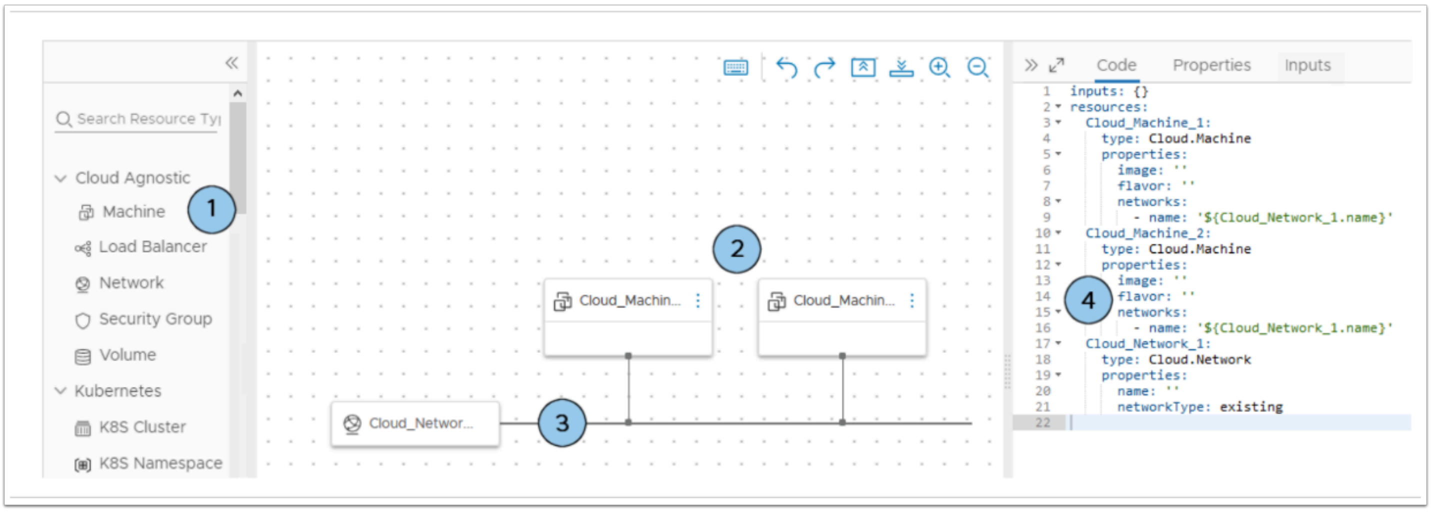
Lab 04 - Advanced Cloud Templates

Introduction

You use the design page to create Aria Automation Assembler template specifications for the machines and applications that you want to provision.

The code editor allows you to type, cut, copy, and paste code directly. If you're uncomfortable editing code, you can click a resource that's already in the design canvas, click the code editor Properties tab, and enter values there. Property values that you enter appear in the code as if you had typed them directly.



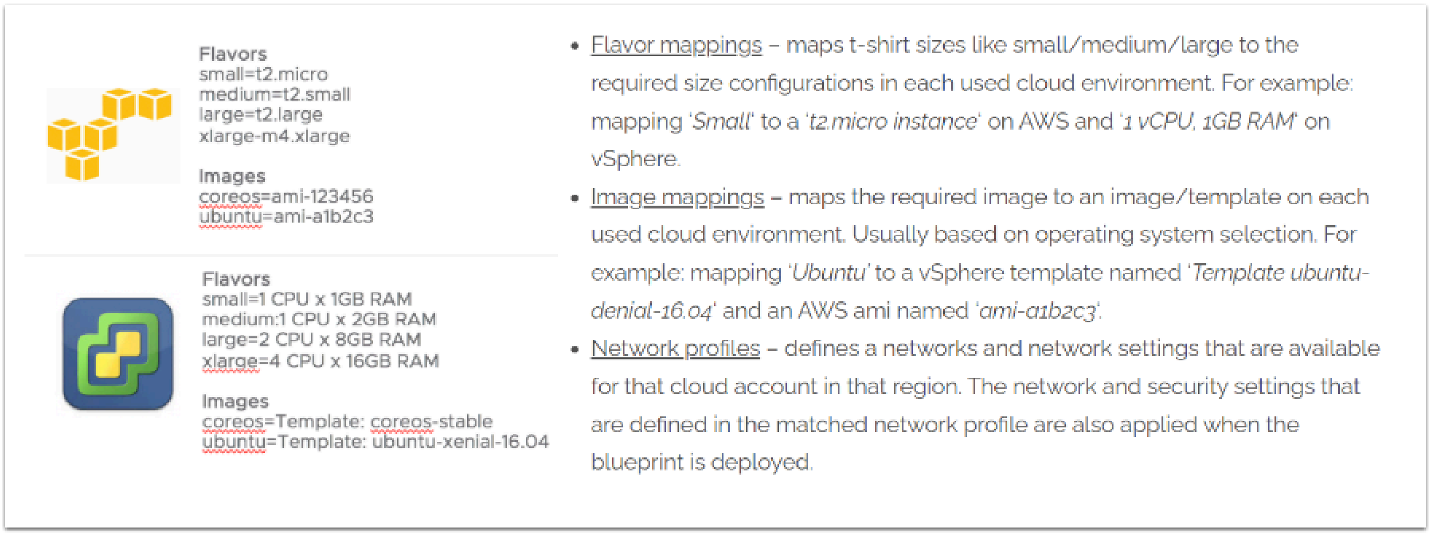
```
inputs: {}
resources:
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: ''
      flavor: ''
      networks:
        - name: '${Cloud_Network_1.name}'
  Cloud_Machine_2:
    type: Cloud.Machine
    properties:
      image: ''
      flavor: ''
      networks:
        - name: '${Cloud_Network_1.name}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: ''
      networkType: existing
```

Cloud Agnostic Templates

A cloud agnostic template is a blueprint that allows you to deploy exactly the same construct to different clouds. Aria Automation Assembler has the intelligence to take this cloud agnostic blueprint and know when to place it where using which specific cloud components. At the time of GA vSphere, AWS, GCP and Azure are supported.

This bring us directly to the limitations of cloud agnostic blueprints. Because Aria Automation Assembler 'translates' the used blueprint components to the cloud specific building blocks, you can only use components that are known to all cloud providers. I can use an Ubuntu template because I pointed to the cloud specific Ubuntu templates using image mappings. I can use t-shirt size deployments like small/medium/large because I defined what I mean with these t-shirts sizes in the different cloud providers using flavor mapping.

Requirements



Flavors
small=t2.micro
medium=t2.small
large=t2.large
xlarge-m4.xlarge

Images
coreos=ami-123456
ubuntu=ami-a1b2c3

Flavors
small=1 CPU x 1GB RAM
medium:1 CPU x 2GB RAM
large=2 CPU x 8GB RAM
xlarge=4 CPU x 16GB RAM

Images
coreos=Template: coreos-stable
ubuntu=Template: ubuntu-xenial-16.04

- Flavor mappings – maps t-shirt sizes like small/medium/large to the required size configurations in each used cloud environment. For example: mapping '*Small*' to a '*t2.micro instance*' on AWS and '*1 vCPU, 1GB RAM*' on vSphere.
- Image mappings – maps the required image to an image/template on each used cloud environment. Usually based on operating system selection. For example: mapping '*Ubuntu*' to a vSphere template named '*Template ubuntu-denial-16.04*' and an AWS ami named '*ami-a1b2c3*'.
- Network profiles – defines a networks and network settings that are available for that cloud account in that region. The network and security settings that are defined in the matched network profile are also applied when the blueprint is deployed.

In the previous lab we created all of these constructs, we will now use them all to illustrate the power of cloud agnostic blueprints.

TASKS

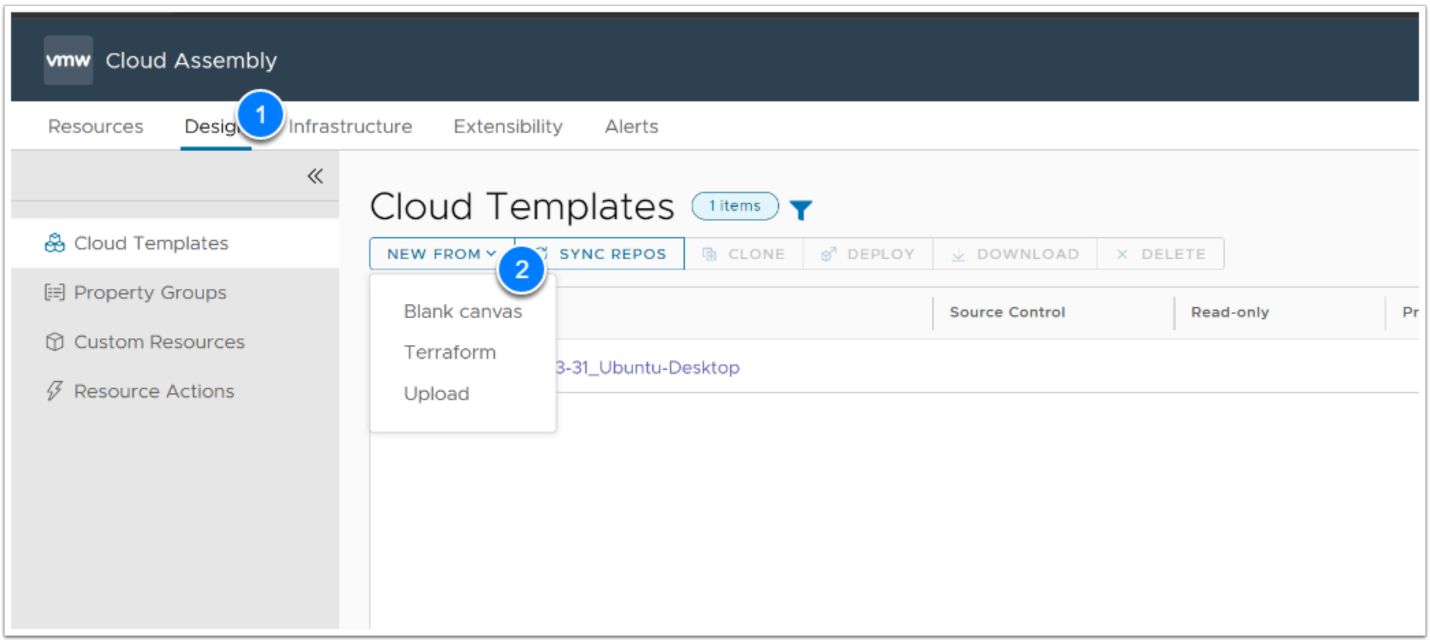
Task 1 - Create a Cloud Agnostic Template

1. From your VDI Desktop, access your Aria Automation Assembler browser tab. If the tab is closed or the session has timed out, then access Aria Automation Assembler from your browser bookmark and login using your student account (**vmcexpert#-xx@vmware-hol.com** | **VMware1!**) if necessary.

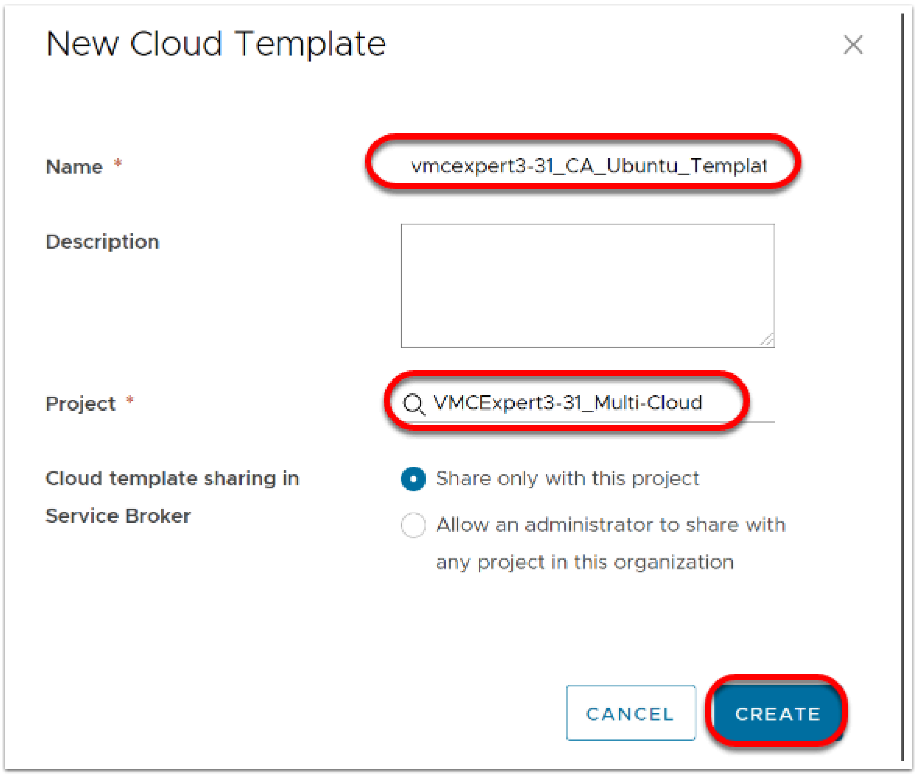
NOTE: You created this bookmark in a previous lab. If you missed the step you can use this link - https://www.mgmt.cloud.vmware.com/. Make sure to bookmark the page for future access

