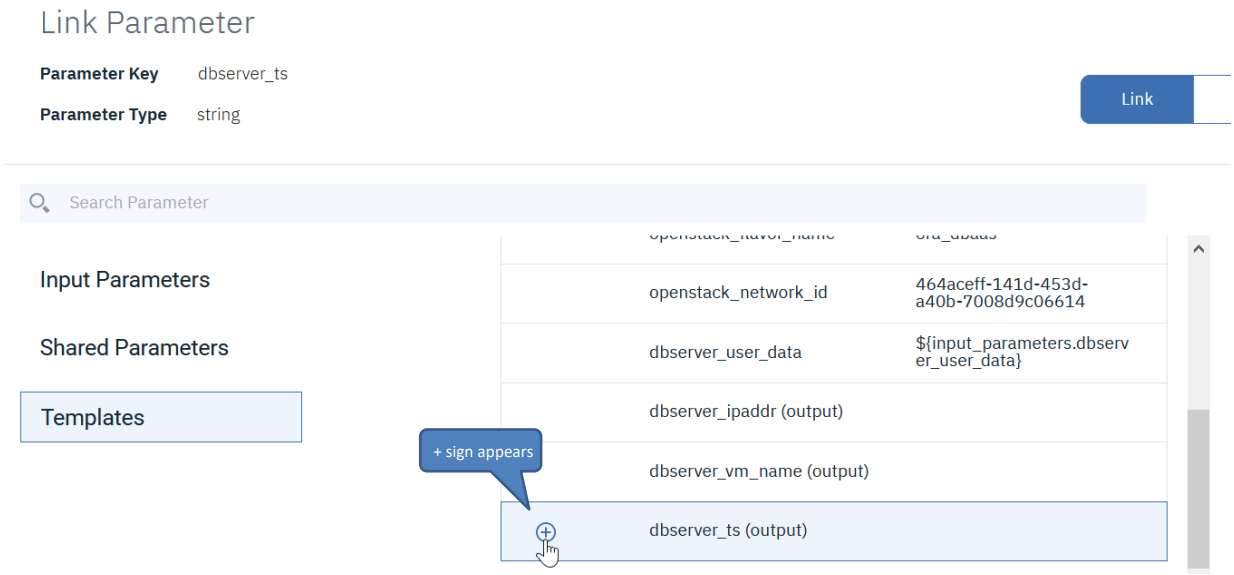
Next, click **dbserver_ts**, then Link Parameter.



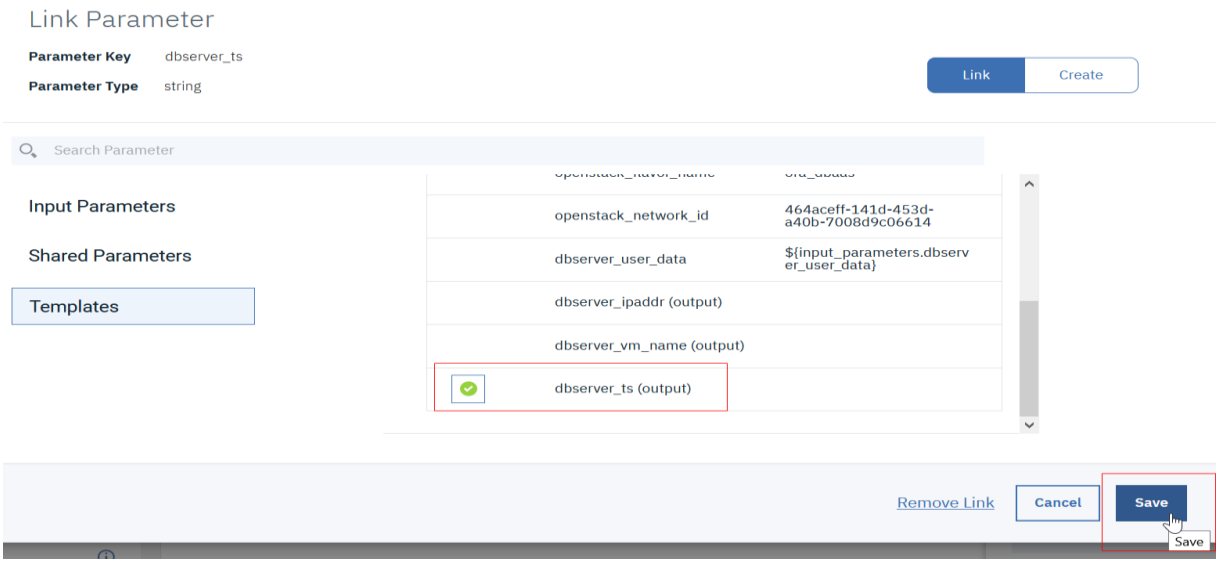
On the pop-up window for Link Parameters, click Templates and expand oradbaas_56f6b8.



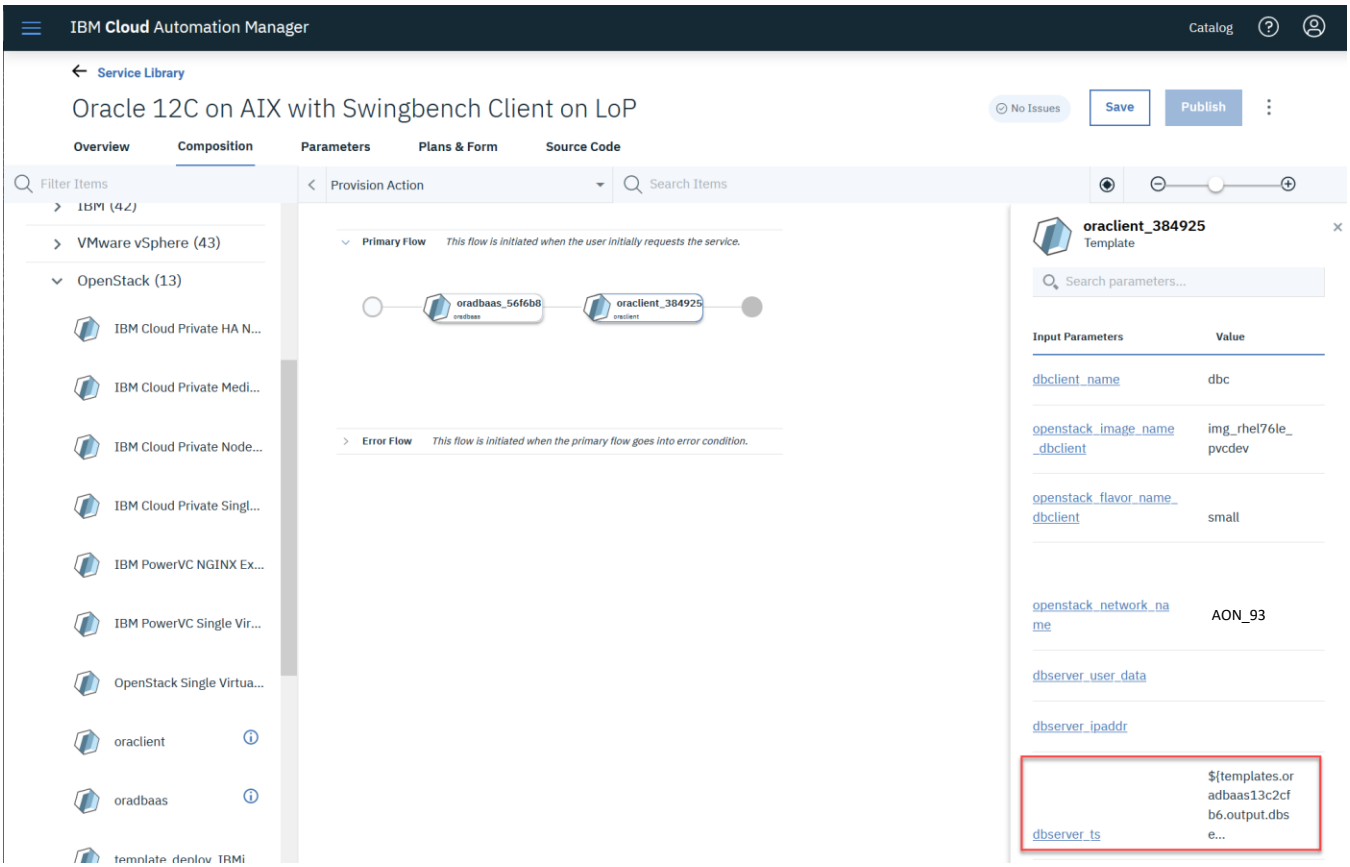
Scroll down to find the **dbserver_ts (output)** key and point the cursor to the dbserver_ts (output) row. The line becomes highlighted and a plus sign appears on the left at the beginning of the line.



Click the plus sign and the check mark appears (signifying selection). Click the Save button.

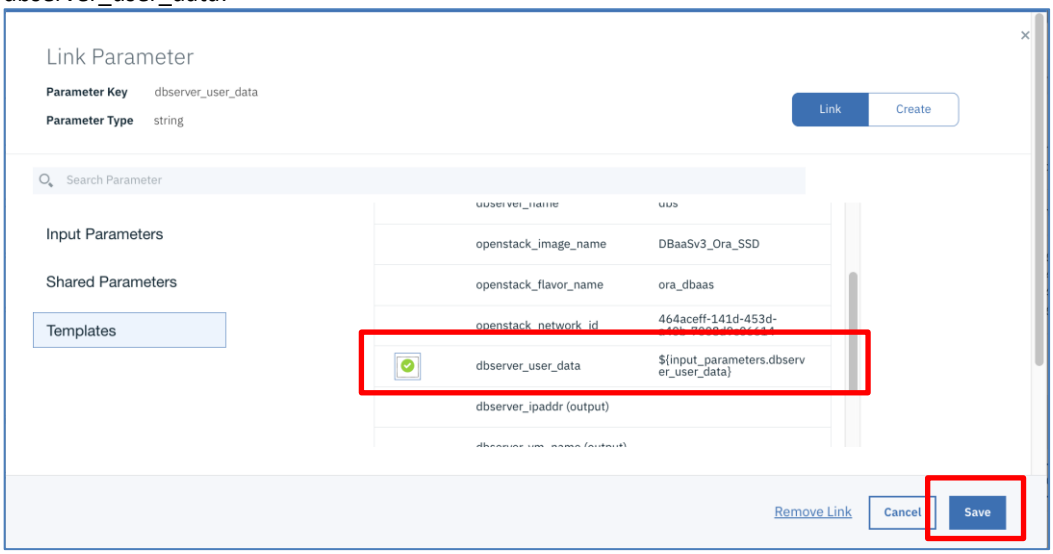


Notice that the value for **dbserver_ts** now points to the Activity ID of the *oradbaas* template. This step connects the output from *oradbaas* template to the input of *oraclient* template, allowing the VM name of the dbclient to have the same timestamp as that of dbserver.

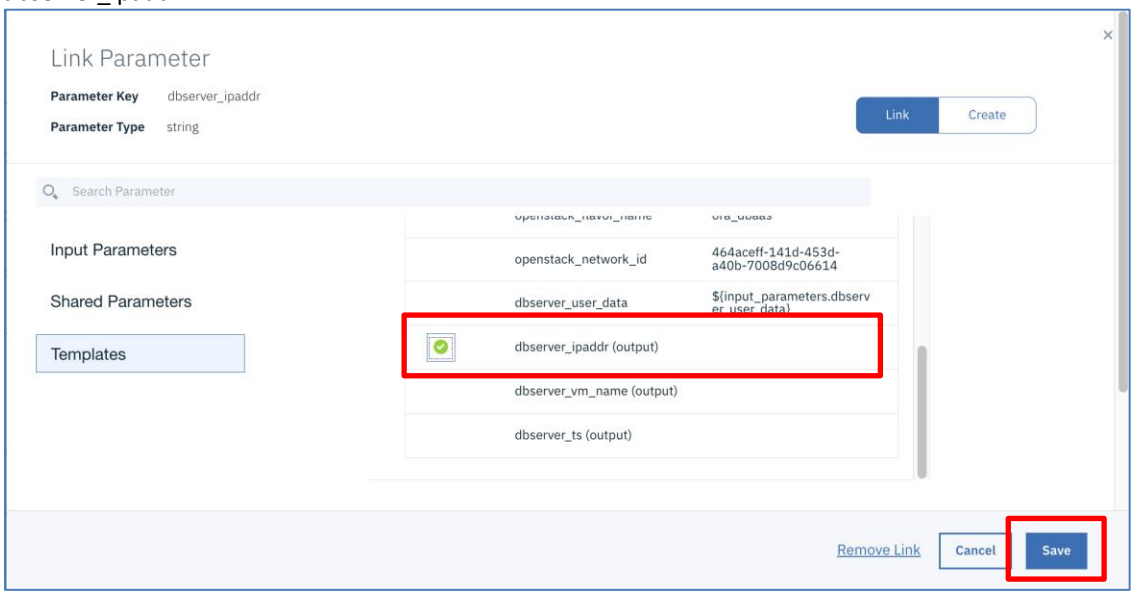


Repeat the above steps for dbserver_user_data and dbserver_ipaddr (only the pop-up windows for the Link Parameter step are shown).

dbserver_user_data:



dbserver_ipaddr:



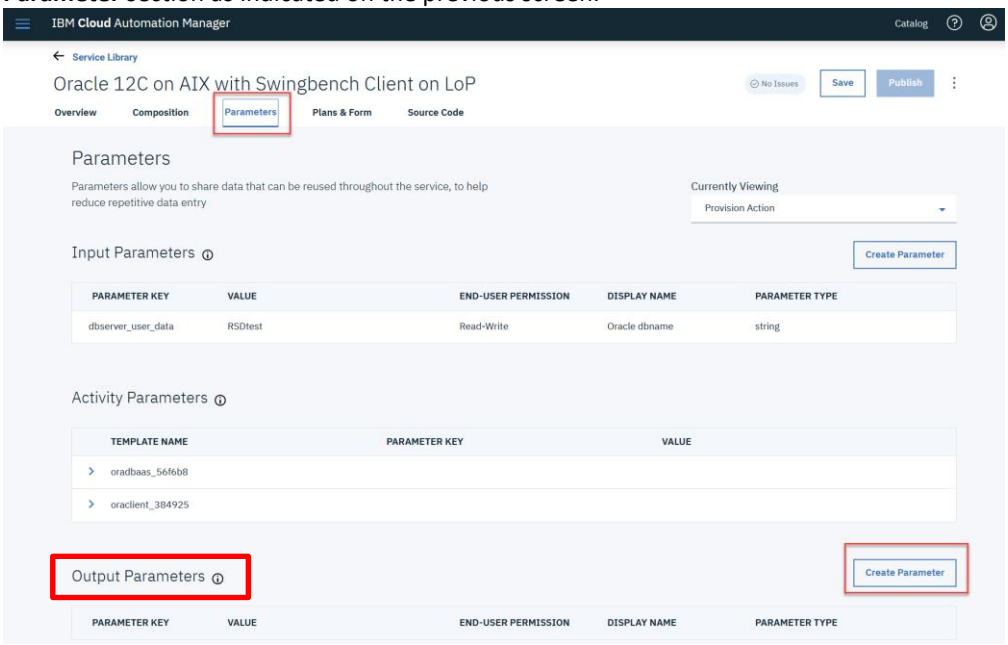
When done, the input parameters for *oraclient* template will look like the screen below.

In the screen shot below you also see the defined Output Parameters. In section 9.2.2, we create the Service's Output Parameters, which will display these parameters on the Service detailed information page.

9.2.2 Create Service Output Parameters

An end user needs to be able to easily find the access information for the deployed service. To provide that data we create service output parameters which will be displayed on the deployed service screen.

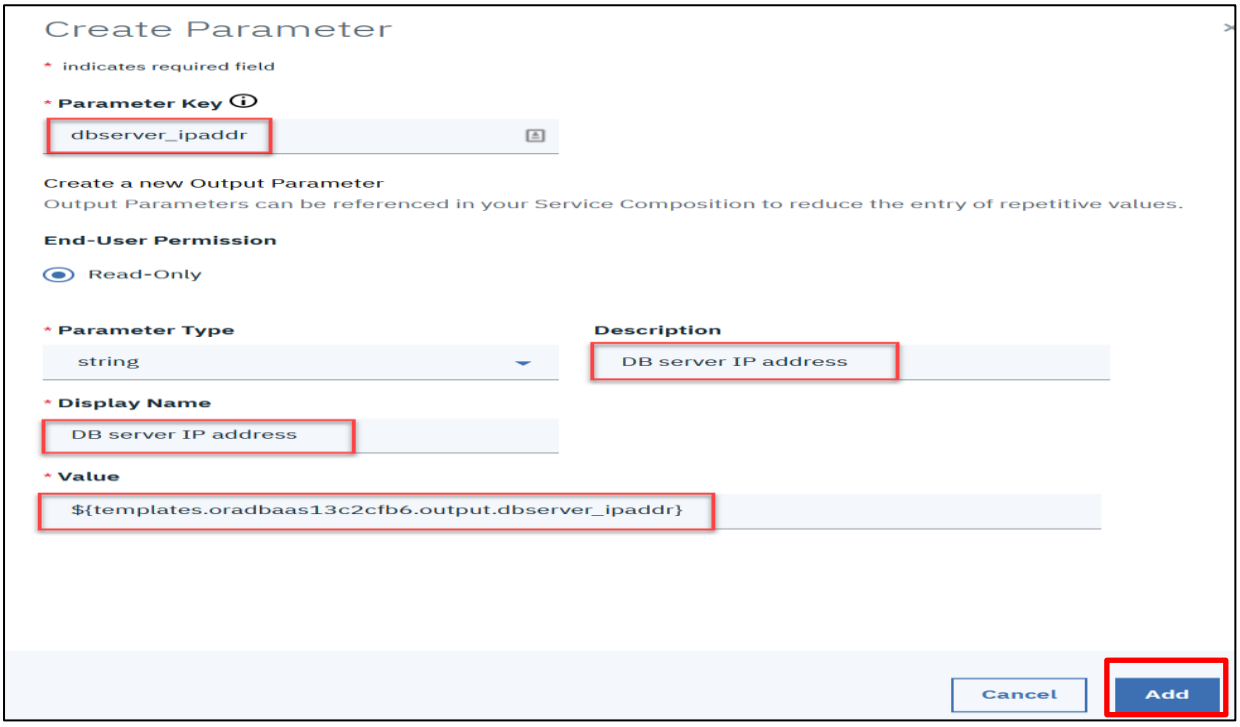
On the service creation screen select the **Parameters** tab and then click the **Create Parameter** button for the **Output Parameter** section as indicated on the previous screen.



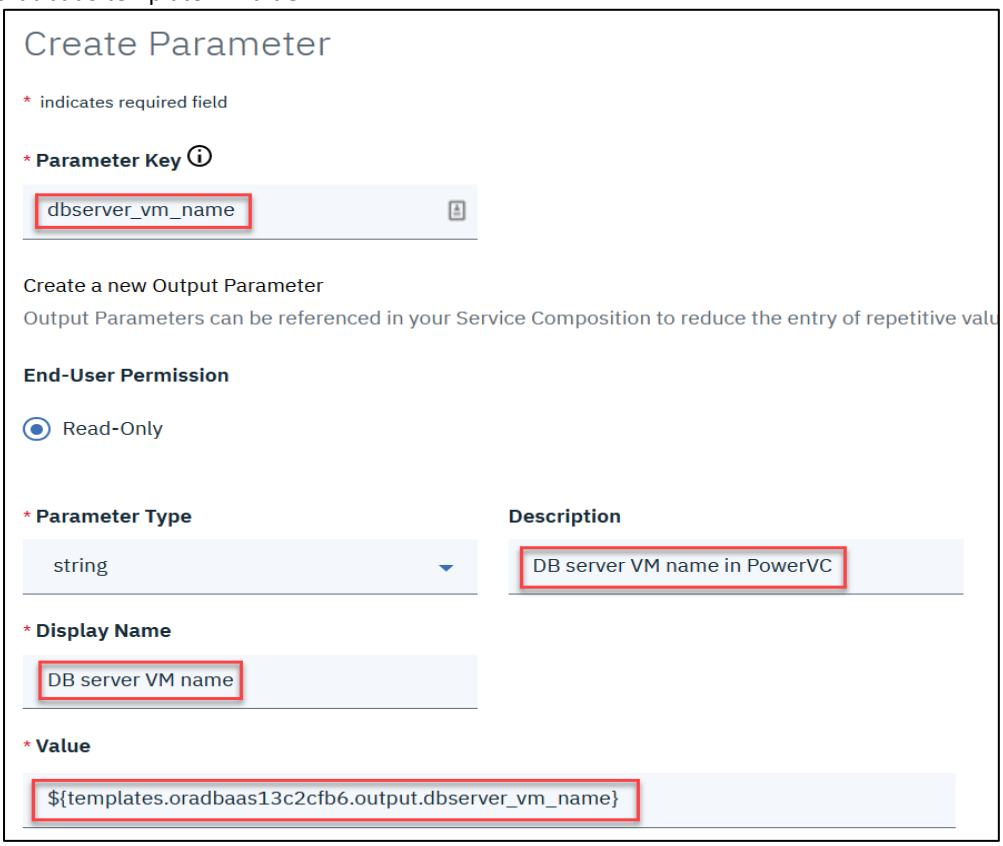
Fill in the parameter shown below. Here we will need the Activity IDs saved earlier. If you need to find them again, go back to the **Composition** tab and select **Basic Information**, after selecting the correct template on the canvas.
Just to repeat – these Activity IDs will be different in your environment and need to be adjusted!

Template	Activity Id
oradbaas	oradbaas13c2cfb6
oraclient	oracliend1356a5d

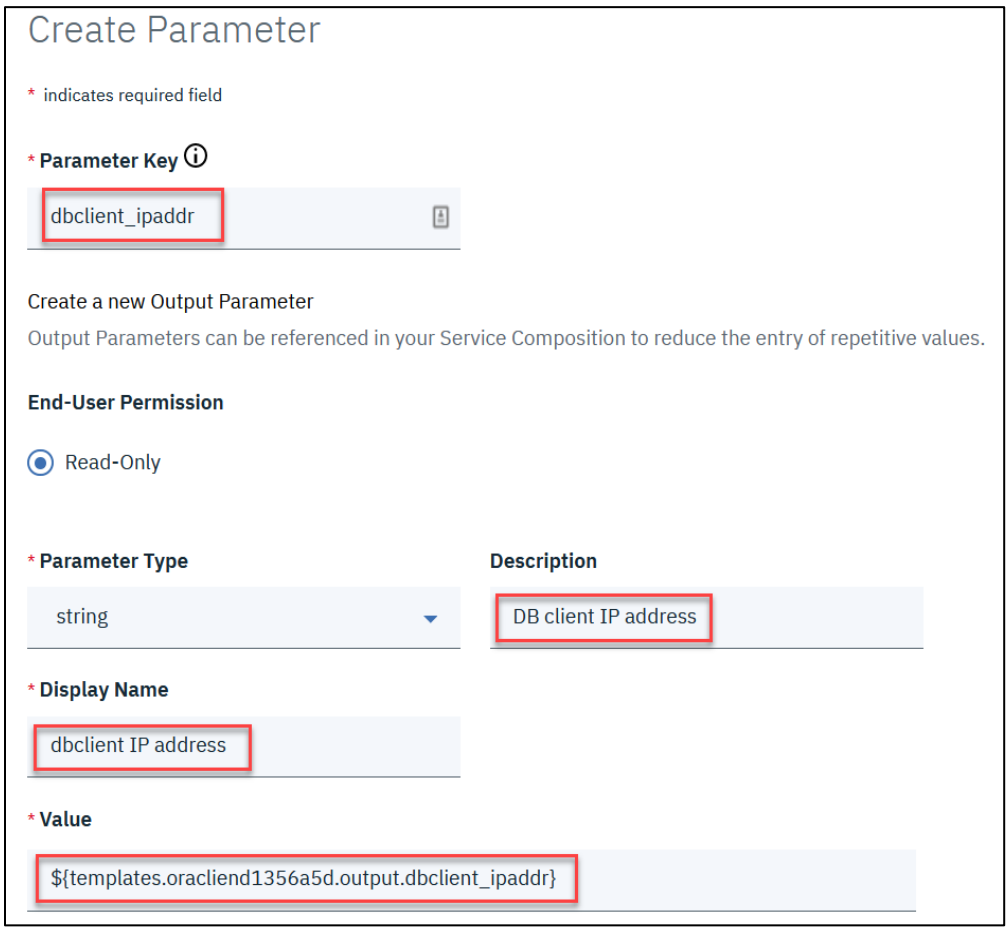
For new output parameter dbserver_ipaddr, complete the definition screen as shown below. Note that you will need to substitute your Activity ID for ***oradbaas*** in the **Value** field. After filling the fields in click **Add**.



