

# Bring Your Own Cloud

Embotics vCommander  
Solution Guide

# Executive Summary

Embotics® vCommander™ provides self-service provisioning automation across private, public and hybrid clouds in single or multi-tenant environments. Leveraging a comprehensive set of multi-hypervisor and cloud management capabilities, organizations at any stage of cloud adoption can build and implement the framework, process automation, and policies that will result in an optimized hybrid cloud environment.






Using the solution described in this document, vCommander administrators can offer public cloud services to Service Portal consumers who are adding their own Amazon Web Services or Azure subscriptions, rather than simply deploying workloads on an existing, integrated subscription.

The focus of this document is on the configuration requirements to allow the addition of the subscriptions to occur. Any other Approval Workflow configuration you wish to configure to enact appropriate governance is beyond the scope here, but you can contact [support@embotics.com](mailto:support@embotics.com) if you wish to discuss this topic further.

## Service Catalog

[1. Add](#)[2. Customize](#)[3. Review](#)

### Request New Service

	<b>Add AWS Public Cloud</b> Adds a managed system based on the specified Amazon Web Service Account Name and Access Key ID. Public	\$0	+
	<b>Add Azure Classic Public Cloud</b> Adds a managed system based on the specified Azure Classic subscription ID. Public	\$0	+
	<b>Add Azure Public Cloud</b> Adds a managed system based on the specified Microsoft Azure Subscription ID. Public	\$0	+
	<b>Amazon Linux</b> Linux, Public	\$61	+
	<b>Azure Linux</b> Linux, Public	\$158	+

## Contents

Executive Summary .....	1
Requirements.....	3
Before You Begin.....	3
Creating an Automation User .....	3
vCommander Scripts.....	4
Brokering Amazon Web Services .....	5
Creating the vCommander Custom Attributes.....	5
Creating the Service Catalog Entry .....	7
Creating the Completion Workflow.....	10
Requesting the Service.....	12
Brokering Microsoft Azure .....	13
Creating the vCommander Custom Attributes.....	13
Creating the Service Catalog Entry .....	15
Creating the Completion Workflow.....	18
Requesting the Service.....	20
Brokering Microsoft Azure Classic .....	21
Downloading the vCommander Azure Certificate .....	21
Creating the vCommander Custom Attributes.....	21
Creating the Service Catalog Entry .....	24
Creating the Approval Workflow .....	26
Creating the Completion Workflow.....	28
Requesting the Service.....	30

# Requirements

Find below the requirements necessary to fulfill the solution as described in this document.

- [Embotics vCommander 6.0.3 or later](#)
- [Embotics vCommander REST API Powershell client \(2.8\) & libraries](#)
- [PowerShell v5 installed on the vCommander application server](#)
- [vCommander scripts](#)
- [Encrypted vCommander credentials with REST API access](#)

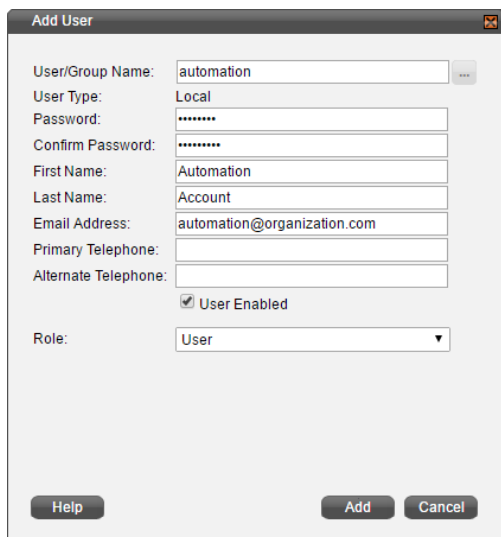
## Before You Begin

As when making any substantial change to a production system, Embotix strongly recommends taking measures to ensure a successful roll-back is possible in case unexpected failures occur. These include making sure there are [current backups](#) of the vCommander SQL database and snapshots of the vCommander application server.

Additionally, you may want to implement and test the integration in a staging environment to make sure it operates as expected prior to putting it into production. Embotix provides customers with staging or lab licenses upon request. Contact your [Customer Advocate](#) for more information about these limited, supplementary licenses.

## Creating an Automation User

As a best practice, you should [create a specific user account](#) to handle the automation activity. Doing so makes it easy to track actions undertaken by this solution by parsing the vCommander events for the account, and provides an account to use for form assignment.



- In most cases, simply using a local vCommander account with the User role will be sufficient. However, you can use directory services (AD/LDAP) accounts and/or Service Portal accounts as necessary.
- Provide a name that will be recognizable in events and logs as being the automation user.
- Provide an email address that will contact an appropriate person should there be any request failures.

