TECHNICAL GUIDE

CloudBolt software | ONEFUSE

Using OneFuse with vRealize Automation 8

CloudBolt software | OneFuse

Table of Contents

TECHNICAL GUIDE

Using OneFuse with vRA 8

Introduction

As companies continue to make the shift towards digital transformation and more cloud infrastructure, teams are under mounting pressure to find efficient and cost-effective solutions to address automation challenges. A growing number of clouds, toolsets, and teams make this increasingly complicated.

"Poor integrations... **take a $500,000 toll** on the business every year."[1]	"52% of custom coded projects cost **189% of their original estimate**."[2]	"57% of **IT processes aren't automated** or integrated."[3]

VMware vRealize Automation (vRA) is a popular choice for many organizations to manage their hybrid cloud infrastructure, but it has some gaps. CloudBolt OneFuse provides the ability to extend platform capabilities as well as reduce the need to write custom code to achieve advanced use cases with integrated technologies. With the OneFuse, organizations can codelessly integrate vRA with other automation and DevOps tools to accelerate automation capabilities and realize a faster time-to-value.

Capabilities	OneFuse	vRA 7	vRA 8
Custom Code Reduction	✓	✗	✗
Integrations Dashboard	✓	✗	✗
Multi-tool Consumption	✓	✗	✗
Auditing	✓	✗	✗
Policy/Template	✓	✗	✗
Migration/Portability	✓	✗	✗
Deep and Wide Integrations	✓	✗	✗

We Make VMware vRA Better...

- Stop building integrations again and again, OneFuse creates reusable integrations
- Gain deeper system integrations, see results and analyze resource flow from integrated systems
- Gain governance around integrations – build policies to dictate usage
- OneFuse can decouple integration services, making integrations easier and simpler to move between platforms

The remainder of this document focuses on how to install and configure OneFuse to complement vRA and fill the gaps to provide a more complete and enterprise ready solution!

Chapter 1: Installing the OneFuse Workflow Package

Here's how easy it is to install the OneFuse workflow package into vRA8. Once the workflow package is installed you will be able to take advantage of all that OneFuse has to offer within vRA8.

*Note that this same workflow package will also work for vRA7 and vRA Cloud.

Once the OneFuse vRA workflow package is installed and configured, it will automatically create the needed vRA Event Broker Subscriptions for each of the available integrations.

Before we begin, there are prerequisites you will want to have ready.

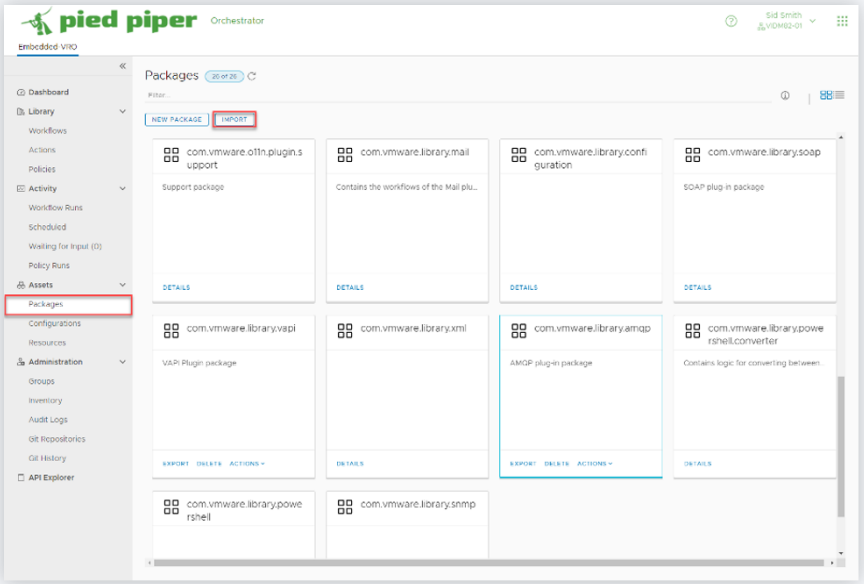
Prerequisites

- Download the appropriate package for the version of OneFuse you are running from.
- The OneFuse appliance should be deployed and configured, see the following articles if you need to walk through the OneFuse deployment and configuration.
  - Deploying the OneFuse Appliance
  - Configuring the OneFuse Appliance
  - Creating a OneFuse Naming Policy
- vRA8 or vRA8 Cloud setup with access to vRO.

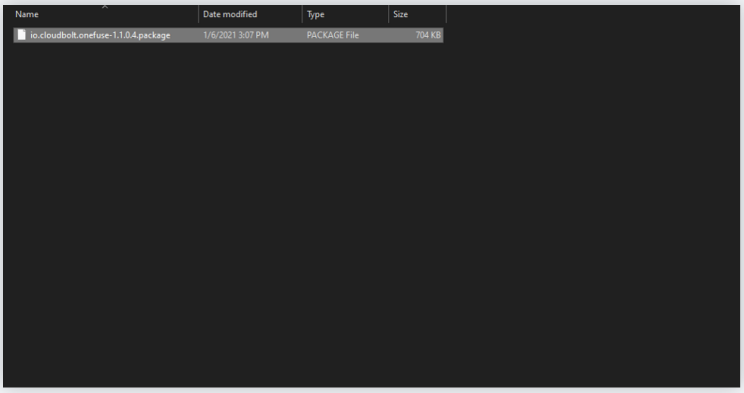
Importing the OneFuse Workflow Package in vRO

First, we need to import the workflow package that was obtained as part of the prerequisites. To import the package follow the steps below:

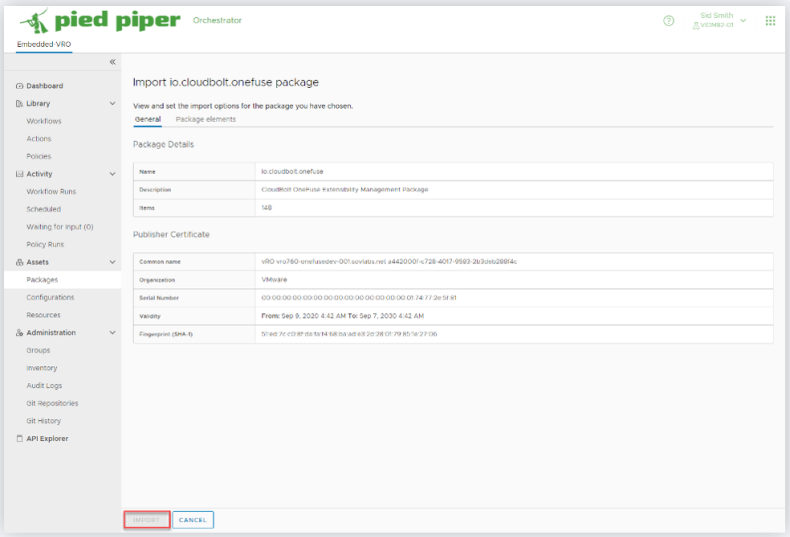
1. From the vRO web based UI navigate to Packages and select "Import."



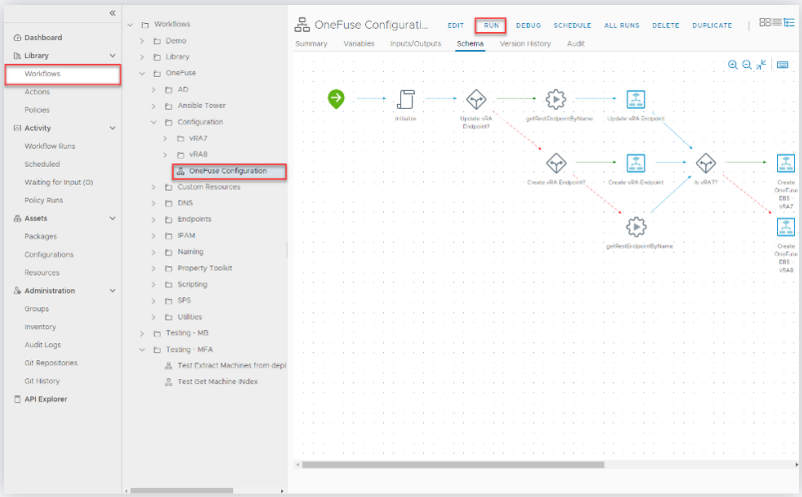
2. Once you have selected import choose the .package file you previously downloaded and select “Open.”



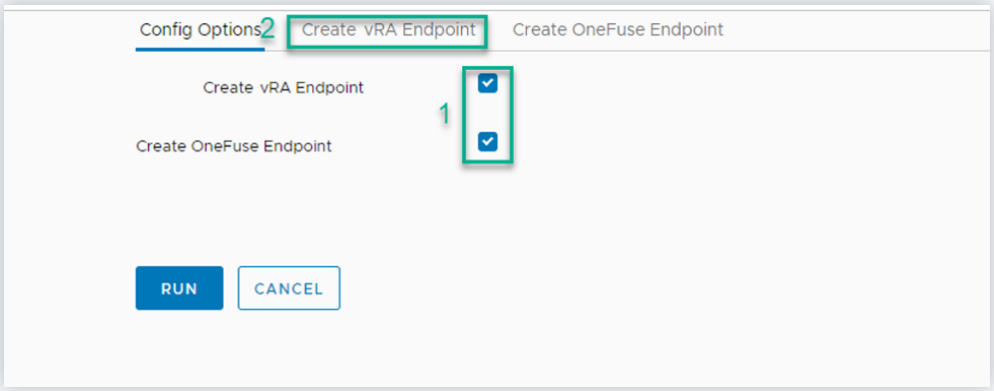
3. After opening the package select “Import.”



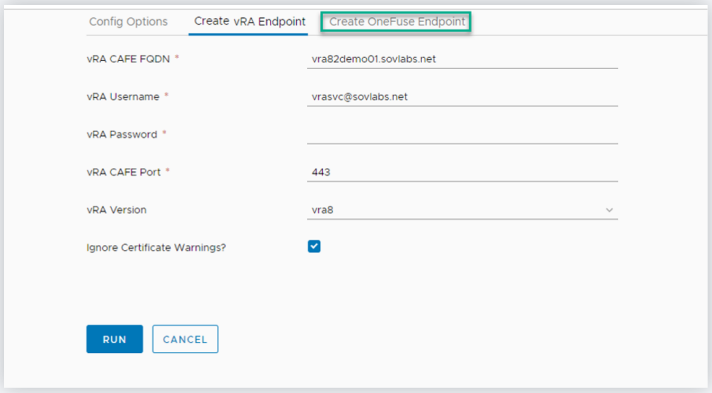
4. Once the Import is complete navigate to “Workflows” and from tree view navigate to OneFuse -> Configuration -> and select “OneFuse Configuration” workflow and then select “RUN.”



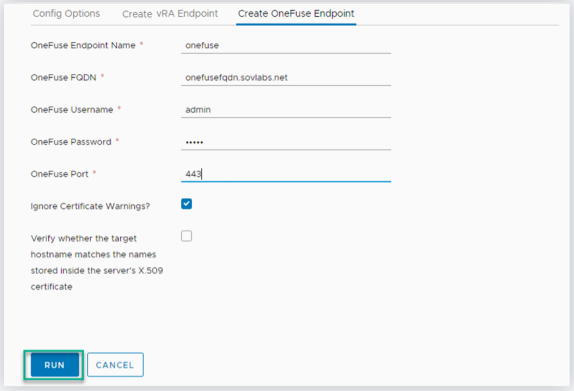
5. When the dialog appears check “Create vRA Endpoint” and “Create OneFuse Endpoint” and then select the “Create vRA Endpoint” page.



6. One the “Create vRA Endpoint” page, enter the FQDN for the vRA appliance, a username, password, port, version, and select “Ignore Certificate Warnings” once completed select the “Create OneFuse Endpoint” page from the top.