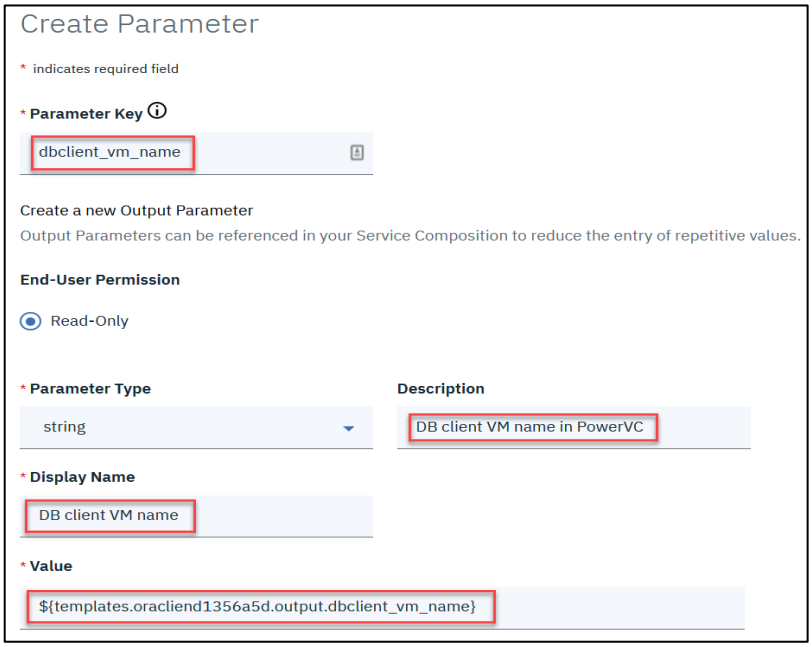
For new output parameter dbserver_vm_name, again ensure to replace the Activity ID with the correct value from your ***oradbaas*** template in **Value**.



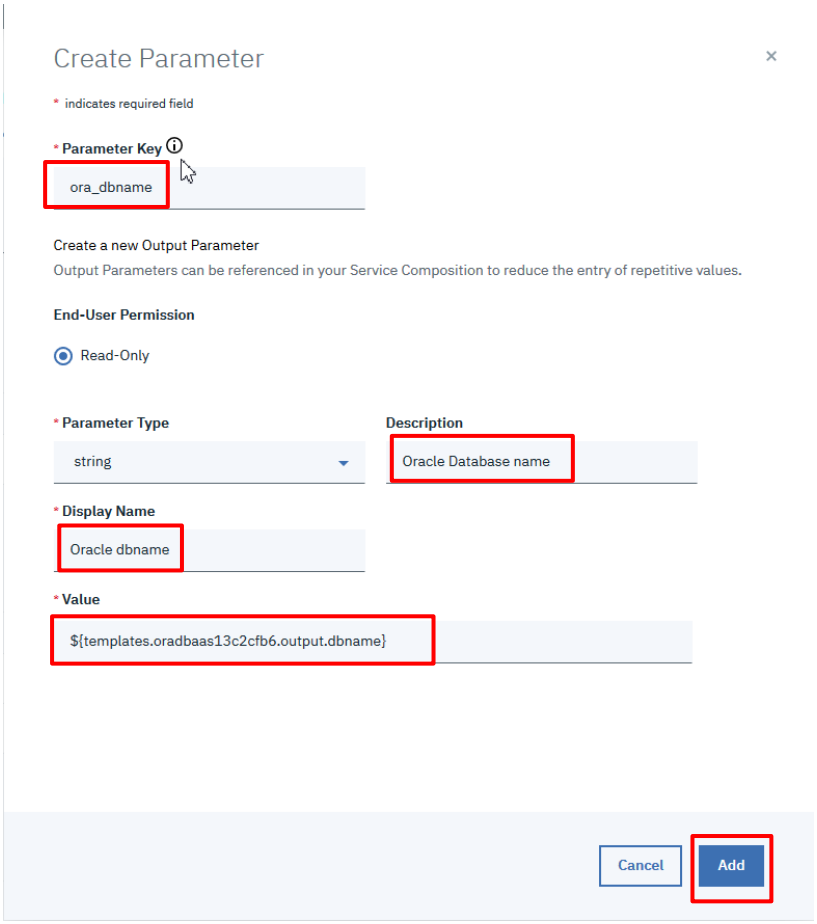
For dbclient_ipaddr, again ensure to replace the Activity ID with the correct value from your ***oraclient*** template in **Value**.



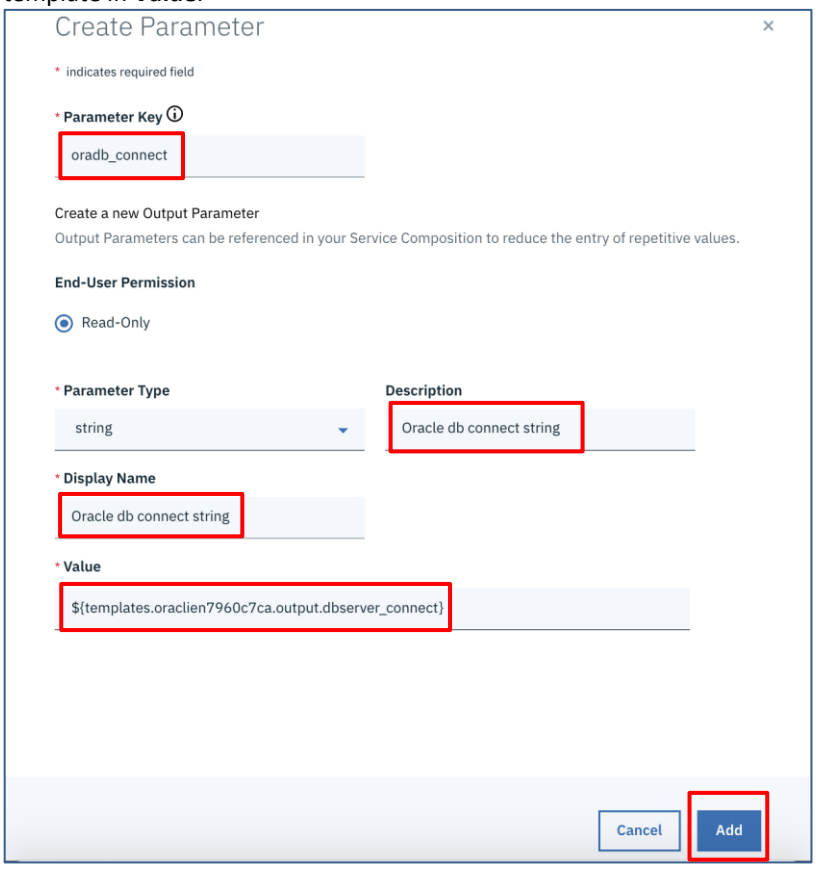
For dbclient_vm_name, again ensure to replace the Activity ID with the correct value from your ***oraclient*** template in **Value**.



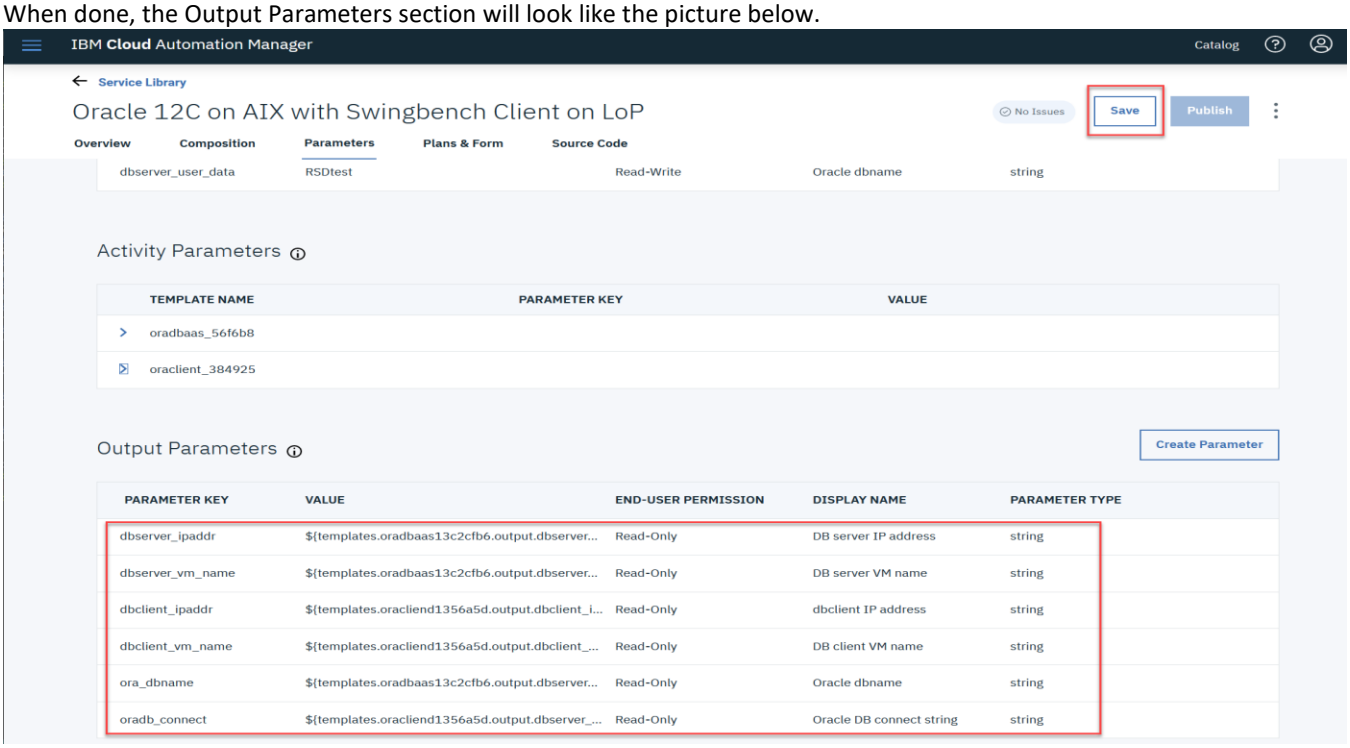
For Oracle dbname, again ensure to replace the Activity ID with the correct value from your ***oradbaas*** template in **Value**.



And finally, for Oracle database connect string, be sure to replace the Activity ID with the correct value from your ***oraclient*** template in **Value**.



When done, the Output Parameters section will look like the picture below.

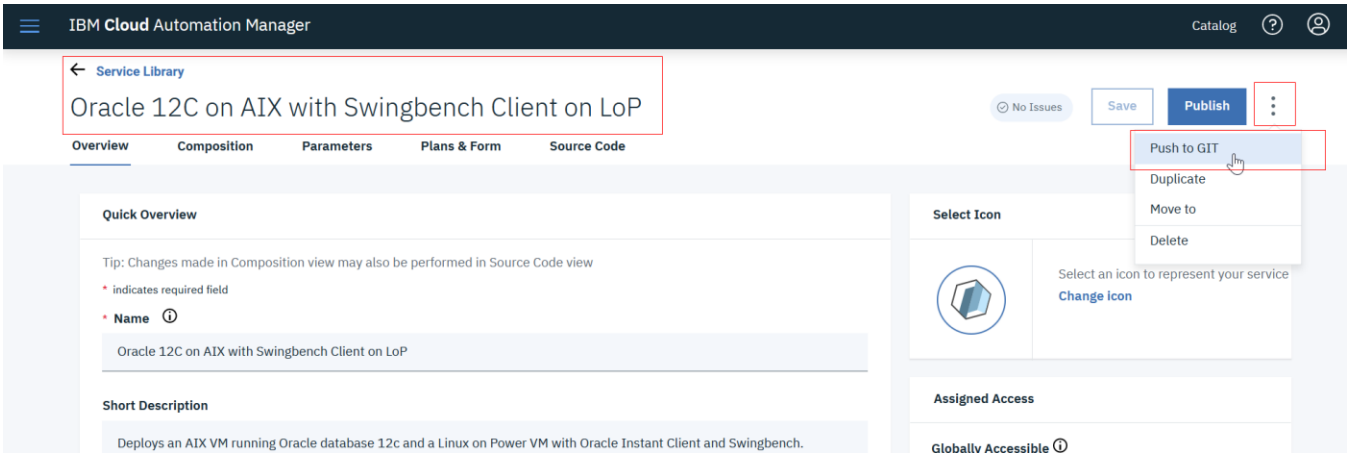


Click the **Save** button as indicated above.
At this point you may want to Save (Push) the Service to a GitHub repository as shown in the next section Push Service to GitHub 9.3.

9.3 Push Service to GitHub

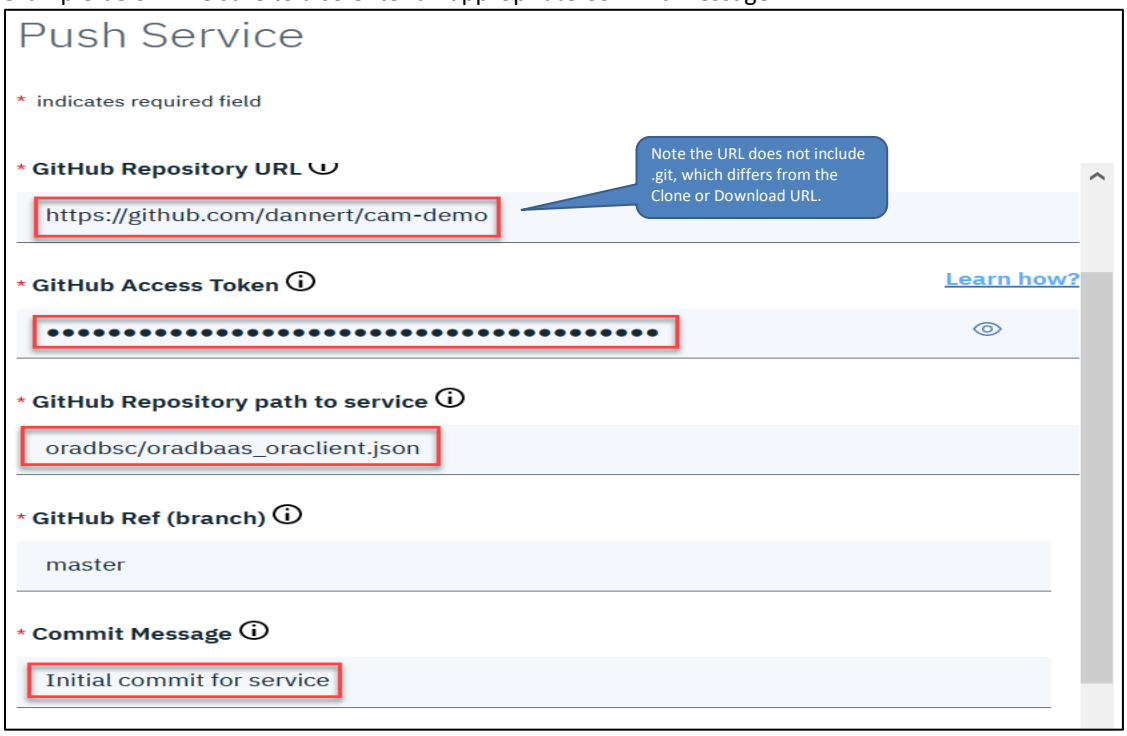
To facilitate distribution and collaboration without having to rebuild the service from scratch each time, push the service to a GitHub repository. To do this, access the Service Library and edit the service.

Click the three-dot menu next to the Publish button and select Push to GIT as shown below.



Since we created the service using Service Composer, rather than importing from Git, we need to provide the Git repository details.

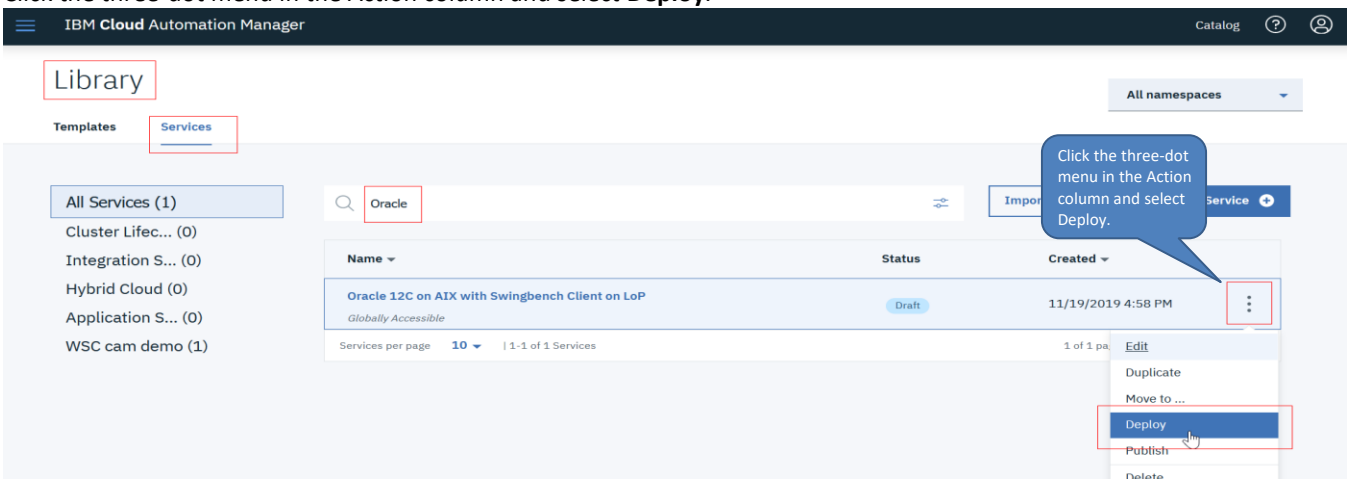
In our scenario, we created a new folder named **oradbsc** in the cam-demo project via GitHub WEB interface first and then performed the **Push to GIT**. On the pop-up screen for Push Service enter the appropriate repository details as shown in the example below. Be sure to also enter an appropriate Commit Message.



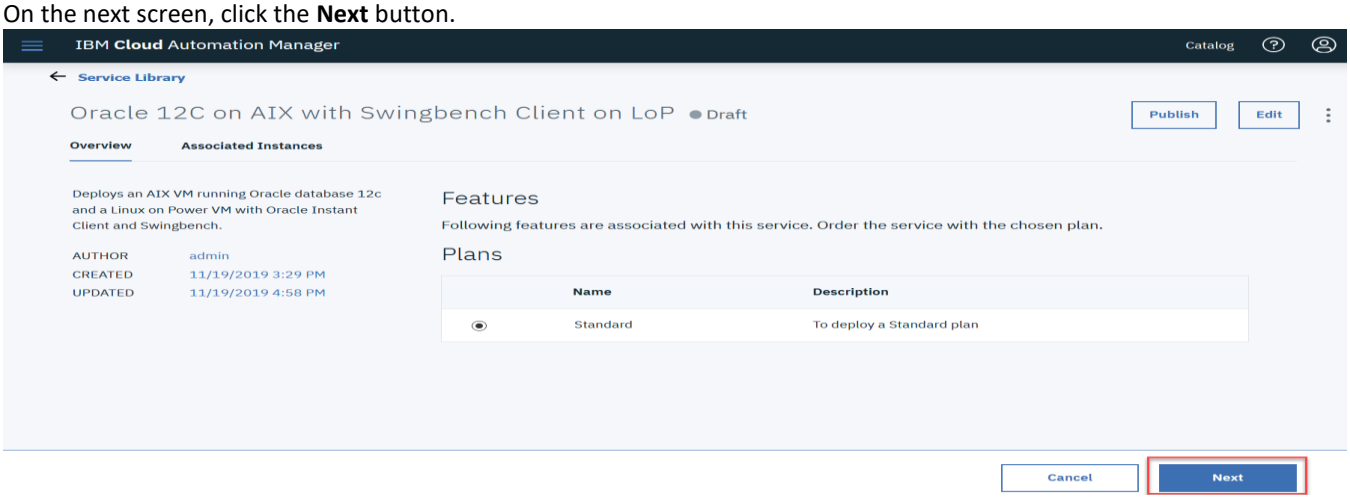
9.4 Deploying the Service

Return to the Service Library by clicking the link on the left-hand upper corner. Alternatively, click the hamburger menu, then Library->Services.

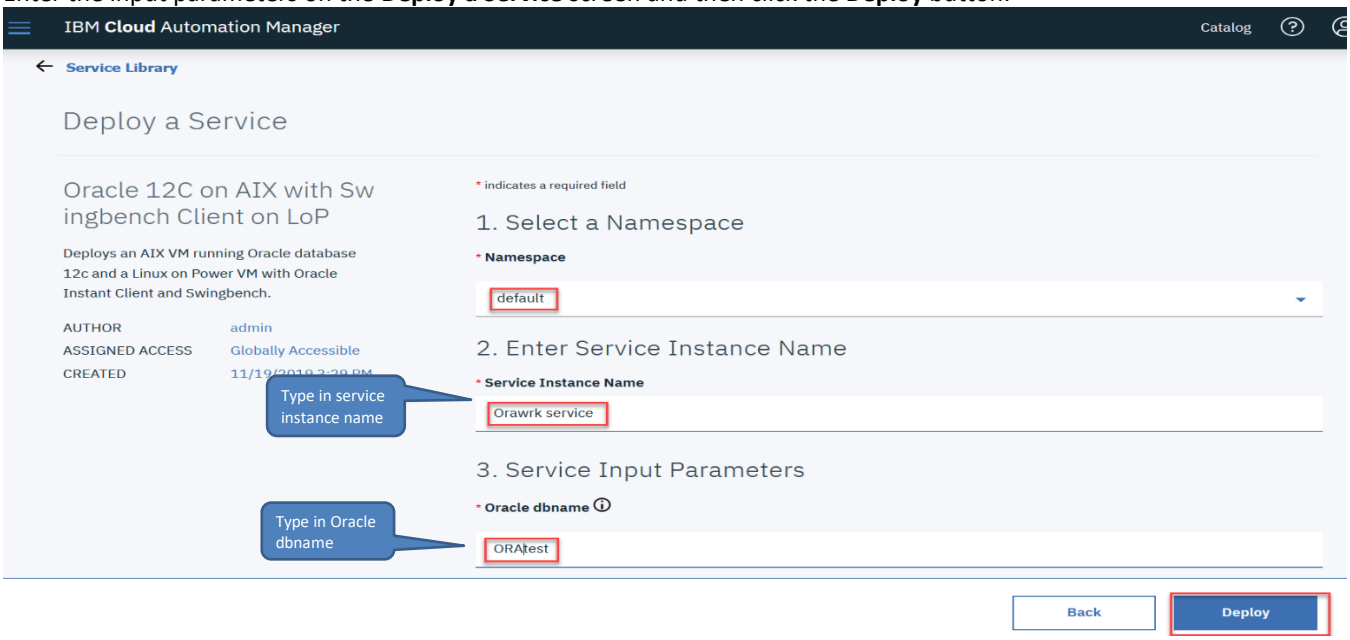
Click the three-dot menu in the Action column and select **Deploy**.



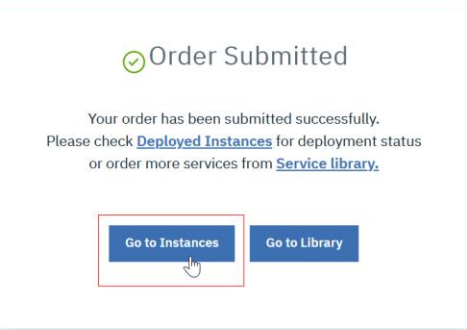
On the next screen, click the **Next** button.



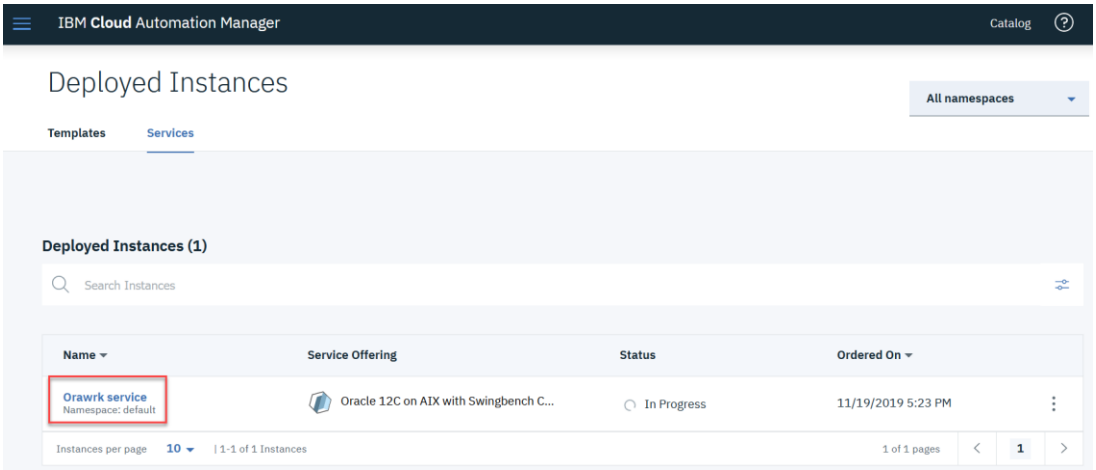
Enter the input parameters on the **Deploy a Service** screen and then click the **Deploy** button.