2. Click the **Design** Tab

3. Click **New From --> Blank Canvas**



4. Name the Template **{Your_User_Name}_CA_Ubuntu_Template**
5. Select your **Project**
6. Click **CREATE**



Name * vmcexpert3-31_CA_Ubuntu_Templat

Description

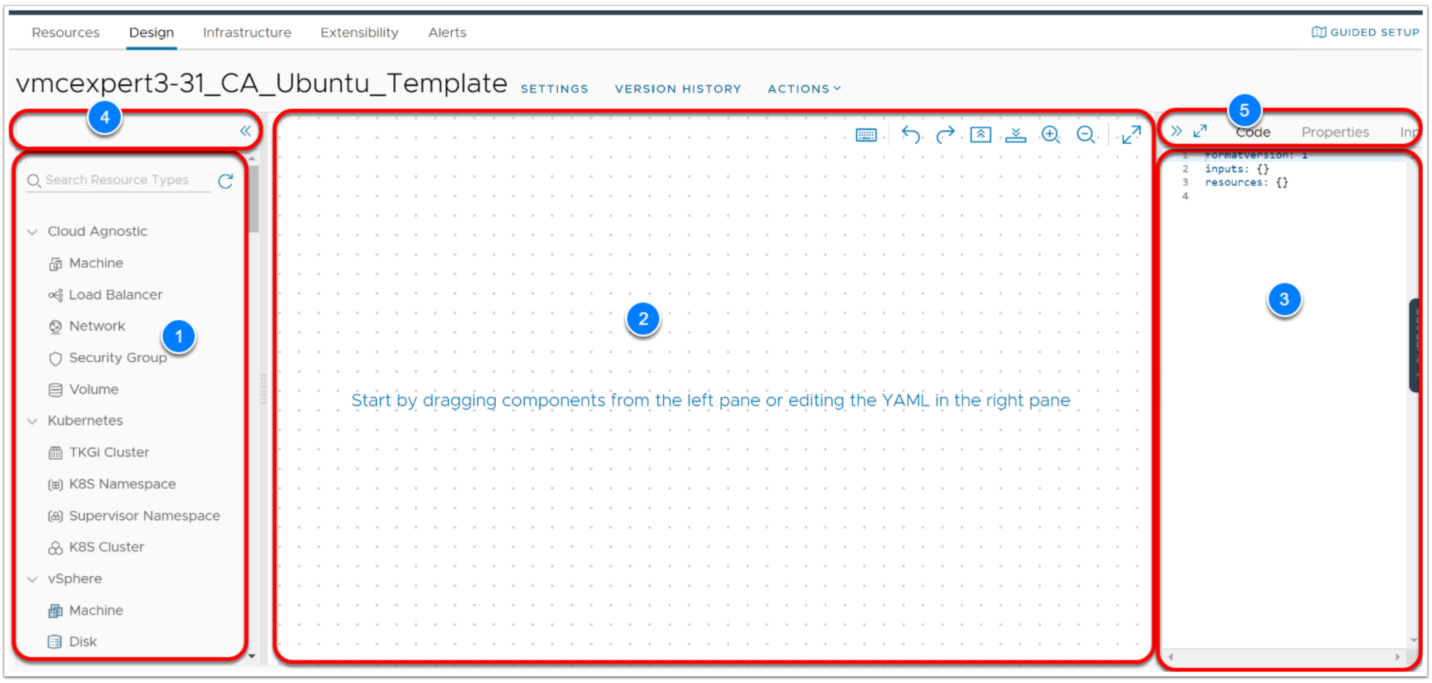
Project * VMCExpert3-31_Multi-Cloud

Cloud template sharing in Service Broker
[✓ Share only with this project]
Allow an administrator to share with any project in this organization

The blueprint canvas is split up into three main sections:

1. The **Components panel**, where you select the components that you want to use for your application.
2. The **Canvas**, where your application infrastructure topology is represented.
3. The **Code/Property Editor** where your YAML will be presented and modified.

   There are two other useful items to be aware of on this page that will help you with managing screen real estate.
4. The Components hide/show button
5. The Editor hide/show button. You may have noticed that the red box expands all the way across the bar, and not just on the button. This has been done because you can click anywhere on the bar to minimize these panels. Hide or show these panels as you see fit throughout the lab to make viewing relevant content easier.



```
formatVersion: 1
inputs: {}
resources: {}
```

7. From the Cloud Agnostic section drag the **Machine** object onto the canvas
8. In the code (YAML) Editor pane rename **Cloud_Machine_1** to **{Your_User_Name}** i.e. **vmcexpert3-31**
   make the following updates also:

```
image:'{Your_User_Name}_Ubuntu'
flavor: {Your_User_Name}_Small
constraints:
 - tag: 'platform:aws'
```

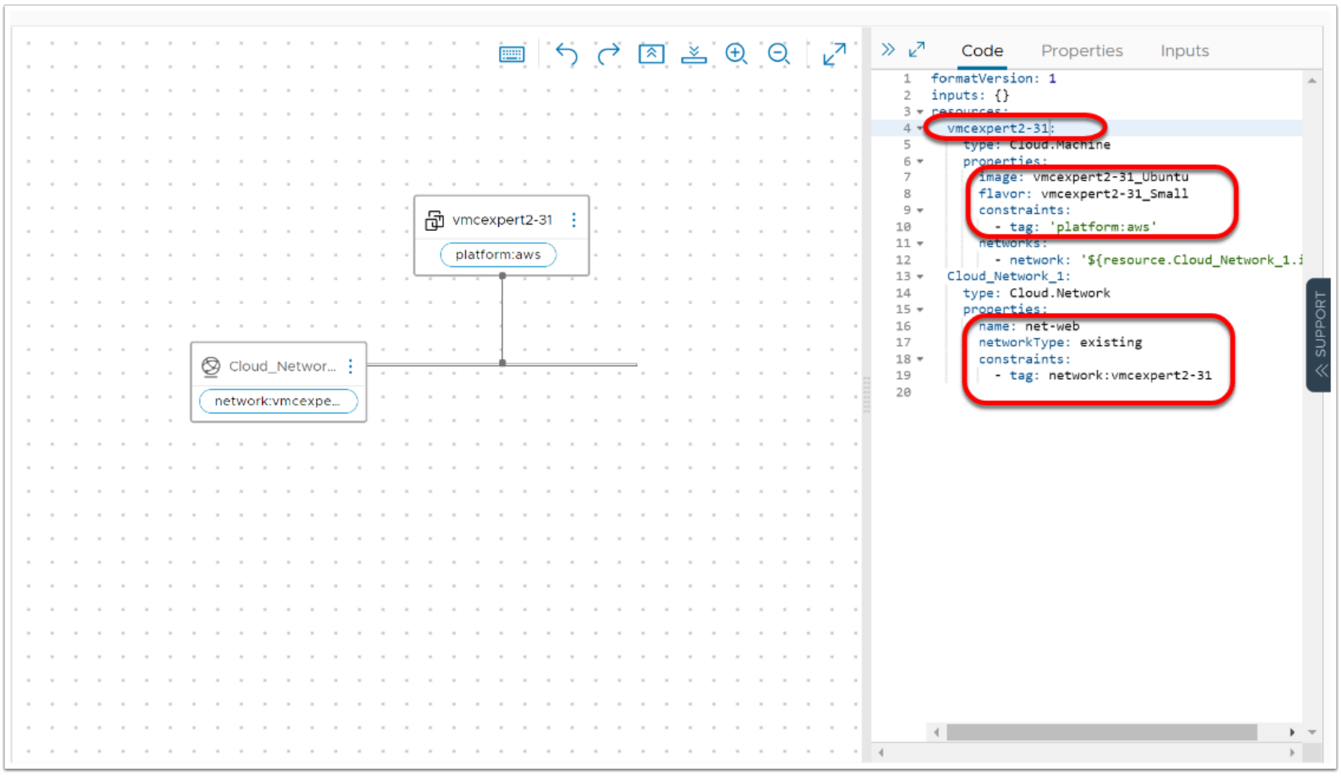
Click to copy

Be aware that YAML is whitespace sensitive, and incorrect indenting may lead to issues with provisioning. If you do make a mistake, you should see a red exclamation appear beside the line where the mistake has been made. Try it out now if you like by adding an extra space before image. Resolve the error and move to the next page.

9. Drag the **Network** object from the Cloud Agnostic section
10. On the Canvas **connect the Cloud Machine to the Cloud Network** by dragging a line from the left corner of the cloud machine to the Cloud Network object
11. In the Code Editor Pane add a **Name** & **Constraint** property to the Cloud Network

```
name: 'net-web'
networkType: existing
constraints:
  - tag: network:vmcexpert2-31
```

Click to copy

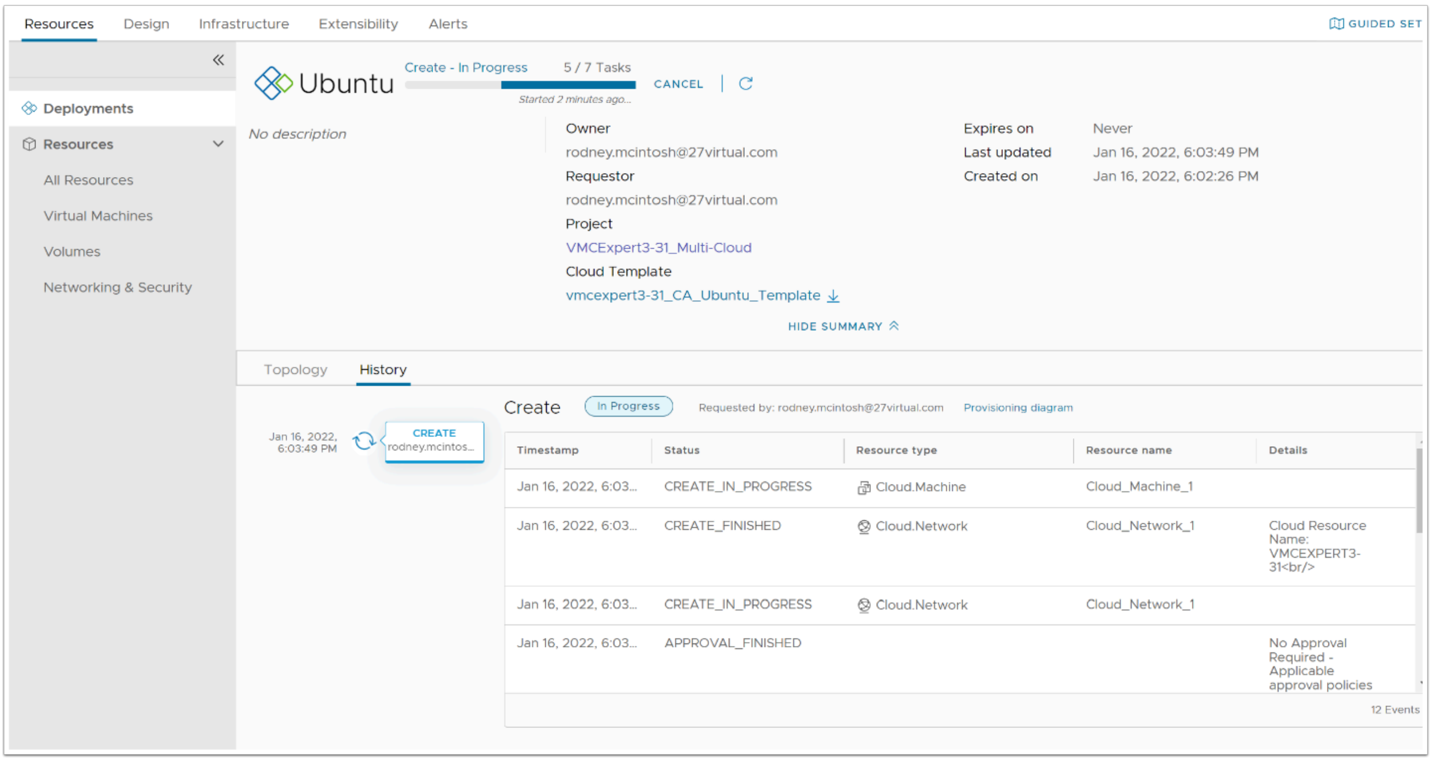


```
formatVersion: 1
inputs: {}
resources:
  vmcexpert2-31:
    type: Cloud.Machine
    properties:
      image: vmcexpert2-31_Ubuntu
      flavor: vmcexpert2-31_Small
      constraints:
        - tag: 'platform:aws'
      networks:
        - network: '${resource.Cloud_Network_1.i
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: net-web
      networkType: existing
      constraints:
        - tag: network:vmcexpert2-31
```

12. Click **TEST** to validate the template
13. Click **VERSION**, set the following values
    - Version: **1**
    - Change Log: **Initial Config**
    - Click **Create**

14. **Click Deploy**

15. Name the Deployment **{Your-User_Name}_Ubuntu**
16. Click **Deploy**
17. NOTE: The deployment takes a few minis to complete. You can click the **History** tab to monitor the steps