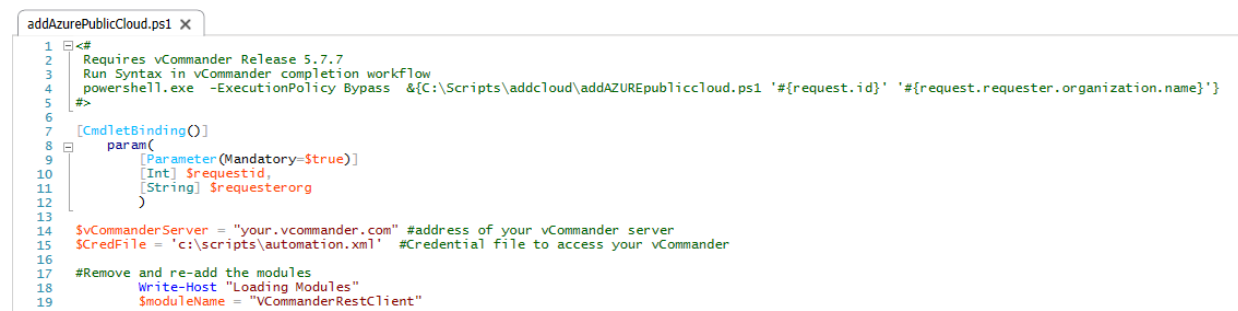
## vCommander Scripts

You should already have downloaded and extracted the scripts to your vCommander application server as detailed in the [Requirements](#) section. Embotics recommends storing all scripts called by vCommander in a single location, using sub-folders to identify the functions of scripts. With the scripts used for this solution extracted to C:\ on the vCommander server, the file system will look like this:

- C:\scripts\addcloud\addAWSPublicCloud.ps1
- C:\scripts\addcloud\addARMPublicCloud.ps1
- C:\scripts\addcloud\addAzurePublicCloud.ps1

These scripts require minor edits before they will work in your system.

Setting	Description
\$vCommanderServer	The hostname or IP address of the vCommander server.
\$CredFile	The credentials file which handles access to your vCommander. For more details, refer to <a href="#">this knowledgebase article</a> .



```

1  <#
2  Requires vCommander Release 5.7.7
3  Run Syntax in vCommander completion workflow
4  powershell.exe -ExecutionPolicy Bypass &{C:\Scripts\addcloud\addAZUREpubliccloud.ps1 '#{request.id}' '#{request.requester.organization.name}'}
5  #>
6
7  [CmdletBinding()]
8  param(
9      [Parameter(Mandatory=$true)]
10     [Int] $requestid,
11     [String] $requesterorg
12 )
13
14 $vCommanderServer = "your.vcommander.com" #address of your vCommander server
15 $CredFile = 'c:\scripts\automation.xml' #Credential file to access your vCommander
16
17 #Remove and re-add the modules
18 Write-Host "Loading Modules"
19 $moduleName = "VCommanderRestClient"
  
```

# Brokering Amazon Web Services

This section covers the configuration needed to allow Service Portal users to add their own Amazon Web Services managed system to vCommander.

## Creating the vCommander Custom Attributes

This solution uses custom attributes to capture the account Name and Access Key ID. Follow the steps below to create these.

1. Under the **Configuration** menu, choose **Custom Attributes**.
2. Click **Add**.
3. Enter *AWS Account Name* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

Configure Attribute

**Define Attribute**  
Define the attribute name, description, and type.

**Define Attribute**  
Configure Attribute

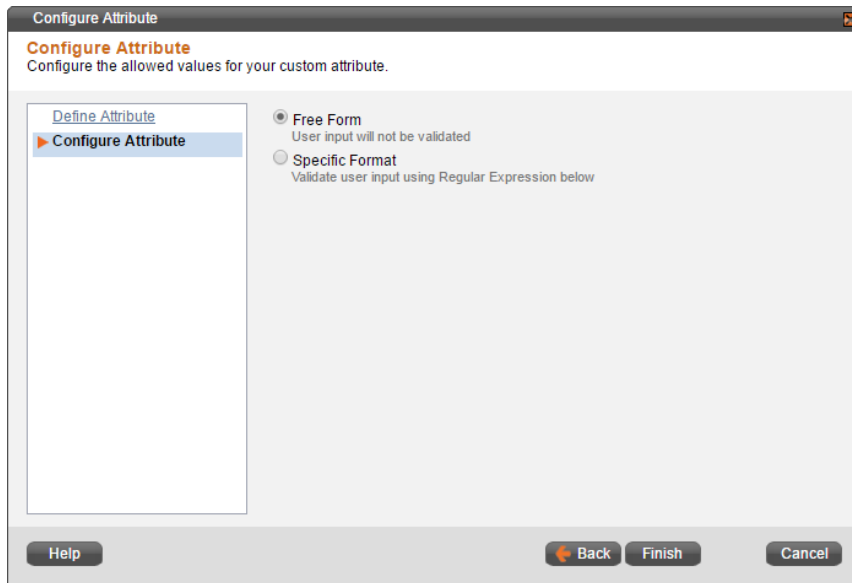
Name: AWS Account Name  
Description: Account Name for Amazon subscription added via the Service Portal