7. Create a name for the Endpoint. Keep it simple as this will be used when consuming the integration from within vRA. I would recommend calling it *onefuse*, or *onefusedev*, *onefuseprod* or something similar. Input the FQDN for the appliance, a username, password, the port, and select “Ignore Certificate Warnings”. Select “RUN once completed.



Once the workflow run is complete the OneFuse vRA package is installed, configured, and ready for use.

Chapter 2: Custom Naming Configuration

Now, we're going to walk through using OneFuse to name a deployed machine in vRA8. To do this we will create a new blueprint within vRA8 and call upon the OneFuse to name it using the naming policy we created as part of "Creating a OneFuse Naming Policy."

By the end of this chapter we will have created a blueprint that will deploy a vSphere machine that is named using the OneFuse Naming module. While this will be a simple example we will build upon this in later chapters to showcase the advanced capabilities offered by OneFuse as a platform.

vRA8 with OneFuse: Naming

Before we begin there are prerequisites you will want to have ready.

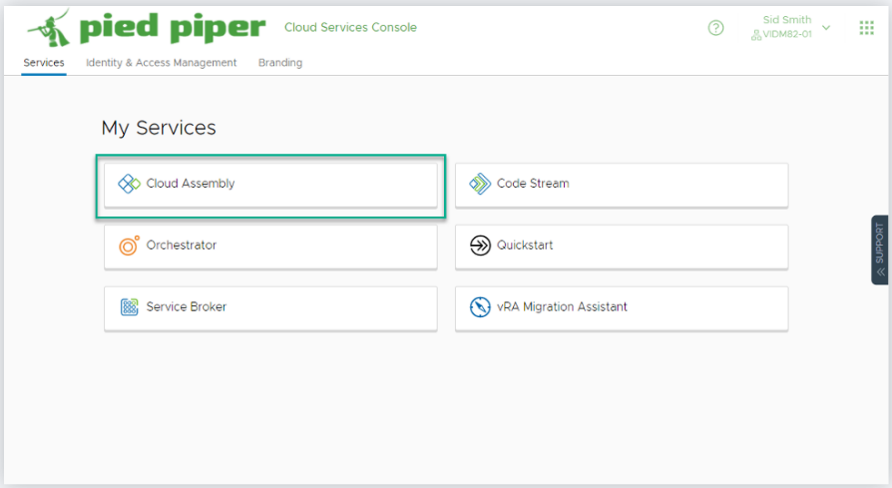
Prerequisites

- The OneFuse appliance should be deployed and configured, see the following articles if you need to walk through the OneFuse deployment and configuration.
  - Deploying the OneFuse Appliance
  - Configuring the OneFuse Appliance
  - Creating a OneFuse Naming Policy
- The OneFuse Workflow package needs to be installed and configured within vRA8. The following article can walk you through is you have not completed this yet.
  - Installing the OneFuse Workflow Package into vRA8
- vRA installed and a working vSphere Blueprint.

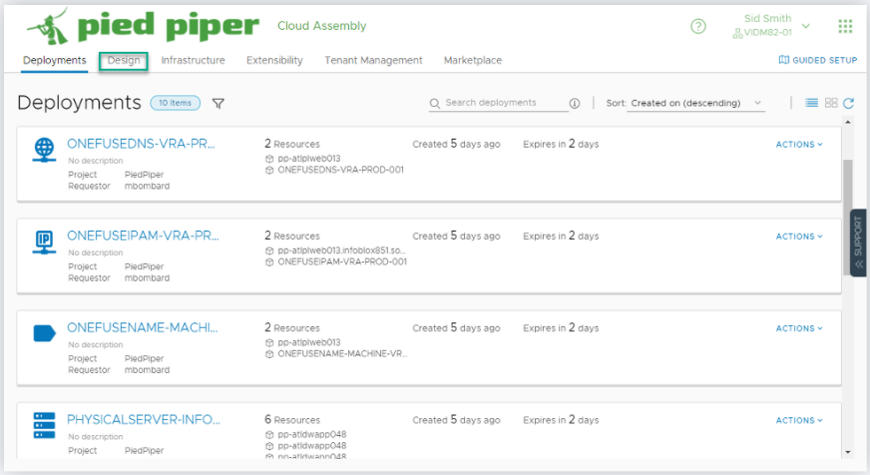
Building the Blueprint

Note: The following Blueprint "OneFuse_Naming_vSphere" example can be found in the following git repo: https://github.com/CloudBoltSoftware/onefuse-examples

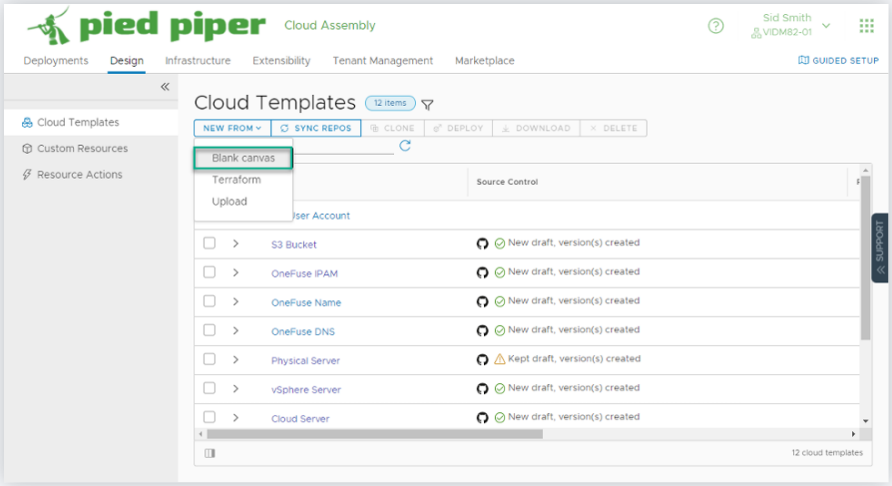
1. To begin, login to vRA8 and launch "Cloud Assembly."



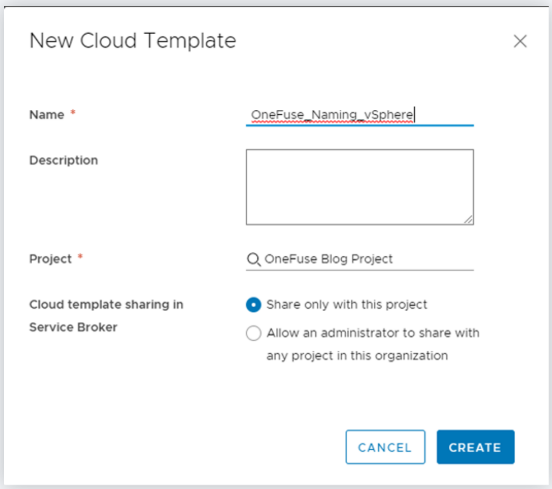
2. Once Cloud Assembly has launched select “Design” from the menu.



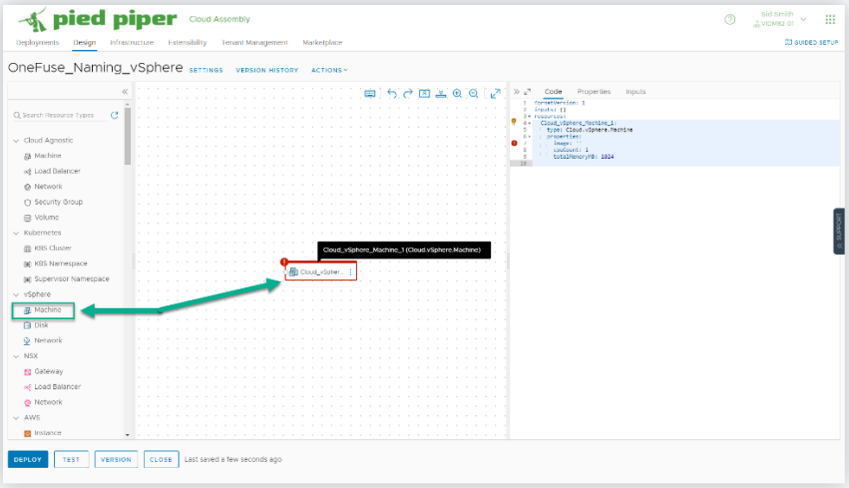
3. Next, select “Blank canvas” for the “New From” menu.



4. Give your new blueprint a name, description, and select the project you would like it associated with and select “Create”.

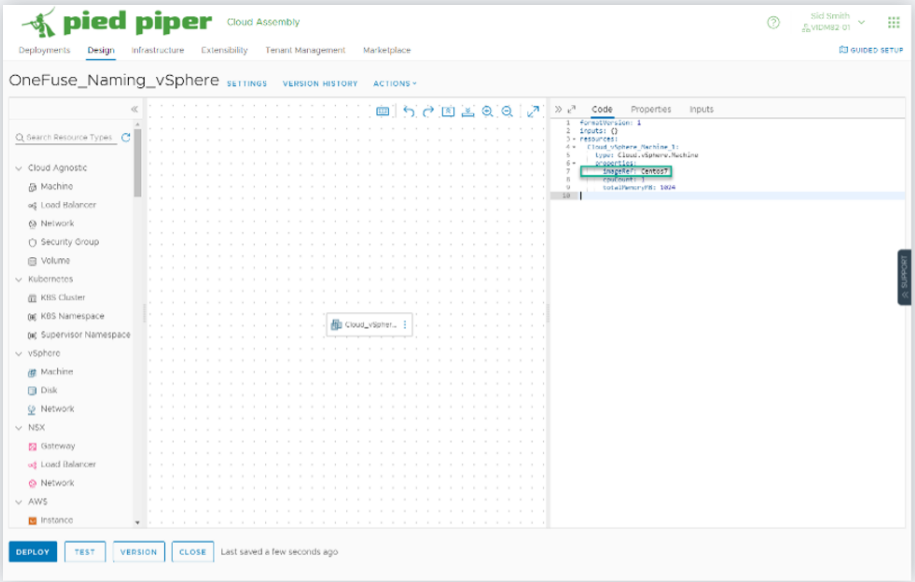


5. Next, drag a vSphere Machine object onto the blank canvas.



6. Next, we need to assign a vSphere template to the vSphere machine. I'm using the ImageRef property and my template is called Centos7. You will want to replace Centos7 with the name of your template. Below is the Yaml for my vSphere Machine:

```
formatVersion: 1
 inputs: {}
 resources:
  Cloud_vSphere_Machine_1:
   type: Cloud.vSphere.Machine
   properties:
    #vRA Properties
    imageRef: Centos7
    cpuCount: 1
    totalMemoryMB: 1024
```



7. Next, we need to add the OneFuse Naming Policy property to the blueprint. The OneFuse Property/Value has certain parameters that need to be set as below:

OneFuse_NamingPolicy:
'OneFuseEndpointName:PolicyName'

OneFuseEndpointName is the name you gave the OneFuse endpoint when you created it and **PolicyName** is the Naming Policy within OneFuse that you would like to assign.

```
#OneFuse Module Properties
OneFuse_NamingPolicy: 'onefuseblog:default'
```