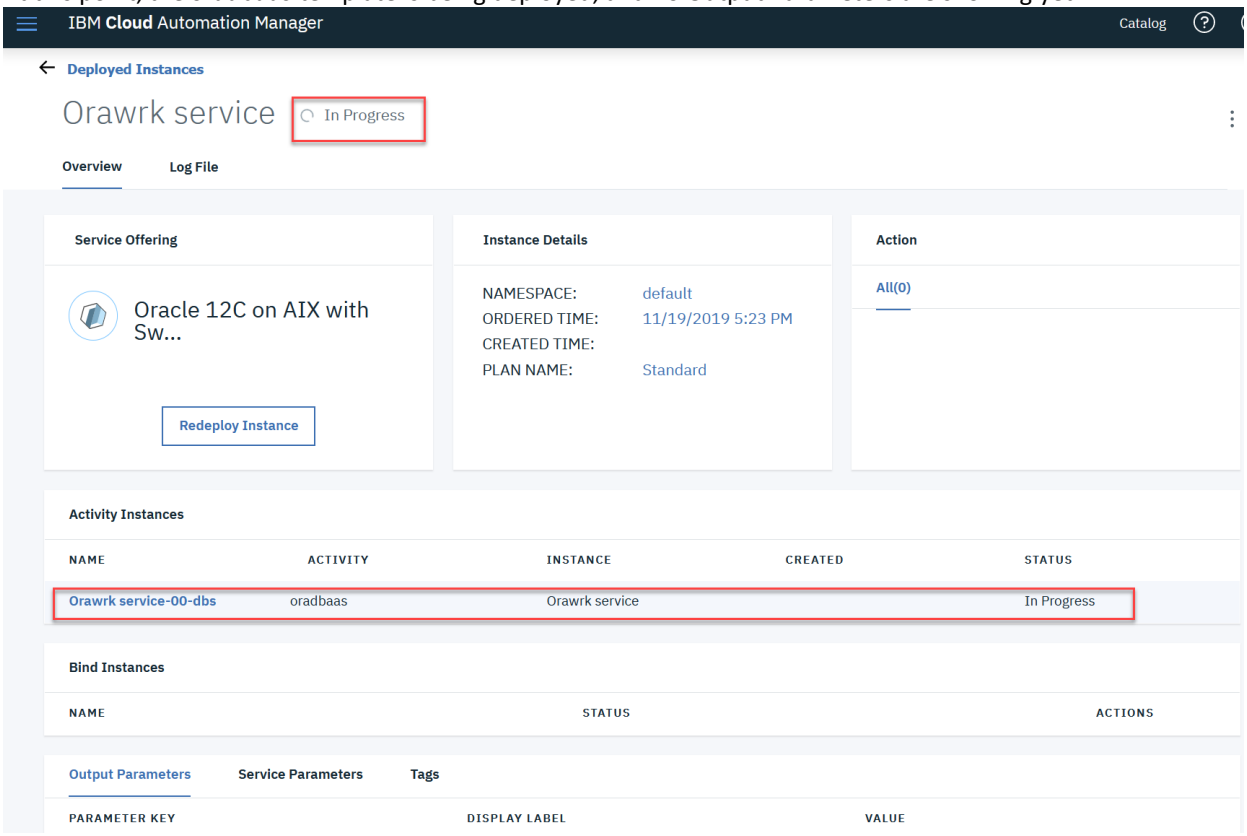
A pop-up window appears to indicate order submission. Click the **Go to Instances** button.



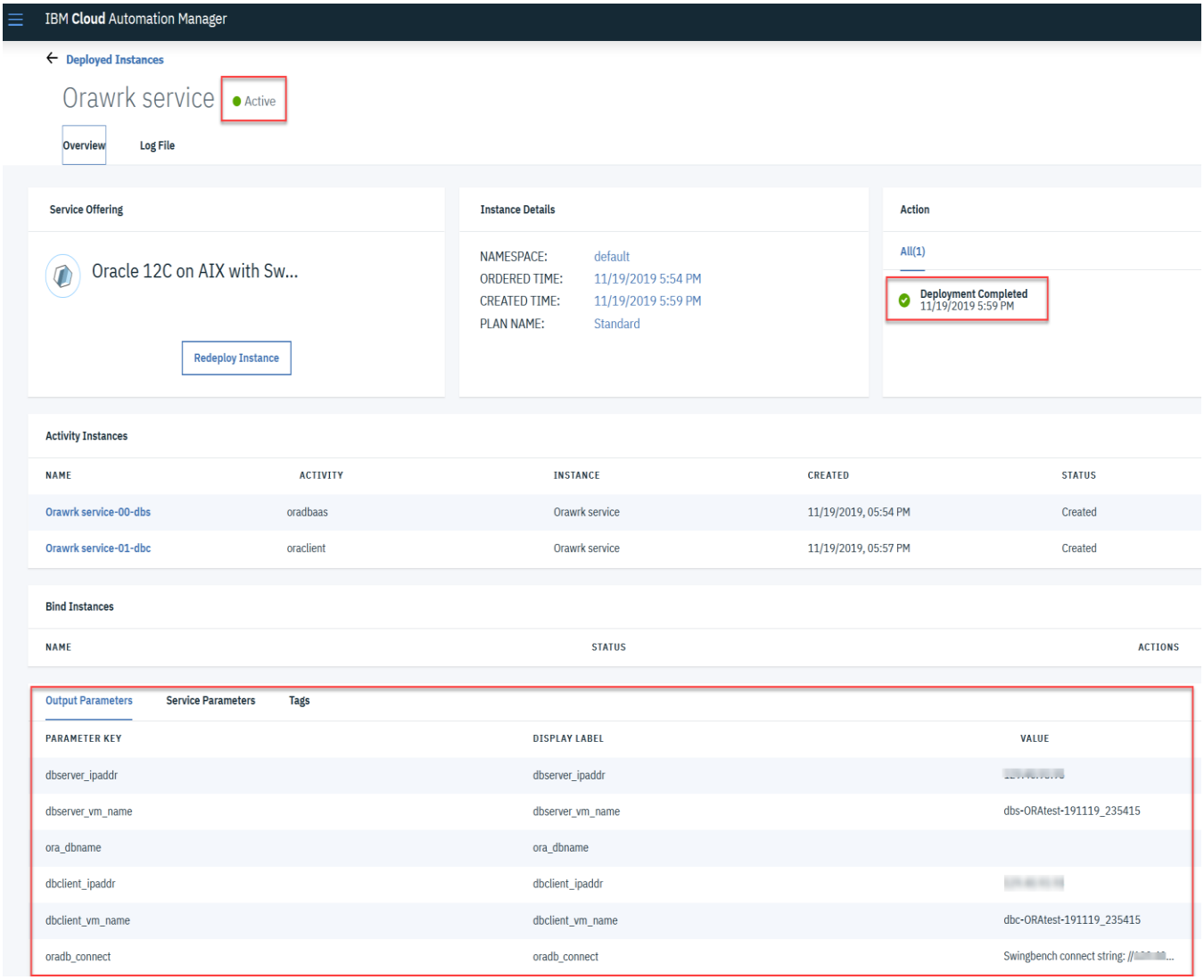
The **Deployed Instances** screen is displayed. Note the name under Name, **Orawrk service**, which we entered earlier. Click on **Orawrk service** for more details.



At this point, the *oradbaas* template is being deployed, and no Output Parameters are showing yet.



After the service is fully deployed and in status Active, the Service details page shows two VMs (from *oradbaas* and *oraclient* templates) deployed and all output parameters are filled in.

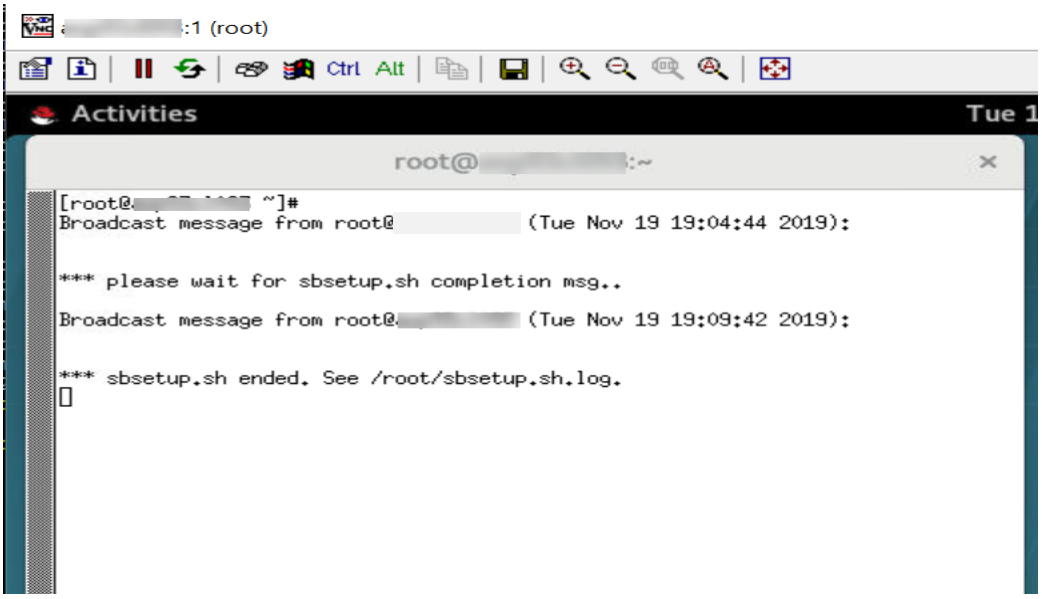


9.5 Test the deployed service

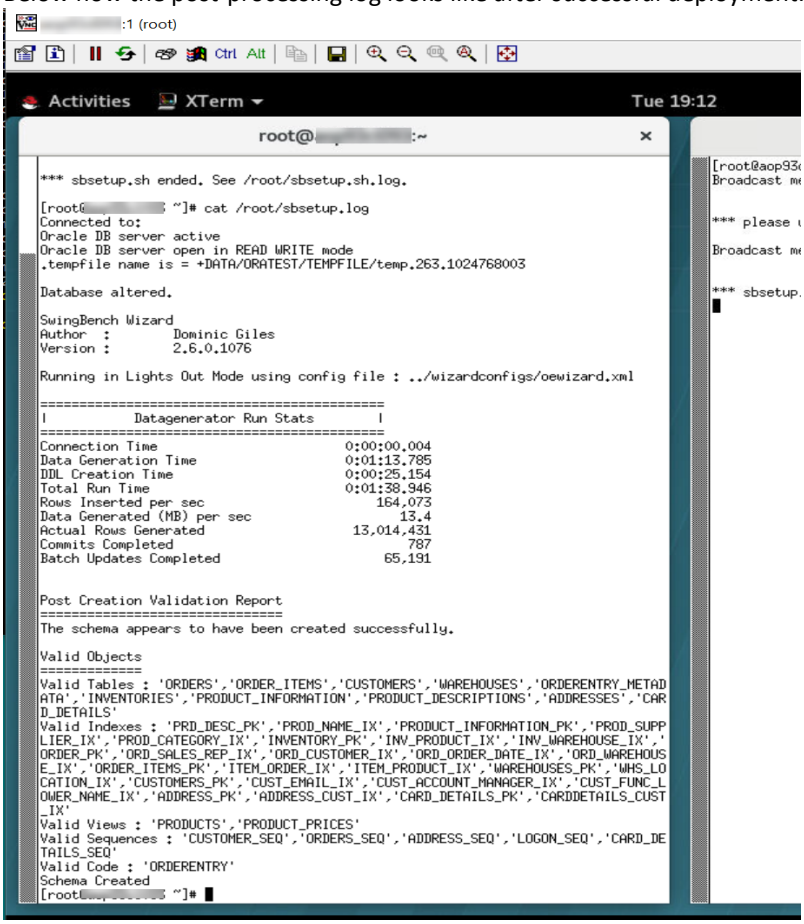
From the parameter output in the services screen we can utilize the value reported in dbclient_ipaddr to open a VNC viewer session to the dbclient server – <ip address>:1. This gets us to the X-session with an xterm running as root user in this case.

Note that the service deployment in CAM will show as complete significantly before all post-processing in the dbclient server is actually complete. The service deployment, from the viewpoint of CAM and PowerVC, is complete with the successful creation and start of the dbclient VM, regardless of the success of any post-deployment steps.

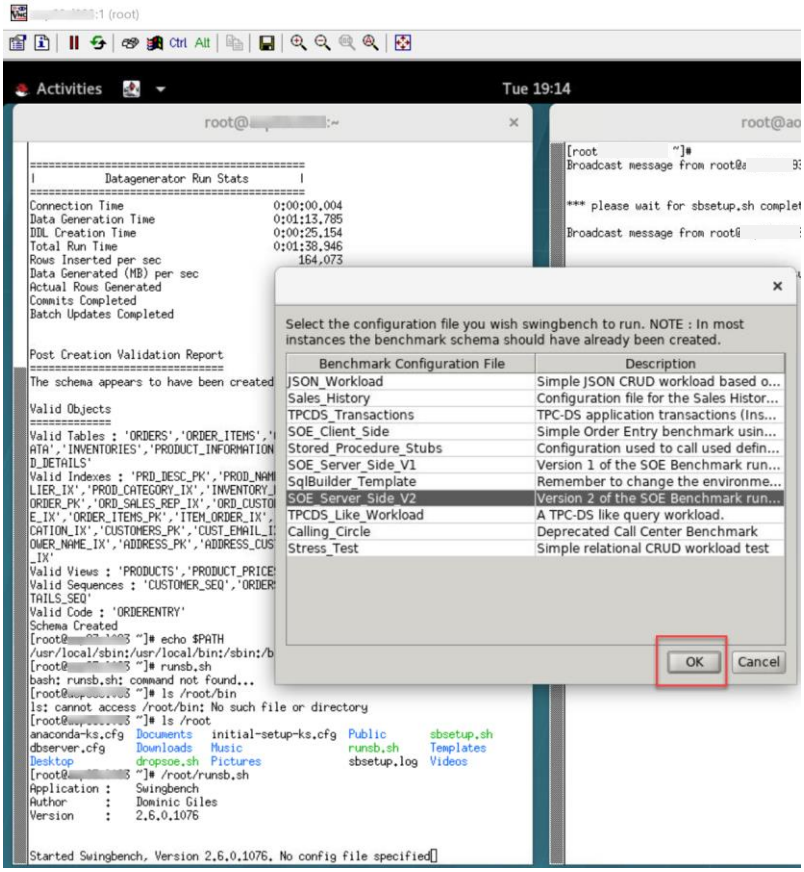
To account for this behavior, the client publishes the start and stop messages of when the data load started and when it completed. Only after completion, as shown below, can the swingbench driver workload be successfully executed.



Below how the post-processing log looks like after successful deployment.

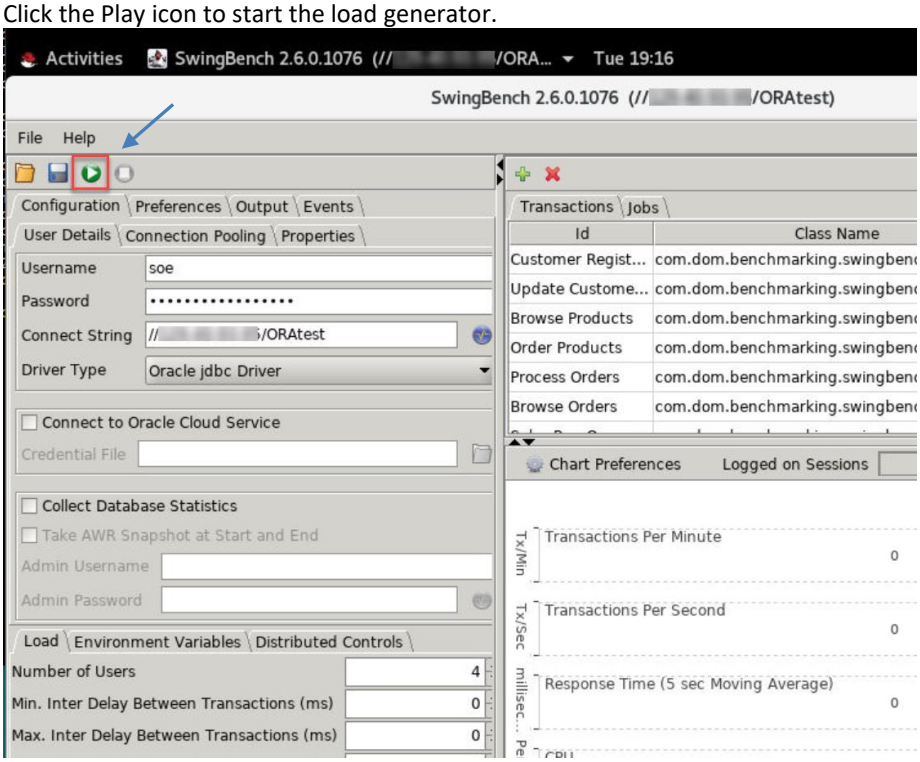


To illustrate that the service deployment achieved the expected end result, we ran the swingbench driver by executing the /root/runsb.sh script:

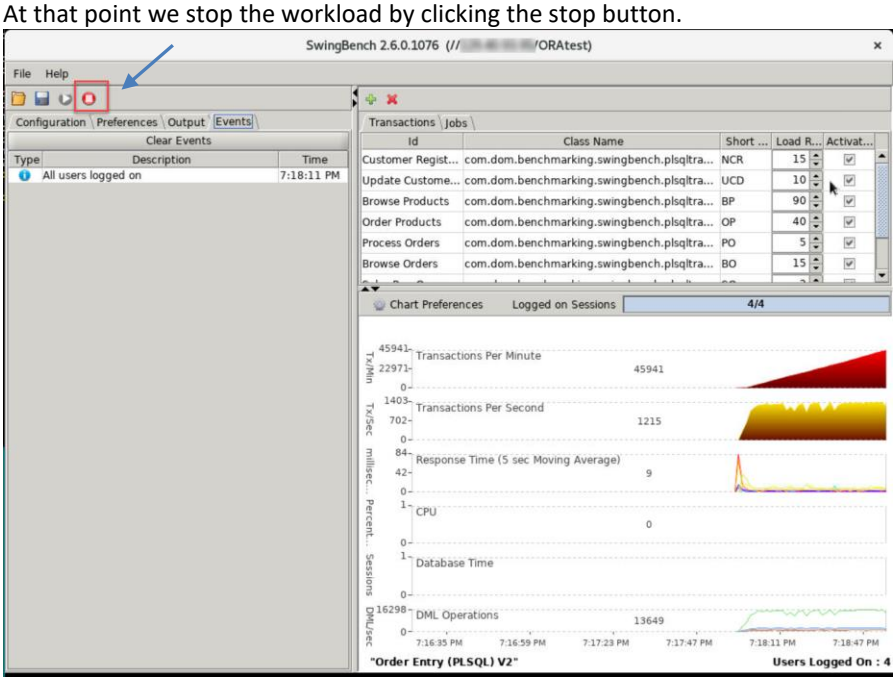


Click the OK button.
Note the connect string and other fields have been pre-filled by the script.

Click the Play icon to start the load generator.



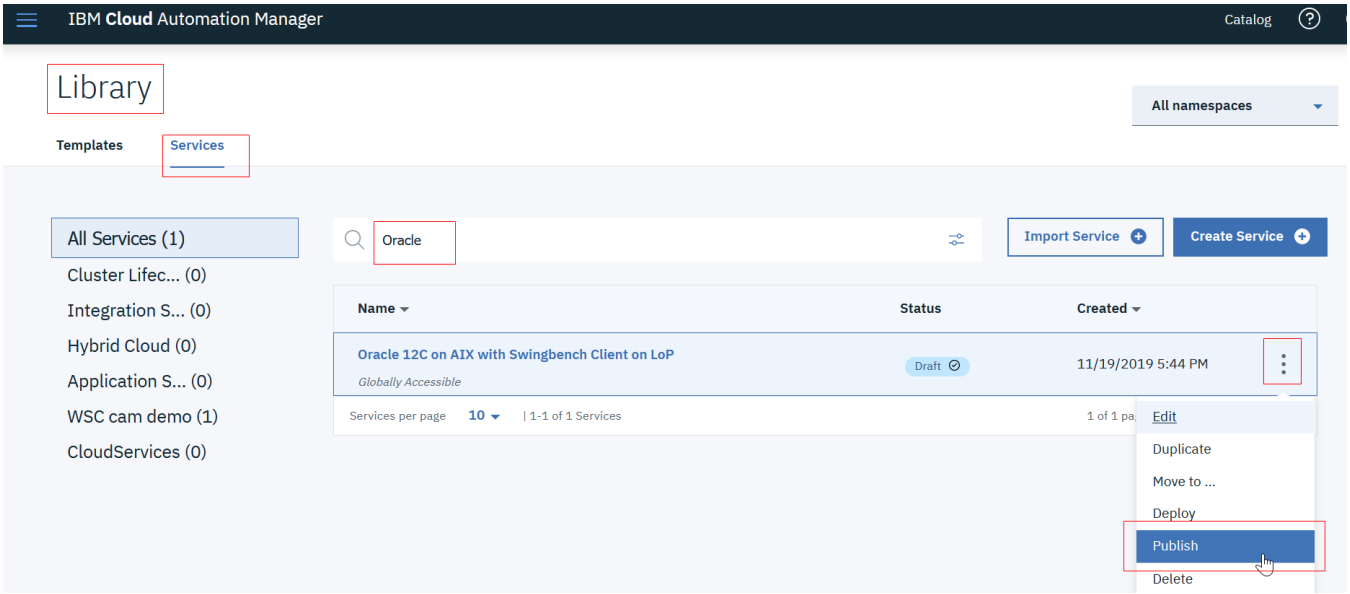
And as a proof that it works you see that all users successfully logged in and are driving transactions against the database. At that point we stop the workload by clicking the stop button.



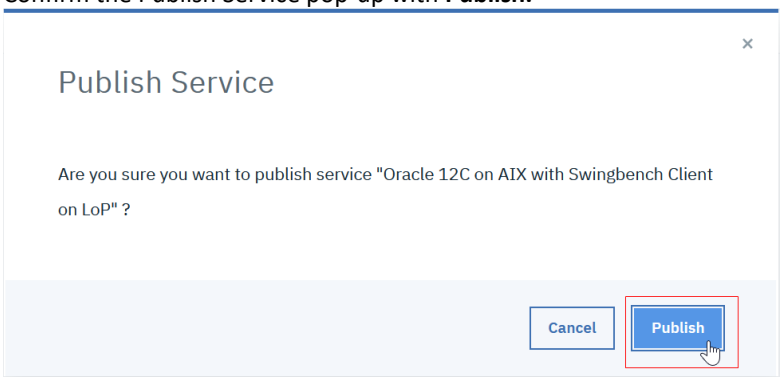
This completes the validation that our service, consisting of the dbserver and the dbclient, was successfully deployed and that parameter hand-over and orchestration worked as expected.

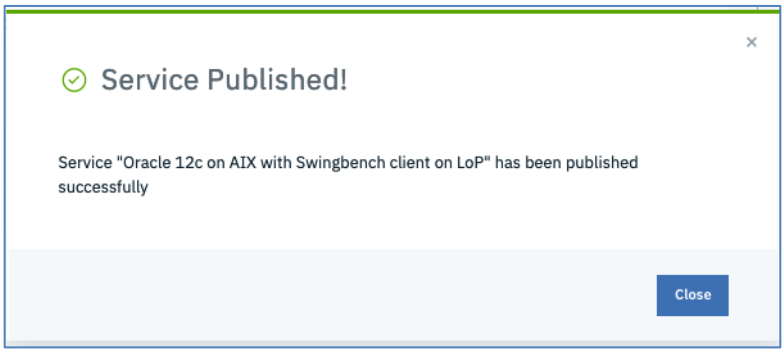
9.6 Publish the service to ICP

The service has so far been successfully deployed from the CAM service library. Now we want to make this service also easily consumable for end-users of ICP via the ICP software catalog.
To publish the service into the ICP catalog go in CAM to Library -> Services, enter Oracle as filter and then click on the three-dot menu to the right of the service and select Publish as shown in the screen shot below.