Type: Text  
Applies To: Form

Edit in Service Portal: ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

Help Back Next Cancel

4. Choose **Free Form** and click **Finish**.



**Configure Attribute**  
Configure the allowed values for your custom attribute.

[Define Attribute](#)  
▶ **Configure Attribute**

☒ **Free Form**  
User input will not be validated

☐ **Specific Format**  
Validate user input using Regular Expression below

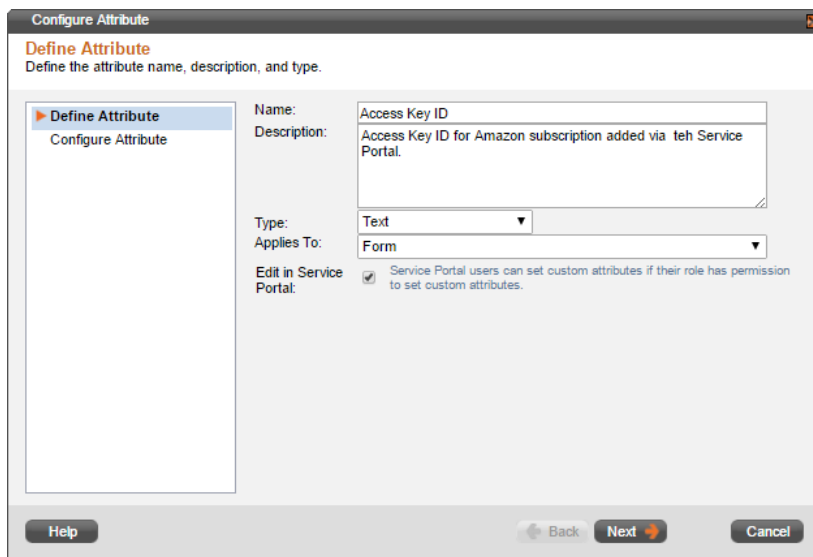
[Help](#) [Back](#) [Finish](#) [Cancel](#)

5. Back on the **Custom Attributes** page, click **Add**.
6. Enter *Access Key ID* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled



**Configure Attribute**  
**Define Attribute**  
Define the attribute name, description, and type.

▶ **Define Attribute**  
Configure Attribute

**Name:** Access Key ID  
**Description:** Access Key ID for Amazon subscription added via teh Service Portal.

**Type:** Text  
**Applies To:** Form

**Edit in Service Portal:** ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

[Help](#) [Back](#) [Next](#) [Cancel](#)

7. Choose **Free Form** and click **Finish**.

## Creating the Service Catalog Entry

Next, create the service catalog entry so that your users have access to the forms they need to request that their AWS subscription be added.

1. Under the **Configuration** menu, choose **Service Request Configuration**.
2. Switch to the **Service Catalog** tab.
3. Click **Add Service**.
4. Configure the options on the **Service Description** page as follows, then click **Next**:

**Name:** Add AWS Public Cloud

**Description:** Adds a managed system based on the specified Amazon Web Services account name and access key ID.

**Icon:** Choose as appropriate.

**Categories:** Choose as appropriate.

5. Now we will use a custom component to add the AWS account to the catalog. On the **Components** page, click **Add > New Component Type**.
6. Configure the new component type as follows, then click **Add to Service**:

**Name:** Add AWS Account

**Description:** Custom component for user-added AWS account.

**Annual Cost:** Set as appropriate (you must use 0.00 to not apply a cost.)

7. Click **Next** and then switch to the **Attributes** tab.



8. Click **Add Attributes** and select **Access Key ID** and **AWS Account Name**. Click **OK**.

Infrastructure | Attributes | Form

Assign metadata with custom attributes and groups. On the Form tab, you can allow requesters to set values for custom attributes.

Attribute Name	Default Value
AWS Account Name:	<input type="text"/>
Access Key ID:	<input type="text"/>

[Add Attributes](#) [Manage Attributes](#)

Group Name	Default Value
------------	---------------

[Add Groups](#) [Manage Groups](#)

[Help](#) [Back](#) [Next](#) [Finish](#) [Cancel](#)

9. Switch to the **Form** tab.
10. In the **Toolbox** section, under Custom Attributes, click **AWS Account Name** and **Access Key ID** to add them to the form.
11. **Edit** each of the attributes and check the **Required** checkbox.

Infrastructure | Attributes | Form

Allow requesters to modify the default component settings configured on the other tabs.

[Back](#) [Next](#) [Finish](#) [Cancel](#)

12. Now use an Input Text Field to capture the Secret Access Key. Add an **Input Text Field** element to the form, configured as follows:

**Display Label:** Secret Access Key  
**Maximum Characters:** 100  
**Number of Lines:** 1  
**Hide User Input:** Enabled  
**Required:** Enabled

Infrastructure Attributes Form

Allow requesters to modify the default component settings configured on the other tabs.

↑  
↑  
↓  
↓

(Custom Attribute) \* Access Key ID Edit Delete

(Custom Attribute) \* AWS Account Name Edit Delete

aa (Input Text Field) Edit Delete

Display Label: Secret Access Key

Maximum Characters: 100

Number of Lines: 1

Hide User Input: ☒ Enable if this is a password field

Required: ☒

OK Cancel

Back Next Finish Cancel

13. Add other elements to the form as appropriate for your use.
14. Sort the form elements using the arrow controls and click **Next**.
15. On the **Deployment** page, choose **None** for the **Completion Workflow**. Click **Next**.
16. On the **Visibility** page, choose to publish globally or to specific organizations, groups and users. Click **Next**.
17. Click **Finish**.

## Creating the Completion Workflow

Finally, create a completion workflow to link to the service. The completion workflow references a script which adds the managed system as requested.