No description

Owner
rodney.mcintosh@27virtual.com
Requestor
rodney.mcintosh@27virtual.com
Project
VMCExpert3-31_Multi-Cloud
Cloud Template
vmcexpert3-31_CA_Ubuntu_Template

Expires on Never
Last updated Jan 16, 2022, 6:03:49 PM
Created on Jan 16, 2022, 6:02:26 PM

Jan 16, 2022, 6:03:49 PM

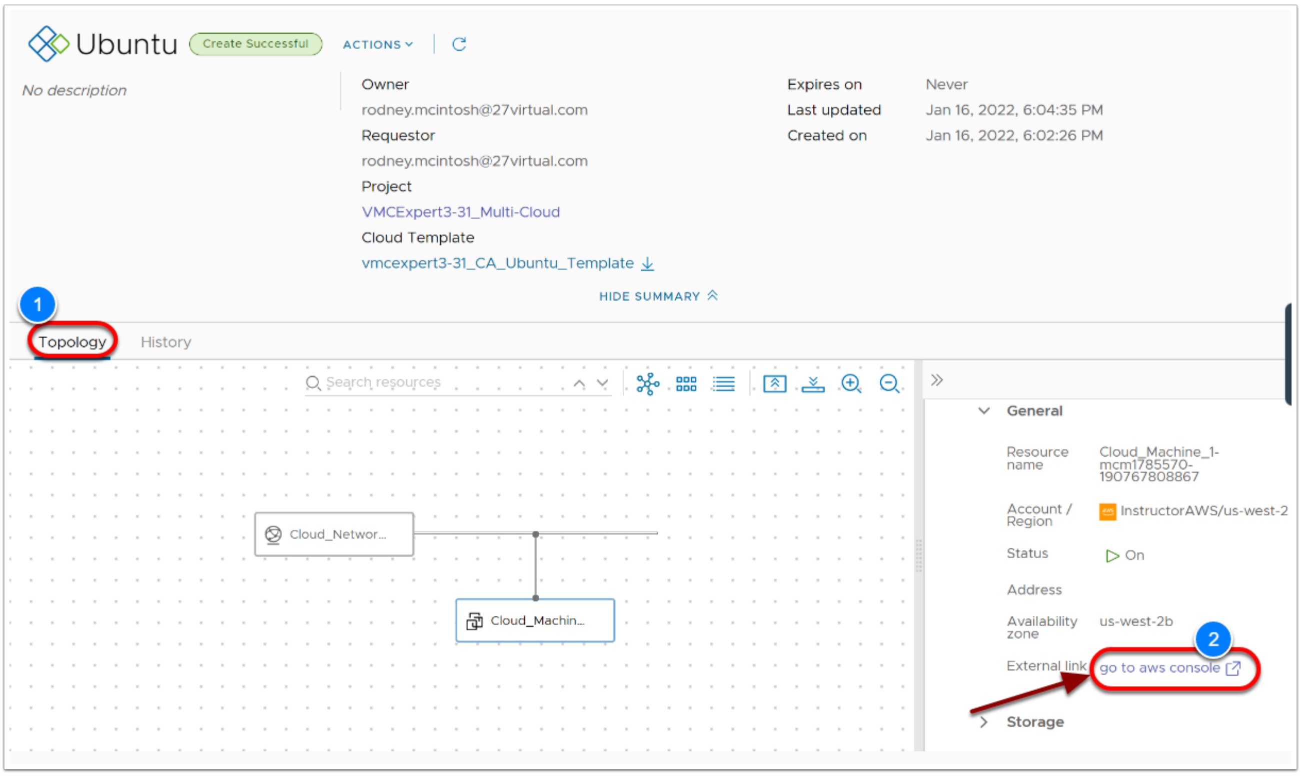
Create In Progress Requested by: rodney.mcintosh@27virtual.com Provisioning diagram

Timestamp	Status	Resource type	Resource name	Details
Jan 16, 2022, 6:03...	CREATE_IN_PROGRESS	Cloud.Machine	Cloud_Machine_1	
Jan 16, 2022, 6:03...	CREATE_FINISHED	Cloud.Network	Cloud_Network_1	Cloud Resource Name: VMCEXPERT3-31<br/>
Jan 16, 2022, 6:03...	CREATE_IN_PROGRESS	Cloud.Network	Cloud_Network_1	
Jan 16, 2022, 6:03...	APPROVAL_FINISHED			No Approval Required - Applicable approval policies

12 Events

We will now log into AWS to observe what was created

18. In the Topology tab, note the resource name (VM name) and click **go to aws console**, a new browser tab will open to the AWS Console


Ubuntu Create Successful ACTIONS

No description

Owner
rodney.mcintosh@27virtual.com
Requestor
rodney.mcintosh@27virtual.com
Project
VMCExpert3-31_Multi-Cloud
Cloud Template
vmcexpert3-31_CA_Ubuntu_Template

Expires on	Never
Last updated	Jan 16, 2022, 6:04:35 PM
Created on	Jan 16, 2022, 6:02:26 PM

HIDE SUMMARY

1 Topology History

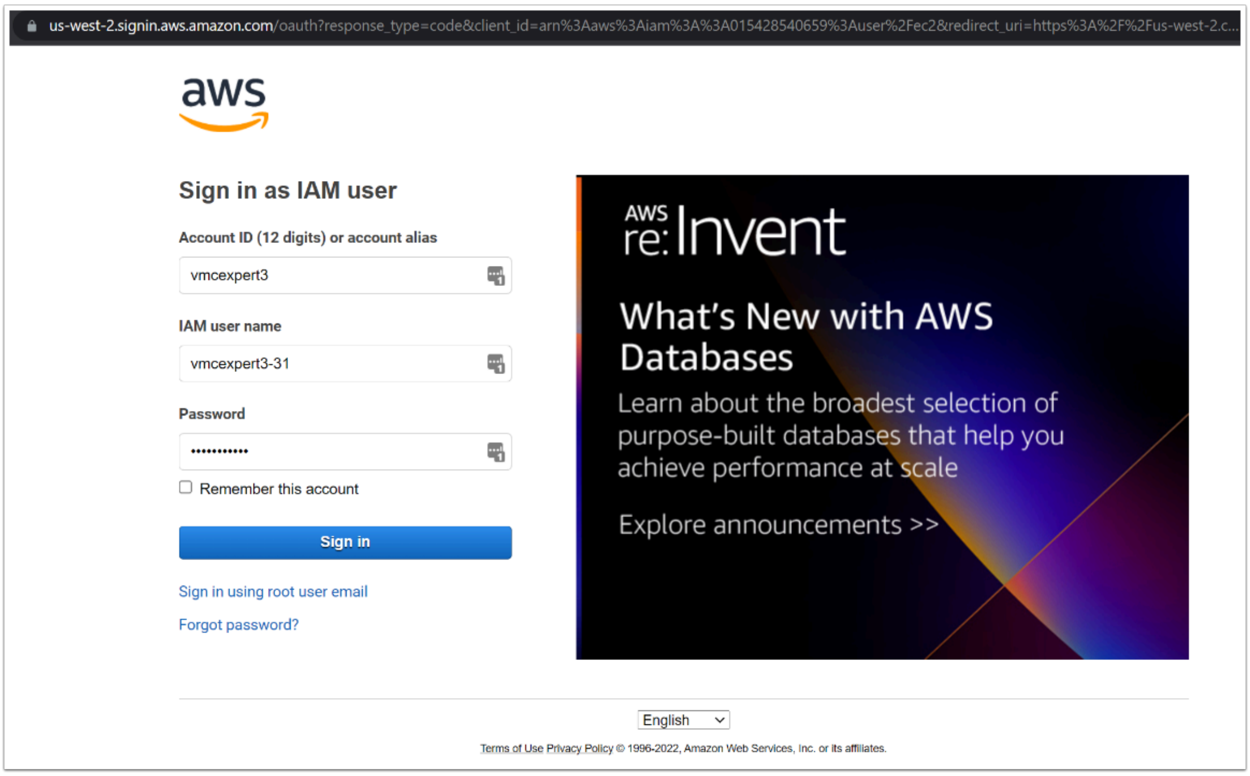
General

Resource name	Cloud_Machine_1-mcm1785570-190767808867
Account / Region	InstructorAWS/us-west-2
Status	On
Address	
Availability zone	us-west-2b
External link	go to aws console

2

Storage

19. Login using the following:
    - Account ID: **{Your Lab Environment} I.E. vmcexpert3**
    - IAM User name: **{Your User Name} I.E. vmcexpert3-31**
    - Password: **{Password assigned by the Instructor}**


us-west-2.signin.aws.amazon.com/oauth?response_type=code&client_id=arn%3Aaws%3Aiam%3A%3A015428540659%3Auser%2Fec2&redirect_uri=https%3A%2F%2Fus-west-2.c...

aws

Sign in as IAM user

Account ID (12 digits) or account alias
vmcexpert3

IAM user name
vmcexpert3-31

Password
••••••••••••

Remember this account

Sign in

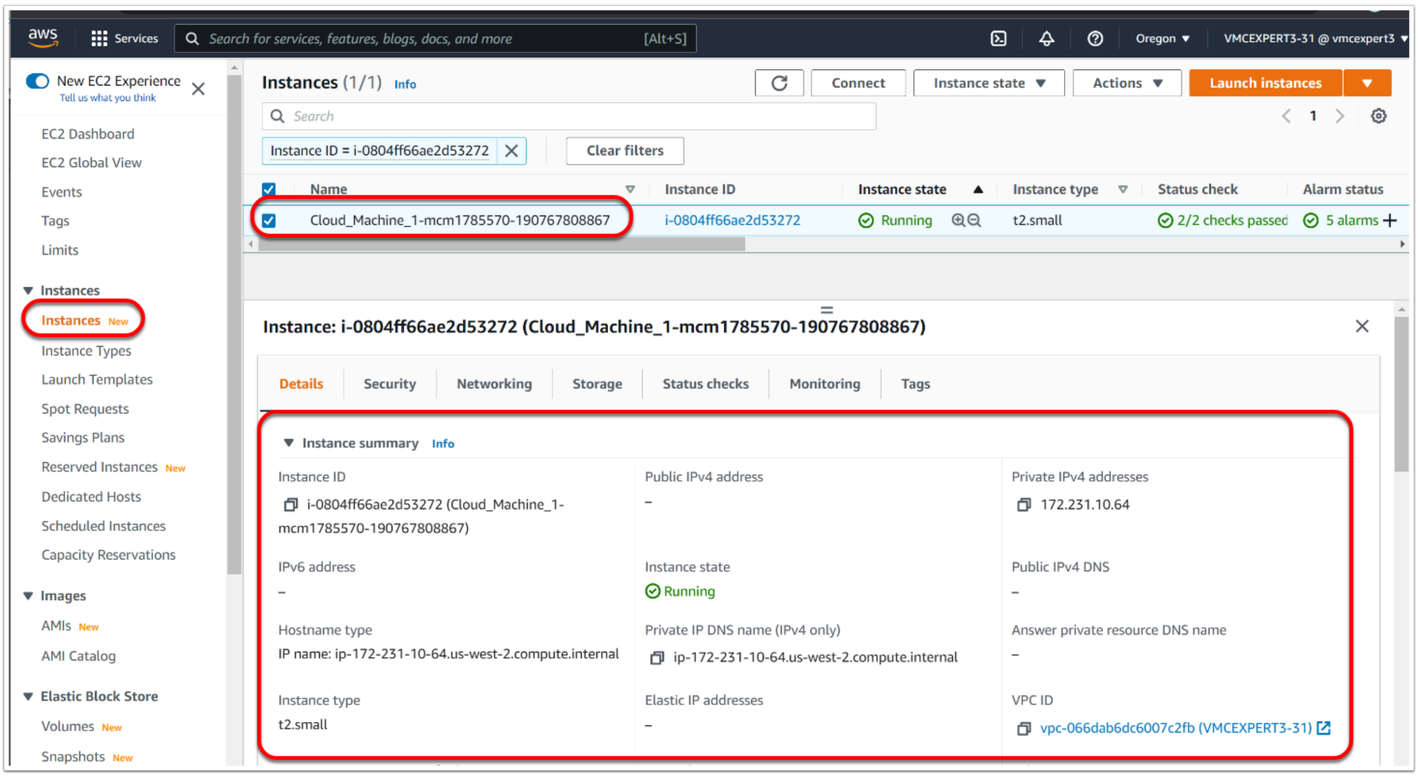
Sign in using root user email
Forgot password?

AWS re:Invent
What's New with AWS Databases
Learn about the broadest selection of purpose-built databases that help you achieve performance at scale
Explore announcements >>

English

Terms of Use Privacy Policy © 1996-2022, Amazon Web Services, Inc. or its affiliates.

20. Once logged in Click **EC2** and then **Instances->Instances**. You'll notice a new t2.small instance running. **Take note of this instance so you can identify it later.** If you do not see your instance, ensure that you are viewing **Oregon** (near top right corner)
21. Select the instance to view its details (IP Address, Platform, Instance Type, etc…) . The name will match the Resource Name in Aria Automation Assembler.