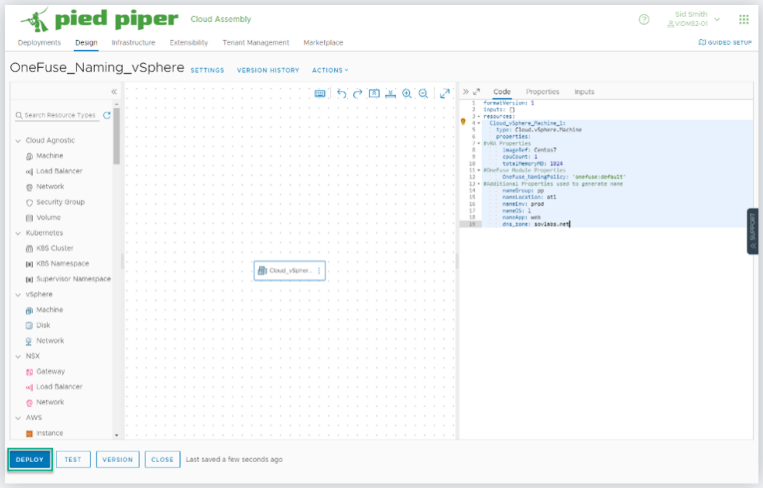
8. The last piece we need to account for is defining additional properties that will be sent to OneFuse and used to create the name. In the article "Configuring the OneFuse Appliance", we created a Static Property Set with simulated inputs. I'm going to use the same properties and values in my example blueprint.

```
#Additional Properties used to generate name
nameGroup: pp
nameLocation: atl
nameEnv: prod
nameOS: l
nameApp: web
dns_suffix: sovlabs.net
```

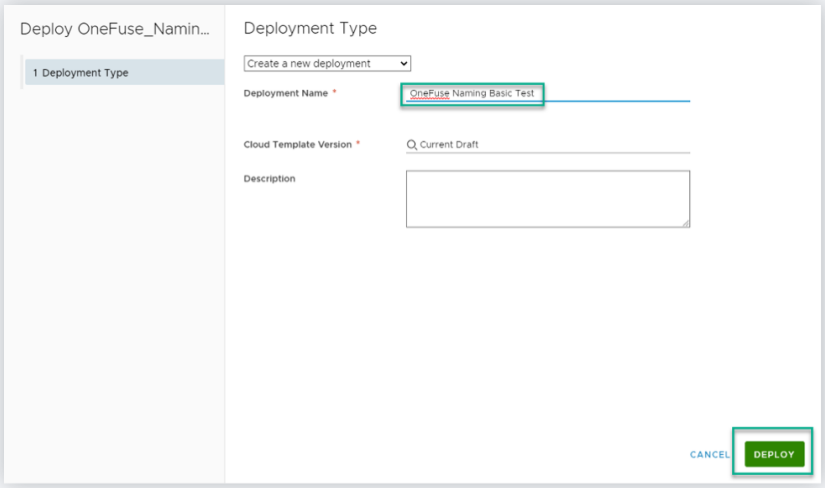
9. The completed YAML for my blueprints is:

```
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      #vRA Properties
      imageRef: Centos7
      cpuCount: 1
      totalMemoryMB: 1024
      #OneFuse Module Properties
      OneFuse_NamingPolicy: 'onefuseblog:default'
      #Additional Properties used to generate name
      nameGroup: pp
      nameLocation: atl
      nameEnv: prod
      nameOS: l
      nameApp: web
      dns_suffix: sovlabs.net
```

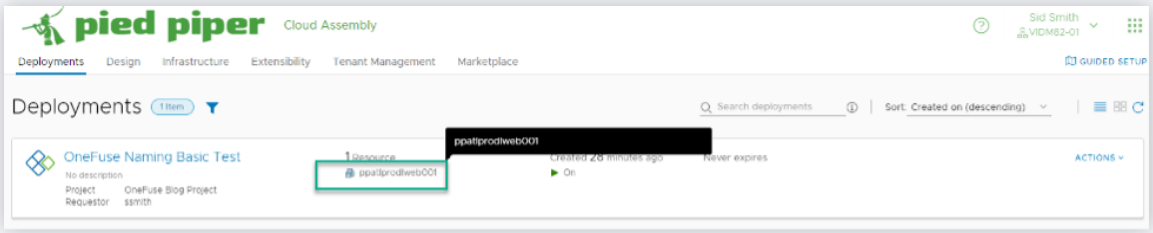
10. Once your Blueprint is complete let's deploy it and give it a test by selecting the "Deploy" button at the bottom left.



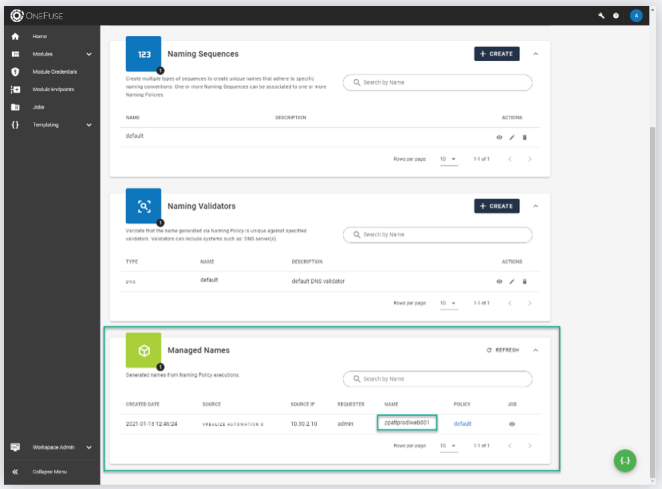
11. Assign a deployment name and select "Deploy."



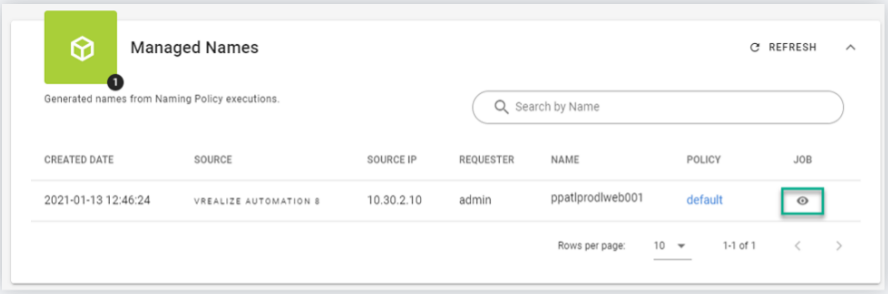
12. Navigate to the Deployments menu and you will see your deployment listed and once completed you will notice the name has been set by the OneFuse Naming module.

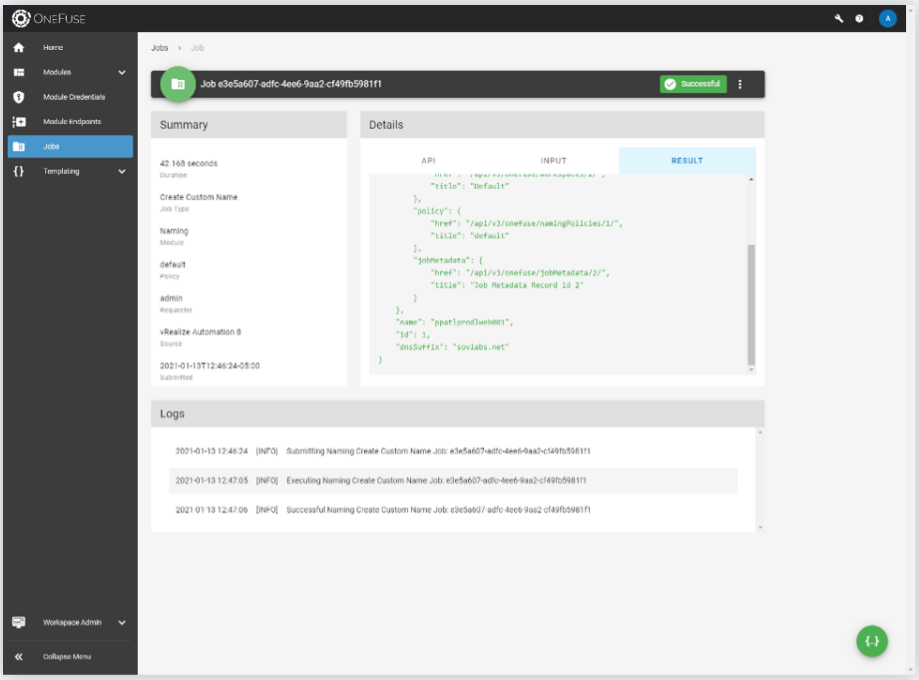


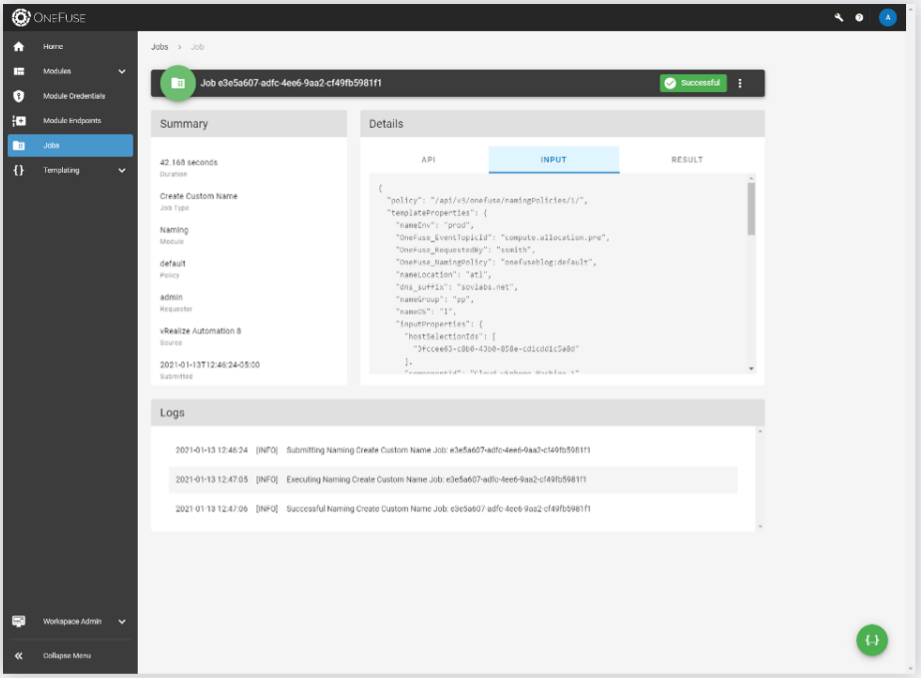
13. Now let's login to the OneFuse appliance and select Naming from the homepage. Once on the Naming page if you scroll all the way down you will see "Managed Names" and you will see your newly created name listed.



14. If you select the “Eye” under Job you can get more information on the job that created the name.







Although this is a very simple example of connecting OneFuse to vRA8 for Custom Naming, you can start to see how things tie together.

Chapter 3: IPAM Configuration

Let's create a vRA8 blueprint that utilizes the OneFuse IPAM module to provide IPAM integration for provisioned workloads. We won't just be supplying IP information to vRA8 in this example, OneFuse will determine the network placement as well.

By the end of this chapter we will have added the OneFuse IPAM policy to a vRA8 blueprint that is deploying a vSphere machine. This will be a simple example that we will build upon in future articles.

vRA8 with OneFuse: IPAM Integration

Before we begin there are prerequisites you will want to have ready.

Prerequisites

- The OneFuse appliance should be deployed and configured, see the following articles if you need to walk through the OneFuse deployment and configuration
  - Deploying the OneFuse Appliance
  - Configuring the OneFuse Appliance
  - Creating an IPAM Policy with OneFuse
- The OneFuse Workflow package needs to be installed and configured within vRA8. The following article can walk you through is you have not completed this yet.
  - Installing the OneFuse workflow package into vRA8
- vRA installed and a working vSphere Blueprint.

While not required to follow along I will be starting off using a blueprint I had previously created in my article vRA8 custom naming with OneFuse. If you want to follow along from where we left off in that article you will want to read the below two articles before continuing.