1. Under the **Configuration** menu, click **Service Request Configuration**.
2. Switch to the **Completion Workflow** tab.
3. Click **Add**.
4. Set the **Name** as *Add AWS Managed System* and choose to **Apply this workflow:** *after a Custom Component is deployed*. Click **Next**.

5. Click **Add > Execute Script** and configure the step as follows, then click **Next**:

**Name:** Execute API Call

**Step Execution:** Always Execute


**Timeout:** 600 seconds

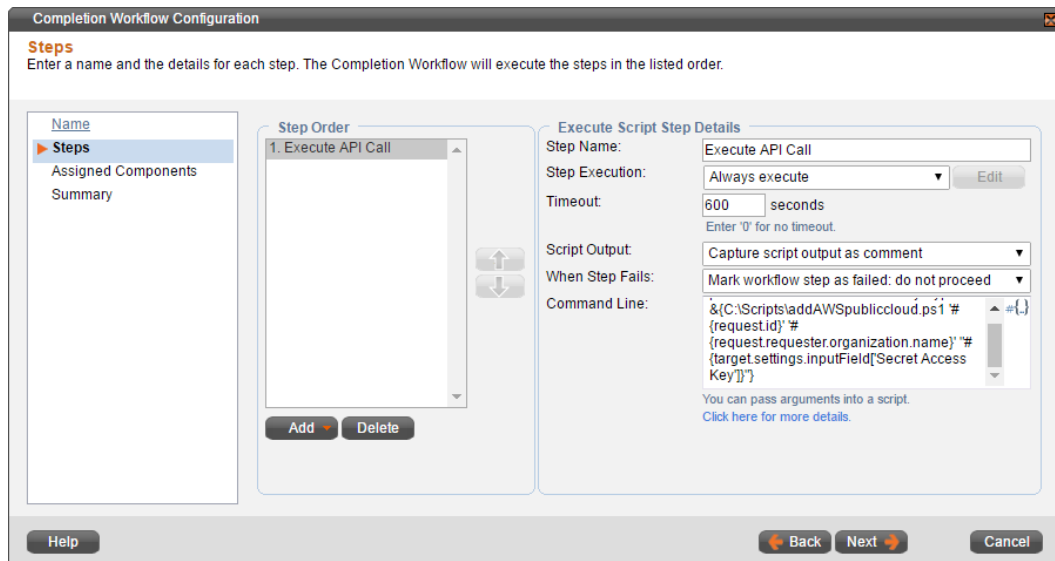
**Script Output:** Capture script output as comment

**When Step Fails:** Mark workflow step as failed: do not proceed

**Command Line:**

```
powershell.exe -ExecutionPolicy Bypass
&{C:\Scripts\addcloud\addAWSpubliccloud.ps1 '#{request.id}'
 '#{request.requester.organization.name}'
 '#{target.settings.inputField['Secret Access Key']]"} }
```

 Note that the **inputField** value is case sensitive. In this case, capitalization of Secret Access Key must match the form label exactly.



**Completion Workflow Configuration**

**Steps**  
Enter a name and the details for each step. The Completion Workflow will execute the steps in the listed order.

**Step Order**

Name
1. Execute API Call

**Execute Script Step Details**

Step Name: Execute API Call

Step Execution: Always execute

Timeout: 600 seconds

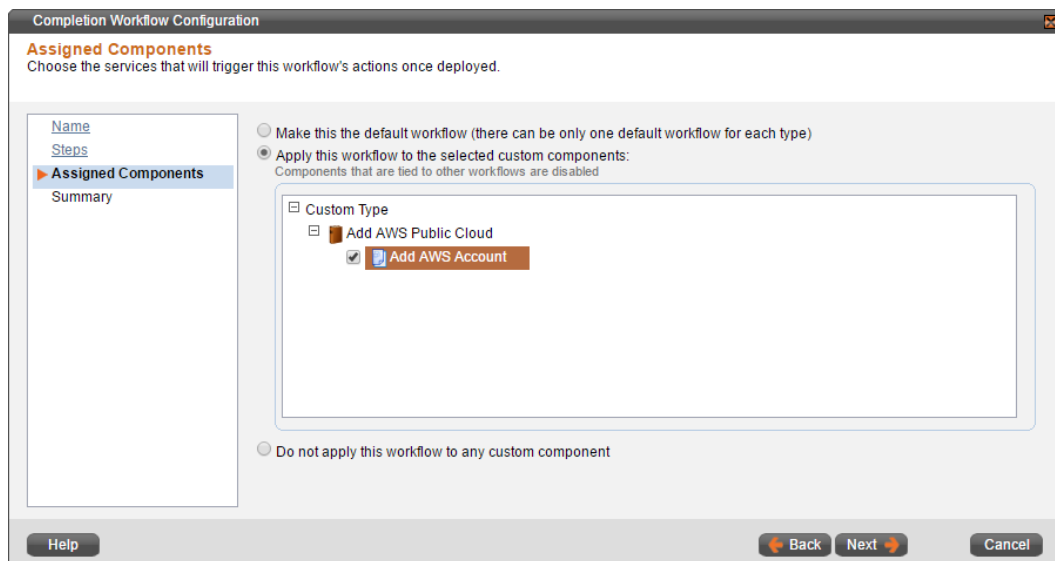
Script Output: Capture script output as comment