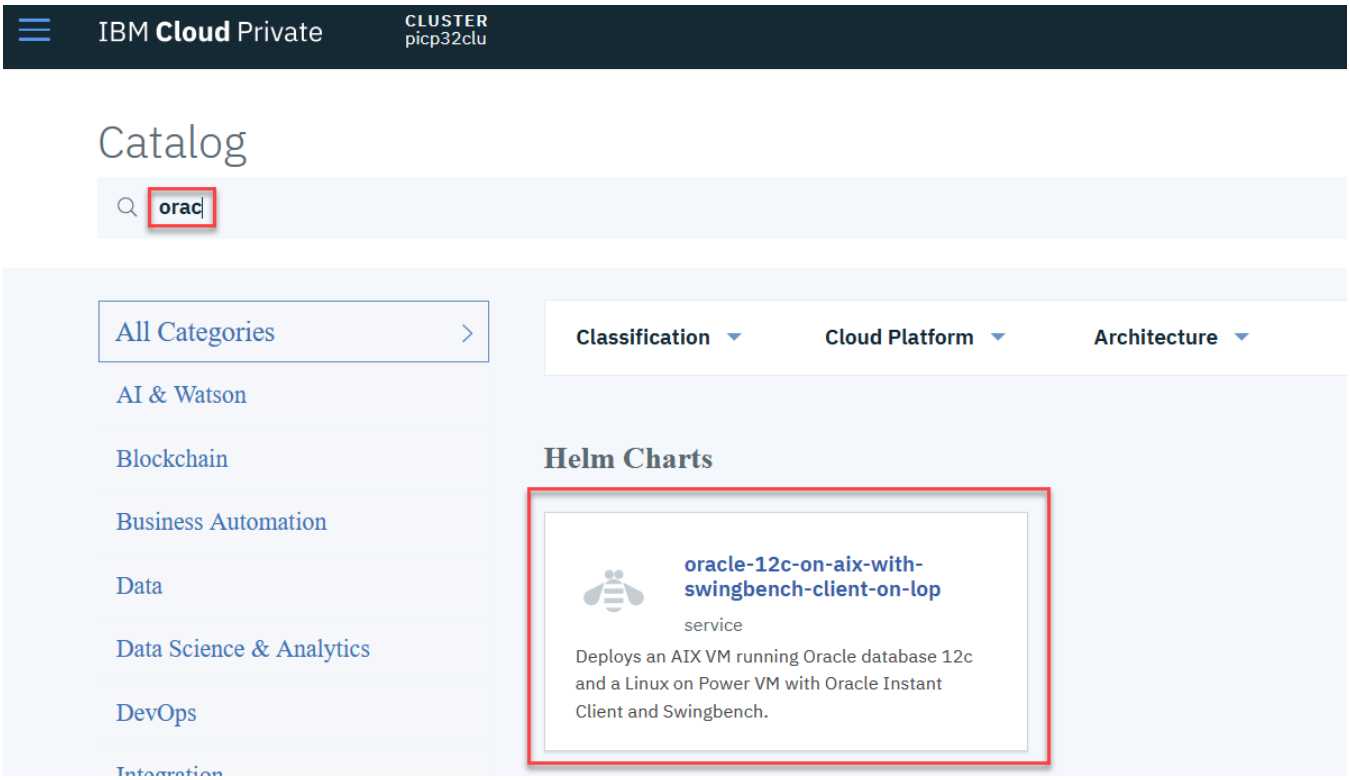


Confirm the Publish Service pop-up with **Publish.**

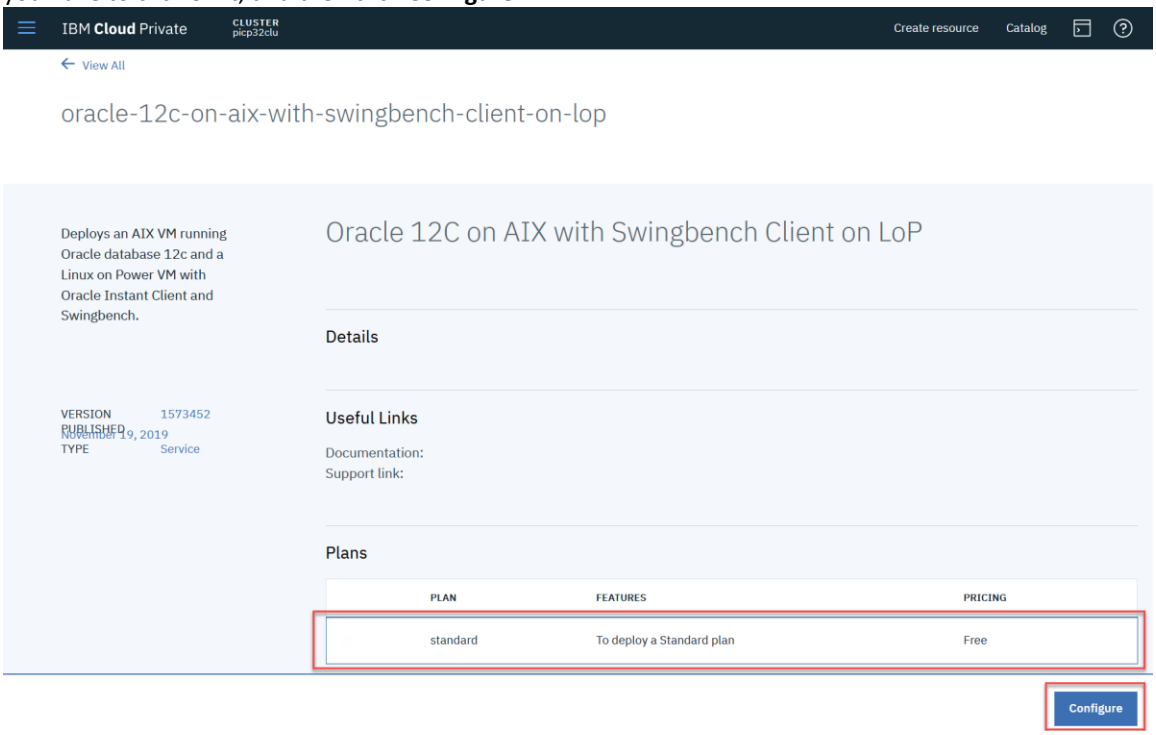




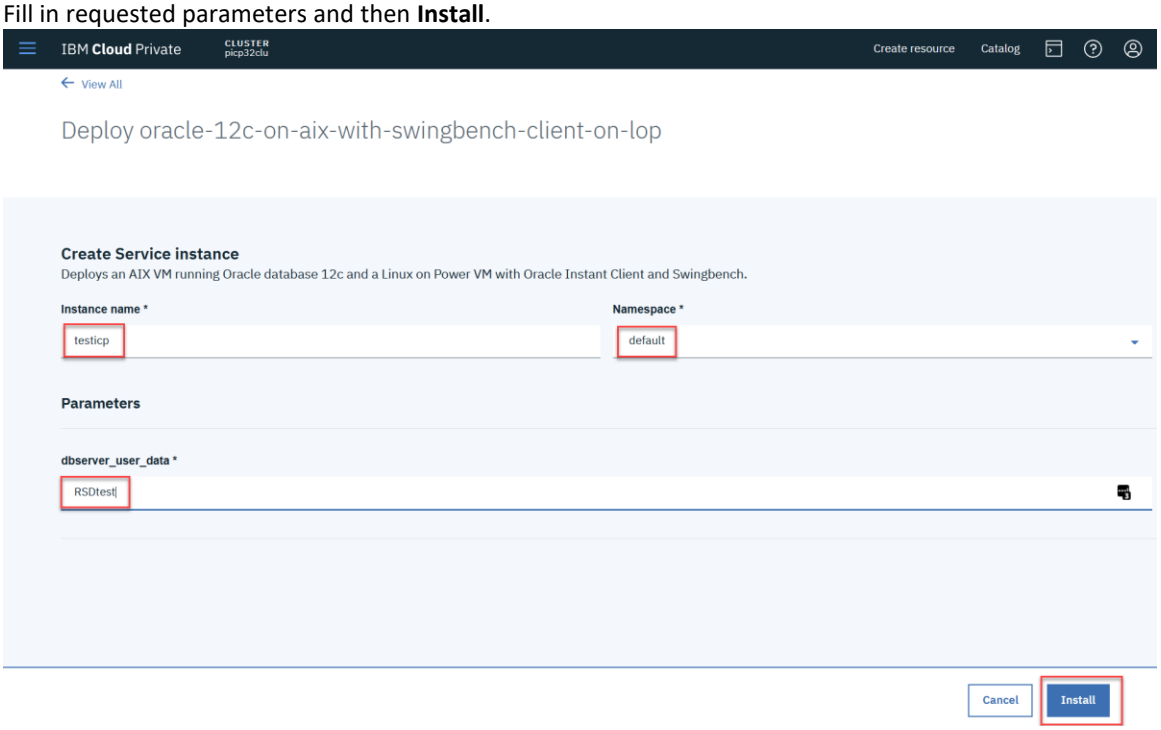
Wait a few minutes and then access the ICP catalog. Enter a search key, e.g. **orac** to more easily find the newly added service – note that the name of the helm chart for our service is all lower-case characters.

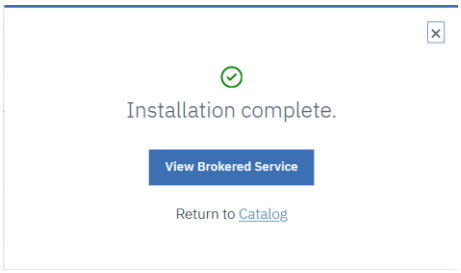


To deploy the service from ICP catalog, click **oracle-12c-on-aix-with-swingbench-client-on-lop**. Select the standard plan, you have to click on it, and then click **Configure**.

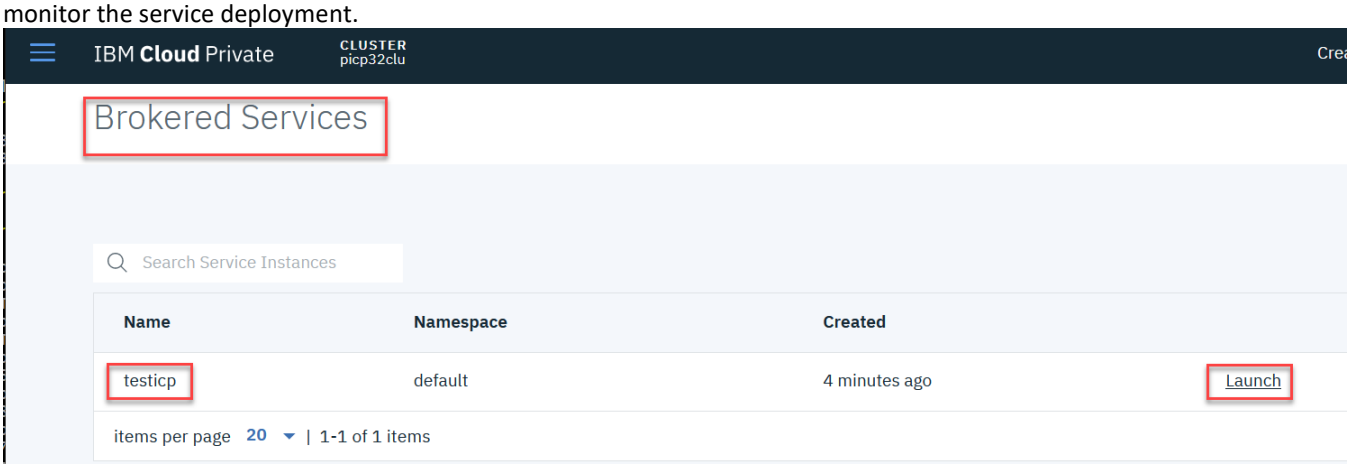


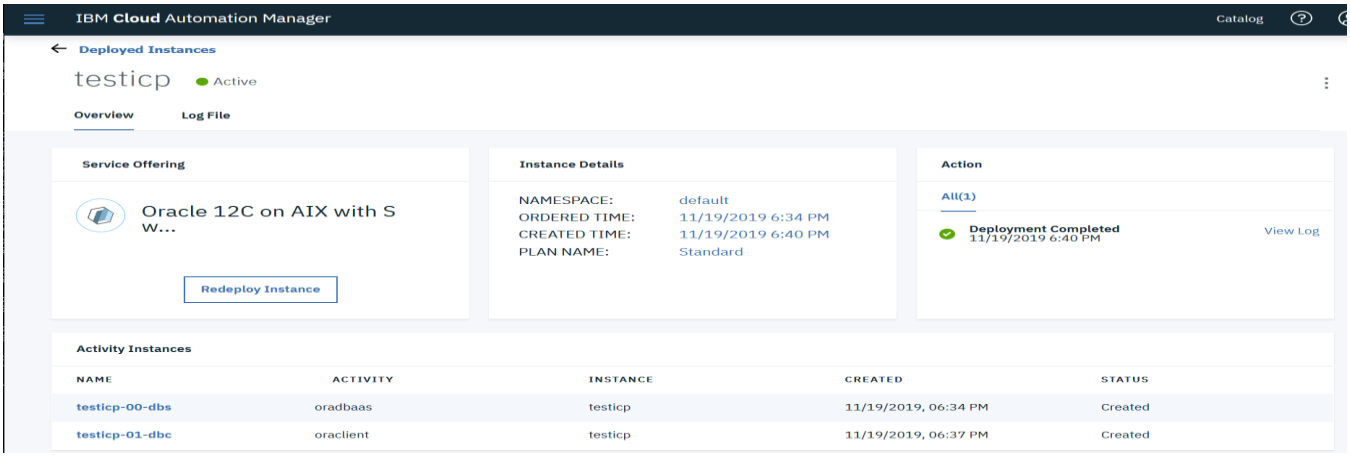
Fill in requested parameters and then **Install**.





Note that the deployment shows up in ICP under “Brokered Services” and not “Helm Releases” under the Workloads menu. To observe the service deployment, click on **Launch** and that switches to a new window with CAM where you then can monitor the service deployment.



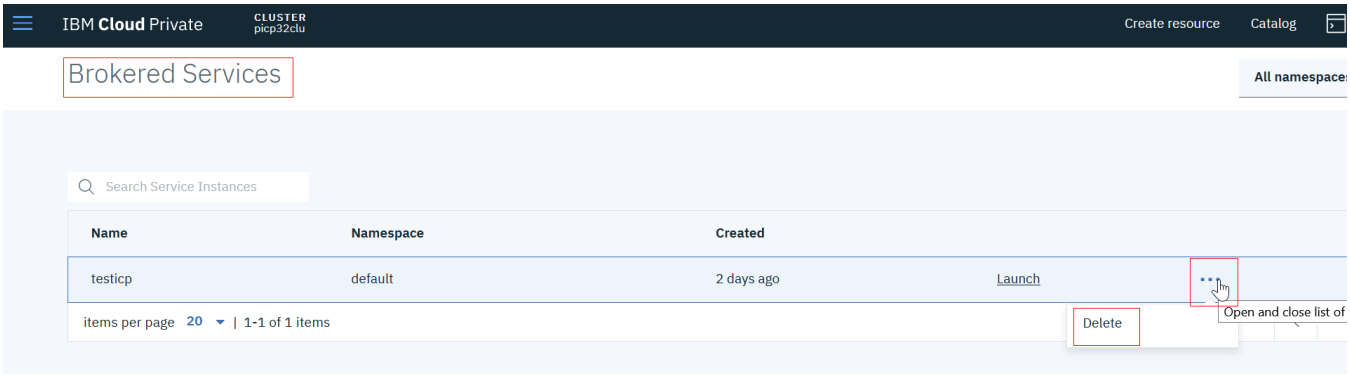


9.7 Cleaning up

This section describes the required steps to delete deployed templates and services. In our testing we found that a specific sequence of steps is required to be able to correctly clean up the environment and not be left with some objects which cannot be deleted.

9.7.1 Deleting a Deployed Service Instance

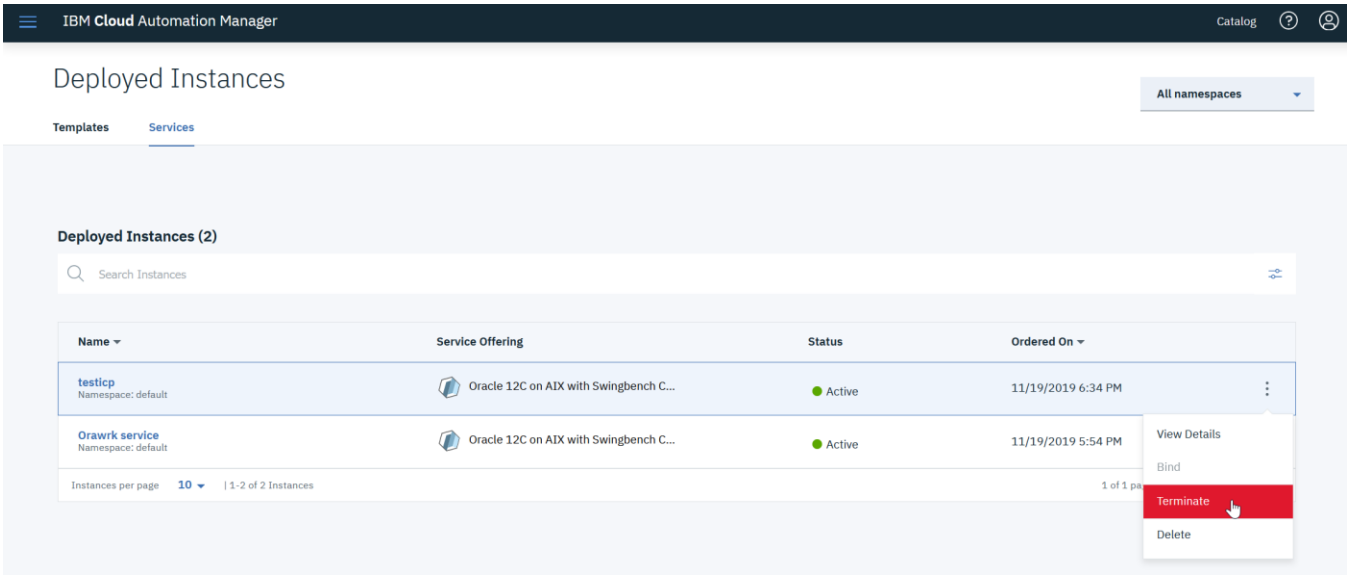
If you deployed the service from the ICP catalog you must start the deletion of the service from ICP as well, and not from CAM. For example, the testicp deploy from ICP needs to be deleted like this – in ICP: Workloads -> Brokered Services. Select the service testicp, then the three-dot menu and select Delete from the drop-down. Accept the pop-up by clicking **Delete ServiceInstance**.



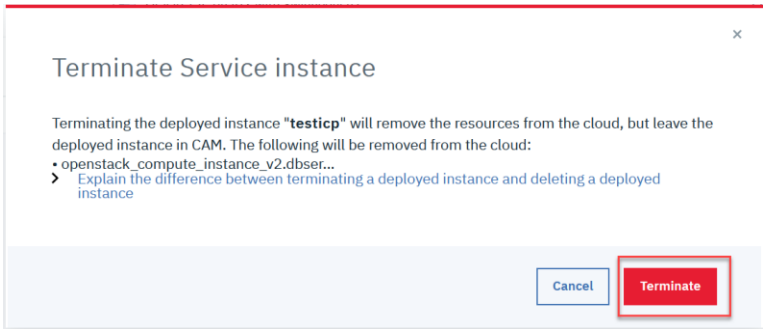
In our environment this correctly removed the service entry from the Brokered Services in ICP and Terminated the service instance in CAM, so all VMs were deleted, but the service was not deleted in CAM itself. That step had to be done manually – see below for details.

Deleting a deployed service instance in CAM requires two steps. The first step is to **Terminate** the service instance, which deletes the VMs in PowerVC. After the instance is terminated, the next step is to **Delete** the service instance which removes the service from CAM. If you do the **Delete** first, the service is removed from CAM, but the deployed VMs will persist and you'd need to delete those VMs from PowerVC instead as CAM has no longer any knowledge of those deployed VMs.

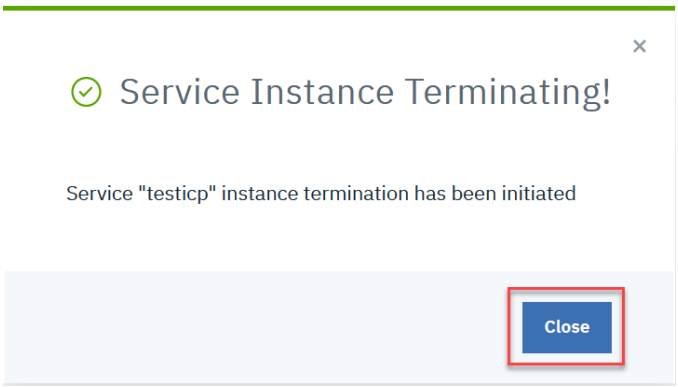
So, to correctly delete a deployed service within CAM go to the Deployed Instances screen for the deployed service you want to delete, click the three-dot menu on the Actions column and then select **Terminate**.



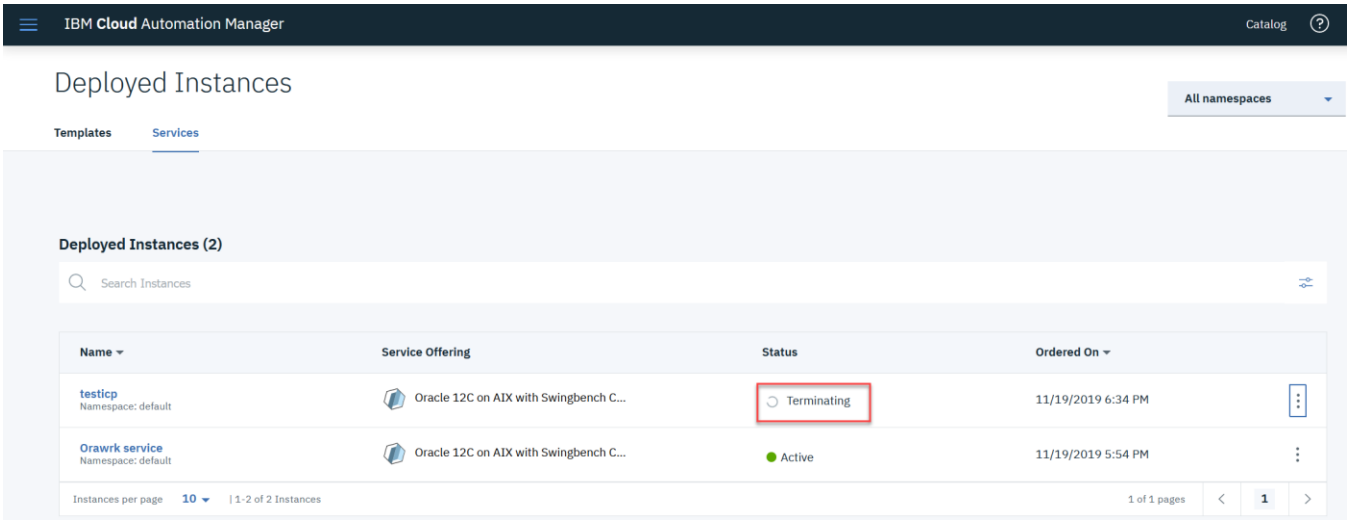
A pop-up window appears, requesting you to confirm. Click the **Terminate** button.



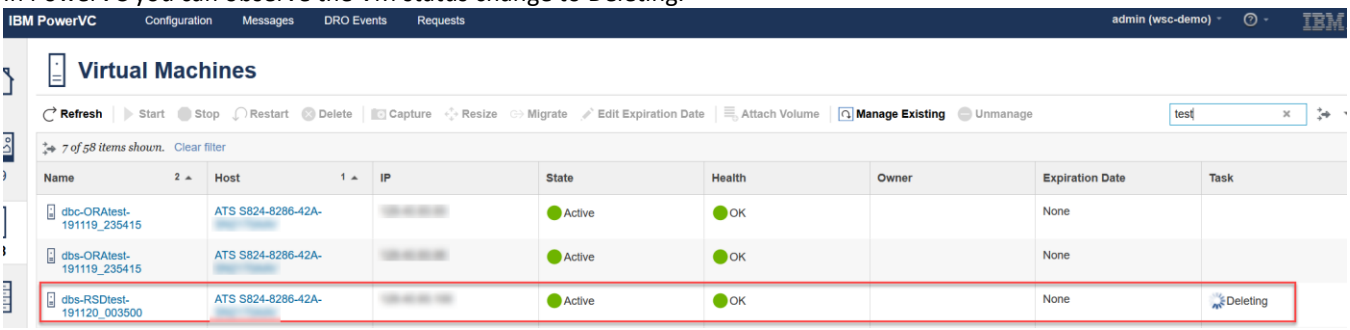
Another pop-up window appears to indicate start of termination.



The Deployed Instances window shows Terminating Status. Depending on the number and complexity of deployed VMs in that service this Terminate action can take several minutes.

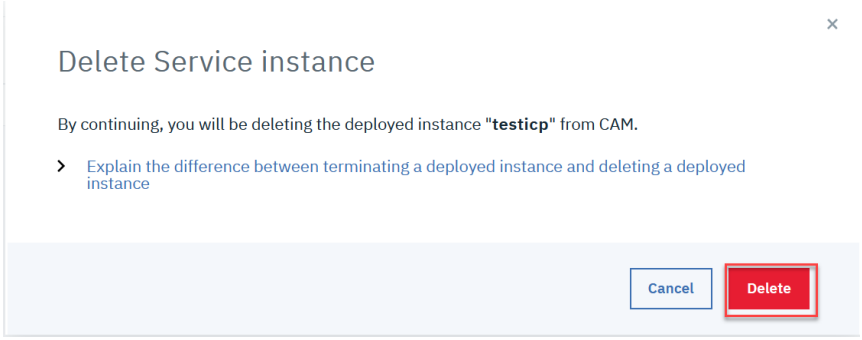


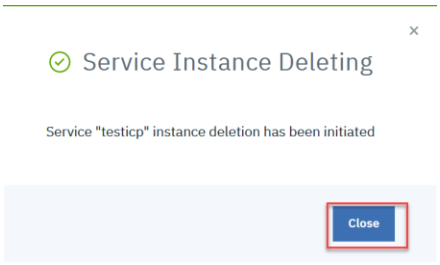
In PowerVC you can observe the VM status change to Deleting.