- Creating a Naming Policy with OneFuse
- vRA8 Custom Naming with OneFuse

Adding IPAM to a vRA8 Blueprint

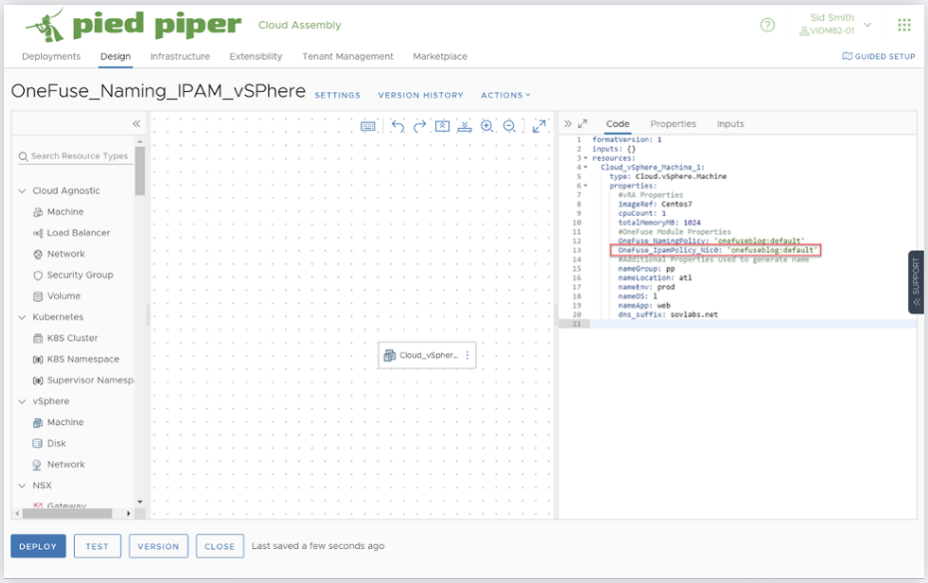
I am going to use the blueprint that I walked through creating in the previous article, vRA8 Custom Naming with OneFuse. You can do the same, clone the existing blueprint or use one you already have available if you like.

To begin you will need to open the blueprint for editing so we can add the appropriate items for consuming our IPAM policy that we created. If you have not yet created your IPAM Policy you can learn how by reading: Creating an IPAM Policy with OneFuse.

1. We need to add the following property to our blueprint
   - *OneFuse_IpamPolicy_Nic0*
   - The property supports 10 NICs. You can set the NIC by changing the number at the end from 0-9.

2. Next we need to set the value for the property. The value is broken up into 2 parts separated by a colon as outlined below.

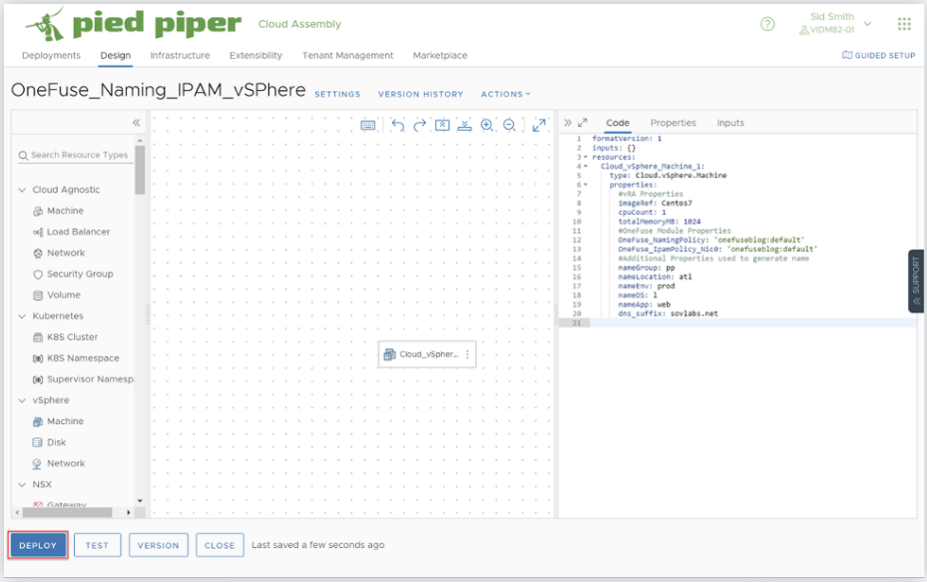
- *OneFuse_Endpoint:Policy_Name*
- For my environment this will be *onefuseblog:default*. This will result in the following being added to the blueprint:
- *OneFuse_IpamPolicy_Nic0: 'onefuseblog:default'*



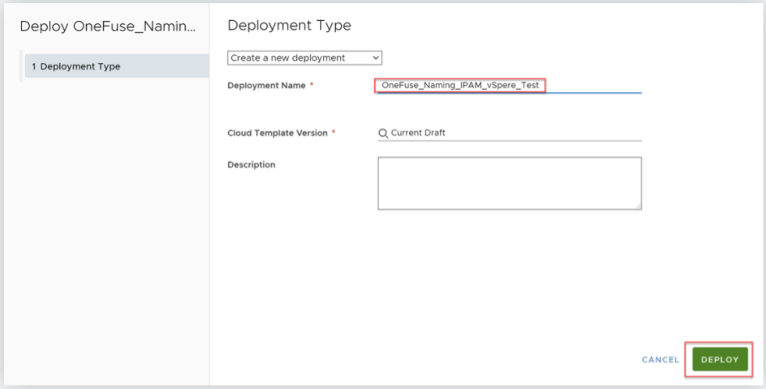
3. The completed yaml code for my blueprint looks like the following:

```
formatVersion: 1
 inputs: {}
 resources:
   Cloud_vSphere_Machine_1:
     type: Cloud.vSphere.Machine
     properties:
       #vRA Properties
       imageRef: Centos7
       cpuCount: 1
       totalMemoryMB: 1024
       #OneFuse Module Properties
       OneFuse_NamingPolicy: 'onefuseblog:default'
       OneFuse_IpamPolicy_Nic0: 'onefuseblog:default'
       #Additional Properties used to generate name
       nameGroup: pp
       nameLocation: atl
       nameEnv: prod
       nameOS: l
       nameApp: web
       dns_suffix: sovlabs.net
```

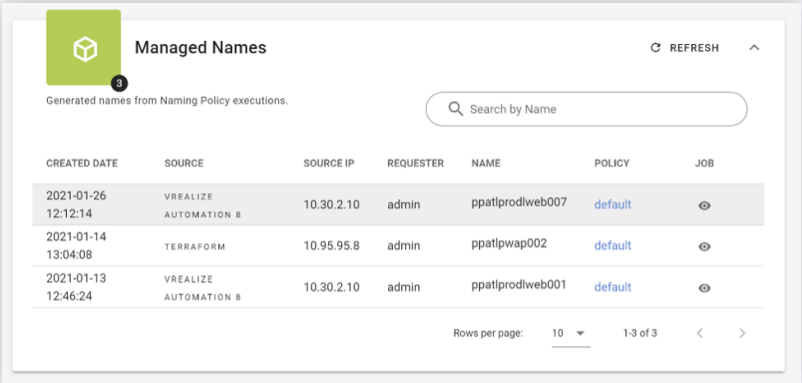
4. Once the blueprint is completed we will do a deploy to test that our IPAM integration is working as expected. Select “Deploy” in the lower left corner.



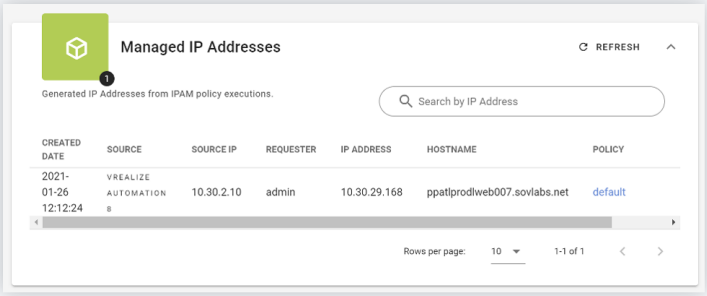
5. When the deployment dialog opens give your deployment a name and select “Deploy”.



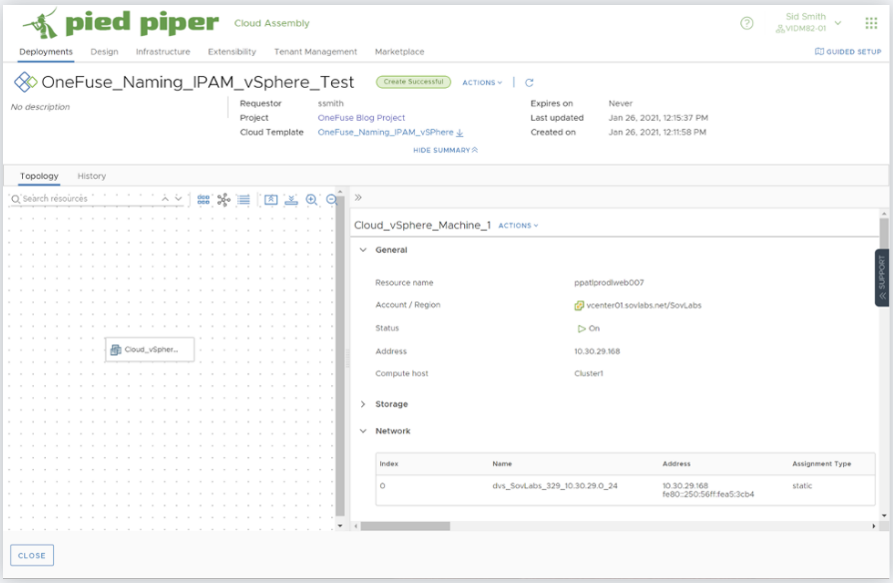
6. Within a few minutes of starting the deployment a name will have been generated and an IP address will have been reserved. To view the name as well as the IP reservation you can login to the OneFuse UI and view the managed objects. You will also b able to see this information within vRA as well. To view the managed objects within OneFuse login to the OneFuse UI and first select naming then scroll down to the section labeled “Managed Names”.



You can then select IPAM from the left menu and under “Managed IP Addresses” you will see the information for your IP Address reservation.



7. Within vRA8 you can view the details of your deployment and view the name and IP Address. If you expand Network you will see the network the workload was assigned to and other details.



8. If you would like to view more detail the entire output from the OneFuse request is also assigned under properties.

- Below is a sample of the information stored with the output:

{"_links":{"self":{"href":"/api/v3/onefuse/ipamReservations/3/","title":"10.30.29.168"},"workspace":{"href":"/api/v3/onefuse/workspaces/2/","title":"Default"},"policy":{"href":"/api/v3/onefuse/ipamPolicies/1/","title":"default"},"jobMetadata":{"href":"/api/v3/onefuse/jobMetadata/27/","title":"Job Metadata Record id 27"}},"id":3,"ipAddress":"10.30.29.168","hostname":"ppatlprodlweb007.sovlabs.net","primaryDns":"10.30.0.11","secondaryDns":"10.30.0.12","dnsSuffix":"sovlabs.net","dnsSearchSuffixes":"infoblox851.sovlabs.net,sovlabs.net","nicLabel":null,"subnet":"10.30.29.0/24","gateway":"10.30.29.1","network":"dvs_SovLabs_329_10.30.29.0_24","netmask":"255.255.255.0","trackingId":"64080c00-db0c-4af5-9c42-afde95b81793","endpoint":"onefuseblog"}

We have now successfully added and tested IPAM within vRA8. While this is a basic example of consuming a static IPAM Policy, it is possible to drive the IPAM integration more flexibly.

Chapter 4: DNS Configuration

We're going to add OneFuse DNS support to a vRA8 blueprint. If you've been following previous chapters you probably have an idea of how this is going to work. We're going to build upon the examples from previous chapters by leveraging the same blueprint that we created in the article "vRA8 with OneFuse: IPAM integration."