When Step Fails: Mark workflow step as failed: do not proceed

Command Line: `&(C:\Scripts\addAWSpubliccloud.ps1' #' {request.id}' #' {request.requester.organization.name}' #' {target.settings.inputField['Secret Access Key']})'`

**Buttons:** Add, Delete, Back, Next, Cancel, Help

6. Check **Apply this workflow to the selected custom components** and enable **Add AWS Account**. Click **Next**.



**Completion Workflow Configuration**

**Assigned Components**  
Choose the services that will trigger this workflow's actions once deployed.

**Assigned Components**

☐ Make this the default workflow (there can be only one default workflow for each type)

☒ Apply this workflow to the selected custom components:  
Components that are tied to other workflows are disabled

☐ Custom Type

- ☒ Add AWS Public Cloud
  - ☒ Add AWS Account

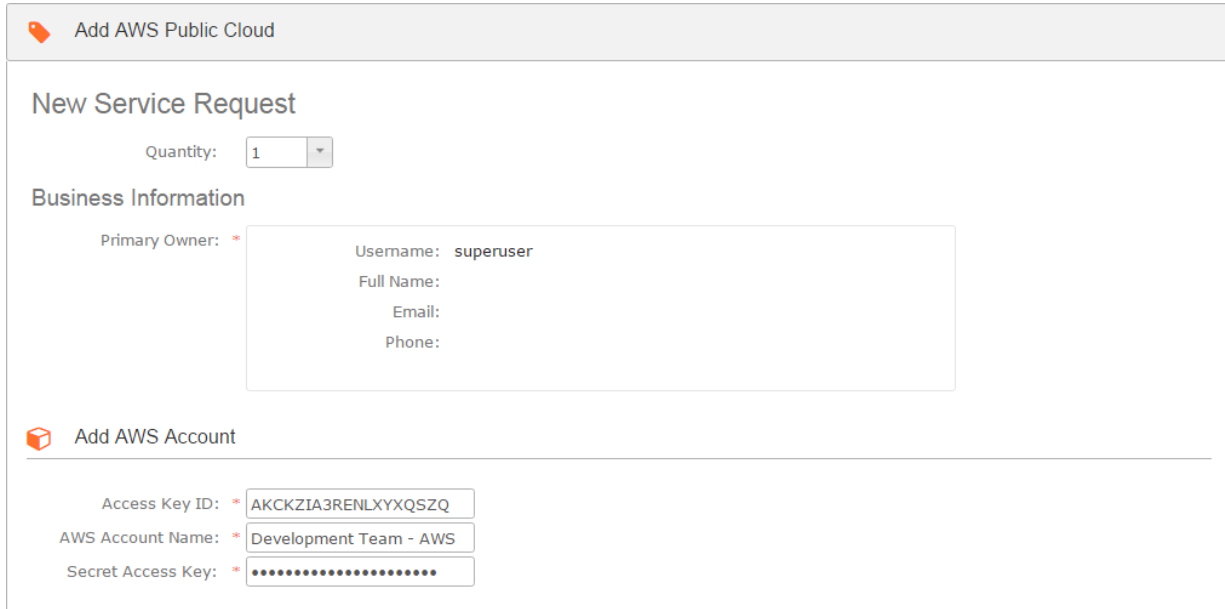
☐ Do not apply this workflow to any custom component

**Buttons:** Back, Next, Cancel, Help

7. Click **Finish**.

## Requesting the Service

When users request the service, they will use the forms as designed in the previous steps.



**Add AWS Public Cloud**

### New Service Request

Quantity: 1

#### Business Information

Primary Owner: \*

Username: superuser

Full Name:

Email:

Phone:

#### Add AWS Account

Access Key ID: \* AKCKZIA3RENLYXQSZQ

AWS Account Name: \* Development Team - AWS

Secret Access Key: \*

The **AWS Account Name** will be used as the label for the managed system wherever it appears in vCommander. Typically, Service Portal users will not often see this label. The **Access Key ID** and **Secret Access Key** are used to connect vCommander to AWS.

Once the completion workflow finishes, the managed system appears in vCommander. All VMs in the managed system are assigned to the requester's organization. This means that any user who does not have the Service Portal permission **Show All Organization Services** will not be able to view the VMs, as users are not assigned individual ownership. If individual ownership is required, it must be assigned afterwards.

## Brokering Microsoft Azure

This section covers the configuration needed to allow Service Portal users to add their own Microsoft Azure managed system to vCommander.

### Creating the vCommander Custom Attributes

This solution uses custom attributes to capture the Subscription Name and Subscription ID. Follow the steps below to create these.

1. Under the **Configuration** menu, choose **Custom Attributes**.
2. Click **Add**.
3. Enter *Subscription ID* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

4. Choose **Free Form** and click **Finish**.
5. Back on the **Custom Attributes** page, click **Add**.
6. Enter *Name* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

**Configure Attribute**

**Define Attribute**  
Define the attribute name, description, and type.

**Define Attribute**  
Configure Attribute

Name: Name  
Description: Application Name for Azure managed systems.