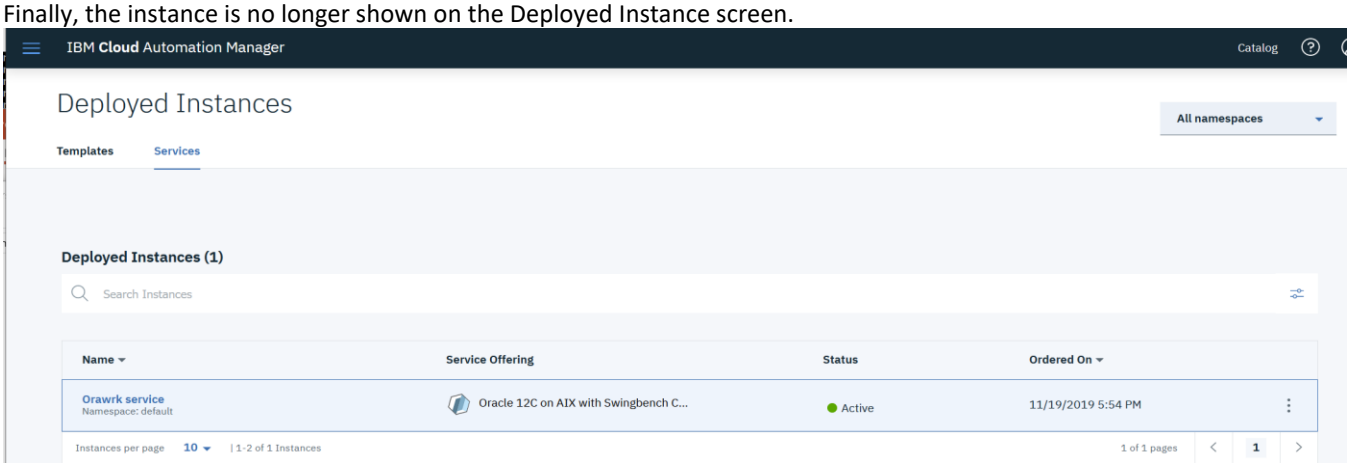
When termination is complete, the service instance can be deleted by selecting the **Delete** option for the service. This **Delete** removes all knowledge about the existence of that service from CAM, including deployment history and logs.

A similar pop-up windows appear to confirm the Delete action. This is typically very quick.

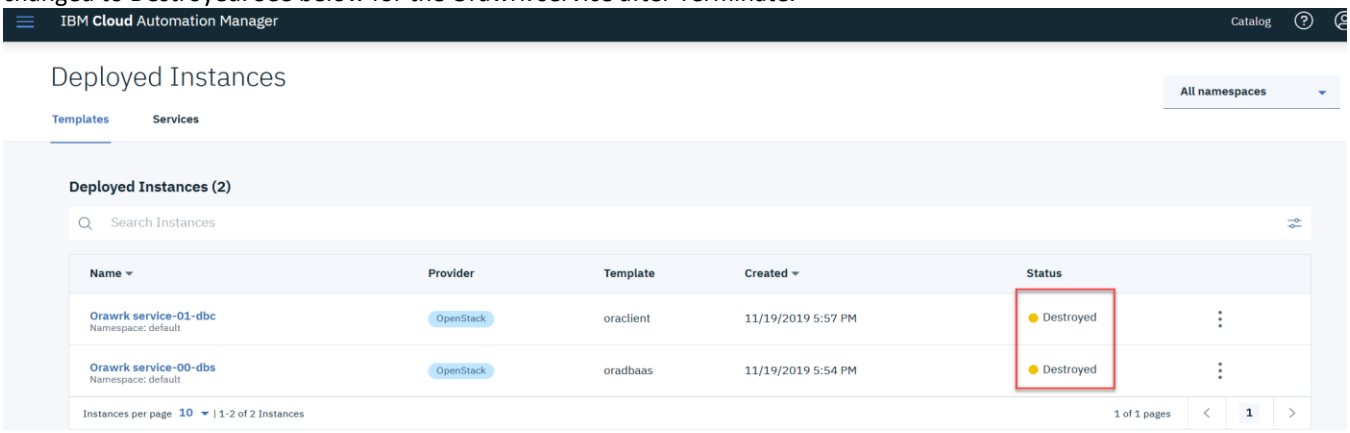




Finally, the instance is no longer shown on the Deployed Instance screen.



Note, after the **Terminate** of the service succeeded the status of the underlying template deployments for a service is changed to Destroyed. See below for the Orawrk service after Terminate.

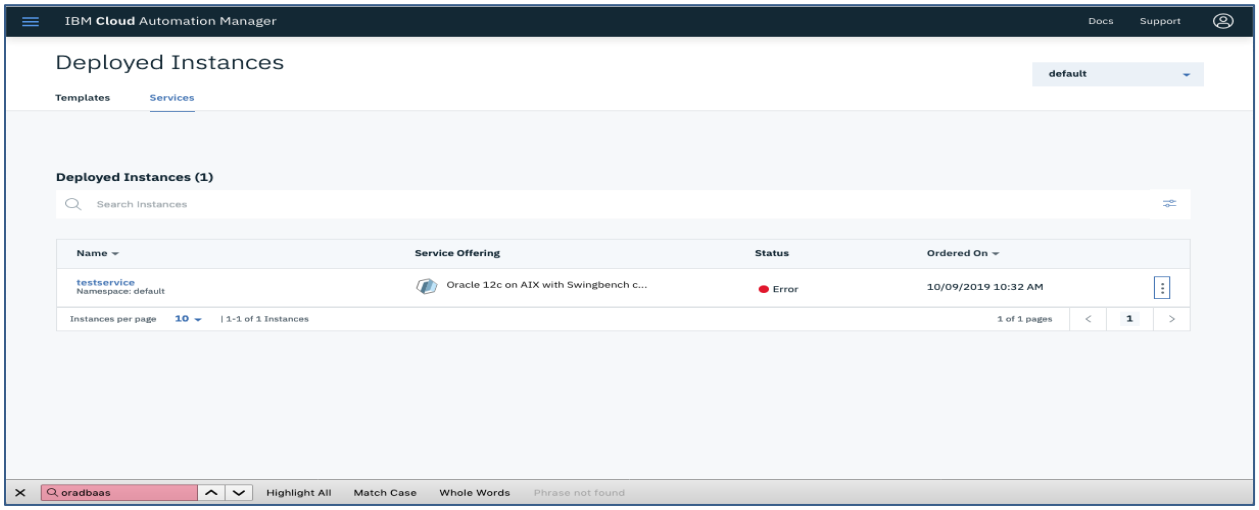


9.7.1.1 Operational Note

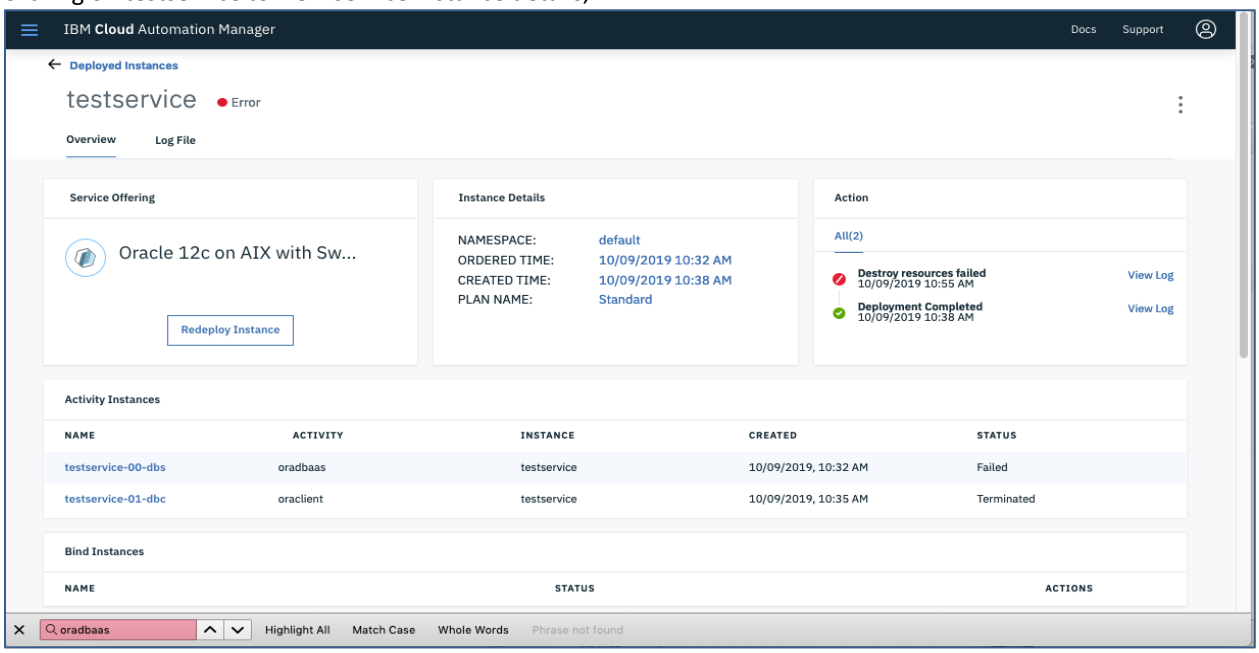
While it is beyond the scope of this paper to discuss considerations, scenarios and steps to modify templates and their use as part of a service deployment, we encountered a situation that warrants mention.

When a service has been deployed and an underlying template is deleted and replaced, you will encounter an error when you try to terminate the deployed service.

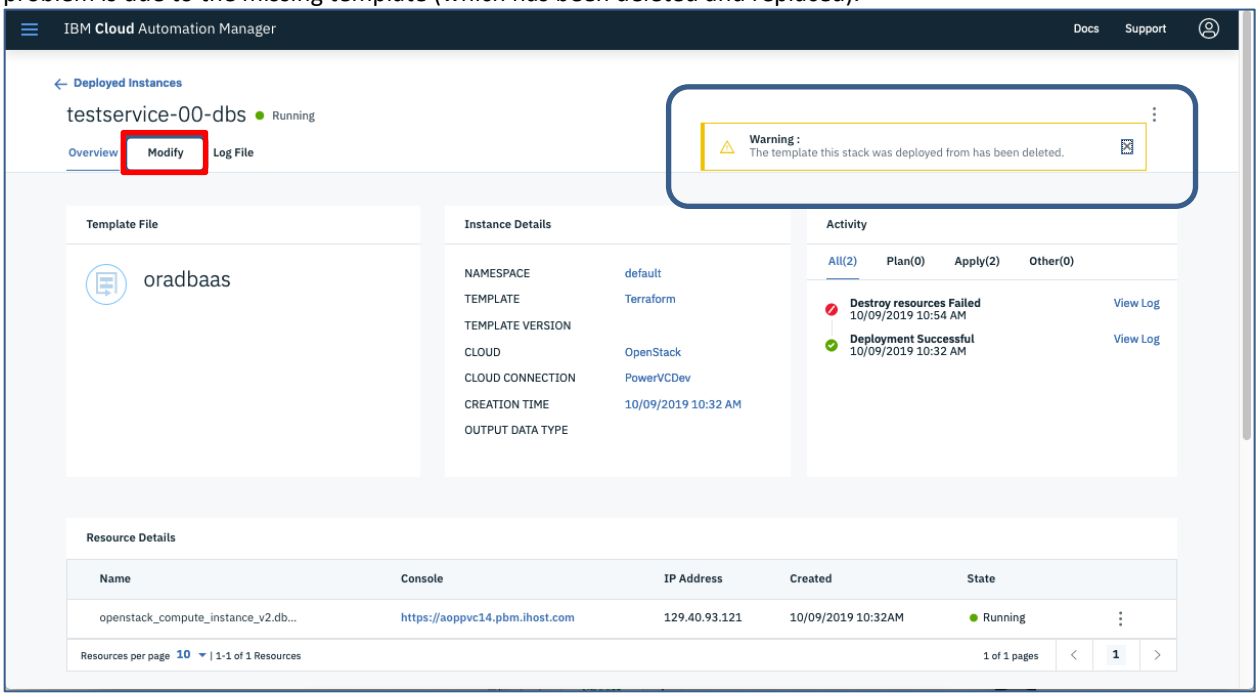
In the scenario described here, we had test deployed our service, then deleted and replaced the *oradbaas* template for a minor change. When trying to terminate the deployed testservice service instance, we got the error shown below.



Clicking on testservice to view service instance details,

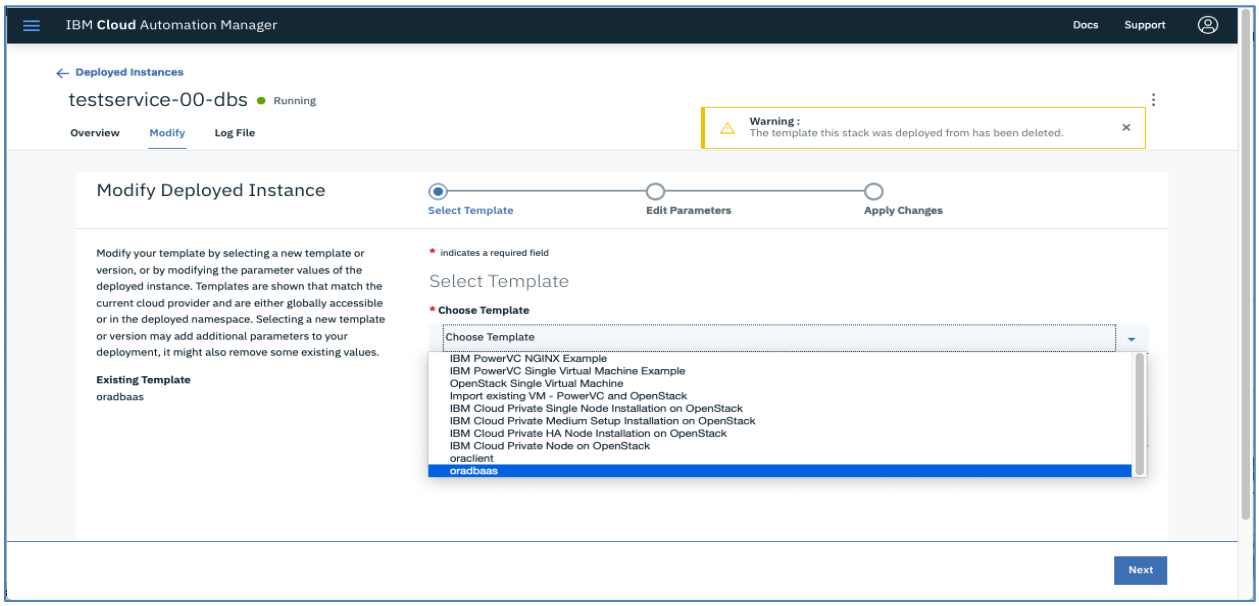


Click on testservice-00-dbs, a new browser tab shows the deployed (underlying) template details. Here we see that the problem is due to the missing template (which has been deleted and replaced).

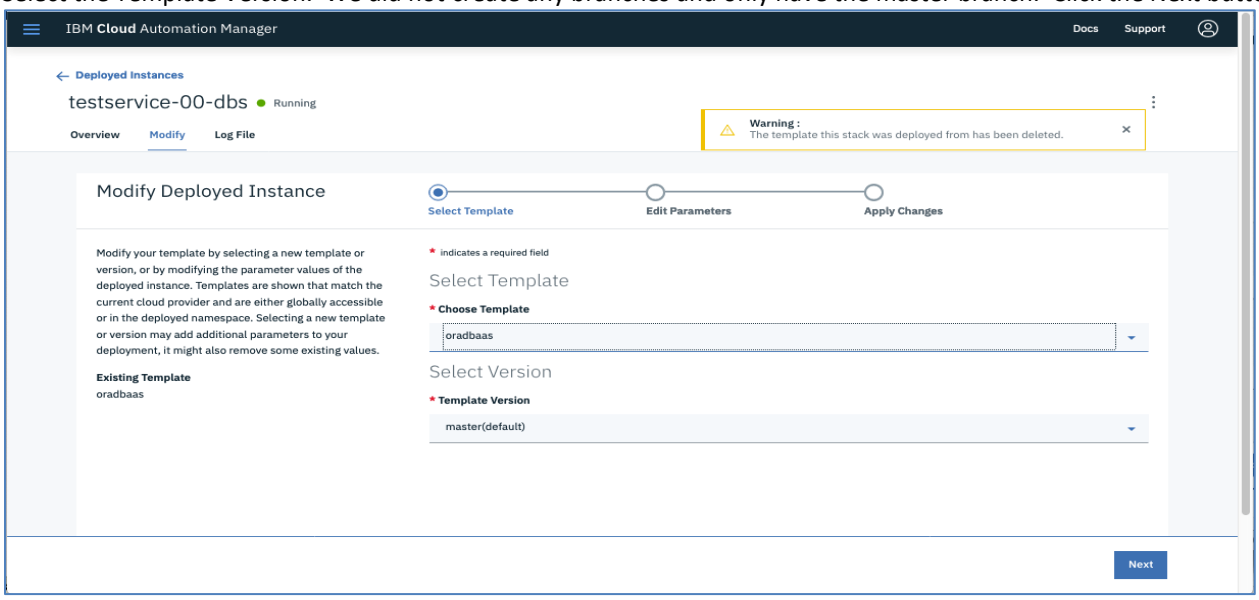


To resolve this situation, click Modify in the above screen.

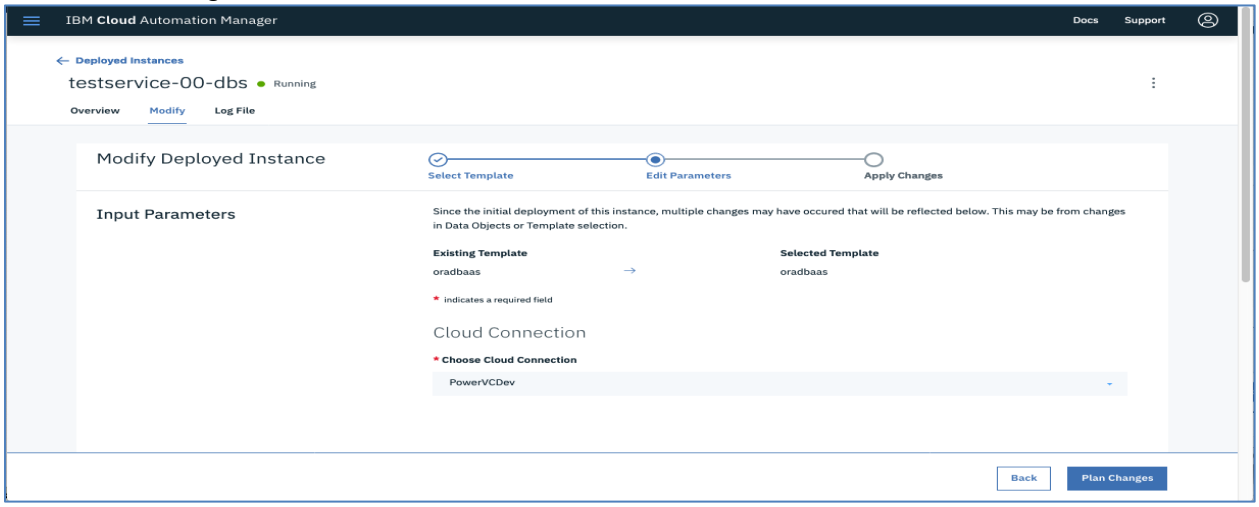
Click Choose Template and select *oradbaas* (the template we replaced),



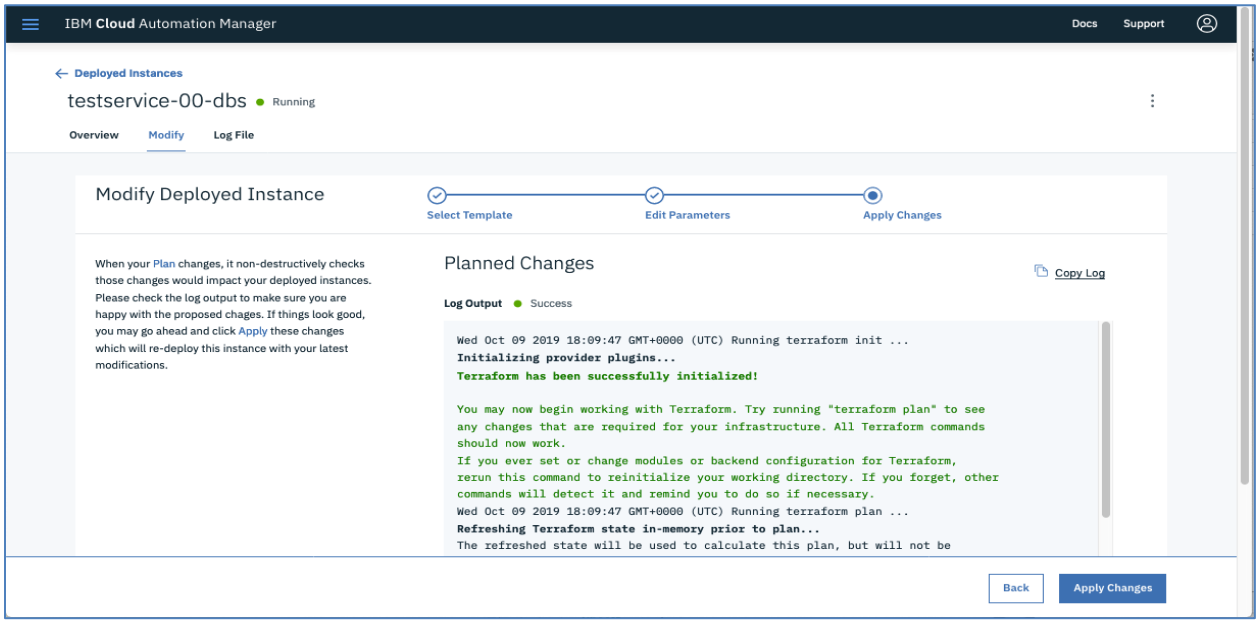
Select the Template Version. We did not create any branches and only have the master branch. Click the Next button.



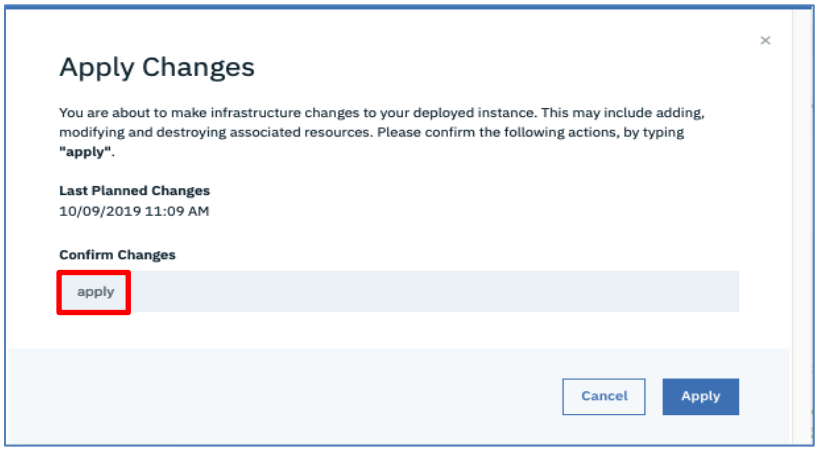
Click the Plan Changes button,



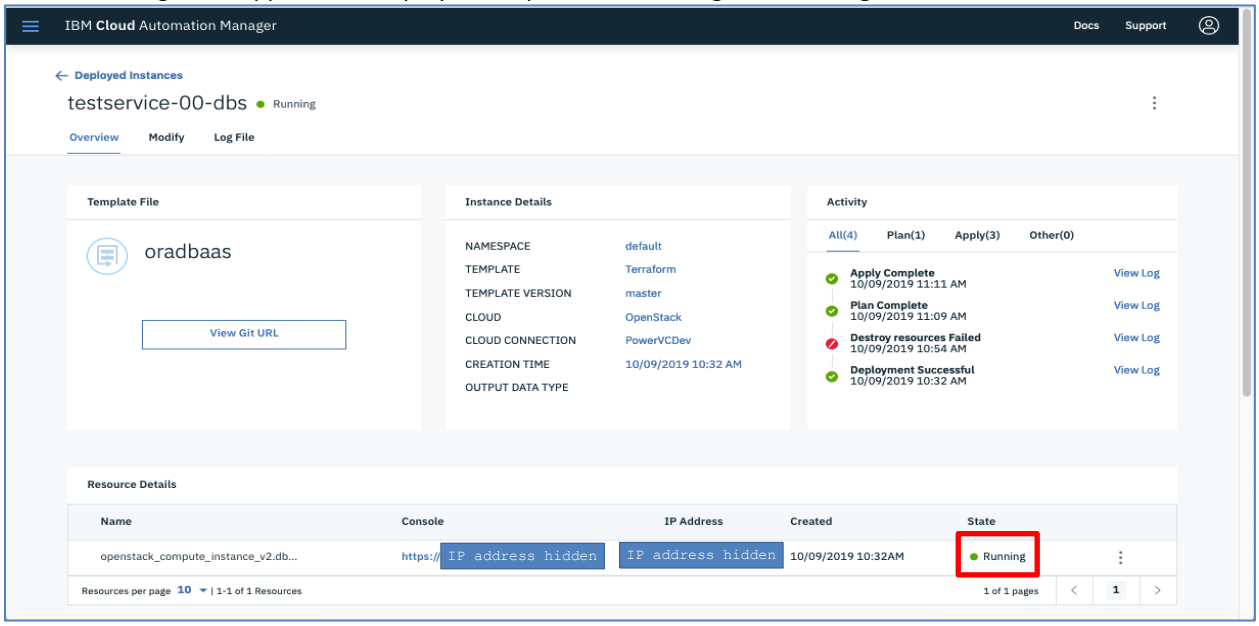
Review the Planned Changes to see what's being changed. Click Apply Changes to continue.