Instances (1/1)

Instance ID = i-0804ff66ae2d53272

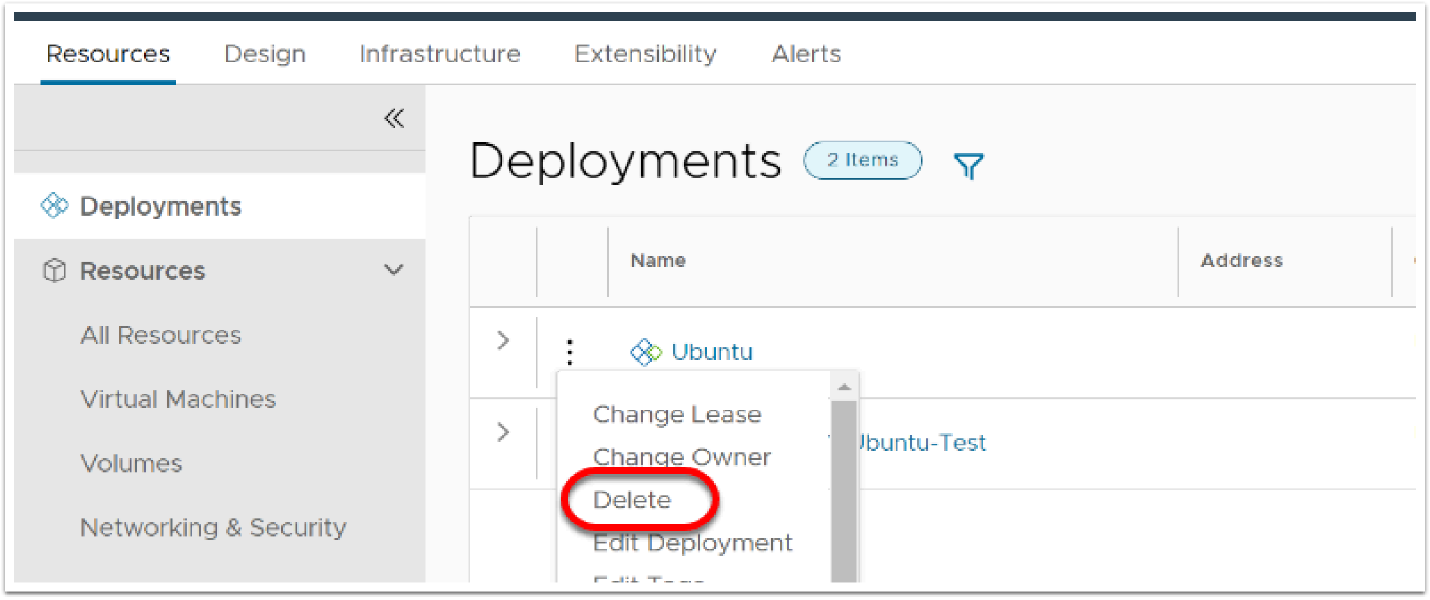
Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Cloud_Machine_1-mcm1785570-190767808867	i-0804ff66ae2d53272	Running	t2.small	2/2 checks passed	5 alarms

Instance: i-0804ff66ae2d53272 (Cloud_Machine_1-mcm1785570-190767808867)

Instance summary

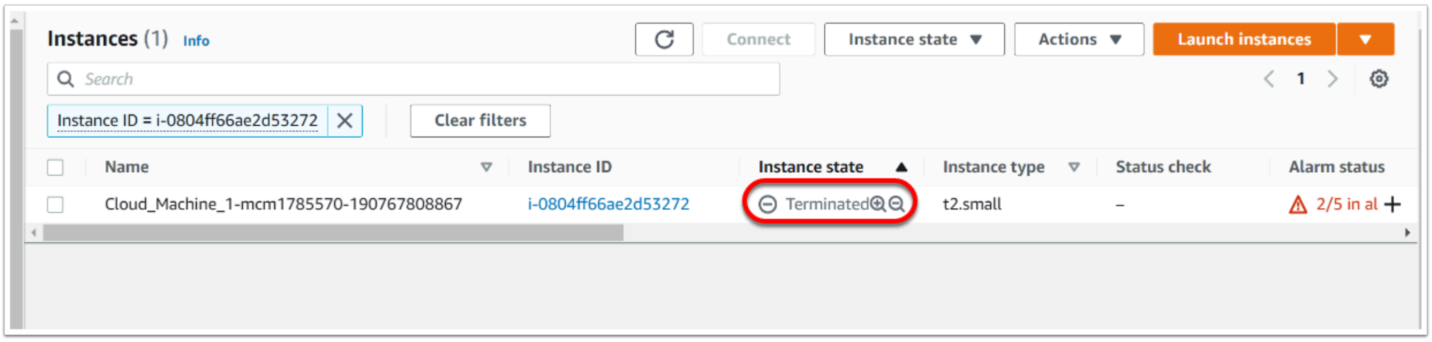
Instance ID: i-0804ff66ae2d53272 (Cloud_Machine_1-mcm1785570-190767808867)	Public IPv4 address: –	Private IPv4 addresses: 172.231.10.64
IPv6 address: –	Instance state: Running	Public IPv4 DNS: –
Hostname type: IP name: ip-172-231-10-64.us-west-2.compute.internal	Private IP DNS name (IPv4 only): ip-172-231-10-64.us-west-2.compute.internal	Answer private resource DNS name: –
Instance type: t2.small	Elastic IP addresses: –	VPC ID: vpc-066dab6dc6007c2fb (VMCEXPERT3-31)

22. Return to the **Aria Automation Assembler** browser tab
23. Click **Close** on the deployment
24. Select the **3 vertical dots** next to the deployment and click **delete**



25. Once the deployment deletion completes, return to the AWS Console browser tab

26. In the AWS Console the VM Instance may still be visible but the state should be "**Terminated**"



Instance ID = i-0804ff66ae2d53272

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Cloud_Machine_1-mcm1785570-190767808867	i-0804ff66ae2d53272	Terminated	t2.small	–	2/5 in al

NOTE: IN AWS Terminated images remain visible in the console for a while. When an instance terminates, the data on any instance store volumes associated with that instance is deleted. AWS Instance Termination

Task 2 - Using CloudConfig to Customize a deployment

With CloudConfig, you can add machine initialization commands that run at deployment time. You use initialization commands to automate the application of data or settings at instance creation time, which can customize users, permissions, installations, or any other command-based operations. Examples include:

- Setting a hostname
- Generating and setting up SSH private keys
- Installing packages

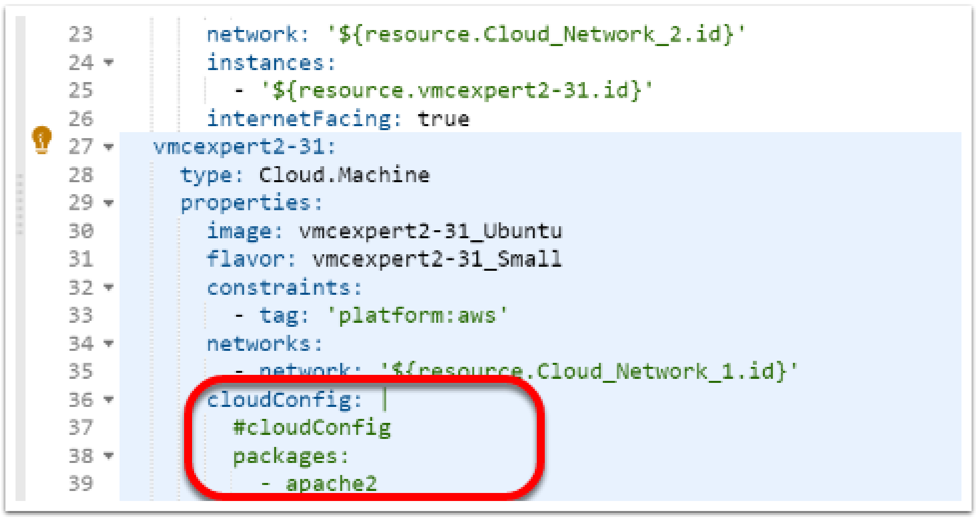
You can add a cloudConfig section to cloud template code, but you can also add one to a machine image in advance, when configuring infrastructure. Then, all cloud templates that reference the source image get the same initialization.

You might have an image map and a cloud template where both contain initialization commands. At deployment time, the commands merge, and Aria Automation Assembler runs the consolidated commands.

1. In the Aria Automation Assembler browser tab Click **Design**
2. Click on your Cloud Agnostic Ubuntu Cloud Template **{Your_User_Name}_CA_Ubuntu_Template**
3. Add the following code in the YAML Edition, just above the "**Cloud_Network_1**" Section

```
cloudConfig:  |
  #cloudconfig
  packages:
    - apache2
```

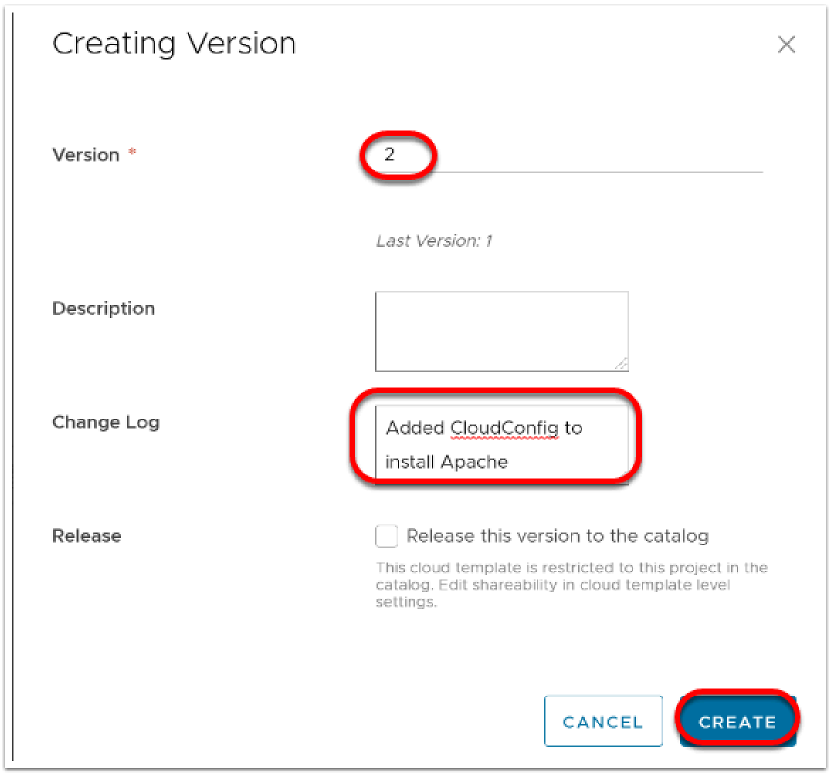
Click to copy



```
      network: '${resource.Cloud_Network_2.id}'
      instances:
        - '${resource.vmcexpert2-31.id}'
      internetFacing: true
  vmcexpert2-31:
    type: Cloud.Machine
    properties:
      image: vmcexpert2-31_Ubuntu
      flavor: vmcexpert2-31_Small
      constraints:
        - tag: 'platform:aws'
      networks:
        - network: '${resource.Cloud_Network_1.id}'
      cloudConfig: |
        #cloudConfig
        packages:
          - apache2
```

Note: Each indent is two spaces (you can use the Tab key on the keyboard). The **cloudConfig: |** aligns directly below **networks:**

4. Click **TEST** to validate the template
5. Click **VERSION** and provide the following inputs
   - Version: **2**
   - Change Log: **Added CloudConfig to Install Apache**
   - Click **Create**


Creating Version

Version * 2

Last Version: 1

Description

Change Log: Added CloudConfig to install Apache

Release: Release this version to the catalog

This cloud template is restricted to this project in the catalog. Edit shareability in cloud template level settings.

6. Click **Deploy**
7. Name the deployment **{Your_User_Name}_Simple Template**
8. Click **Deploy**
9. Feel free to review the deployment in the AWS Console as you did in the previous task after the deployment tasks completes
10. In Aria Automation Assembler Click **Close** to Close the deployment pane

We will now update the template to deploy multiple machine instances and a load-balancer.