By the end of this chapter, we will have a blueprint that leverages OneFuse to generate a name, assign Network/IP Address as well as create DNS records for the deployed machine. Although these examples are simple and static, they are setting the foundation for future examples where we will dive into creating more flexible and dynamic blueprints.

vRA8 with OneFuse: DNS Integration

Prerequisites

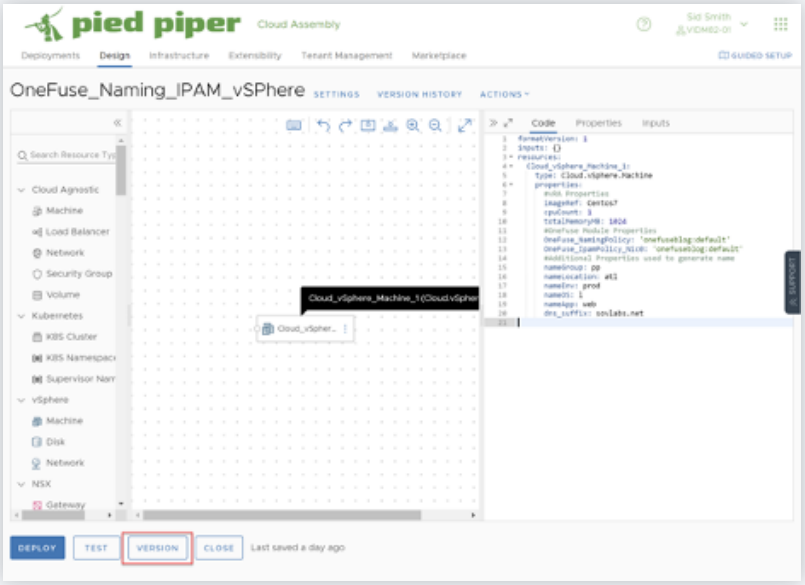
- The OneFuse appliance should be deployed and configured, see the following articles if you need to walk through the OneFuse deployment and configuration.
    - Deploying the OneFuse Appliance
    - Configuring the OneFuse Appliance
    - Creating a OneFuse DNS Policy
- The OneFuse Workflow package needs to be installed and configured within vRA8. The following article can walk you through is you have not completed this yet.
    - Installing the OneFuse Workflow Package into vRA8
- vRA installed and a working vSphere Blueprint

Adding DNS to a vRA8 Blueprint

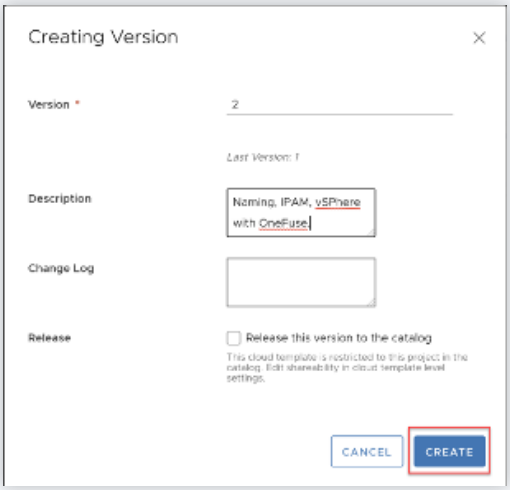
We'll use the blueprint that we walked through creating in the previous chapter, vRA8 IPAM Integration with OneFuse. In the previous chapter we cloned a blueprint that was previously used for my naming article. In this example we won't clone the blueprint but utilize the version control that exists within vRA8.

To begin you will need to open the blueprint for editing so we can add the appropriate items for consuming our DNS policy that we created. If you have not yet created your IPAM Policy, you can learn how by reading: Creating a OneFuse DNS Policy.

Before we go ahead and add the needed configuration to our blueprint, we are going to create a version so we can make a record of its current state. To do this we need to open the blueprint and select "version" from the lower menu.



Once the version dialog opens give it a description and optionally change log information and select "Create".

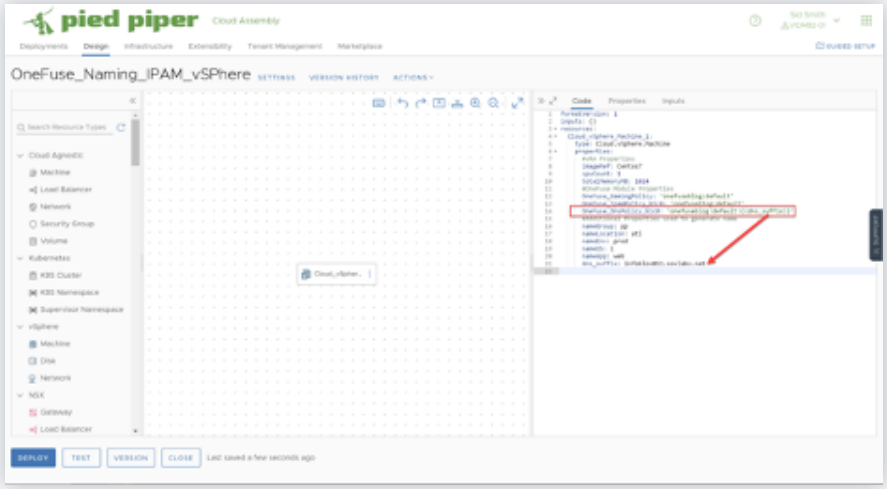


Now that we have a tracked version of the current state we can go ahead and add the properties needed to integrate DNS into our blueprint.

1. We need to add the following property to our blueprint:
   - *OneFuse_DnsPolicy_Nic0*
     - The property supports 10 NICs.
     - You can set the NIC by changing the number at the end from 0-9.
2. Next, we need to set the value for the property. The value is broken up into 3 parts separated by a colon as outlined below.
   0. *OneFuse_Endpoint:Policy_Name:DNS_Suffix*
   1. For my environment this will be *onefuseblog:default:{{dns_suffix}}*

   This will result in the following being added to the blueprint:
   - *OneFuse_DnsPolicy_Nic0:*
     'onefuseblog:default:{{dns_suffix}}'

2.



3. You will notice the {{dns_suffix}} is getting it's value from a property that already exists and has been used in the previous examples for naming and IPAM.

- It's important to note that this is not vA8 Blueprint expression syntax, but it is OneFuse Jinja 2 template syntax.
- It's also important to note that you can use blueprint expression syntax or a combination of both blueprint expression and Jinja 2 template syntax with OneFuse properties. There are reasons you may choose to utilize one over the other, however that is a conversation for another article.

4. For the purposes of this article, it is convenient to use the same property that has provided the DNS suffix to the Naming and IPAM modules, however you may not always want to do this. You also may want to ensure you are using the exact suffix that was leveraged for IPAM and you can. You can specify to use the DNS Suffix from the IPAM configuration by using the following value:

5. *{{OneFuse_Ipam_Nic0.dnsSuffix}}*

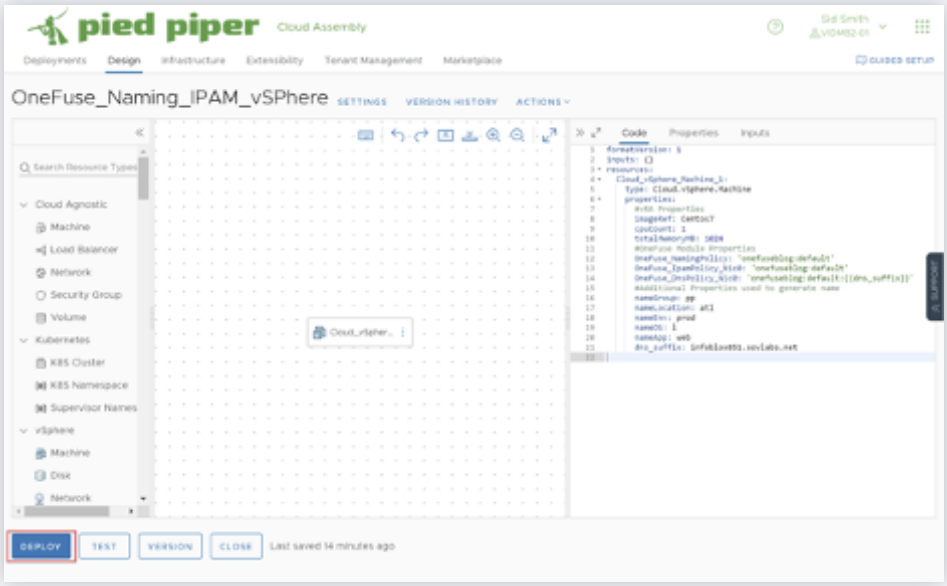
6. This tells OneFuse to use the DNS Suffix associated with the assigned Nic0 value.

3. Below is the blueprint yaml for this example:

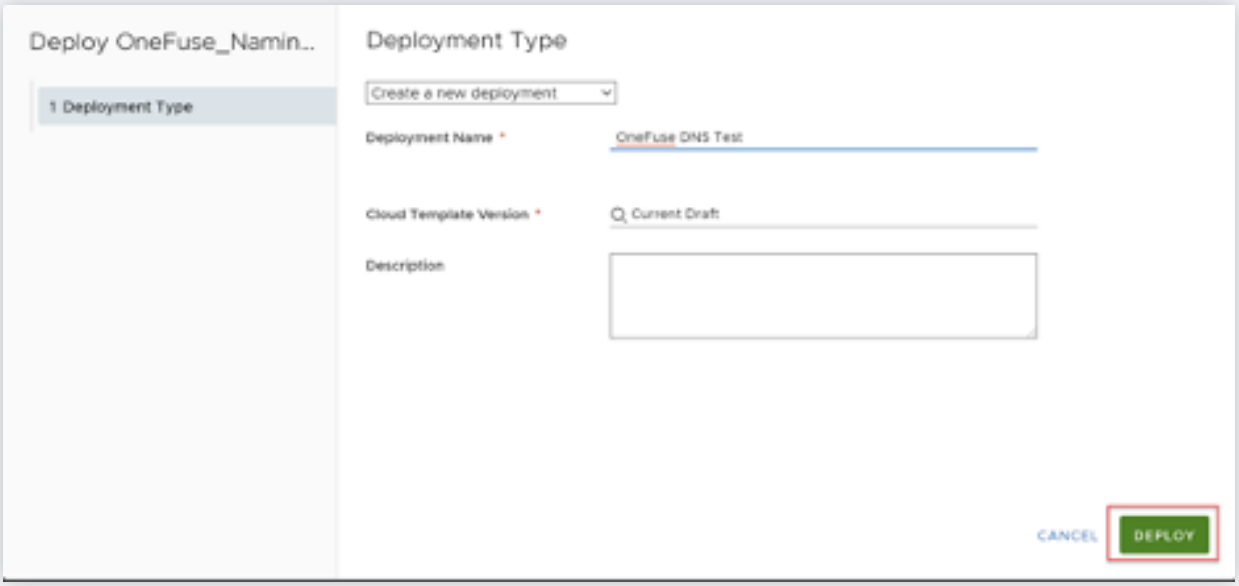
```
formatVersion: 1
 inputs: {}
 resources:
   Cloud_vSphere_Machine_1:
     type: Cloud.vSphere.Machine
     properties:
       #vRA Properties
       imageRef: Centos7
       cpuCount: 1
       totalMemoryMB: 1024
       #OneFuse Module Properties
       OneFuse_NamingPolicy: 'onefuseblog:default'
       OneFuse_IpamPolicy_Nic0: 'onefuseblog:default'
```

```
OneFuse_DnsPolicy_Nic0: 'onefuseblog:default:{{dns_suffix}}'
#Additional Properties used to generate name
nameGroup: pp
nameLocation: atl
nameEnv: prod
nameOS: l
nameApp: web
dns_suffix: infoblox851.sovlabs.net
```

4. Once the blueprint is completed, we will do a deploy to test that our DNS integration is working as expected. Select "Deploy" in the lower left corner.