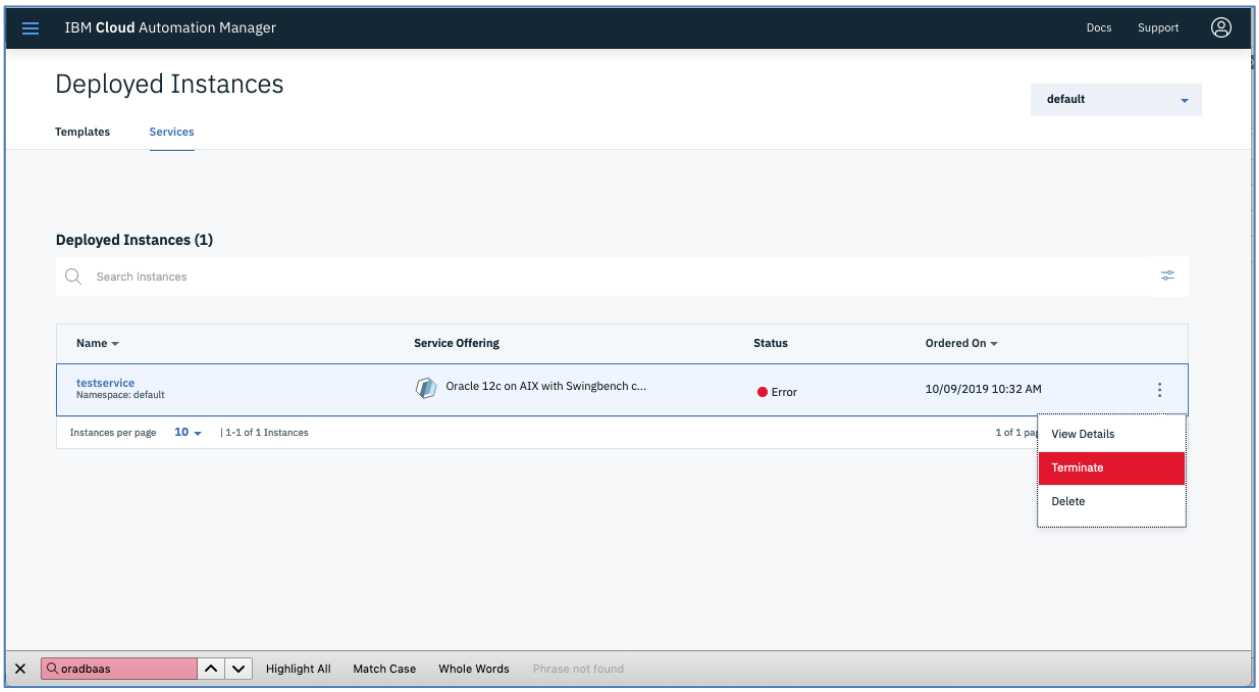
Confirm by typing in apply and click the Apply button.



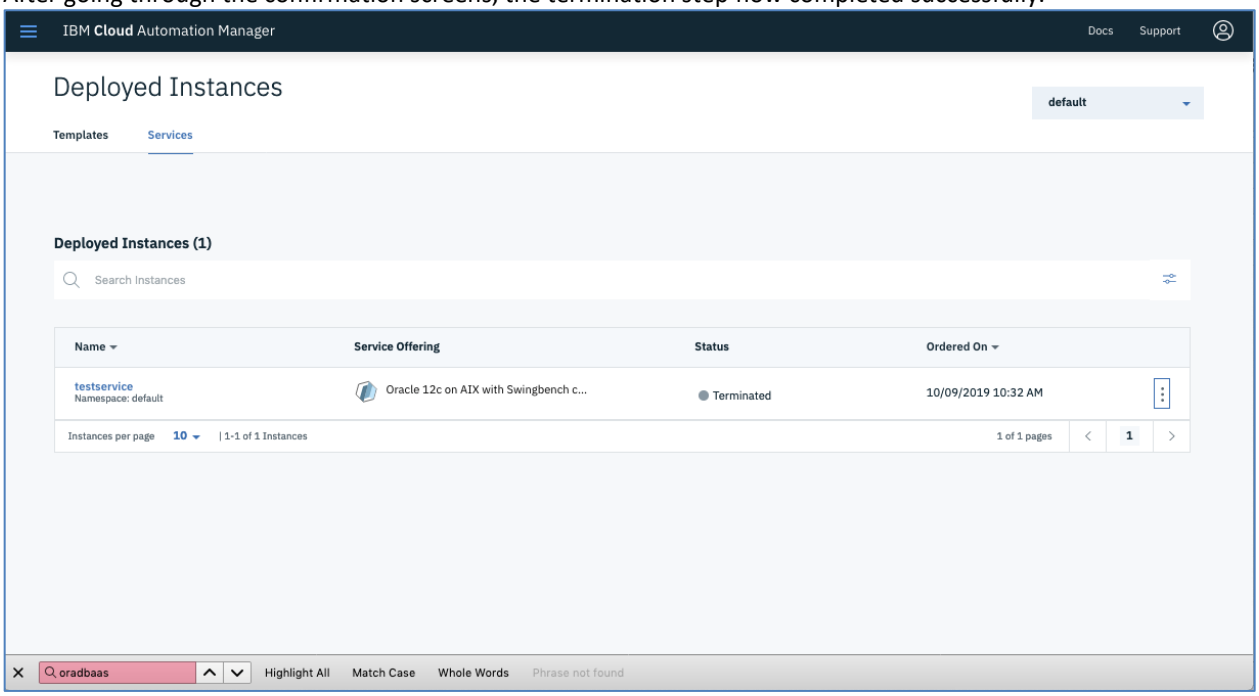
After the changes are applied, the deployed template status changes to Running.



With that change in place we now try to Terminate the service again. Return to the deployed service tab and select Terminate.



After going through the confirmation screens, the termination step now completed successfully.



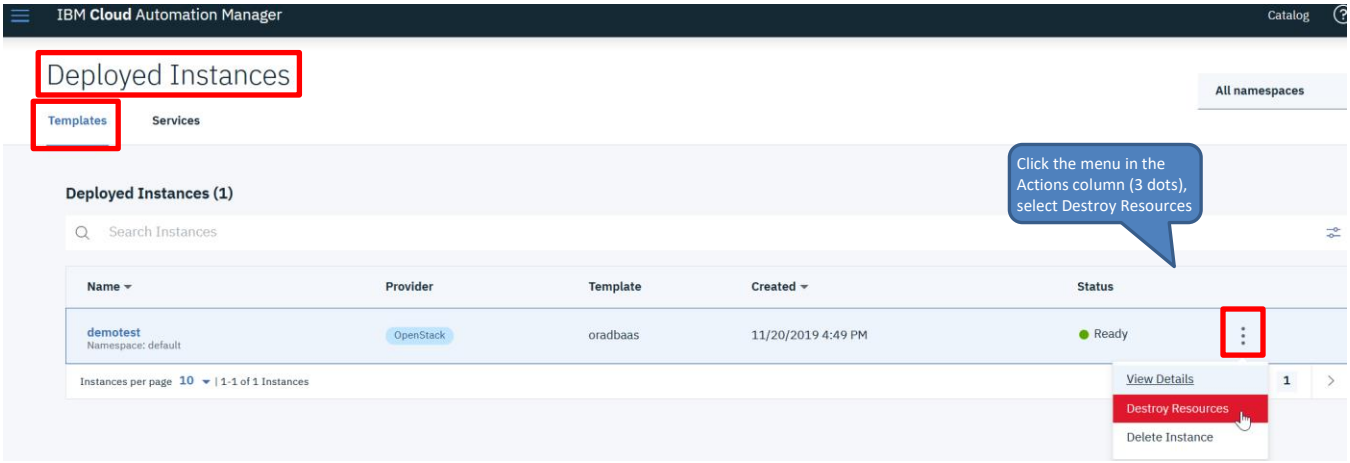
The instance can now be deleted as described earlier.

9.7.2 Deleting a Deployed Template Instance

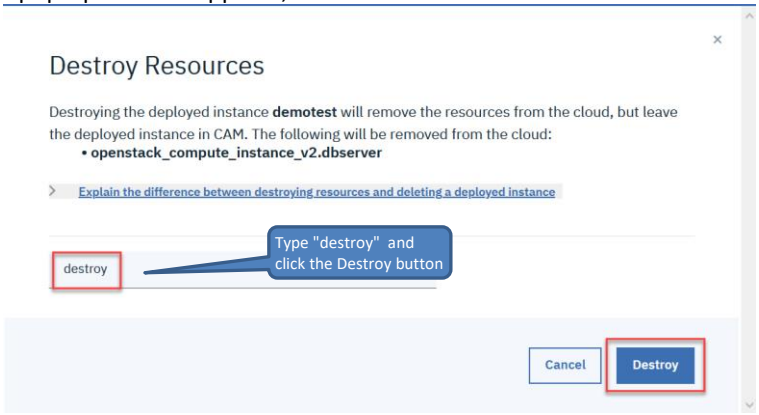
Deleting the deployed template instance requires two steps as well: Destroying the Resources and Deleting the Instance.

9.7.2.1 Destroy Resources

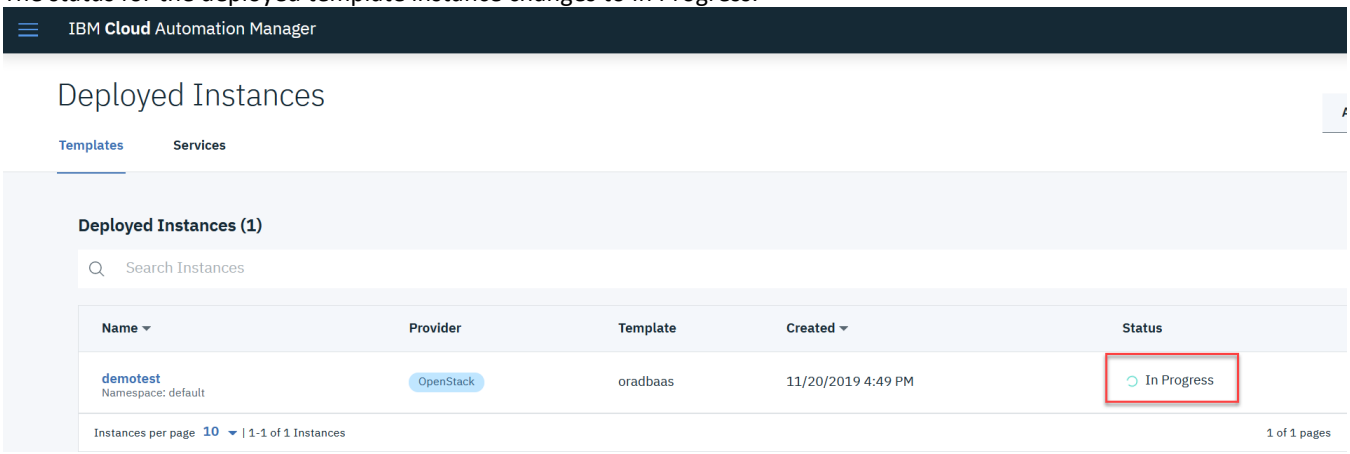
Go back to the CAM screen, click **Deployed Instances**, or use the upper left hamburger menu, select Deployed Instances -> Templates. The **Destroy Resources** is similar to the **Terminate** for a Service in that it deletes the underlying VM(s) for the template in PowerVC but keeps the information about that template deployment intact in CAM.



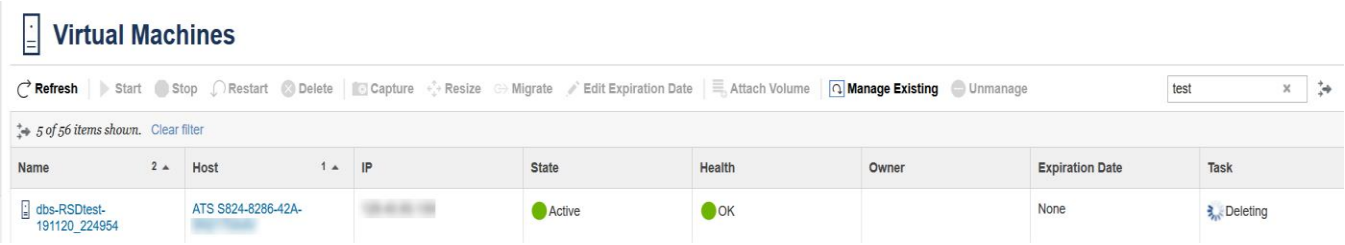
A pop-up window appears,



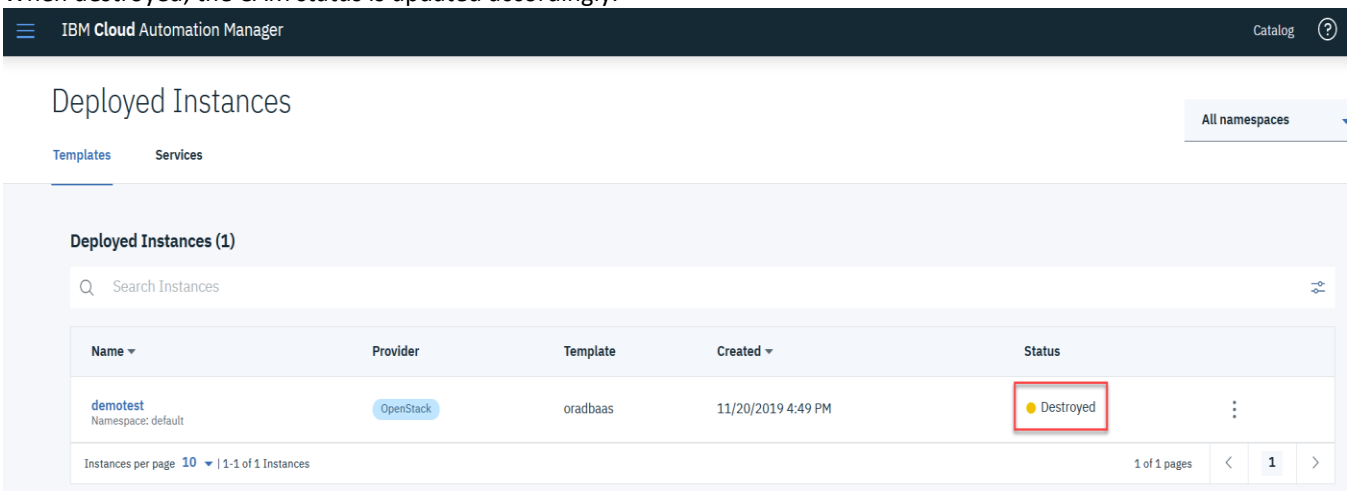
The status for the deployed template instance changes to In Progress.



Looking at PowerVC, we see the VM being deleted.



When destroyed, the CAM status is updated accordingly.

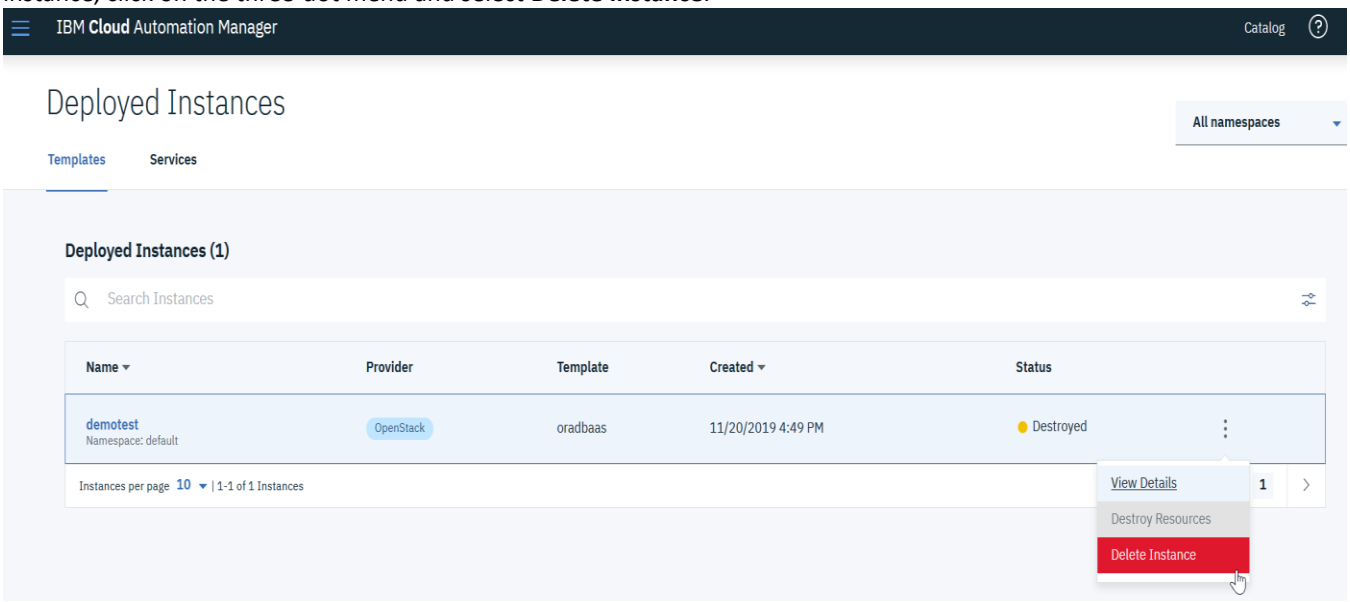


The deployed instance may now be deleted.

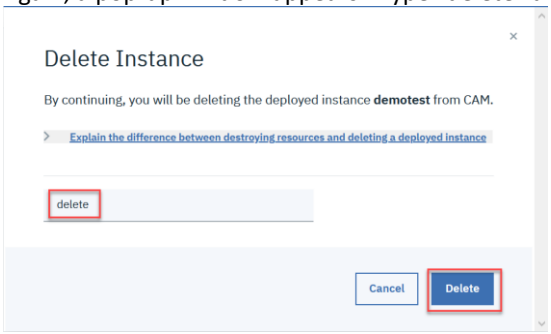
Tip: If you have a template with many parameters and you just want to re-deploy the template with a minor parameter change, you can Destroy the deployed template, then go to the “deployed template”, click on the details and edit the deployment, re-plan it – means changing any parameters needed – and then activate the plan. This will then trigger a new deployment with the same deployment name and the changed parameters, without having to go through the full template deployment process.

9.7.2.2 Delete Deployed Instance

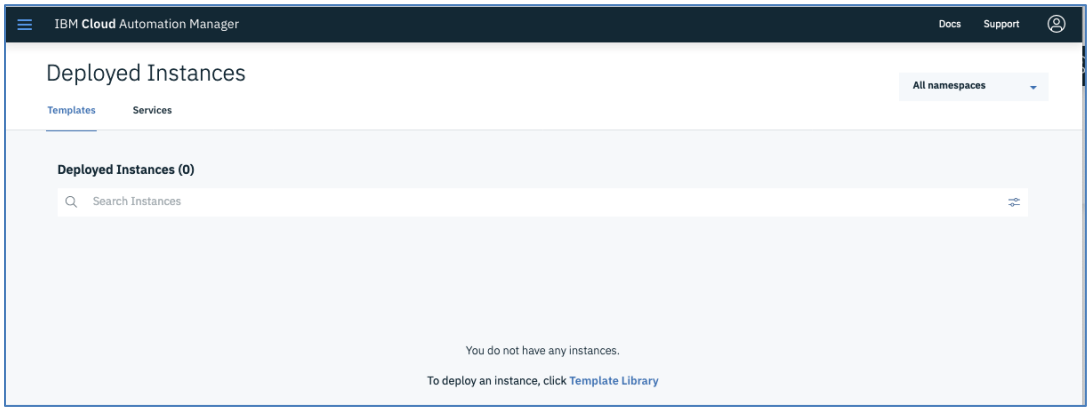
Once the resources have been destroyed, the deployed instance can be deleted. Select the demotest deployed template instance, click on the three-dot menu and select **Delete Instance**.



Again, a pop-up window appears. Type "delete" and click the **Delete** button.



The instance has been deleted.



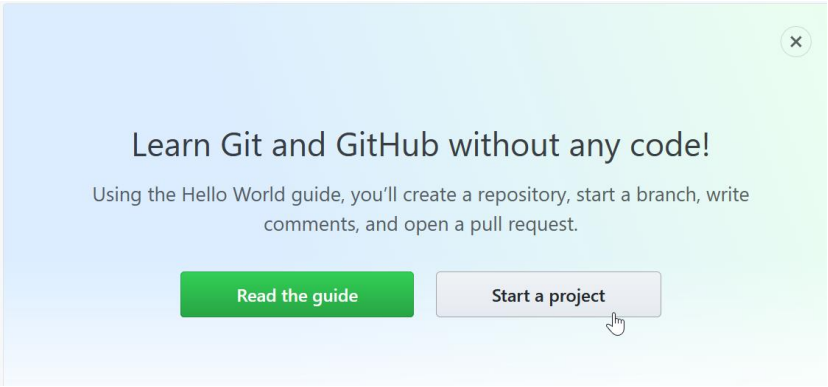
10 Appendix

10.1 Working with GitHub

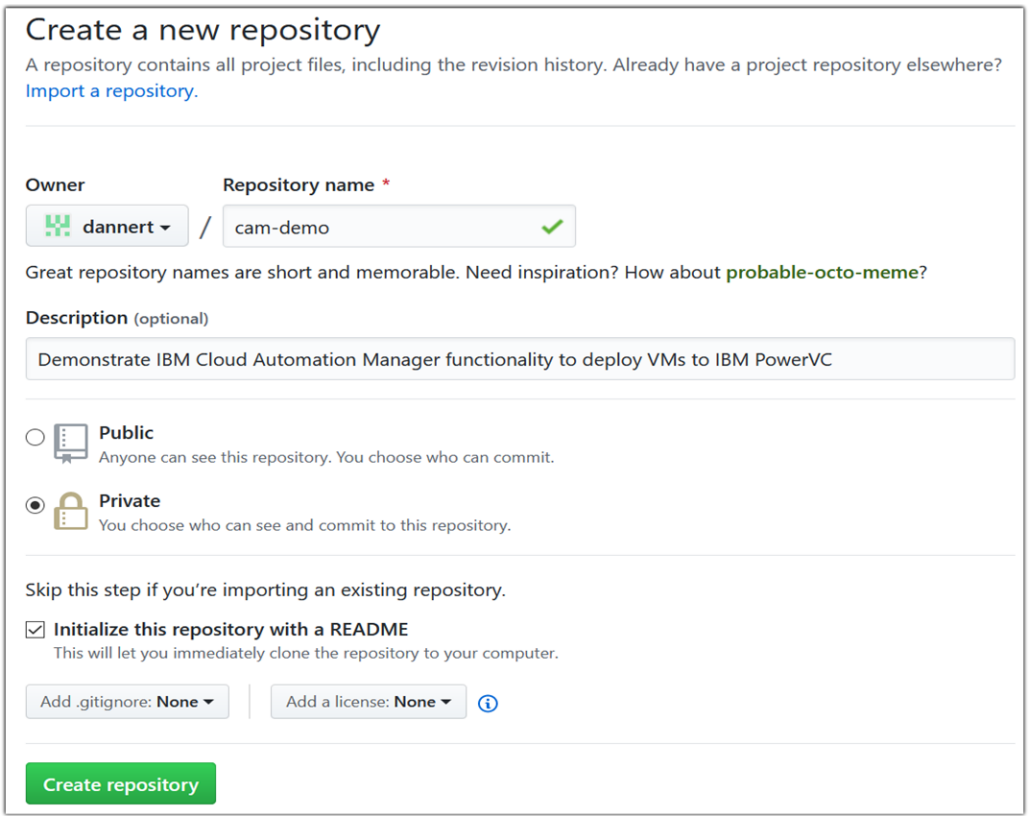
10.1.1 Create GitHub repository

Access https://github.com. Sign up for an account on github.com or use your existing account and log in to GitHub.