Task 3 - Create and Deploy a Load-Balanced Web Application

1. In the Aria Automation Assembler browser tab Click **Design** Tab
2. Click on your Cloud Agnostic Ubuntu Cloud Template **{Your_User_Name}_CA_Ubuntu_Template**
3. Drag a **Cloud Agnostic Load Balancer** object onto the canvas
4. Drag a 2nd **Cloud Agnostic Network** onto the Canvas
5. Connect the **Load Balancer** to the 2nd **Network**
6. Select the 2nd **Network** and Set it's Name Property to **net-lb**
7. In the Code Editor Pane add a **Name & Constraint** property to the 2nd **Cloud Network**

```
name: 'net-lb'
```

```
networkType: existing
constraints:
  - tag: network:vmcexpert2-31
```

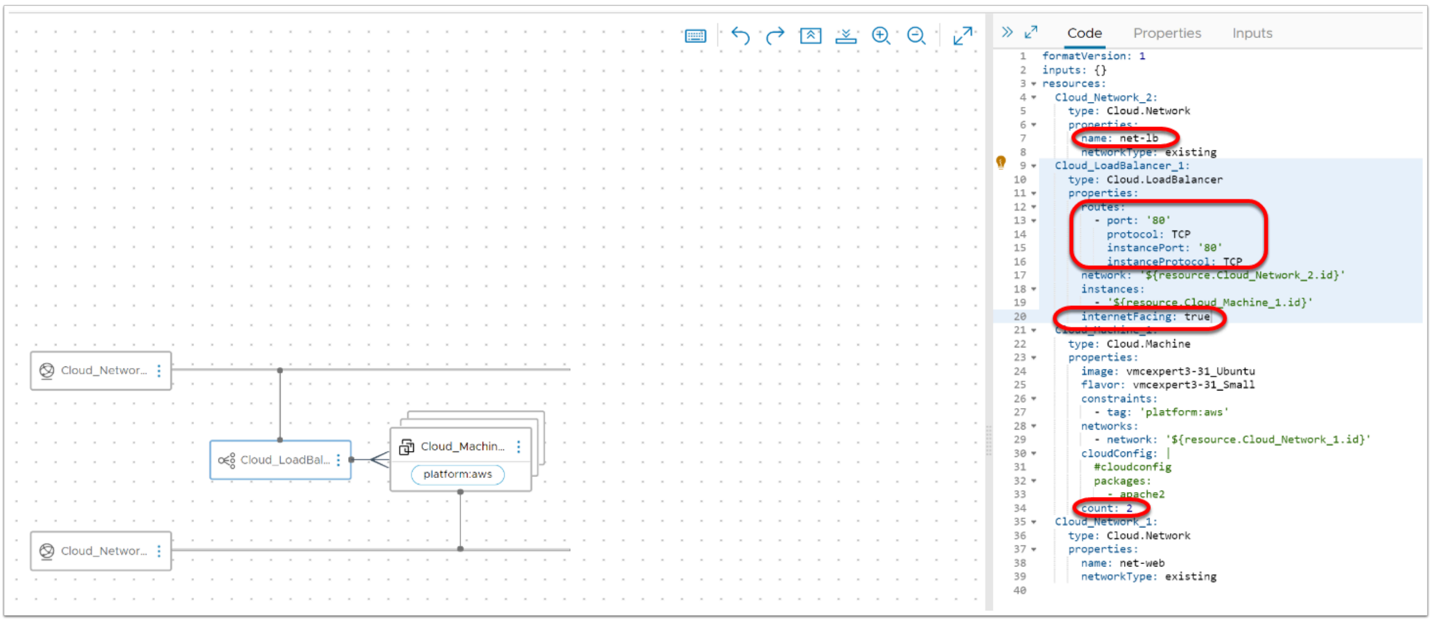
Click to copy

8. Reconnect the Load Balancer to the 2nd network.
9. Select the **Load Balancer** and set the following vales in the Routes Property

```
routes:
  - port: '80'
    protocol: 'TCP'
    instancePort: '80'
    instanceProtocol: 'TCP'
```

Click to copy

10. Change the internetFacing property value to true
11. Connect the **Cloud Machine** to the **Load Balancer**
12. Select the **Cloud Machine a**nd add a **count property** on the YAML. Set it's value to **2**



```
formatVersion: 1
inputs: {}
resources:
  Cloud_Network_2:
    type: Cloud.Network
    properties:
      name: net-lb
      networkType: existing
  Cloud_LoadBalancer_1:
    type: Cloud.LoadBalancer
    properties:
      routes:
        - port: '80'
          protocol: TCP
          instancePort: '80'
          instanceProtocol: TCP
      network: '${resource.Cloud_Network_2.id}'
      instances:
        - '${resource.Cloud_Machine_1.id}'
      internetFacing: true
  Cloud_Machine_1:
    type: Cloud.Machine
    properties:
      image: vmcexpert3-31_Ubuntu
      flavor: vmcexpert3-31_Small
      constraints:
        - tag: 'platform:aws'
      networks:
        - network: '${resource.Cloud_Network_1.id}'
      cloudConfig: |
        #cloudconfig
        packages:
          - apache2
      count: 2
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: net-web
      networkType: existing
```

13. Click **Test** to validate the Template

Finally, we will modify the template by adding a input field allowing the requestor to determine how many machine instances they want instead of having 2 machines deployed every time.

14. In the YAML Editor add the following properties and values in the **Input** section

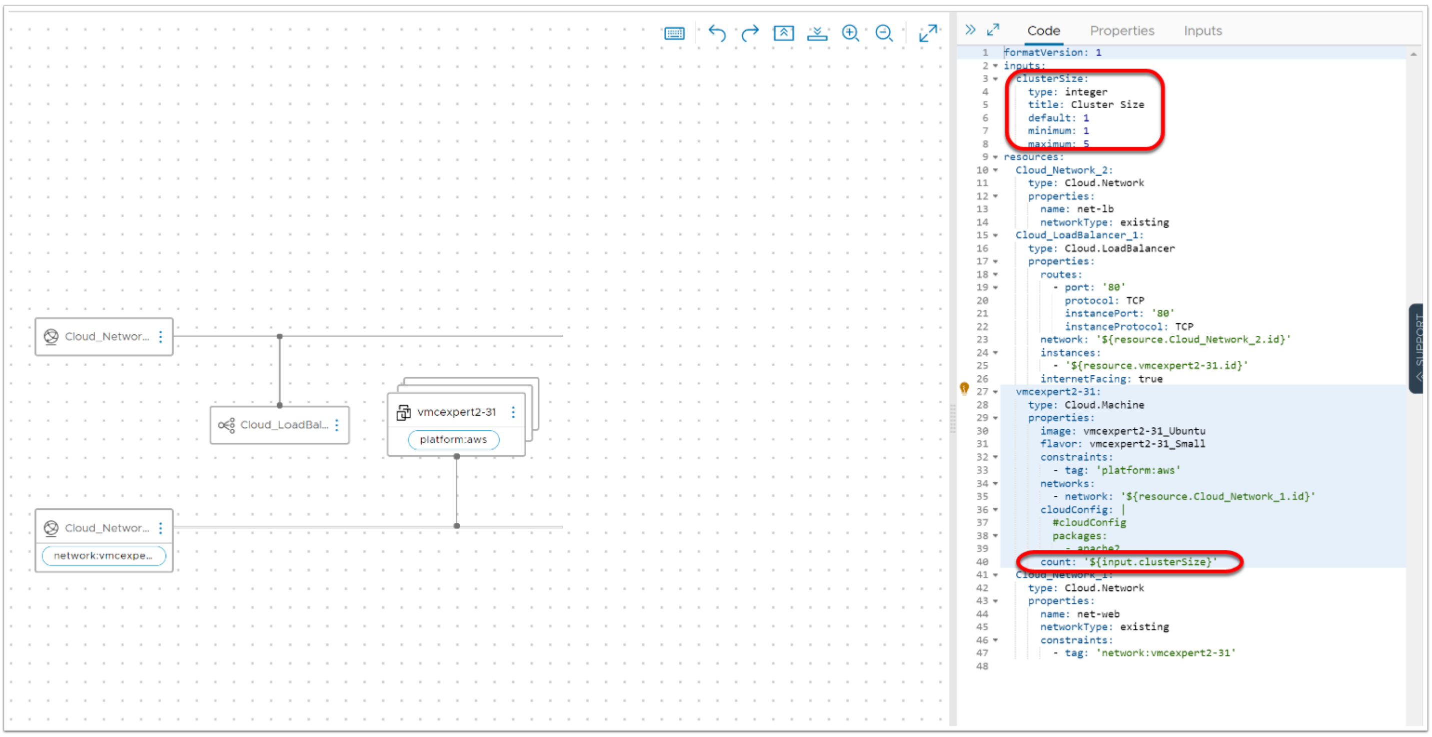
```
inputs:
  clusterSize:
    type:  integer
    title: Cluster Size
    default: 1
    minimum: 1
    maximum: 5
```

Click to copy

15. In the Cloud_machine_1 section, change the **count** value from **2** to **'${input.clusterSize}'**

```
count:  '${input.clusterSize}'
```

Click to copy

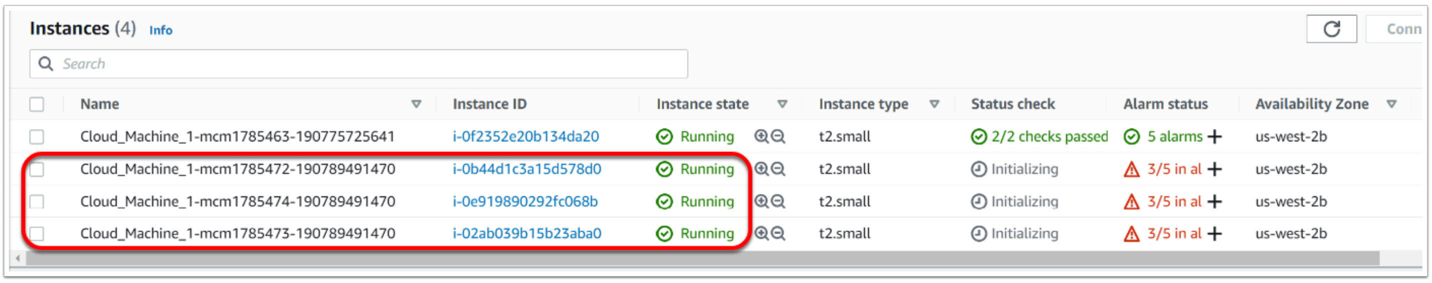


```
formatVersion: 1
inputs:
  clusterSize:
    type: integer
    title: Cluster Size
    default: 1
    minimum: 1
    maximum: 5
resources:
  Cloud_Network_2:
    type: Cloud.Network
    properties:
      name: net-lb
      networkType: existing
  Cloud_LoadBalancer_1:
    type: Cloud.LoadBalancer
    properties:
      routes:
        - port: '80'
          protocol: TCP
          instancePort: '80'
          instanceProtocol: TCP
      network: '${resource.Cloud_Network_2.id}'
      instances:
        - '${resource.vmcexpert2-31.id}'
      internetFacing: true
  vmcexpert2-31:
    type: Cloud.Machine
    properties:
      image: vmcexpert2-31_Ubuntu
      flavor: vmcexpert2-31_Small
      constraints:
        - tag: 'platform:aws'
      networks:
        - network: '${resource.Cloud_Network_1.id}'
      cloudConfig: |
        #cloudConfig
        packages:
          - apache2
      count: '${input.clusterSize}'
  Cloud_Network_1:
    type: Cloud.Network
    properties:
      name: net-web
      networkType: existing
      constraints:
        - tag: 'network:vmcexpert2-31'
```

16. Click **Test**, to validate the template
17. Click **Version** and provide the following inputs
    - Version: 3
    - Change Log: **Added Inputs for number of instance**
    - Click **Create**

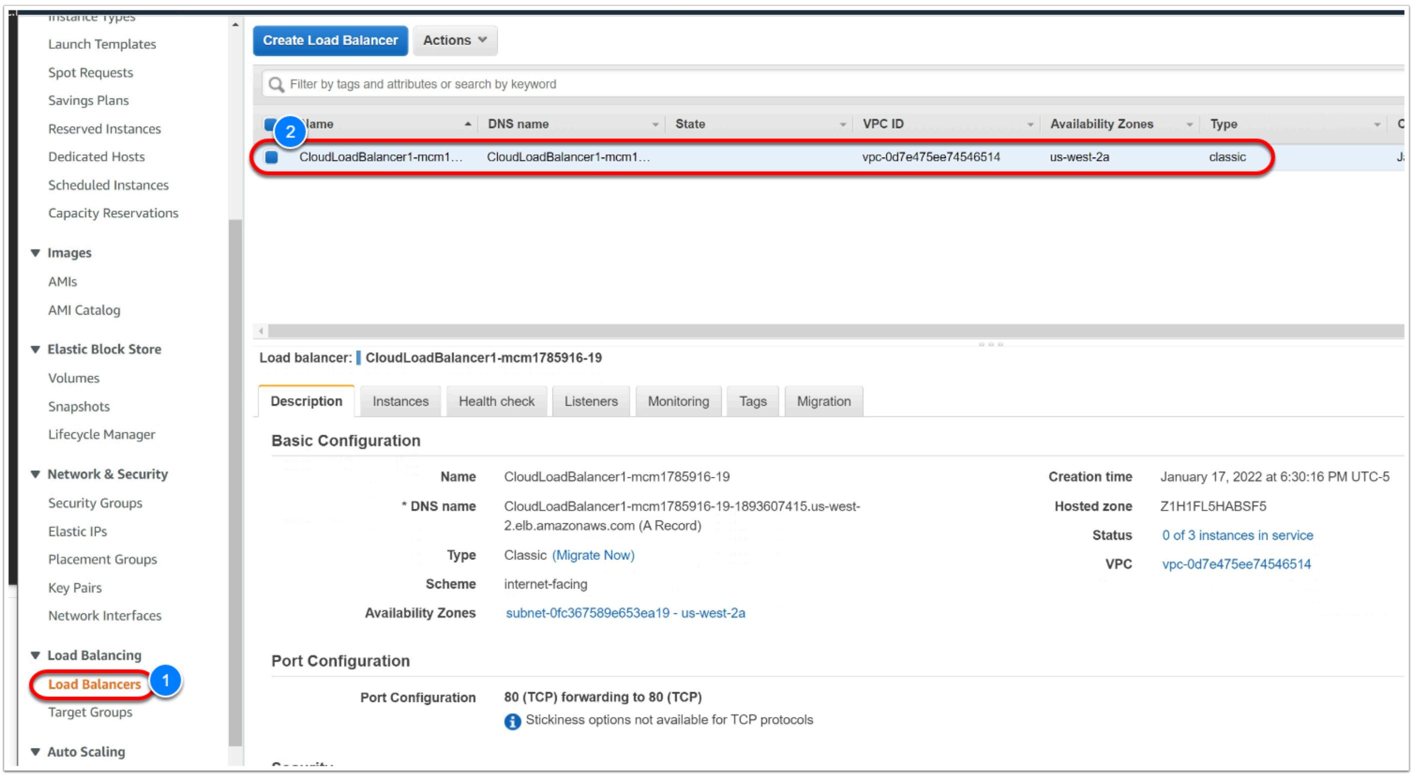
18. Click **Deploy** enter the following inputs
    - Deployment Name: **{Your_User_Name}_Flexible_Template**
    - Cloud Template Version: **Current Draft**
    - Click **Next**
    - Cluster Size: **3**
    - **Deploy**