5. When the deployment dialog opens give your deployment a name and select "Deploy"

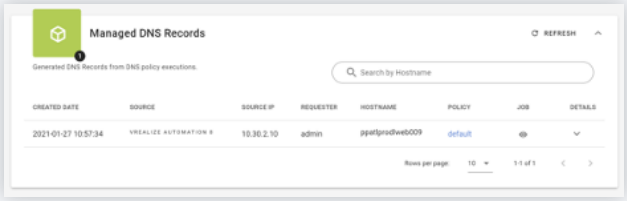


6. Within a few minutes of starting the deployment a name will have been generated, an IP address will have been reserved, and DNS records will have been created. To view the name, IP reservation, and DNS records you can login to the OneFuse UI and view the managed objects. You will also be able to see this information within vRA as well.

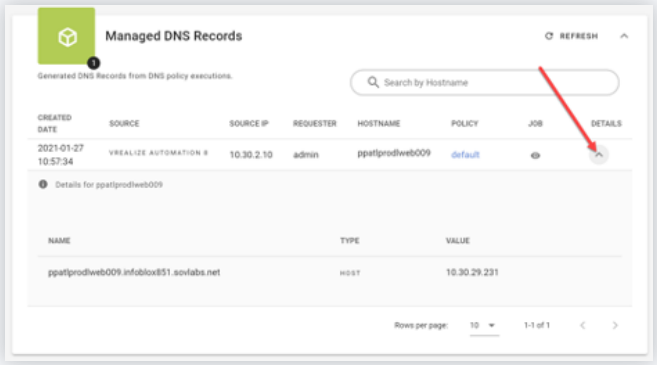
0. To view the managed objects within OneFuse login to the OneFuse UI navigate to the module and view the Managed object for that module.

1. In this example, we will have a Naming object, an IPAM object, and a DNS object.

2.

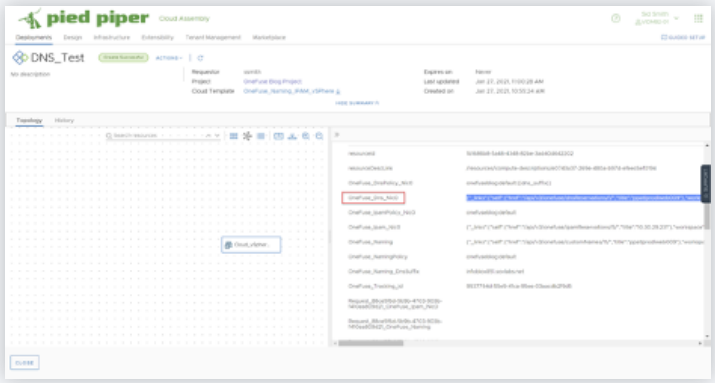


3. On the DNS Object record you can expand details to view the created DNS records.



7. Within vRA8 you can view the details of your deployment and view the name and IP Address. If you expand Network you will see the network the workload was assigned to and other details. You can view the entire output from these as well as DNS under properties.

0.



1. Below is a sample of the information stored with the output:

{"_links":{"self":{"href":"/api/v3/onefuse/dnsReservations/1/","title":"ppatlprodlweb009"},"workspace":{"href":"/api/v3/onefuse/workspaces/2/","title":"Default"},"policy":{"href":"/api/v3/onefuse/dnsPolicies/1/","title":"default"},"jobMetadata":{"href":"/api/v3/onefuse/jobMetadata/33/","title":"Job Metadata Record id

33"}],"name":"ppatlprodlweb009","id":1,"records":[{"type":"host","name":"ppatlprodlweb009.infoblox851.sovlabs.net","value":"10.30.29.231"}],"trackingId":"9537794d-55e9-41ca-95ee-03aacdb2f9d5","endpoint":"onefuseblog"}

We have now successfully added and tested DNS within vRA8. While this is a basic example of consuming a static DNS Policy, it is possible to drive the DNS integration more flexibly without having to know which specific DNS technology/vendor is being used –policy abstraction at its best!

Chapter 5: Active Directory Configuration

We'll now create a vRA8 blueprint that utilizes the OneFuse Active Directory module to provide AD integration for provisioned workloads. We'll utilize the Active Directory (AD) policy that we created in the "Creating a OneFuse Active Directory Policy" article. I will also be leveraging the same blueprint that was used in the "vRA8 with OneFuse: DNS Integration" article to build on the previous example.

By the end of this chapter we will have added the OneFuse Active Directory (AD) policy to a vRA8 blueprint that is deploying a vSphere machine. This will be a simple example that we will build upon in future articles.

Before we begin, there are prerequisites you will want to have ready.

Prerequisites

- The OneFuse appliance should be deployed and configured, see the following articles if you need to walk through the OneFuse deployment and configuration.
  - Deploying the OneFuse Appliance
  - Configuring the OneFuse Appliance
  - Creating a OneFuse Active Directory Policy
- The OneFuse Workflow package needs to be installed and configured within vRA8. The following article can walk you through is you have not completed this yet.
  - Installing the OneFuse Workflow Package into vRA8
- vRA installed and a working vSphere Blueprint

Adding the OneFuse Active Directory Integration to a vRA8 Blueprint

We'll use the blueprint we walked through in the previous article on DNS integration. To begin you will need to open the blueprint for editing so we can add the appropriate items for consuming the Active Directory integration. Before we begin we will want to create a new version so we can make a record of the blueprints current state.

If you are not familiar with how to do this please see the previous article "vRA8 with OneFuse: DNS Integration" for step by step instructions.

Now that we have a tracked version of the current state we can go ahead and add the properties needed to integrate Active Directory into our blueprint.

1. We need to add the following property to our blueprint: *OneFuse_ADPolicy*

2. Next we need to set the value for the property. The value is broken up into 2 parts separated by a colon as outlined below.

1. OneFuse_Endpoint:Policy_Name

    • For my environment this will be onefuseblog:default This will result in the following being added to the blueprint:

    • OneFuse_ADPolicy: 'onefuseblog:default'

2. If you recall in the article "Creating a OneFuse Active Directory Policy" some new inputs were introduced within my AD policy and they were flagged as required. In this example I'm configuring these as properties on my blueprint so they are passed in as part of the request. There are many different ways to handle these including taking them as vRA8 inputs into the blueprint. We will dive deeper into the various options in future articles.

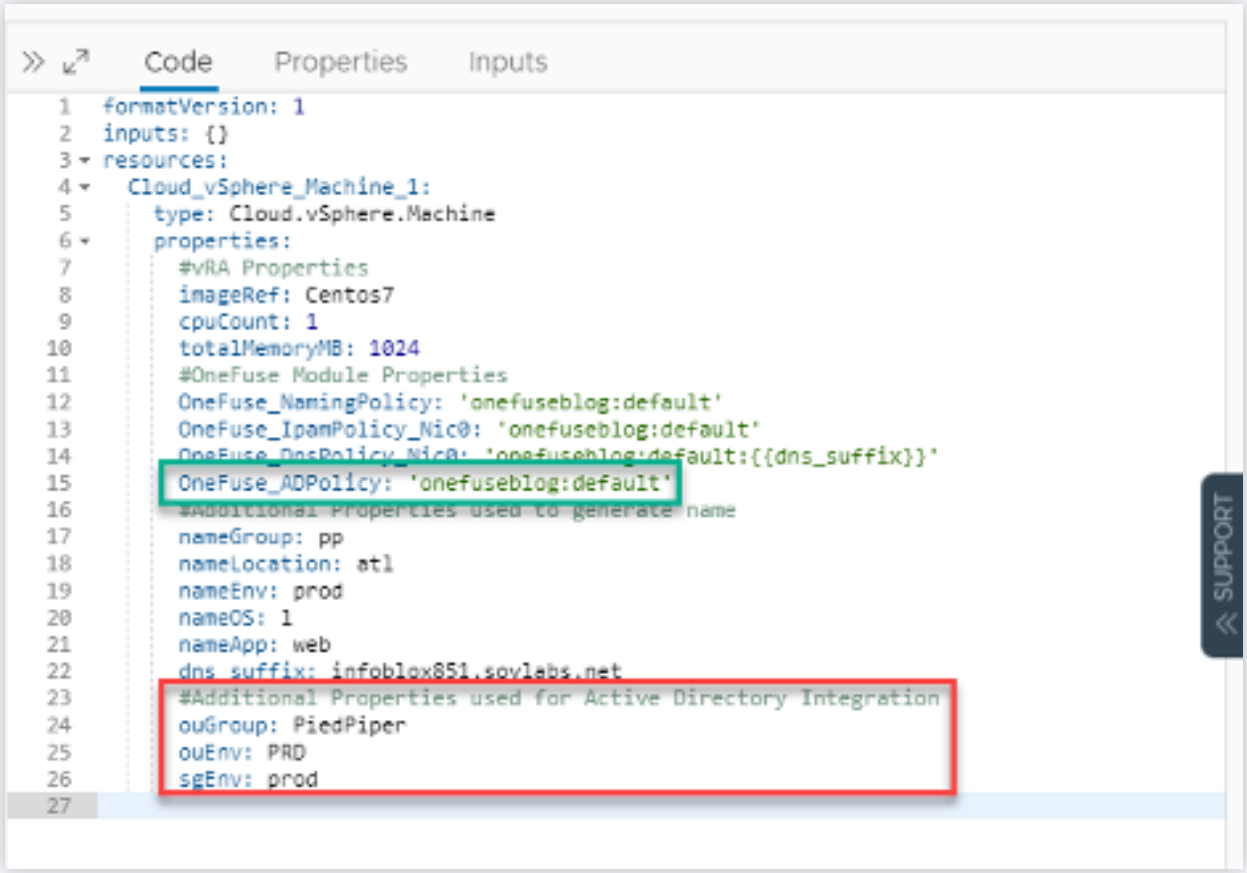
    • The inputs I need to add are:

        - ouGroup

        - ouEnv

        - sgEnv

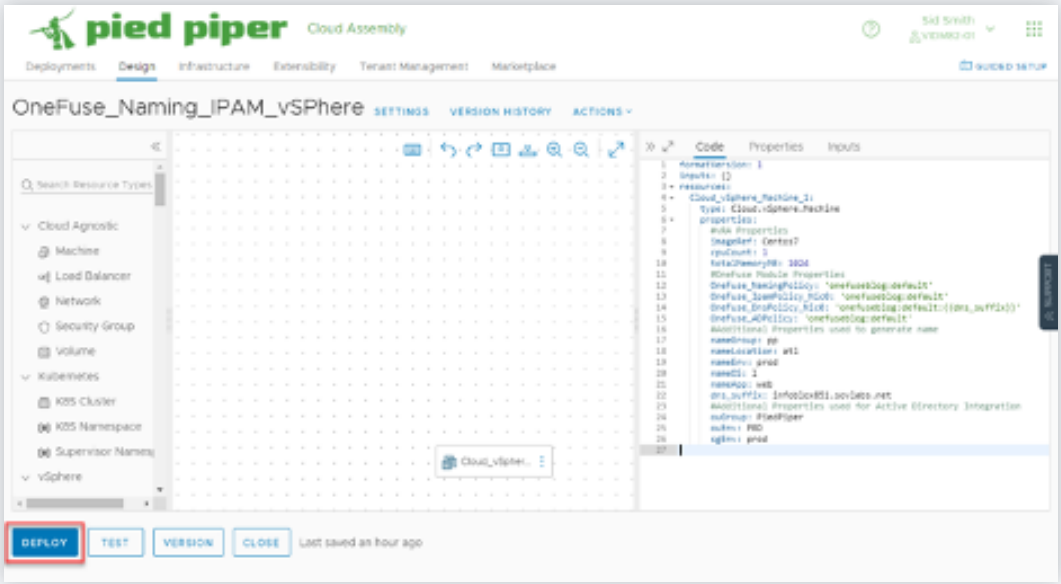
3. You can see both the OneFuse_ADPolicy and the added properties in my screenshot below.