Type: Text  
Applies To: Form

Edit in Service Portal: ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

Help Back Next Cancel

7. Choose **Free Form** and click **Finish**.
8. Back on the **Custom Attributes** page, click **Add**.
9. Enter *Tenant ID* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

**Configure Attribute**

**Define Attribute**  
Define the attribute name, description, and type.

**Define Attribute**  
Configure Attribute

Name: Tenant ID  
Description: Tenant ID for Microsoft Azure.

Type: Text  
Applies To: Form

Edit in Service Portal: ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

Help Back Next Cancel

10. Choose **Free Form** and click **Finish**.
11. Back on the **Custom Attributes** page, click **Add**.
12. Enter *Application ID* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

**Configure Attribute**

**Define Attribute**  
Define the attribute name, description, and type.

**Define Attribute**  
Configure Attribute

Name: Application ID

Description: Application ID for Microsoft Azure.

Type: Text

Applies To: Form

Edit in Service Portal: ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

Help Back Next Cancel

13. Choose **Free Form** and click **Finish**.

## Creating the Service Catalog Entry

Next, create the service catalog entry so that your users have access to the forms they need to request that their Azure subscription be added.

1. Under the **Configuration** menu, choose **Service Request Configuration**.
2. Switch to the **Service Catalog** tab.
3. Click **Add Service**.
4. Configure the options on the **Service Description** page as follows, then click **Next**:

**Name:** Add Azure Public Cloud

**Description:** Adds a managed system based on the specified Azure subscription ID.

**Icon:** Choose as appropriate.

**Categories:** Choose as appropriate.



The screenshot shows the 'Add Service: Add Azure Public Cloud' form. The left sidebar contains a menu with 'Service Description' selected. The main form area has the following fields and options:

- Name:** Add Azure Public Cloud
- Description:** Adds a managed system based on the specified Microsoft Azure Subscription ID.
- Icon:** Manage Icons. A row of icons is shown, including a multi-colored star, a red 'N' logo, the AWS logo, a blue cloud icon, and a share icon.
- Categories:** Manage Categories. A grid of checkboxes is shown:
  - ☐ Windows
  - ☐ Linux
  - ☐ Database
  - ☐ Production
  - ☐ Development
  - ☐ Multi Tier
  - ☒ Public
  - ☐ Private
  - ☐ RHEL

At the bottom of the form are buttons for 'Help', 'Back', 'Next', 'Finish', and 'Cancel'.

5. On the **Components** page, click **Add** > **New Component Type**.
6. Configure the new component type as follows, then click **Add to Service**:

**Name:** Add Azure Subscription

**Description:** Custom component for user-added Azure subscription.

**Annual Cost:** Set as appropriate (you must use 0.00 to not apply a cost.)

The screenshot shows the 'Create New Component Type' dialog box. It contains the following fields and buttons:

- Name:** Add Azure Subscription
- Description:** Custom component for user-added Azure subscription.
- Annual Cost:** 0.00

At the bottom are buttons for 'Help', 'Add to Service', and 'Cancel'.

7. Click **Next** and then switch to the **Attributes** tab.
8. Click **Add Attributes** and select **Application ID**, **Name**, **Subscription ID** and **Tenant ID**. Click **OK**.

**Edit Service: Add Azure Public Cloud**

Service Description  
Component Blueprints  
**Add Azure Subscription**  
Deployment  
Visibility  
Summary

**Infrastructure** **Attributes** **Form**

Assign metadata with custom attributes and groups. On the Form tab, you can allow requesters to set values for custom attributes.

Attribute Name	Default Value
Application ID:	<input type="text"/>
Name:	<input type="text"/>
Subscription ID:	<input type="text"/>
Tenant ID:	<input type="text"/>

[Add Attributes](#) [Manage Attributes](#)

Group Name	Default Value
------------	---------------

[Add Groups](#) [Manage Groups](#)

[Help](#) [Back](#) [Next](#) [Finish](#) [Cancel](#)

9. Switch to the **Form** tab.
10. Under the **Toolbox** section click **Name**, **Application ID**, **Subscription ID** and **Tenant ID** to add them to the form. Mark them as **Required**.
11. Again, under the **Toolbox**, click **Input Text field**. Enter the **Display Label API Key**, and enable **Hide User Input** and **Required**.

**Infrastructure** **Attributes** **Form**

Allow requesters to modify the default component settings configured on the other tabs.

↑  
↑  
↓  
↓

(Custom Attribute) \* [Edit](#) [Delete](#)  
**Name**

(Custom Attribute) [Edit](#) [Delete](#)  
**Application ID**

(Custom Attribute) \* [Edit](#) [Delete](#)  
**Subscription ID**

(Custom Attribute) \* [Edit](#) [Delete](#)  
**Tenant ID**

aa (Input Text Field) \* [Edit](#) [Delete](#)  
**API Key**

Display Label:

Maximum Characters:

Number of Lines:

Hide User Input: ☒ Enable if this is a password field

Required: ☒

[OK](#) [Cancel](#)

[Back](#) [Next](#) [Finish](#) [Cancel](#)

12. Sort the form elements using the arrow controls and click **Next**.
13. On the **Deployment** page, choose **None** for the **Completion Workflow**. Click **Next**.

14. On the **Visibility** page, choose to publish globally or to specific organizations, groups and users. Click **Next**.
15. Click **Finish**.

## Creating the Completion Workflow

Finally, create a completion workflow to link to the service and perform automation.