19. Monitor and review the deployment steps
20. In the AWS Console you should now see the 3 additional instances and a Load Balancer



Instances (4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Cloud_Machine_1-mcm1785463-190775725641	i-0f2352e20b134da20	Running	t2.small	2/2 checks passed	5 alarms	us-west-2b
Cloud_Machine_1-mcm1785472-190789491470	i-0b44d1c3a15d578d0	Running	t2.small	Initializing	3/5 in al	us-west-2b
Cloud_Machine_1-mcm1785474-190789491470	i-0e919890292fc068b	Running	t2.small	Initializing	3/5 in al	us-west-2b
Cloud_Machine_1-mcm1785473-190789491470	i-02ab039b15b23aba0	Running	t2.small	Initializing	3/5 in al	us-west-2b



Name	DNS name	State	VPC ID	Availability Zones	Type
CloudLoadBalancer1-mcm1…	CloudLoadBalancer1-mcm1…		vpc-0d7e475ee74546514	us-west-2a	classic

Load balancer: CloudLoadBalancer1-mcm1785916-19

Basic Configuration

Name	CloudLoadBalancer1-mcm1785916-19	Creation time	January 17, 2022 at 6:30:16 PM UTC-5
* DNS name	CloudLoadBalancer1-mcm1785916-19-1893607415.us-west-2.elb.amazonaws.com (A Record)	Hosted zone	Z1H1FL5HABSF5
		Status	0 of 3 instances in service
Type	Classic (Migrate Now)	VPC	vpc-0d7e475ee74546514
Scheme	internet-facing		
Availability Zones	subnet-0fc367589e653ea19 - us-west-2a		

Port Configuration

Port Configuration 80 (TCP) forwarding to 80 (TCP)
Stickiness options not available for TCP protocols

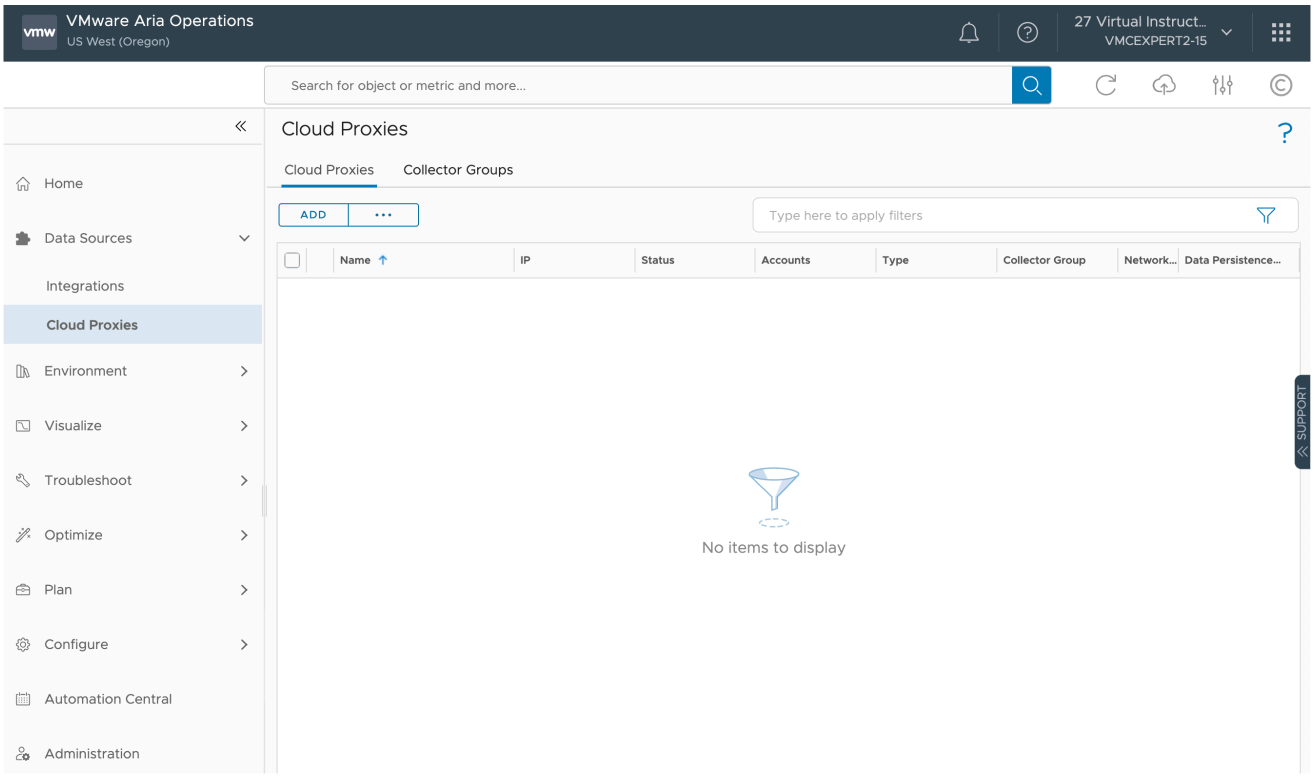
21. Back in the Browser tab for Aria Automation Assembler, Click **Close** to exit the Deployment page
22. Click the **Design.**
23. Click **Close**, to Exit the cloud Template. We will continue from here in the next task

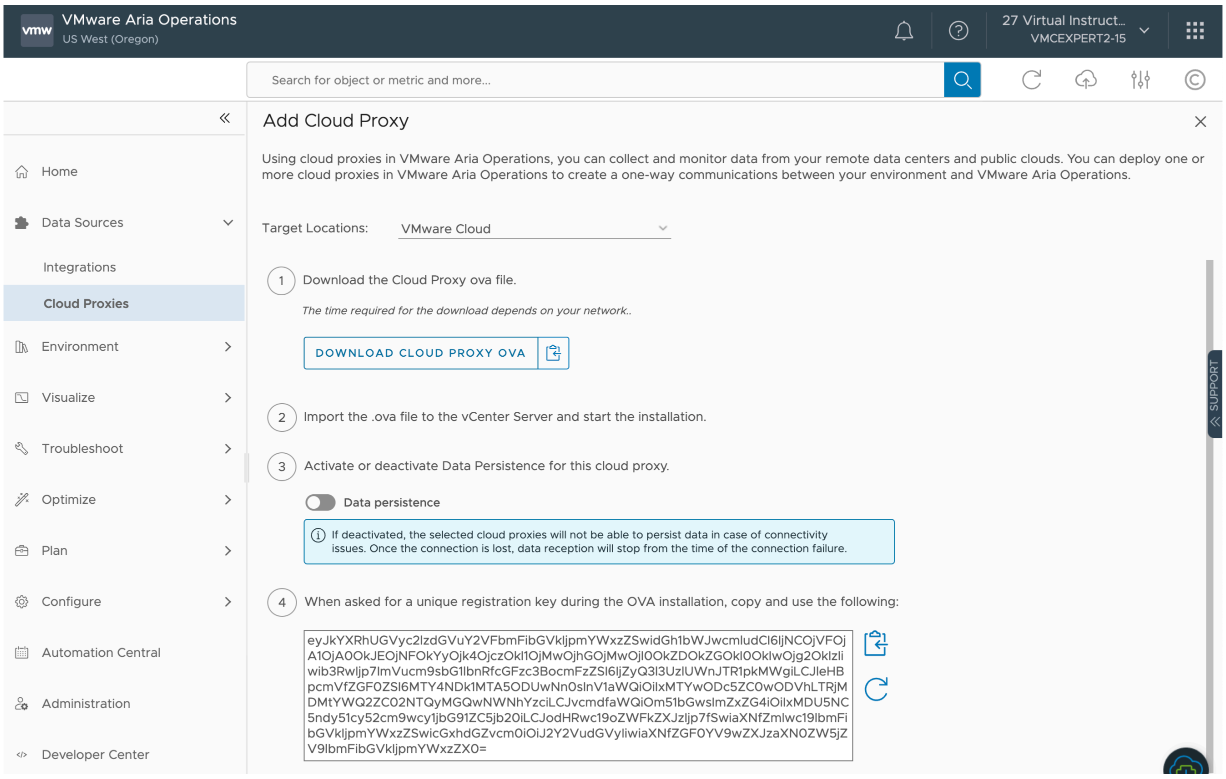
Task 4 - Import an Advanced Cloud Template

Task 4.1 - Deploy Aria Operations Cloud Proxy

1. In another new browser tab access the **Cloud Services Portal**, and login if required https://console.cloud.vmware.com/csp/gateway/portal
2. Bookmark this page for easier future access
3. Click the Aria **Operations** Tile
4. In the Left pane Expand **Data Sources**

5. Click **Cloud Proxies**
6. Click **ADD** to add a new Cloud Proxy
7. Copy the **Unique Registration Key**. We will use it to deploy the Aria Operations Cloud Proxy appliance





Add Cloud Proxy

Using cloud proxies in VMware Aria Operations, you can collect and monitor data from your remote data centers and public clouds. You can deploy one or more cloud proxies in VMware Aria Operations to create a one-way communications between your environment and VMware Aria Operations.

Target Locations: VMware Cloud

1 Download the Cloud Proxy ova file.

The time required for the download depends on your network..

DOWNLOAD CLOUD PROXY OVA

2 Import the .ova file to the vCenter Server and start the installation.

3 Activate or deactivate Data Persistence for this cloud proxy.

Data persistence

If deactivated, the selected cloud proxies will not be able to persist data in case of connectivity issues. Once the connection is lost, data reception will stop from the time of the connection failure.

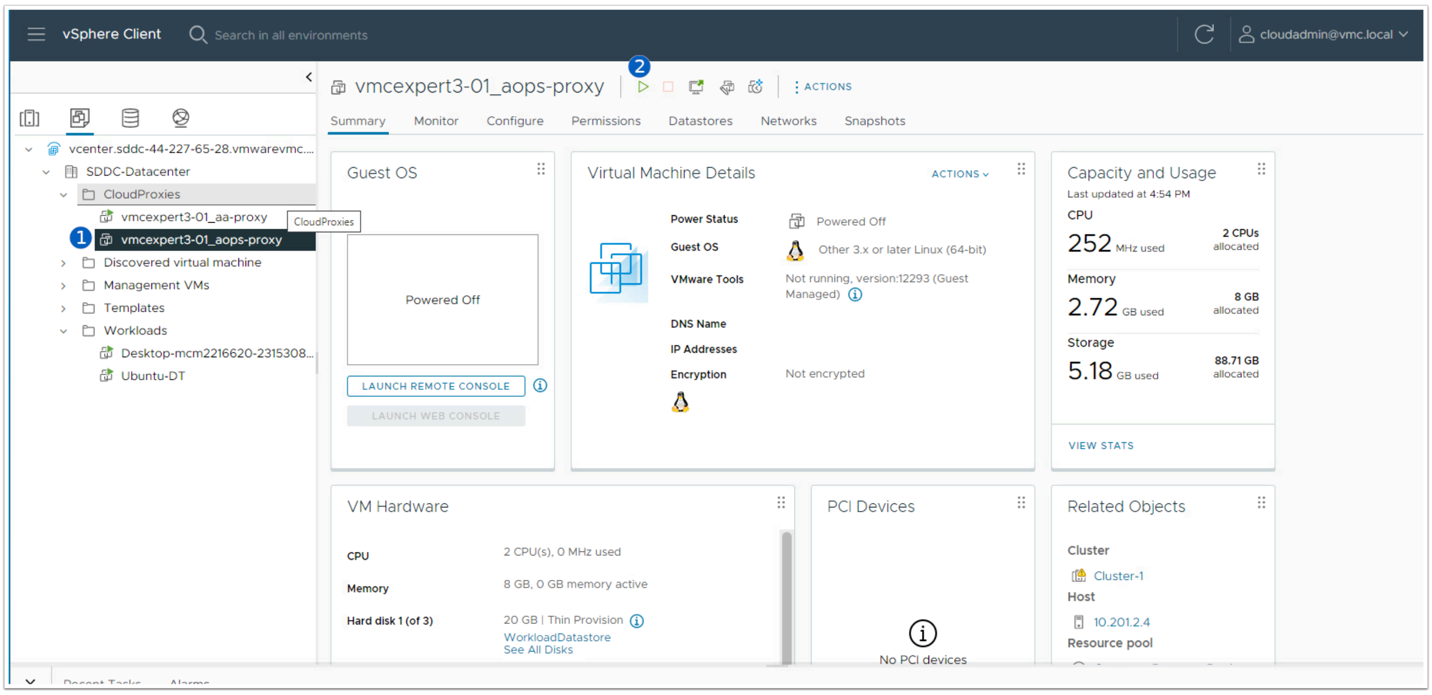
4 When asked for a unique registration key during the OVA installation, copy and use the following:

eyJkYXRhUGVyc2lzdGVuY2VFbmFibGVkljpmYWxzZSwidGh1bWJwcmludCI6IjNCOjVFOj
A1OjA0OkJEOjNFOkYyOjk4OjczOkI1OjMwOjhGOjMwOjI0OkZDOkZGOkI0OkIwOjg2OkIzli
wib3Rwljp7ImVucm9sbG1lbnRfcGFzc3BocmFzZSI6IjZyQ3l3UzlUWnJTR1pkMWgiLCJleHB
pcmVfZGF0ZSI6MTY4NDk1MTA5ODUwNn0sInV1aWQiOiIxMTYwODc5ZC0wODVhLTRjM
DMtYWQ2ZC02NTQyMGQwNWNhYzciLCJvcmdfaWQiOm51bGwsImZxZG4iOiIxMDU5NC
5ndy51cy52cm9wcy1jbG91ZC5jb20iLCJodHRwc19oZWFkZXJzIjp7fSwiaXNfZmlwc19lbmFi
bGVkIjpmYWxzZSwicGxhdGZvcm0iOiJ2Y2VudGVyIiwiaXNfZGF0YV9wZXJzaXN0ZW5jZ
V9lbmFibGVkIjpmYWxzZX0=