Click the **New** button next to Repositories or click on **Start a project** to create a new repository.



Type in the name of your repository. In this example, we use the name **cam-demo**. Select **Private** and **Initialize this repository with a README** and then click **Create Repository**.

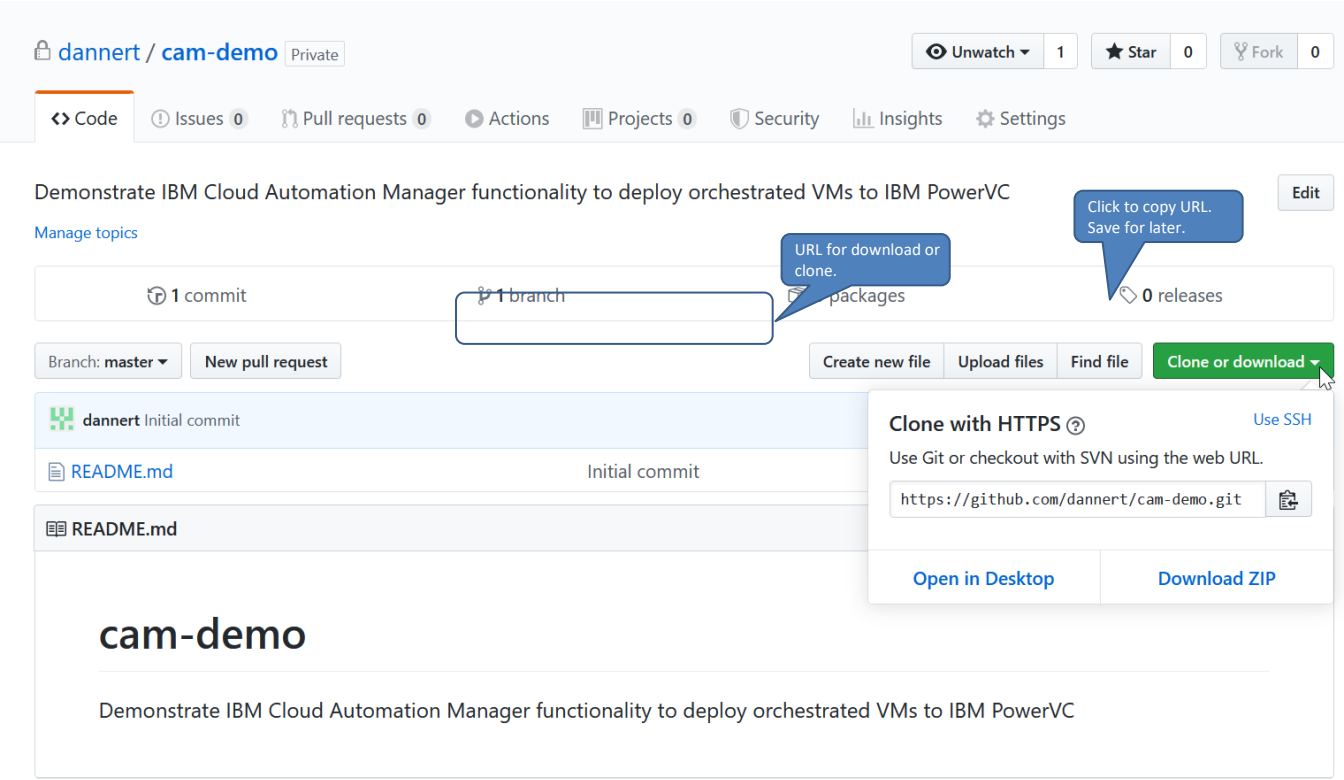


Information on Private repository may be found on:
https://help.github.com/en/articles/githubs-products

Review
https://github.com/pricing

for features comparison of the various GitHub offerings.

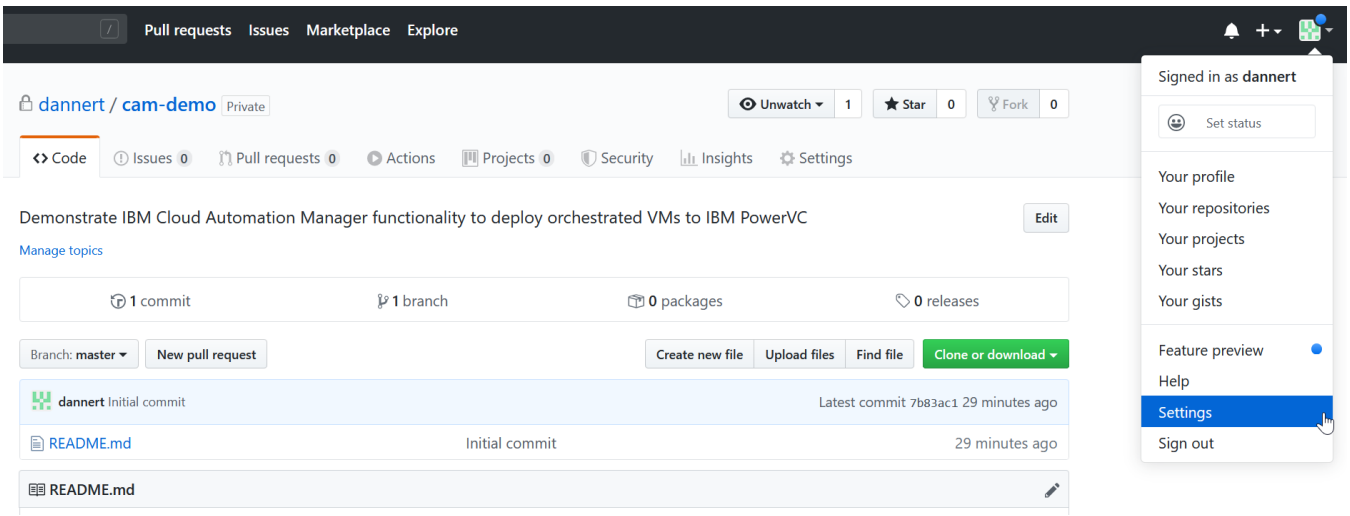
Opening the new repoistory then looks like this:



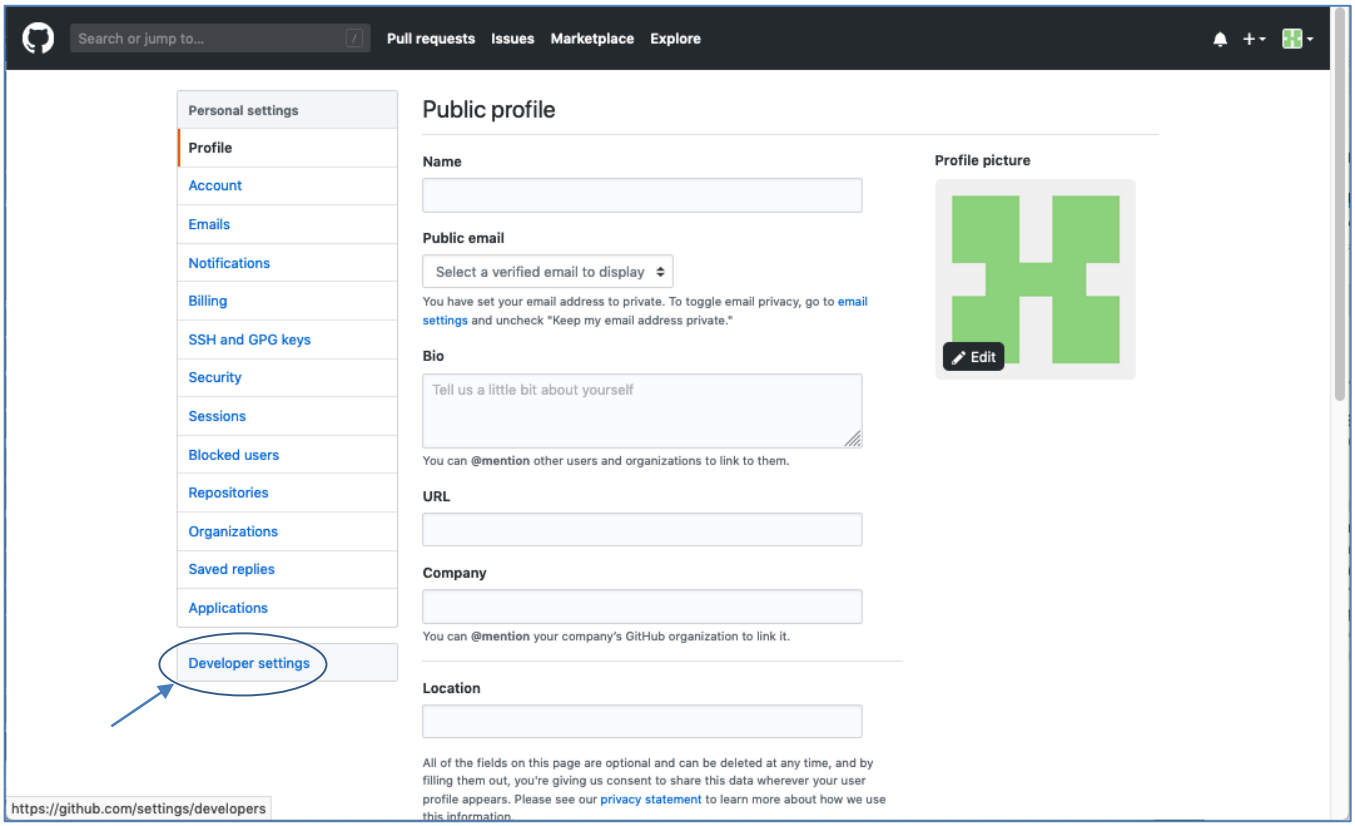
Copy the HTTPS URL shown as you will need that later in CAM to register this GitHub repository for use in the project.

10.1.2 Create personal access token

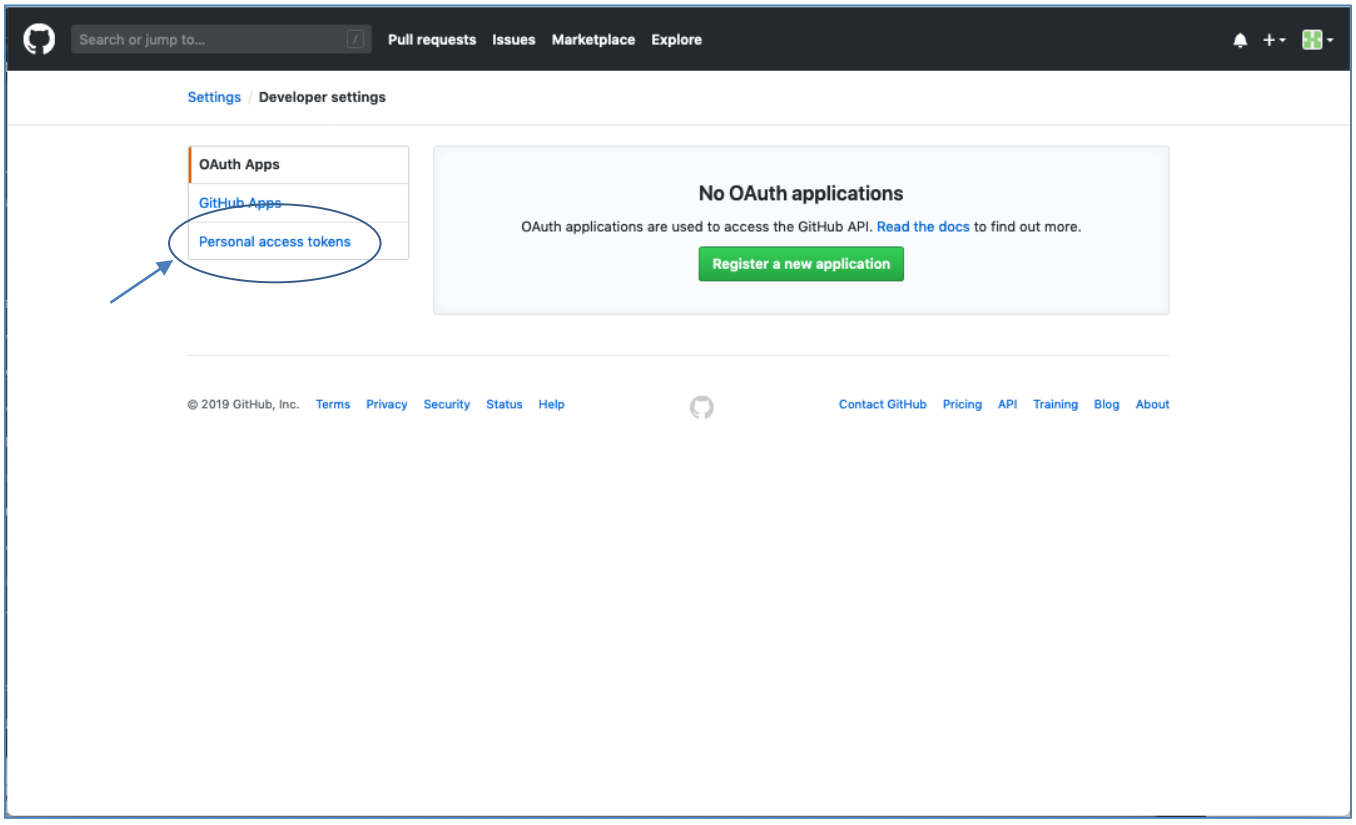
Click the icon (a profile picture that can be changed on the **Settings** screen) on the upper right-hand corner to reveal and access the drop-down submenu. See picture below. Select **Settings**.



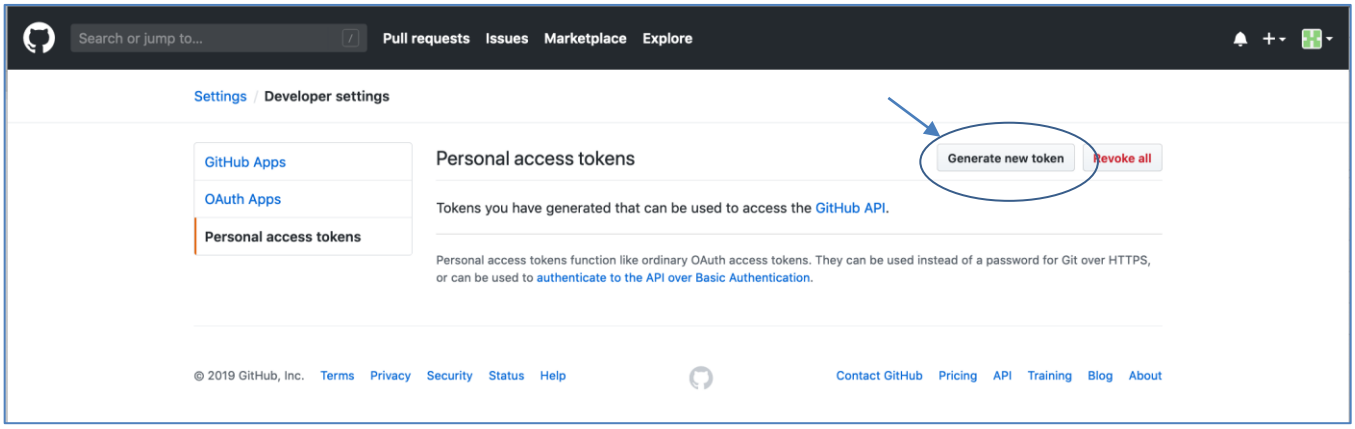
Select **Developer settings** on the left.



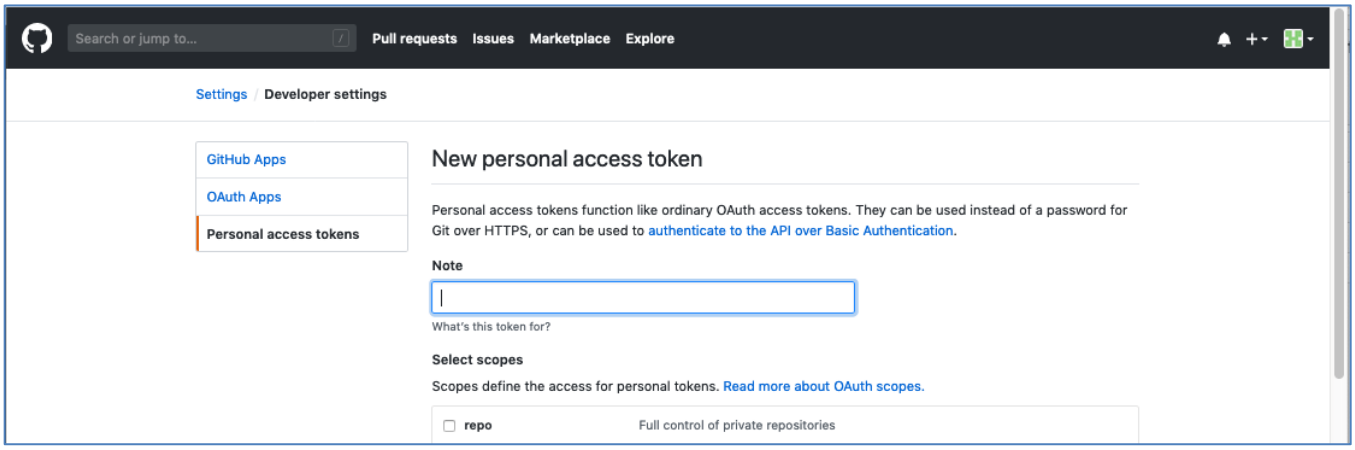
Select **Personal access tokens**.



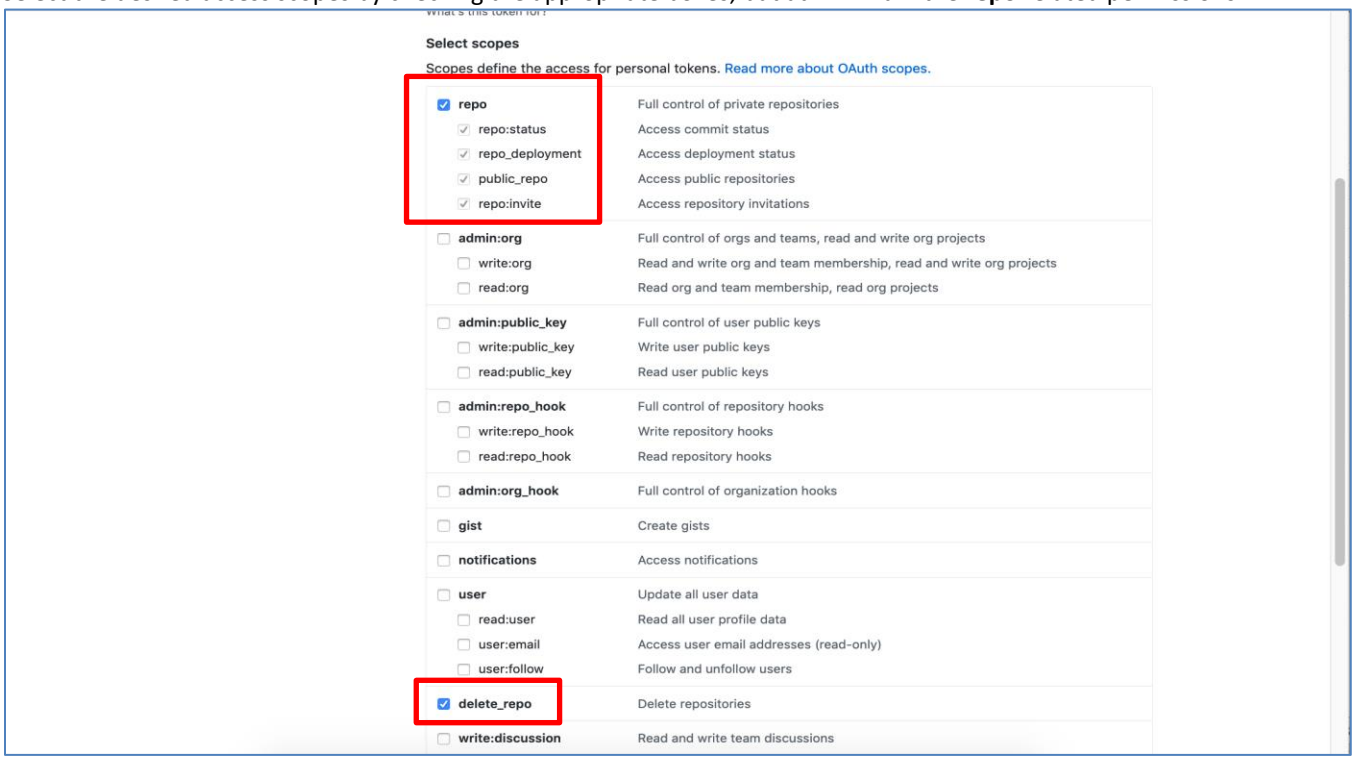
Now click **Generate new token**.



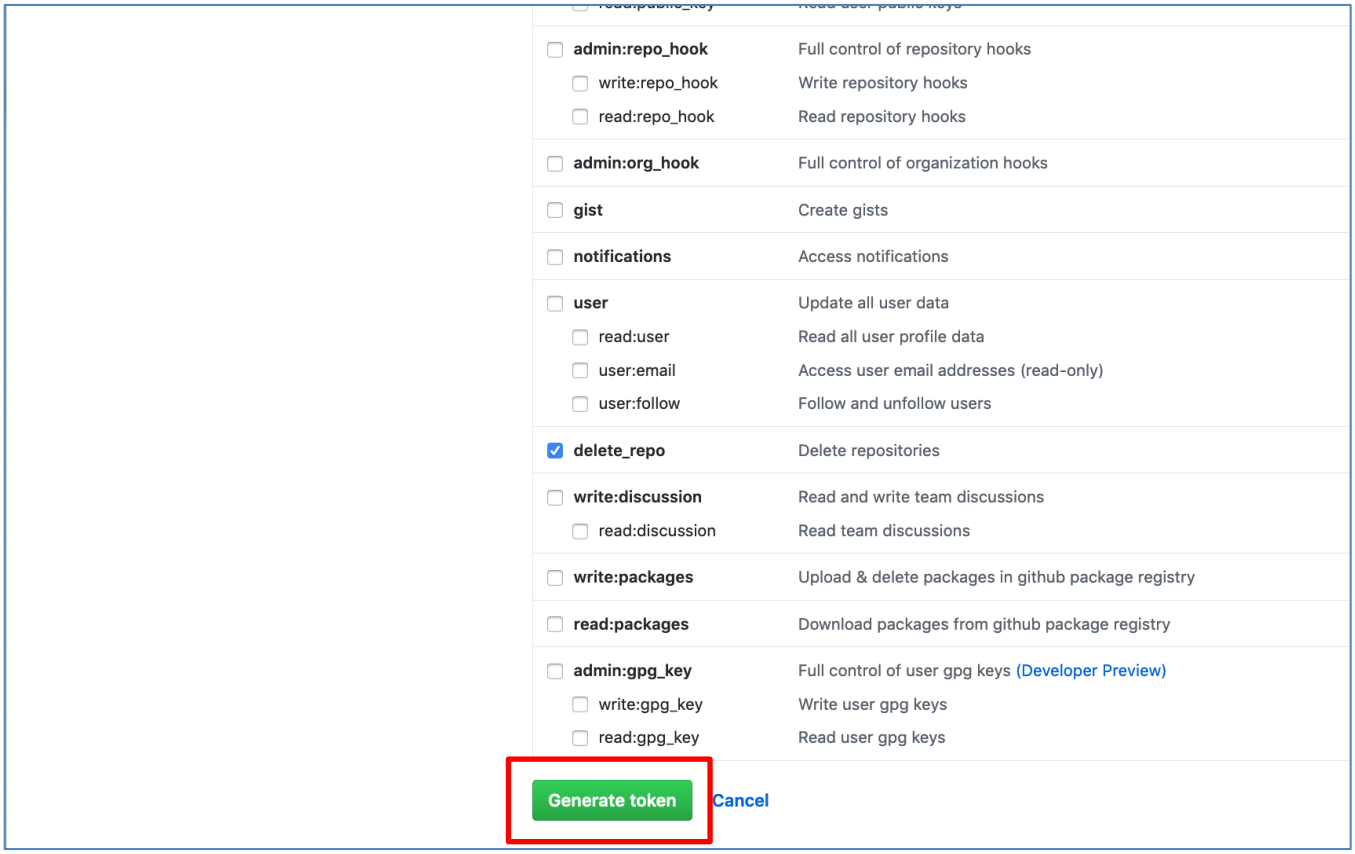
Confirm the user password if requested to get to the next screen as shown below.
Type in an appropriate description for this token and scroll down.