1. Under the **Configuration** menu, click **Service Request Configuration**.
2. Switch to the **Completion Workflow** tab.
3. Click **Add**.
4. Set the **Name** as *Add Azure Managed System* and choose to **Apply this workflow: after a Custom Component is deployed**. Click **Next**.

**Completion Workflow Configuration**

**Name**  
Provide a name for this workflow.

**Name**  
Add Azure Managed System

**Apply this workflow:**  
after a Custom Component is deployed

Allows you to specify actions to be carried out after an custom component is deployed.

Help Back Next Cancel

5. Click **Add > Execute Script** and configure the step as follows, then click **Next**:

**Name:** Execute API Call

**Step Execution:** Always Execute

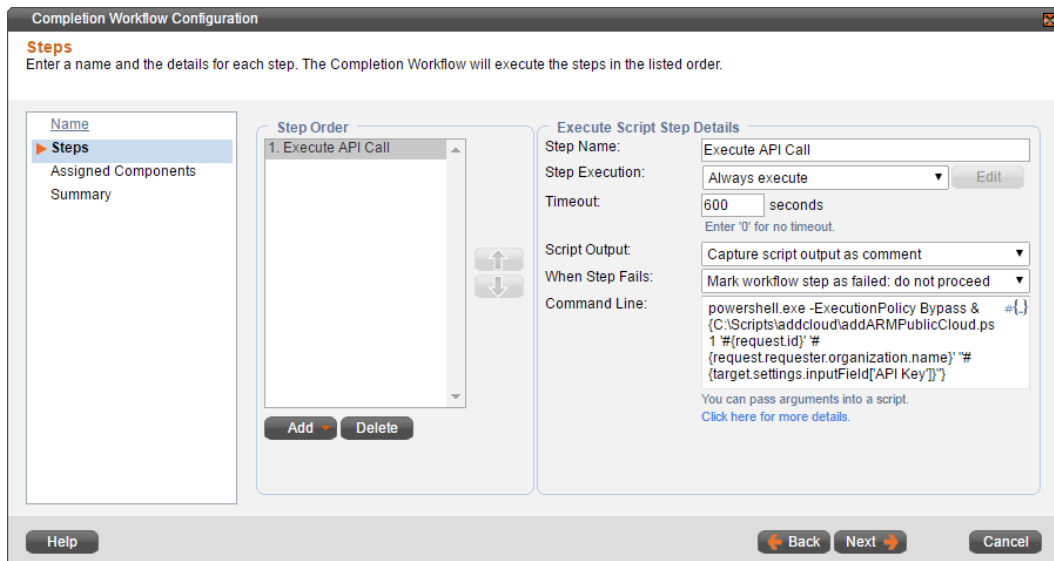
**Timeout:** 600 seconds

**Script Output:** Capture script output as comment

**When Step Fails:** Mark workflow step as failed: do not proceed

**Command Line:**

```
powershell.exe -ExecutionPolicy Bypass
&{C:\Scripts\addcloud\addARMPublicCloud.ps1 '#{request.id}'}
 '#{request.requester.organization.name}'
 '#{target.settings.inputField['API Key']}]"} }
```



**Completion Workflow Configuration**

**Steps**  
Enter a name and the details for each step. The Completion Workflow will execute the steps in the listed order.

**Left Panel:** Name, Steps (selected), Assigned Components, Summary

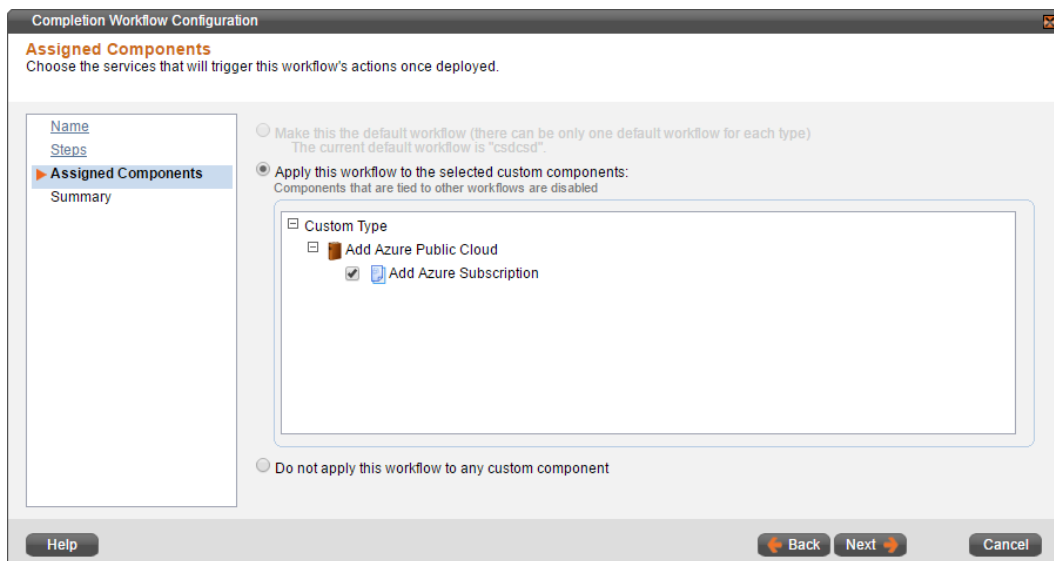
**Step Order:** 1. Execute API Call

**Execute Script Step Details:**

- Step Name: Execute API Call
- Step Execution: Always execute
- Timeout: 600 seconds
- Script Output: Capture script output as comment
- When Step Fails: Mark workflow step as failed: do not proceed
- Command Line: powershell.exe -ExecutionPolicy Bypass & {C:\Scripts\addcloud\addARMPublicCloud.ps 1 #{request.id} # {request.requester.organization.name} # {target.settings.inputField[API Key]}}