4. The complete yaml for my blueprint is below:

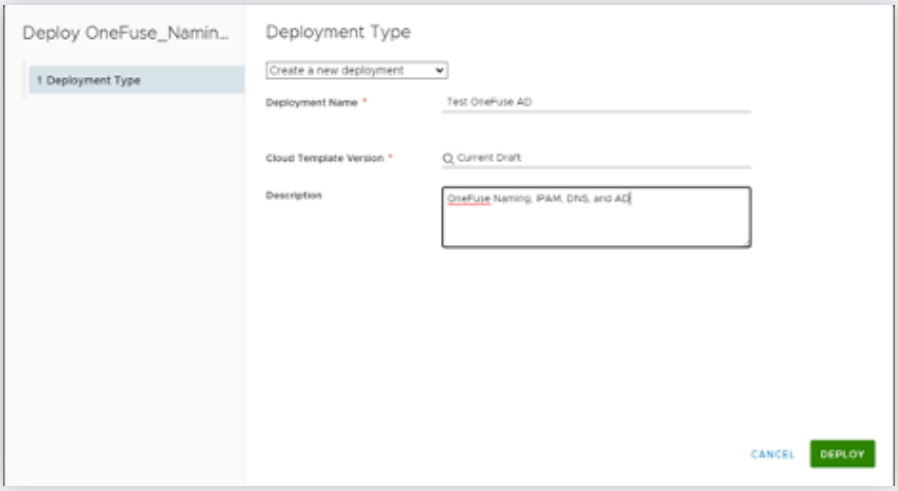
```
formatVersion: 1
inputs: {}
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      #vRA Properties
      imageRef: Centos7
      cpuCount: 1
      totalMemoryMB: 1024
      #OneFuse Module Properties
      OneFuse_NamingPolicy: 'onefuseblog:default'
      OneFuse_IpamPolicy_Nic0: 'onefuseblog:default'
      OneFuse_DnsPolicy_Nic0: 'onefuseblog:default:{{dns_suffix}}'
      OneFuse_ADPolicy: 'onefuseblog:default'
      #Additional Properties used to generate name
      nameGroup: pp
      nameLocation: atl
      nameEnv: prod
      nameOS: l
      nameApp: web
      dns_suffix: infoblox851.sovlabs.net
      #Additional Properties used for Active Directory Integration
      ouGroup: PiedPiper
      ouEnv: PRD
      sgEnv: prod
```

3. Once you have completed adding the properties to your blueprint select “Deploy”



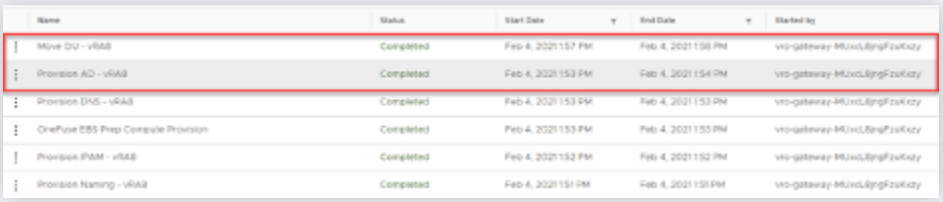
4. Give your deployment a name and select “Deploy”

0.



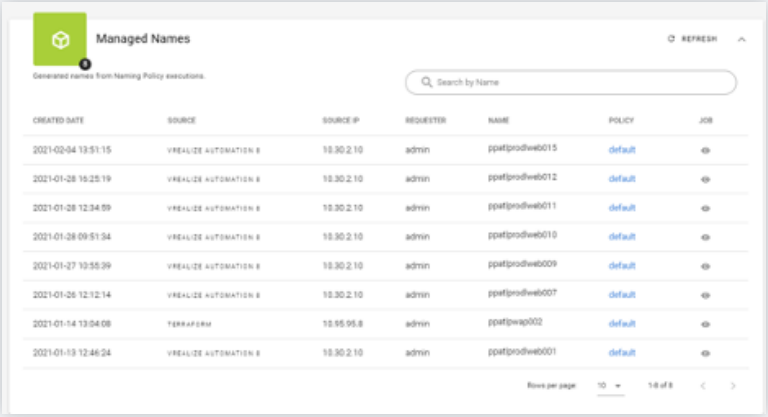
1. Now that the deployment has been started, let’s discuss vRealize Orchestrator. If you launch the vRO web UI and navigate to “Workflow Runs” you will be able to see the OneFuse workflow runs as they execute. This is because the Active Directory Policy includes the build OU option which will initially place the AD computer object in a temporary build OU and then move it to it’s final OU once the deployment is complete. You can see this in the screenshot below. You will also see the Naming, IPAM, and DNS workflow runs as well.

2.

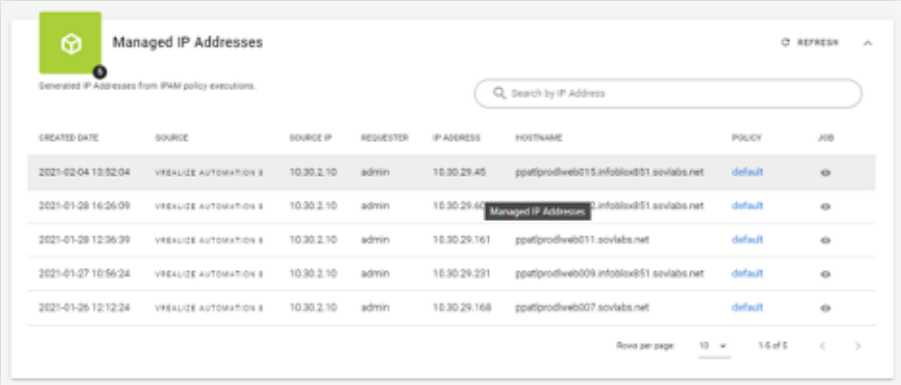


5. Now let’s switch back to the OneFuse Appliance and have a look at what objects we have available for each of the modules.

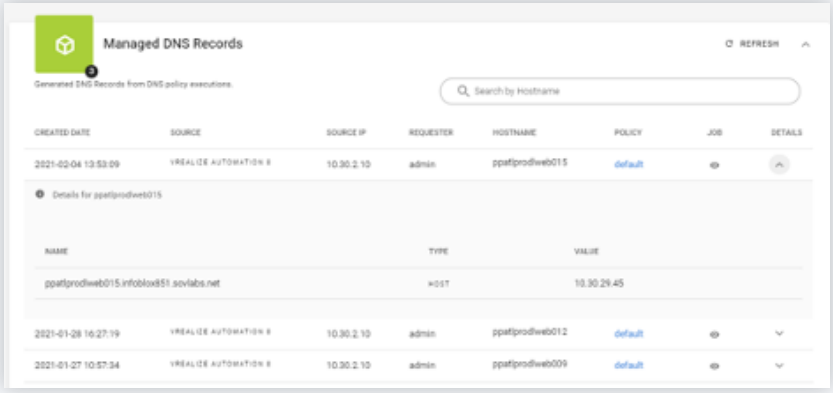
0. First we will look at naming and see the name that was generated for my deployment which is ppatlprodweb015.



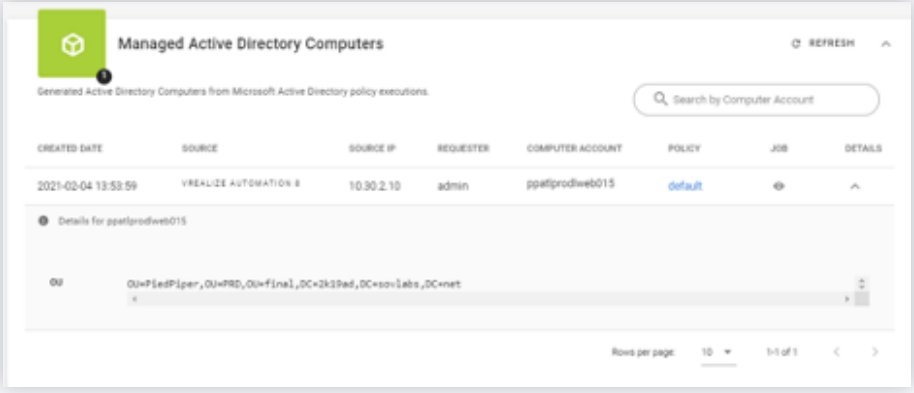
1. If we next look at IPAM for this deployment we will see the IP Address of 10.30.29.45 reserved for the machine ppatlprodweb015.



2. We can then jump over to DNS and see that a host record has been created for the machine ppatlprodweb015 with it's IP Address of 10.30.29.45.



3. Finally if we look at the Managed Active Directory Computer We can see the final OU that the computer object is located within.



4. If we look at the "Job" details by selecting the "eye" under the job column we can get additional details including the build OU it was in as well as any security groups that it has been joined to.

5. Within vRA8 you can view the details of your deployment and view the name, IP Address, DNS Records, and AD OU that the workload was placed into under the properties sections of the deployment. Below is a sample of the output data that is returned to vRA8 from OneFuse.

{"_links":{"self":{"href":"/api/v3/onefuse/microsoftADComputerAccounts/1/","title":"ppatlprodlweb015"},"workspace":{"href":"/api/v3/onefuse/workspaces/2/","title":"Default"},"policy":{"href":"/api/v3/onefuse/microsoftADPolicies/1/","title":"default"},"jobMetadata":{"href":"/api/v3/onefuse/jobMetadata/57/","title":"Job Metadata Record id 57"}},"name":"ppatlprodlweb015","id":1,"state":"final","buildOu":"OU=PiedPiper,OU=PRD,OU=build,DC=2k19ad,DC=sovlabs,DC=net","finalOu":"OU=PiedPiper,OU=PRD,OU=final,DC=2k19ad,DC=sovlabs,DC=net","securityGroups":["CN=prodDemoComputers,OU=Groups,OU=SovLabs,DC=2k19ad,DC=sovlabs,DC=net"],"trackingId":"18238259-1687-4b4c-b3ca-e26ab2fa7a43","endpoint":"onefuseblog"}

We have now successfully added and tested Active Directory within vRA8. While this is a basic example of consuming a simple Active Directory Policy, it is possible to drive the Active Directory (AD) integration more flexibly.

Chapter 6: Property Toolkit

Let's look at some of the ways the Property Toolkit can be utilized within vRA8.

vRA8 Property Stack

vRA8, much like its predecessor vRA7, has the concept of a property stack that lives with the deployed machine throughout its lifecycle. Although there are some significant differences between how the two versions manage the property stack across different lifecycles, we are not going to dig into those details in this article. For this article we are going to talk about the property stack in general terms.

Within the current release of vRA8 there are two places properties can be defined. They can be defined on a project or within a blueprint. The concepts in this article apply to both.

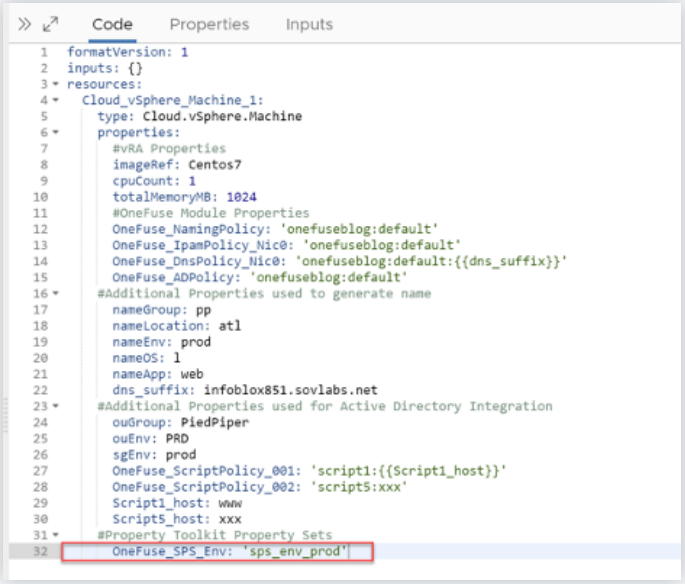
Assign OneFuse Property Sets using static properties

Adding a OneFuse Property Toolkit Property Set to a vRA8 blueprint is as easy as defining a standard property on the blueprint. There are a few things to be aware of as outlined below.