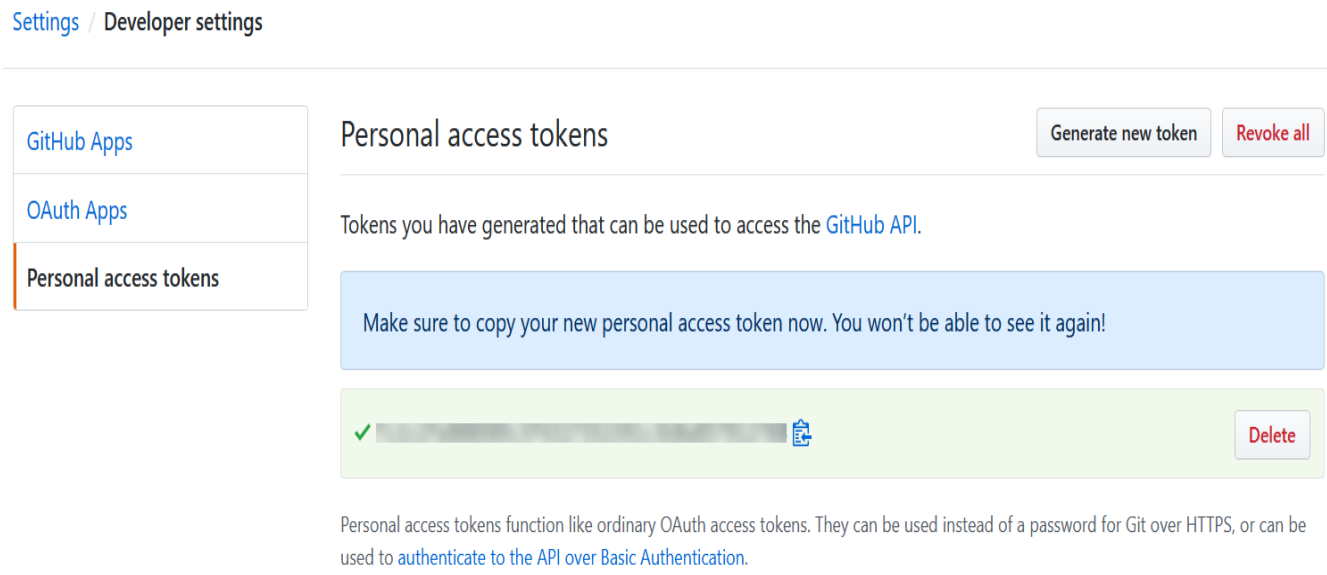
Select the desired access scopes by checking the appropriate boxes, but at minimum the **repo** related permissions.



Scroll down and click **Generate token.**



IMPORTANT: Save the personal access token for use later as CAM will request that token for any interaction with the GitHub repository.



10.2 *oradbaas* Template files

10.2.1 main.tf

```
#########################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#------------------------------------------------------------------------
#########################################################################
##
## Deploy Oracle database as a service
##
#########################################################################
## REFERENCE {"openstack_network":{"type": "openstack_reference_network"}}

terraform {
  required_version = "> 0.8.0"
}

provider "openstack" {
  insecure    = true
  version = "~> 1.2"
}

resource "openstack_compute_instance_v2" "dbserver" {
  name      = "${format("${var.dbserver_name}-${var.dbserver_user_data}-
${local.local_dbserver_ts}")}"
  image_name  = "${var.openstack_image_name}"
  flavor_name = "${var.openstack_flavor_name}"
  user_data = "${format("DBNAME=%s",var.dbserver_user_data)}"
  network {
    uuid = "${var.openstack_network_id}"
  }
}
```

10.2.2 variables.tf

```
#########################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

```
#
#-----------------------------------------------------------------------
#######################################################################
##
## Deploy Oracle database as a service
##
#######################################################################

variable "dbserver_name" {
  type = "string"
  description = " db server VM name prefix. Oracle dbname and a timestamp will be appended in
the PowerVC VM name."
}

variable "openstack_image_name" {
  type = "string"
  description = "PowerVC image name"
}

variable "openstack_flavor_name" {
  type = "string"
  description = "flavor name, or PowerVC compute template)"
}

variable "openstack_network_id" {
  type = "string"
  description = "Network id in PowerVC"
}

variable "dbserver_user_data" {
  type = "string"
  description = "Oracle DB name"
}

locals {
  local_dbserver_ts = "${replace(replace(replace(substr(timestamp(),2,17),"-
","")","T","_"),":","")}"
}
```

10.2.3 outputs.tf

```
#######################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#-----------------------------------------------------------------------
#######################################################################
##
## Deploy Oracle database as a service
##
#######################################################################

output "dbserver_ipaddr" {
  value = "${format("%s",join(",",
openstack_compute_instance_v2.dbserver.*.network.0.fixed_ip_v4))}"
```

```
}

output "dbserver_vm_name" {
  value = "${format("%s",join(",", openstack_compute_instance_v2.dbserver.*.name))}"
  description = "VM name in PowerVC"
}

output "dbserver_ts" {
  value = "${format("${local.local_dbserver_ts}")}"
}

output "dbname" {
  value = "${var.dbserver_user_data}"
}
```

10.2.4 camtemplate.json

```
{
   "name": "oradbaas",
   "description": "Description for oradbaas.",
   "type": "userCreated",
   "manifest": {
     "template_type": "Terraform",
     "template_format": "HCL",
     "template_provider": "OpenStack"
   },
   "metadata": {
     "bullets": []
   }
}
```

10.2.5 camvariables.json

```
{
  "input_datatypes": [ ],
  "input_namespaces": [ ],
  "output_namespace": "",
  "input_groups": [
    {
      "name": "Input-parameters",
      "label": "Input parameters"
    }
  ],
  "output_groups": [
     {
      "name": "Output-parameters",
      "label": "Output parameters"
    }
  ],
  "template_input_params": [
    {
      "name": "dbserver_name",
      "label": "db server VM name prefix",
      "description": " db server VM name prefix. Oracle dbname and a timestamp will be
appended in the PowerVC VM name.",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_image_name",
      "label": "Openstack Image Name",
      "description": "PowerVC image name",
      "type": "string",
      "default": "",
```

```
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_flavor_name",
      "label": "Openstack Flavor Name",
      "description": "flavor name, or PowerVC compute template)",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_network_id",
      "label": "Openstack Network Id",
      "description": "Network id in PowerVC",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "dbserver_user_data",
      "label": "Dbserver User Data",
      "description": "Oracle DB name",
      "type": "string",
      "default": "RSDtest",
      "validation": "",
      "group_name": "Input-parameters",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    }
  ],
  "template_output_params": [
    {
      "name": "dbserver_ipaddr",
      "label": "dbserver IP address",
      "description": null,
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string"
    },
    {
      "name": "dbserver_vm_name",
      "label": "dbserver VM name",
      "description": "VM name in PowerVC",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": false,
```

```
      "shortname": "",
      "type": "string"
    },
    {
      "name": "dbserver_ts",
      "label": "dbserver ts",
      "description": "internal only - timestamp for output to dbclient",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": true,
      "shortname": "",
      "type": "string"
    },
    {
      "name": "dbname",
      "label": "dbname",
      "description": "Database name",
      "group_name": "Output-parameters",
      "secured": false,
      "hidden": true,
      "shortname": "",
      "type": "string"
    }
  ]
}
```

10.3 *oraclient* Template files

10.3.1 main.tf

Note that you need to replace the text <MY_NFS_SERVER_IP> to reflect the IP of your NFS server.

```
#########################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#-------------------------------------------------------------------------
#########################################################################
##
##      Deploy LOP with Oracle client and Swingbench for oraclient
##
#########################################################################
#
# Note that the text <MY_NFS_SERVER_IP> needs to be replaced with the
# hostname or IP of your NFS server in the line:
# - mount <MY_NFS_SERVER_IP>:/export/stage /mnt
#########################################################################

terraform {
  required_version = "> 0.8.0"
}

provider "openstack" {
  insecure    = true
  version = "~> 1.2"
}

resource "openstack_compute_instance_v2" "dbclient" {
  name  = "${format("${var.dbclient_name}-${var.dbserver_user_data}-${var.dbserver_ts}")}"
  image_name  = "${var.openstack_image_name_dbclient}"
  flavor_name = "${var.openstack_flavor_name_dbclient}"
  user_data = <<EOT
#cloud-config
write_files:
  - content: |
      # dbserver parameters
      ${format("export DBNAME=%s",var.dbserver_user_data)}
      ${format("export DBSERVERIPADDR=%s",var.dbserver_ipaddr)}
      export ORASYSPW=oracle
      export SWINGBENCHBINDIR=/opt/swingbench/bin
    path: /root/dbserver.cfg
  - content: |
      cd ~
      . ./.bash_profile
      . ./dbserver.cfg
      DBACTIVE=false
      MAXLOOP=30
      LOOPDELAY=10
      i=1
      while [ "$DBACTIVE" = false ] && [ "$i" -le $MAXLOOP ] ; do
        sqlplus system/$ORASYSPW@$DBSERVERIPADDR/$DBNAME < /dev/null | grep "Connected to"
```

```
      if [ "$?" -eq 0 ]; then
          echo "Oracle DB server active"
          TEMP=`sqlplus -s system/$ORASYSPW@$DBSERVERIPADDR/$DBNAME << EOF1
          set pages 0
          set head off
          set feed off
          select open_mode, database_status, shutdown_pending from v\\$database,
v\\$instance;
          exit
     EOF1`
       TEMP2=`echo $TEMP | tr -d ' '`
       if [ "$TEMP2" = "READWRITEACTIVENO" ]; then
            DBACTIVE=true
            echo "Oracle DB server open in READ WRITE mode"
       fi
       else
          if [ "$i" -eq 1 ]; then
                 /bin/echo -e "Waiting for Oracle DB server\c"
          fi
       fi
       sleep $LOOPDELAY
       /bin/echo -e ".\c"
       i=$(( $i + 1 ))
     done
     if [ "$DBACTIVE" = true ]; then
       TEMPFILE_NAME=`sqlplus -s system/$ORASYSPW@$DBSERVERIPADDR/$DBNAME << EOF
       set pages 0
       set head off
       set feed off
       select name from v\\$tempfile;
       exit
     EOF`
       echo tempfile name is = $TEMPFILE_NAME
       sqlplus -s system/$ORASYSPW@$DBSERVERIPADDR/$DBNAME << EOF2
       alter database tempfile '$TEMPFILE_NAME' resize 2G;
       exit
     EOF2
       cd $SWINGBENCHBINDIR
       ./oewizard -c ../wizardconfigs/oewizard.xml -bigfile -cl -create -cs
//$DBSERVERIPADDR/$DBNAME -dbap $ORASYSPW -scale 1 -u soe -p soe -ts soe -tc 16
     else
       echo "Unable to connect to database.. perhaps not active or incorrect connect
information"
       echo "Retries = $i"
     fi
    path: /root/sbsetup.sh
    permissions: '0755'
  - content: |
     cd
     . ./dbserver.cfg
     cd $SWINGBENCHBINDIR
     ./oewizard -c ../wizardconfigs/oewizard.xml -cl -drop -cs //$DBSERVERIPADDR/$DBNAME -
dbap $ORASYSPW -scale 1 -u soe -p soe -ts soe
    path: /root/dropsoe.sh
    permissions: '0755'
  - content: |
     cd
     . ./dbserver.cfg
     cd $SWINGBENCHBINDIR
     ./swingbench -cs "//$DBSERVERIPADDR/$DBNAME"
     cd
    path: /root/runsb.sh
    permissions: '0755'
runcmd:
  - systemctl restart vncserver@:1
  - mount <MY_NFS_SERVER_IP>:/export/stage /mnt
  - if [ `uname -p` = "ppc64le" ]; then cp /mnt/client/12g/instantclient*.leppc64.*.zip
/root; elif [[ `uname -p` = "ppc64" ]]; then cp /mnt/client/12g/instantclient*.ppc64.*.zip
/root; fi
```

```
  - unzip -o /mnt/Swingbench/swingbench261076.zip -d /opt
  - cd /root; for i in `ls | grep instantclient`; do unzip -o $i -d /opt; rm -f $i; done
  - chmod -R 755 /opt/swingbench /opt/instantclient_12_1
  - /bin/echo -e "export PATH=\$PATH:/opt/`ls /opt | grep instantclient` \nexport
LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:/opt/`ls /opt | grep instantclient`" >> /root/.bash_profile
  - umount /mnt
  - cd /root; rm -f instantclient*.zip
  - /bin/echo -e "\n*** please wait for sbsetup.sh completion msg.." | wall
  - sleep 180; /root/sbsetup.sh >> /root/sbsetup.log 2>&1
  - /bin/echo -e "\n*** sbsetup.sh ended. See /root/sbsetup.sh.log." | wall
EOT

  network {
    name = "${var.openstack_network_name}"
  }
}
```

10.3.2 variables.tf

```
#####################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#--------------------------------------------------------------------
#####################################################################
##
##      Deploy LOP with Oracle client and Swingbench for oraclient
##
#####################################################################

variable "dbclient_name" {
  type = "string"
  description = "DB client name (VM) on PowerVC"
  default = "dbc"
}

variable "openstack_image_name_dbclient" {
  type = "string"
  description = "PowerVC image name"
}

variable "openstack_flavor_name_dbclient" {
  type = "string"
  description = "PowerVC Compute Template name"
}

variable "openstack_network_name" {
  type = "string"
  description = "PowerVC network name"
}

variable "dbserver_user_data" {
  type = "string"
  description = "Oracle DB name"
```

```
}

variable "dbserver_ipaddr" {
  type = "string"
  description = "DB server IP address"
}

variable "dbserver_ts" {
  type = "string"
  description = "Unique id to be appended to db client name"
}
```

10.3.3 outputs.tf

```
#########################################################################
# Copyright: IBM Corp., 2020
#
# Written by: Stephen Poon, Ralf Schmidt-Dannert
# IBM North America Systems Hardware
# Power Technical Team, Oracle
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#------------------------------------------------------------------------
#########################################################################
##
##      Deploy LOP with Oracle client and Swingbench for oraclient
##
#########################################################################

output "dbclient_ipaddr" {
  value = "${format("%s",join(",",
openstack_compute_instance_v2.dbclient.*.network.0.fixed_ip_v4))}"
}

output "dbclient_vm_name" {
  value = "${format("%s",join(",", openstack_compute_instance_v2.dbclient.*.name))}"
  description = "dbclient VM name"
}

output "dbserver_connect" {
  value = "${format("Swingbench connect string: //%s/%s",var.dbserver_ipaddr,
var.dbserver_user_data)}"
}
```

10.3.4 camtemplate.json

```
{
   "name": "oraclient",
   "description": "Description for oraclient.",
   "type": "userCreated",
   "manifest": {
     "template_type": "Terraform",
     "template_format": "HCL",
     "template_provider": "OpenStack"
   },
   "metadata": {
     "bullets": []
   }
```

```
}
```

10.3.5 camvariables.json

```
{
  "input_datatypes": [],
  "input_namespaces": [],
  "output_namespace": "",
  "input_groups": [
    {
      "name": "input-parms",
      "label": "Input parameters"
    }
  ],
  "output_groups": [
    {
      "name": "output_parms",
      "label": "Output parmeters"
    }
  ],
  "template_input_params": [
    {
      "name": "dbclient_name",
      "label": "db client VM name prefix",
      "description": "DB client VM name prefix on PowerVC",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_image_name",
      "label": "Openstack Image Name",
      "description": "PowerVC image name",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_flavor_name",
      "label": "Openstack Flavor Name",
      "description": "flavor name, or PowerVC Compute Template name",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "openstack_network_id",
      "label": "Openstack Network id",
      "description": "PowerVC network id",
      "type": "string",
      "default": "",
      "validation": "",
```

```
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "dbserver_user_data",
      "label": "DB Server's DB name",
      "description": "Oracle DB name",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "dbserver_ipaddr",
      "label": "DB Server ip address",
      "description": "DB server IP address",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": true,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": true
    },
    {
      "name": "dbserver_ts",
      "label": "unique id in dbclient VM name",
      "description": "Unique id to be appended to db client VM name prefix",
      "type": "string",
      "default": "",
      "validation": "",
      "group_name": "input-params",
      "required": false,
      "secured": false,
      "hidden": false,
      "immutable": false,
      "immutable_after_create": false
    }
  ],
  "template_output_params": [
    {
      "name": "dbclient_ipaddr",
      "label": "DB client IP address",
      "description": "dbclient IP address",
      "group_name": "output_parms",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string",
      "default": "",
      "validation": "",
      "required": true,
      "immutable": false,
      "immutable_after_create": false
    },
    {
      "name": "dbclient_vm_name",
      "label": "DB client VM name",
```

```
      "description": "dbclient VM name",
      "group_name": "output_parms",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string",
      "default": "",
      "validation": "",
      "required": true,
      "immutable": false,
      "immutable_after_create": false
    },
    {
      "name": "dbserver_connect",
      "label": "Oracle db jdbc connect string",
      "description": "Oracle database jdbc connect string",
      "group_name": "output_parms",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string",
      "default": "",
      "validation": "",
      "required": true,
      "immutable": false,
      "immutable_after_create": false
    },
    {
      "name": "dbclient_vnc",
      "label": "dbclient VNC host display",
      "description": "dbclient VNC server and display number",
      "group_name": "output_parms",
      "secured": false,
      "hidden": false,
      "shortname": "",
      "type": "string",
      "default": "",
      "validation": "",
      "required": true,
      "immutable": false,
      "immutable_after_create": false
    }
  ]
}
```

10.4 Service definition

Note that the below requires significant adjustments to match your specific environment and is only meant as illustration how the service configuration will look like. See the chapter on creating the service for more details.

10.4.1 oradbaas_oraclient.json

```
{
  "service": {
    "specVersion": "v3",
    "catalog_metadata": {
      "name": "Oracle 12C on AIX with Swingbench Client on LoP",
      "description": "Deploys an AIX VM running Oracle database 12c and a Linux on Power VM
with Oracle Instant Client and Swingbench.",
      "image": "serviceicon_1.svg",
      "category": "WSC cam demo",
      "bullets": [],
      "providerDisplayName": "IBM",
      "longDescription": "",
      "documentationUrl": "",
      "supportUrl": "",
      "bindable": false,
      "updatable": "",
      "systemTags": true,
      "author": "admin"
    },
    "tags": [
      {
        "name": "request_group",
        "label": "request_group",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one
group and this identifies the context of which group the user made the request.",
        "isSystemTag": true,
        "default": "${svc_instance.group}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "request_user",
        "label": "request_user",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The user id of the current user that requested the provisioning of
the template.",
        "isSystemTag": true,
        "default": "${svc_instance.owner}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_name",
        "label": "service_name",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Name provided for the service instance by the end user at time of
request",
        "isSystemTag": true,
        "default": "${svc_instance.name}",
        "customtype": "string",
```

```
        "permission": "Read-Only"
      },
      {
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the service
instance of the template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "actions": [
      {
        "type": "provision",
        "name": "Provision",
        "description": "Default action for deployment of service",
        "input_parameters": [
          {
            "name": "dbserver_user_data",
            "label": "Oracle dbname",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": true,
            "secured": false,
            "description": "Oracle DB name",
            "view": "create new",
            "default": "RSDtest"
          }
        ],
        "flow": {
          "conditions": [],
          "templates": [
            {
              "oradbaas": {
                "title": "oradbaas_56f6b8",
                "template_name": "oradbaas",
                "version": "master",
                "id": "oradbaas13c2cfb6",
                "template_type": "Terraform",
                "template_content_type": "OpenStack",
                "template_provider": "OpenStack",
                "instance_name": "dbs",
                "cloud_connection_name": "PowerVCDev",
                "template_data_objects": {},
                "template_params": {
                  "dbserver_name": "dbs",
                  "openstack_image_name": "DBaaSv3_Ora_SSD",
                  "openstack_flavor_name": "ora_dbaas",
                  "openstack_network_id": "464aceff-141d-453d-a40b-7008d9c06614",
                  "dbserver_user_data": "${input_parameters.dbserver_user_data}"
                },
                "outputs": [
                  "dbserver_ipaddr",
                  "dbserver_vm_name",
                  "dbserver_ts"
                ],
                "error": false,
                "isErrorFlow": false,
                "warning": false
              }
```

```
    },
    {
      "oraclient": {
        "title": "oraclient_384925",
        "template_name": "oraclient",
        "version": "master",
        "id": "oracliend1356a5d",
        "template_type": "Terraform",
        "template_content_type": "OpenStack",
        "template_provider": "OpenStack",
        "instance_name": "dbc",
        "cloud_connection_name": "PowerVCDev",
        "template_data_objects": {},
        "template_params": {
          "dbclient_name": "dbc",
          "openstack_image_name_dbclient": "img_rhel76le_pvcdev",
          "openstack_flavor_name_dbclient": "small",
          "openstack_network_name": "AON_93",
          "dbserver_user_data": "${templates.oradbaas13c2cfb6.dbserver_user_data}",
          "dbserver_ipaddr": "${templates.oradbaas13c2cfb6.output.dbserver_ipaddr}",
          "dbserver_ts": "${templates.oradbaas13c2cfb6.output.dbserver_ts}"
        },
        "outputs": [
          "dbclient_ipaddr",
          "dbclient_vm_name",
          "dbserver_connect"
        ],
        "error": false,
        "isErrorFlow": false,
        "warning": false
      }
    }
  ],
  "resthooks": [],
  "notifications": [],
  "sequence": {
    "0": "oradbaas13c2cfb6",
    "1": "oracliend1356a5d"
  },
  "error_sequence": {}
},
"output_parameters": [
  {
    "name": "dbserver_ipaddr",
    "label": "DB server IP address",
    "customtype": "string",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "DB server IP address",
    "default": "${templates.oradbaas13c2cfb6.output.dbserver_ipaddr}"
  },
  {
    "name": "dbserver_vm_name",
    "label": "DB server VM name",
    "customtype": "string",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "DB server VM name in PowerVC",
    "default": "${templates.oradbaas13c2cfb6.output.dbserver_vm_name}"
  },
  {
    "name": "dbclient_ipaddr",
    "label": "dbclient IP address",
```

```
            "customtype": "string",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "DB client IP address",
            "default": "${templates.oracliend1356a5d.output.dbclient_ipaddr}"
          },
          {
            "name": "dbclient_vm_name",
            "label": "DB client VM name",
            "customtype": "string",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "DB client VM name in PowerVC",
            "default": "${templates.oracliend1356a5d.output.dbclient_vm_name}"
          },
          {
            "name": "ora_dbname",
            "label": "Oracle dbname",
            "customtype": "string",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Oracle Database name",
            "default": "${templates.oradbaas13c2cfb6.output.dbname}"
          },
          {
            "name": "oradb_connect",
            "label": "Oracle DB connect string",
            "customtype": "string",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Oracle DB connect string",
            "default": "${templates.oracliend1356a5d.output.dbserver_connect}"
          }
        ]
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
      }
    ]
  }
}
```

11 References

- Containers, 12 August 2019, IBM Cloud Education
  https://www.ibm.com/cloud/learn/containers

- IBM Cloud Automation Manager documentation
  https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.0.0/kc_welcome.html

- Concepts and key components of Cloud Automation Manager
  https://www.ibm.com/cloud/garage/content/course/ibm-cloud-private-automation-manager/3

- HashiCorp Terraform Documentation
  https://www.terraform.io/docs/index.html

- HashiCorp Introduction to Terraform
  https://www.terraform.io/intro/index.html

- HashiCorp Terraform Configuration Syntax
  https://www.terraform.io/docs/configuration/syntax.html

- HashiCorp Terraform Documentation - openstack_compute_instance_v2
  https://www.terraform.io/docs/providers/openstack/r/compute_instance_v2.html

- Canonical Ltd., cloud-init Documentation
  https://cloudinit.readthedocs.io/en/latest/

- Canonical Ltd., Cloud Config Examples
  https://cloudinit.readthedocs.io/en/latest/topics/examples.html

- Oracle Instant Client, Oracle Corporation
  https://www.oracle.com/database/technologies/instant-client.html

- Dominic Giles, Swingbench
  http://www.dominicgiles.com/swingbench.html

- Technical Whitepaper, Deploying Oracle Database as a Service with IBM Cloud PowerVC Manager 1.3.1, Stephen R. Poon, Ralf Schmidt-Dannert, May 6, 2017
  http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102708

- Export and Import of IBM Cloud Automation Manager service to and from GIT repositories
  https://developer.ibm.com/cloudautomation/tutorials/export-import-ibm-cloud-automation-manager-service-git-repositories/

- Creating Cloud Automation Manager service instances from IBM Cloud Private
  https://www.ibm.com/support/knowledgecenter/en/SS2L37_3.2.1.0/cam_icp_cam_serviceinst.html

- Infrastructure as Code: Terraform meets PowerVC, Joe Cropper, Published on June 29, 2017 / Updated on June 30, 2017
  https://developer.ibm.com/powervc/2017/06/29/infrastructure-code-terraform-meets-powervc/

12 Notices and Trademarks

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice and represent goals and objectives only.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. The sample programs are licensed under:
Apache License, Version 2.0 (the "License");
You may not use these files except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided “AS IS”, without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, in the United States, other countries, or both. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of other companies:
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.