8. Access your **vCenter Browser** tab and log in if required (cloudadmin@vmc.local).
9. If the browser tab is closed, open a new tab and launch it from the bookmark

10. In the Host and Clusters Inventory View, right-click **Compute-ResourcePool**
11. Click **New Virtual Machine**
12. Choose **Deploy from template**, Click **Next**
13. Select **Aria Operations**, Click **Next**
14. Name the virtual Machine **{Your_Login_Name}_aops-proxy** (**I.E. vmcexpert3-31_aops-proxy**)
15. Expand **SDDC-Datacenter**, Select **CloudProxies**, and click **Next**
16. Click **Next**, on the Select a Compute Resource page
17. Click **Next o**n the Review details page
18. Check **I accept all license agreements** on the license agreement page and click **Next**
19. Select **Small Cloud Proxy** and Click **Next**
20. Select **WorkloadDatastore** on the select storage page and click **Next**
21. Confirm **sddc-cgw-network-1** network is selected and click **Next** on the Select network page
27. On the Customize template page input the following values:
    - VMware Cloud Service OTK: **Paste in the Unique Registration Key** you copied in step 7 If you no longer have the key repeat steps 2 - 7, where appropriate
    - Friendly Name: **aops-proxy_{your student number} I.E. aops-proxy_3-31**
    - NTP Server List: **0.us.pool.ntp.org,1.us.pool.ntp.org** . Ensure that you did not copy the "." list character from this manual.
    - Leave all other fields blank

28. Click **Next**
29. Click **Finish**
30. Monitor the appliance deployment. Once it completes **Power-On** the VM



Power Status: Powered Off
Guest OS: Other 3.x or later Linux (64-bit)
VMware Tools: Not running, version:12293 (Guest Managed)
DNS Name
IP Addresses
Encryption: Not encrypted

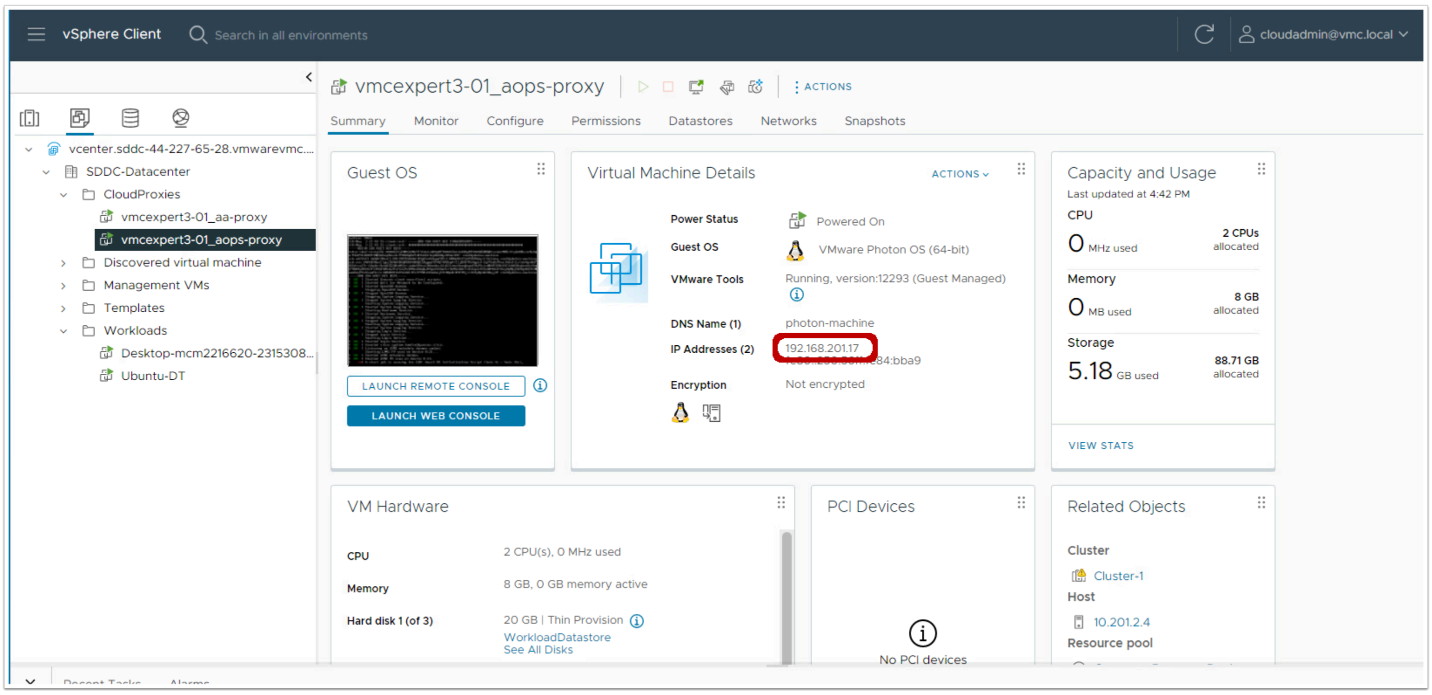
Last updated at 4:54 PM
CPU: 252 MHz used, 2 CPUs allocated
Memory: 2.72 GB used, 8 GB allocated
Storage: 5.18 GB used, 88.71 GB allocated

CPU: 2 CPU(s), 0 MHz used
Memory: 8 GB, 0 GB memory active
Hard disk 1 (of 3): 20 GB | Thin Provision, WorkloadDatastore, See All Disks

No PCI devices

Cluster: Cluster-1
Host: 10.201.2.4
Resource pool

31. Wait for the VM to complete its power-on operation. Copy the VMs IP address, you'll use it later when deploying from the cloud template



Power Status: Powered On
Guest OS: VMware Photon OS (64-bit)
VMware Tools: Running, version:12293 (Guest Managed)
DNS Name (1): photon-machine
IP Addresses (2): 192.168.201.17
Encryption: Not encrypted

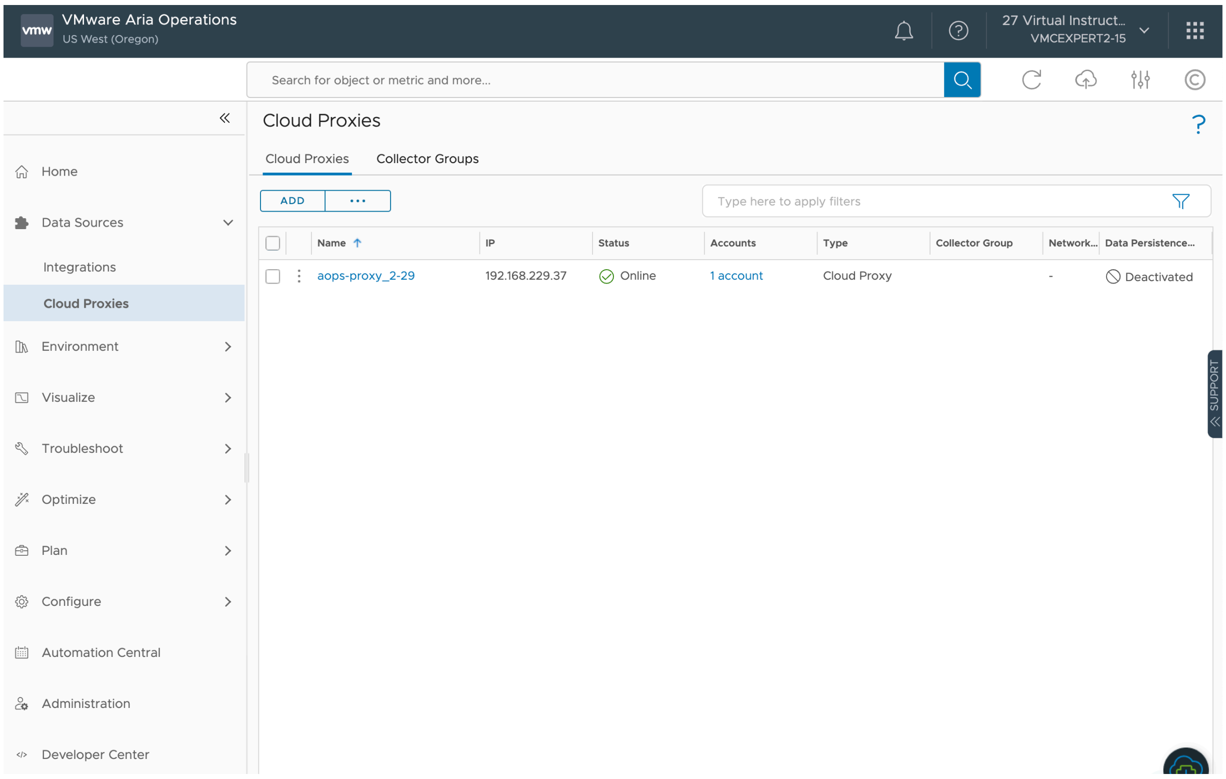
Capacity and Usage
Last updated at 4:42 PM
CPU: 0 MHz used, 2 CPUs allocated
Memory: 0 MB used, 8 GB allocated
Storage: 5.18 GB used, 88.71 GB allocated

VM Hardware
CPU: 2 CPU(s), 0 MHz used
Memory: 8 GB, 0 GB memory active
Hard disk 1 (of 3): 20 GB | Thin Provision
WorkloadDatastore
See All Disks

Related Objects
Cluster: Cluster-1
Host: 10.201.2.4
Resource pool

32. Return to the Aria Automation Operations Cloud browser tab, and refresh the Cloud Proxies page

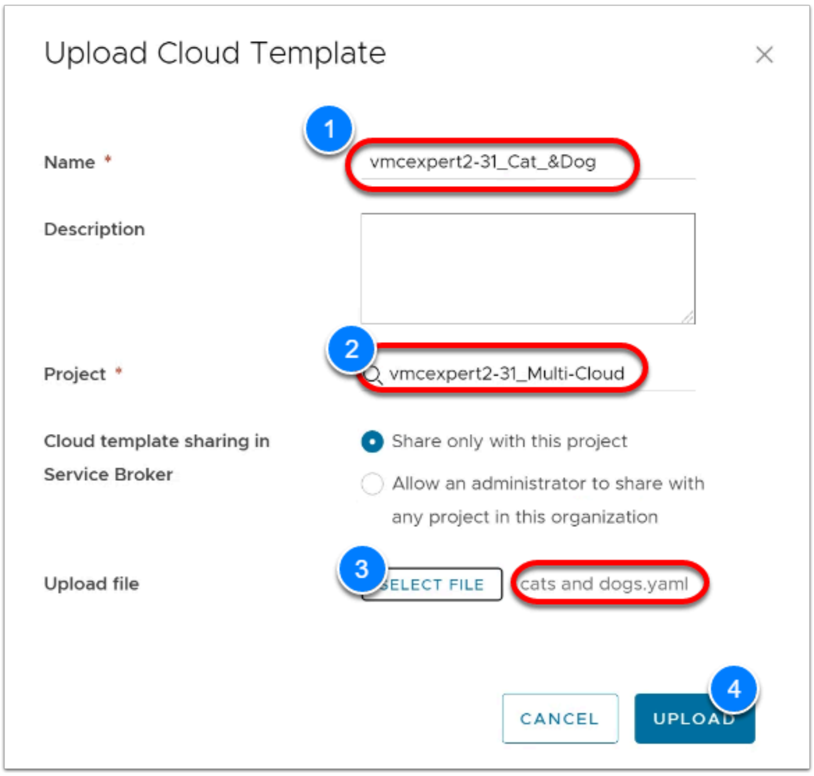
NOTE: the cloud proxy can take up to 15 mins to initialize, register and show up in the console