The property name must be formatted as *OneFuse_SPS_{something_unique}*. The *OneFuse_SPS_* portion of the property name tells the OneFuse workflows that this is a OneFuse property toolkit property that will reference a OneFuse Property Set and the *{something_unique}* part allows us to define multiples of them. The value for the property will be the name of the Static Property Set that you have created in OneFuse.

We have a property group named *sps_env_prod* and this will be the value we will use. It would look something like the following:

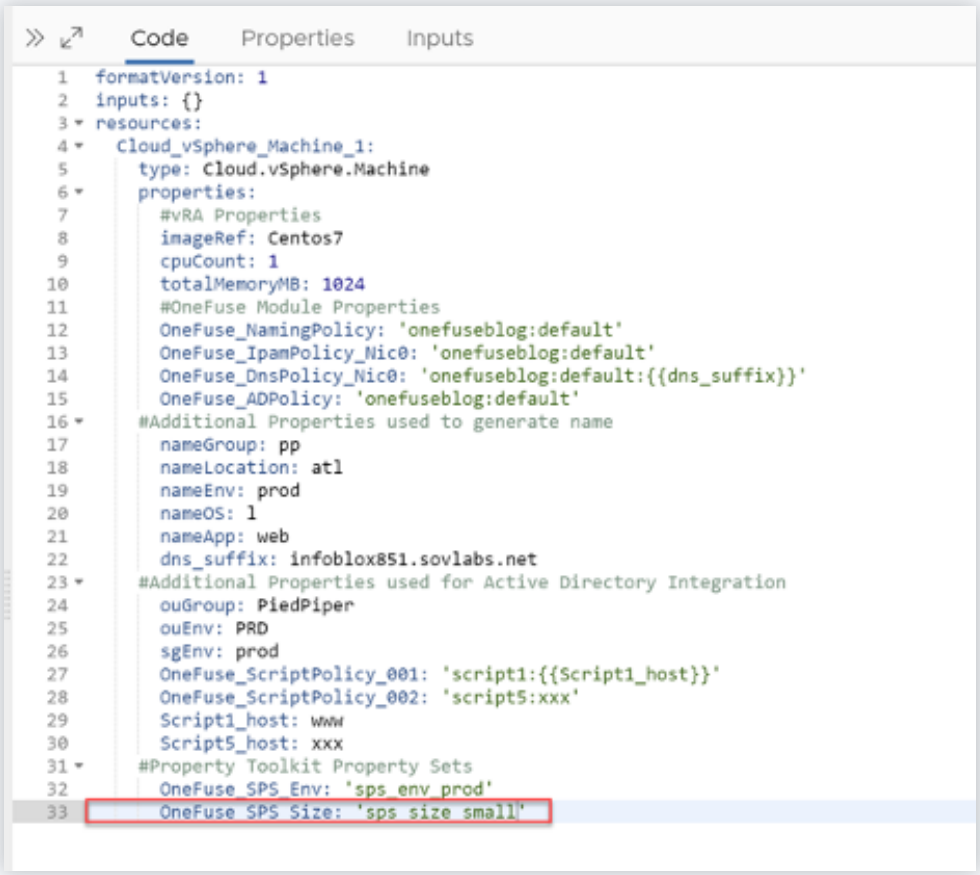
- Property Name: *OneFuse_SPS_Env*
- Property Value: *sps_env_prod*



Adding this property to the blueprint will add every property defined within the *sps_env_prod* property set to the property stack for any deployments requested from this blueprint. In our case it will add the following:

```
{
  "nameEnv": "p",
  "folderEnv": "PROD",
  "ipamEnv": "prod",
  "ouEnv": "PRD",
  "sgEnv": "prod",
  "dnsPolicy": "prod",
  "adPolicy": "prod",
  "location": "atl",
  "dnsSuffix": "infoblox851.sovlabs.net",
  "deployNameEnv": "Prod",
  "s3NameEnv": "production"
}
```

Because this is a flat property set everything within it will be added to the property stack within vRA. Our next property set that we will add to vRA8 will be a property to define sizing. Within our sizing property set we have nested properties. These are nested under parents and one of those parents is a reserved property that tells vRA8 which properties to include for its platform.

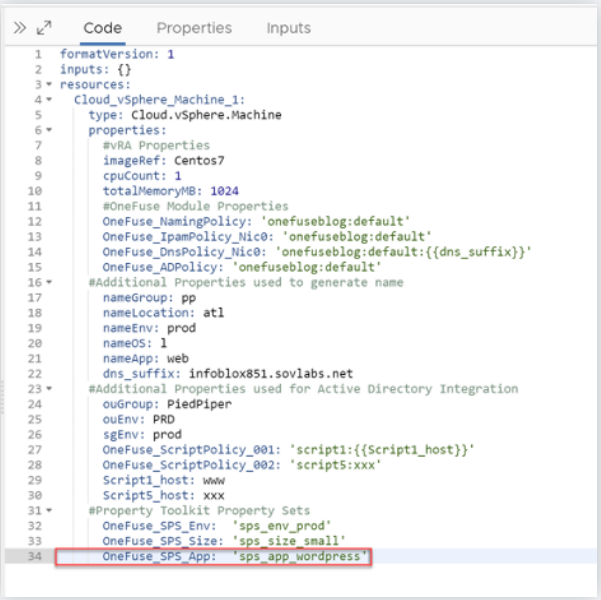


The contents of the *sps_size_small* property set are:

```
{
  "size": "small",
  "cpuCount": "1",
  "memoryMB": "1024",
  "memoryGB": "1",

   "OneFuse_VRA7_Props": {
     "VirtualMachine.CPU.Count": "{{cpuCount}}",
     "VirtualMachine.Memory.Size": "{{memoryMB}}"
   },
   "OneFuse_VRA8_Props": {
     "flavor": "{{size}}"
   },
   "OneFuse_TF_Props": {
     "cpu": "{{cpuCount}}",
     "memMb": "{{memoryMB}}"
   },
   "OneFuse_TF_Props": {
     "vcpu": "{{cpuCount}}",
     "mem_size": "{{memoryGB}}"
   }
}
```

Unlike the *sps_env_prod* property set not all of these properties will be written against the deployments property stack. Only the parent level properties; cpuCount, memoryMB, and memoryGB as well as the properties located under OneFuse_VRA8_Props; flavor will be written against the deployments property stack. The final property set we will add in this example is one that we use for defining applications. In this case WordPress will be the application. The property set will be *sps_app_wordpress*.



The contents of *sps_app_wordpress* are:

If you look closely you will see that within this property set is another property set property and value. This will result in the contents of the *sps_os_centos7* property set also being pulled in as part of the deployments property stack.

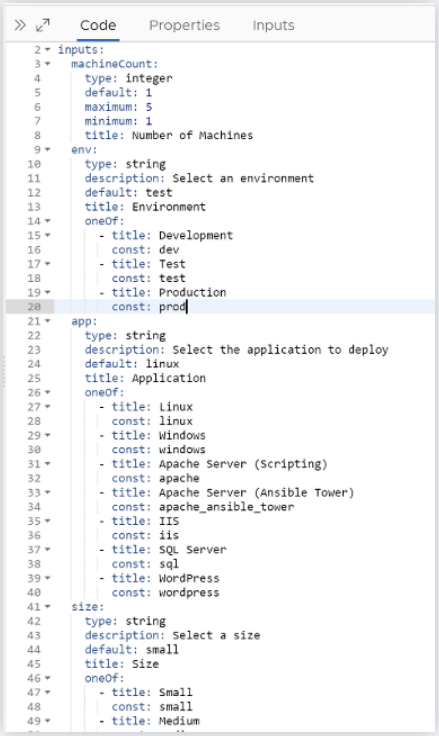
The contents of *sps_os_centos7* are:

```
{
  "nameApp": "web",
  "ipamApp": "web",
  "OneFuse_SPS_OS": "sps_os_centos7",
  "deployNameApp": "WORDPRESS"
}
```

Just like with sps_size_small only the parent level properties as well as the properties located under *OneFuse_VRA*_Props* will be added to the deployments property stack. At this point all the relevant properties from each group will become part of the deployments resultant property stack and impact the deployment. In this case some affect vRA8 capabilities and others will impact OneFuse module integrations.

Assign OneFuse Property Sets using inputs

Assigning OneFuse Property Sets using vRA8 inputs is easy. If we use the same property sets we have already looked at earlier in this chapter we only need to make some minor changes.
To start we need inputs for the users to make selections.



Next, we need to be able to use those inputs to drive my OneFuse Static Property Set selections. We do this through the use of blueprint expression syntax.

```
#Property Toolkit Property Sets
OneFuse_SPS_Env: 'sps_env_${input.env}'
OneFuse_SPS_Size: 'sps_size_${input.size}'
OneFuse_SPS_App: 'sps_app_${input.app}'
```

When a user selects an environment it will complete the sps_env_ with whatever they selected to attach the appropriate OneFuse Property Set. The full blueprint yaml is below:

```
formatVersion: 1
inputs:
  machineCount:
    type: integer
    default: 1
    maximum: 5
    minimum: 1
    title: Number of Machines
  env:
    type: string
    description: Select an environment
    default: test
    title: Environment
    oneOf:
      - title: Development
        const: dev
      - title: Test
        const: test
      - title: Production
        const: prod
  app:
    type: string
    description: Select the application to deploy
    default: linux
    title: Application
    oneOf:
      - title: Linux
        const: linux
      - title: Windows
        const: windows
      - title: Apache Server (Scripting)
        const: apache
      - title: Apache Server (Ansible Tower)
        const: apache_ansible_tower
      - title: IIS
        const: iis
```

```
  - title: SQL Server
      const: sql
    - title: WordPress
      const: wordpress
  size:
    type: string
    description: Select a size
    default: small
    title: Size
    oneOf:
      - title: Small
        const: small
      - title: Medium
        const: medium
      - title: Large
        const: large
resources:
  Cloud_vSphere_Machine_1:
    type: Cloud.vSphere.Machine
    properties:
      #vRA Properties
      imageRef: Centos7
      cpuCount: 1
      totalMemoryMB: 1024
      #OneFuse Module Properties
      OneFuse_NamingPolicy: 'onefuseblog:default'
      OneFuse_IpamPolicy_Nic0: 'onefuseblog:default'
      OneFuse_DnsPolicy_Nic0: 'onefuseblog:default:{{dns_suffix}}'
      OneFuse_ADPolicy: 'onefuseblog:default'
      #Property Toolkit Property Sets
      OneFuse_SPS_Env: 'sps_env_${input.env}'
      OneFuse_SPS_Size: 'sps_size_${input.size}'
      OneFuse_SPS_App: 'sps_app_${input.app}'
```

Conclusion

All organizations are seeking to automate process to save time, money, and be more productive and efficient. But automation requires an abundance of custom coded integrations and with that these top-ranked challenges:

- Require domain knowledge - Require coding expertise - Time-consuming projects	- Unmanageable as intro more tools - Expensive projects - Human-error, visibility & governance

OneFuse delivers cloud automation through abstracting underlying integration complexity and presenting varying cloud infrastructure integrations as services that can be re-used again and again. Policy ensures governance conformity, and you don't need domain or coding expertise. Build processes faster and get more done with pluggable and modular integration services.

How much are IT integrations costing you?

Find out with our Free ROI Calculator.

Ready to try OneFuse in your environment?

Download the Free Community Edition.

See OneFuse in action!

Just book and attend a CloudBolt demo.

1. www.networkcomputing.com/networking/integration-challenges-lead-half-million-dollar-year-losses
2. www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2018-report/The-Cost-of-Poor-Quality-Software-in-the-US-2018-Report.pdf
3. www.gartner.com/en/documents/3888587/rethink-your-internal-private-cloud