Name	IP	Status	Accounts	Type	Collector Group	Network...	Data Persistence...
aops-proxy_2-29	192.168.229.37	Online	1 account	Cloud Proxy		-	Deactivated

Task 4.2 - Import and deploy an Advanced Template

1. In the Aria Automation Assembler browser tab Click **Design**
2. Click From New --> Upload
3. Define the Template as follows:
    - Name: {Your_User_Name}_Cat_&_Dog
    - Project: {Select_Your_Project}
    - Upload File: **E:\Lab_Files\Day-2\Cats & Dogs.yml**
4. Click **Upload**


Upload Cloud Template

Name * vmcexpert2-31_Cat_&Dog

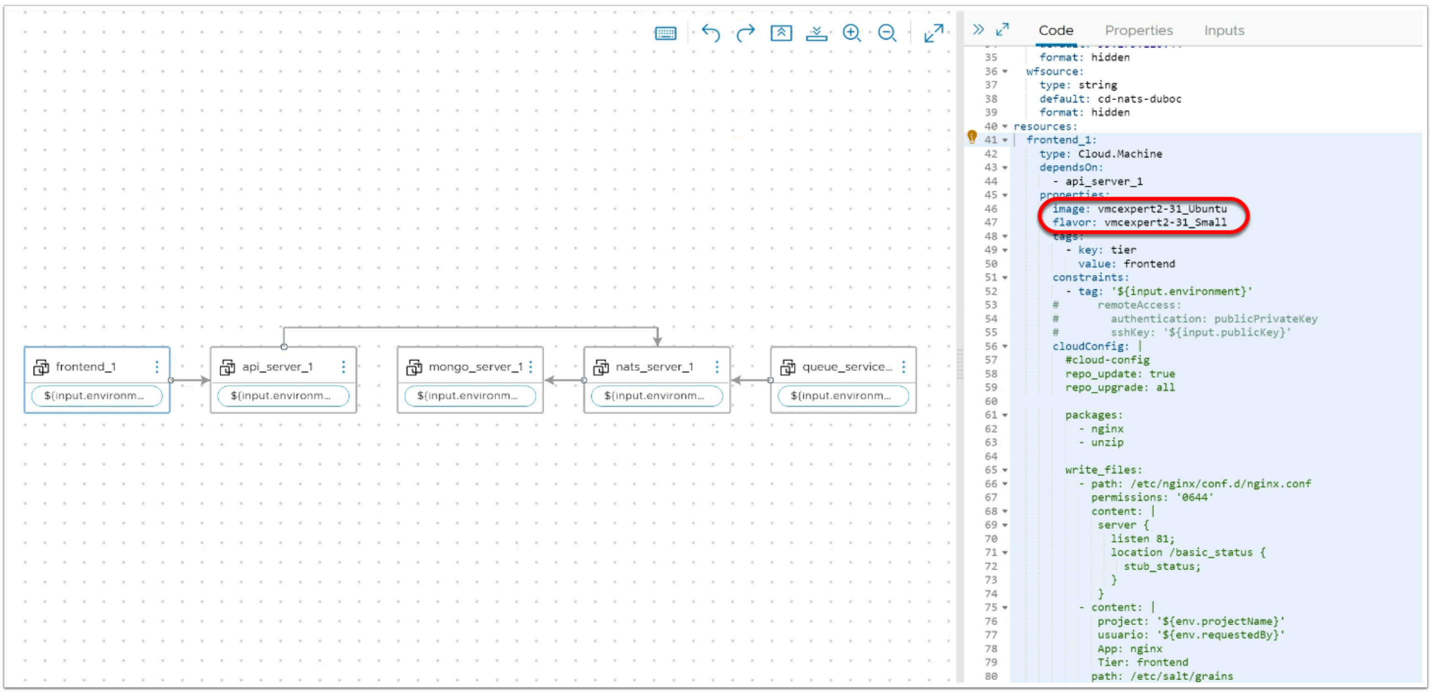
Description

Project * vmcexpert2-31_Multi-Cloud

Cloud template sharing in Service Broker: [✓ Share only with this project] Allow an administrator to share with any project in this organization

Upload file: cats and dogs.yaml

5. Click the **template** to open it.
6. Review the layout of the **template** and the **YAML**
7. Select the mongo_server_1 instance and change the following:
    - image: **{Your_User_Name}_Ubuntu**
    - flavor: **{Your_User_Name}_Small**

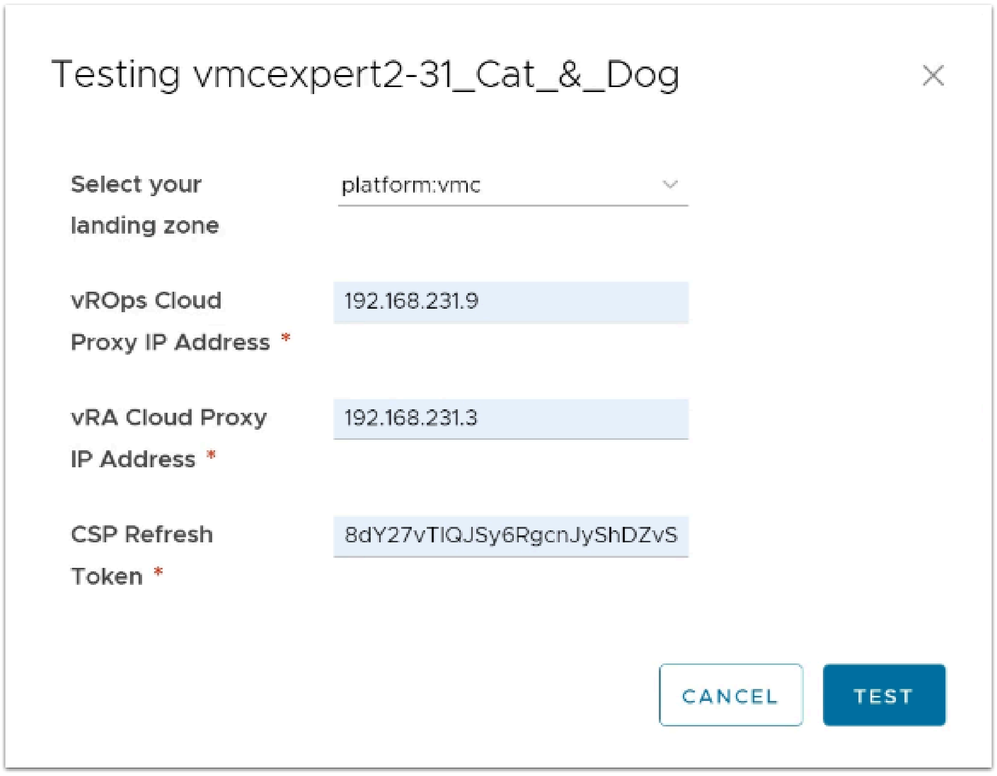


```
    format: hidden
  wfsource:
    type: string
    default: cd-nats-duboc
    format: hidden
resources:
  frontend_1:
    type: Cloud.Machine
    dependsOn:
      - api_server_1
    properties:
      image: vmcexpert2-31_Ubuntu
      flavor: vmcexpert2-31_Small
      tags:
        - key: tier
          value: frontend
      constraints:
        - tag: '${input.environment}'
      #      remoteAccess:
      #        authentication: publicPrivateKey
      #        sshKey: '${input.publicKey}'
      cloudConfig: |
        #cloud-config
        repo_update: true
        repo_upgrade: all

        packages:
          - nginx
          - unzip

        write_files:
          - path: /etc/nginx/conf.d/nginx.conf
            permissions: '0644'
            content: |
             server {
               listen 81;
               location /basic_status {
                 stub_status;
               }
             }
          - content: |
             project: '${env.projectName}'
             usuario: '${env.requestedBy}'
             App: nginx
             Tier: frontend
            path: /etc/salt/grains
```

8. Click **Test** to validate the template
9. Enter the following inputs:
   - Select your landing zone: **platform:vmc**
   - vRops Cloud Proxy IP Address: **{Your_vROps_Proxy_IP}** **i.e. 192.168.X.Y**
   - vRA Cloud Proxy IP Address: **{Your_vRA_Proxy_IP}** **i.e. 192.168.X.Y**
   - CSP Refresh Token: **{Your_API_Token}**

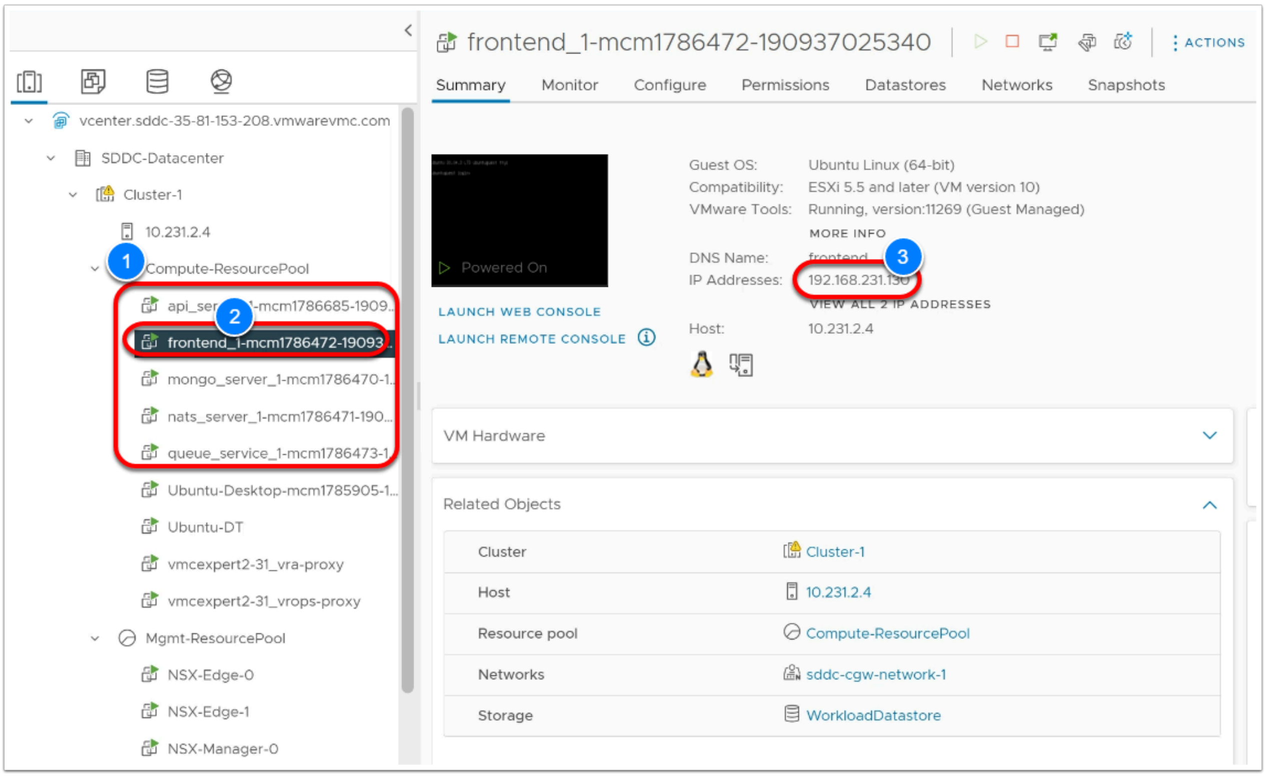


Testing vmcexpert2-31_Cat_&_Dog

Select your landing zone	platform:vmc
vROps Cloud Proxy IP Address *	192.168.231.9
vRA Cloud Proxy IP Address *	192.168.231.3
CSP Refresh Token *	8dY27vTIQJSy6RgcnJyShDZvS

10. Click **Test**

11. Once the test succeeds, Click **Deploy** to deploy the template
12. Name the Deployment **{Your_User_Name}_Cats_&_Dogs_VMC**
13. Click **Next**
14. Enter the same inputs form step 9
15. Click **Deploy**
16. Monitor the deployment and once it completes successfully, return to the vCenter browser tab and review the machines that were deployed.
17. Click the **frontend VM** and record its **IP address**



Guest OS: Ubuntu Linux (64-bit)
Compatibility: ESXi 5.5 and later (VM version 10)
VMware Tools: Running, version:11269 (Guest Managed)
DNS Name: frontend
IP Addresses: 192.168.231.130
Host: 10.231.2.4

Cluster	Cluster-1
Host	10.231.2.4
Resource pool	Compute-ResourcePool
Networks	sddc-cgw-network-1
Storage	WorkloadDatastore

18. Select the **Ubuntu-DT** VM and click **Open Console**
19. Enter the Password of **VMware1!** if Prompted
20. Launch the Firefox browser in the Ubunt-DT VM and type in **{the address of your Frontend VM}** for the Cats & Dogs Application
21. Click the **Gato & Cachorro** buttons multiple times until the images that appears is a hedgehog.

Conclusion