Buttons: Add, Delete, Back, Next, Cancel, Help

6. Check **Apply this workflow to the selected custom components** and enable **Add Azure Subscription**. Click **Next**.



**Completion Workflow Configuration**

**Assigned Components**  
Choose the services that will trigger this workflow's actions once deployed.

**Left Panel:** Name, Steps, Assigned Components (selected), Summary

**Options:**

- ☐ Make this the default workflow (there can be only one default workflow for each type). The current default workflow is "csdcscd".
- ☒ Apply this workflow to the selected custom components: Components that are tied to other workflows are disabled.
- ☐ Do not apply this workflow to any custom component

**Custom Type:**

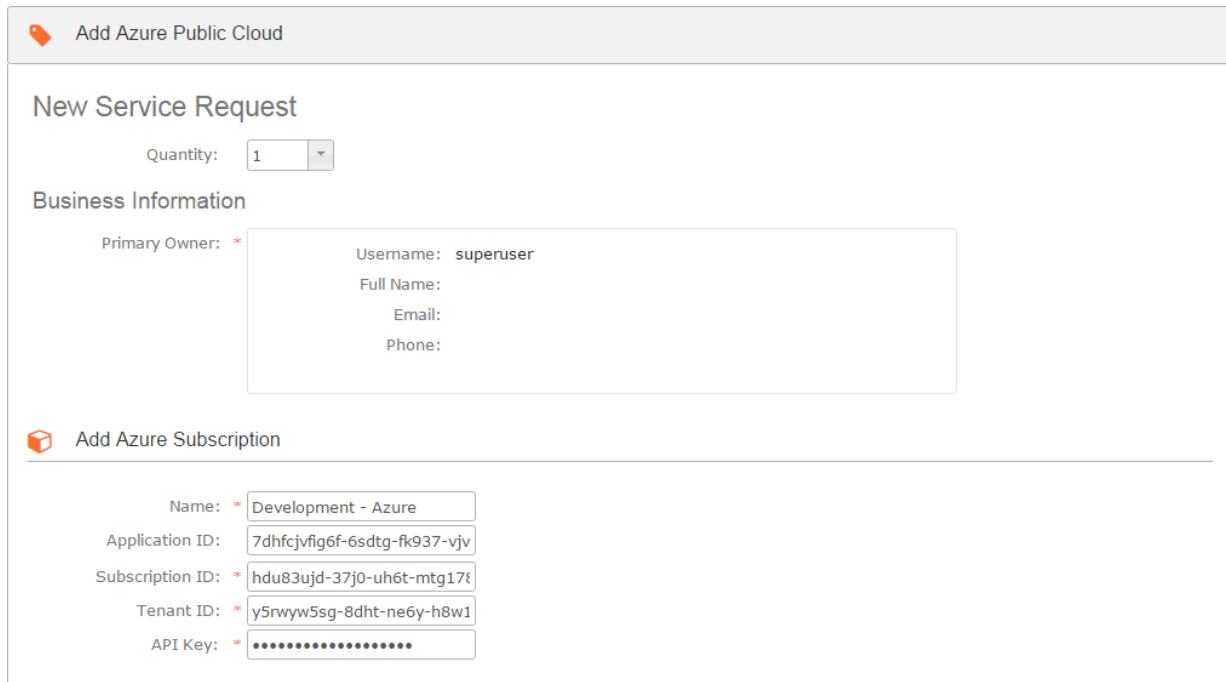
- ☒ Add Azure Public Cloud
- ☒ Add Azure Subscription

Buttons: Back, Next, Cancel, Help

7. Click **Finish**.

## Requesting the Service

When users request the service, they will use the forms as designed in the previous steps.



**Add Azure Public Cloud**

### New Service Request

Quantity: 1

#### Business Information

Primary Owner: \*

Username: superuser

Full Name:

Email:

Phone:

#### Add Azure Subscription

Name: \* Development - Azure

Application ID: 7dhfcjvfig6f-6sdtg-fk937-vjv

Subscription ID: \* hdu83ujd-37j0-uh6t-mtg17f

Tenant ID: \* y5rwyw5sg-8dht-ne6y-h8w1

API Key: \*

The **Name** will be used as the label for the managed system wherever it appears in vCommander. Typically, Service Portal users will not often see this label. The other values are used to connect vCommander to Azure.

Once the completion workflow finishes, the managed system appears in vCommander. All VMs in the managed system are assigned to the requester's organization. This means that any user who does not have the Service Portal permission **Show All Organization Services** will not be able to view the VMs, as users are not assigned individual ownership. If individual ownership is required, it must be assigned afterwards.

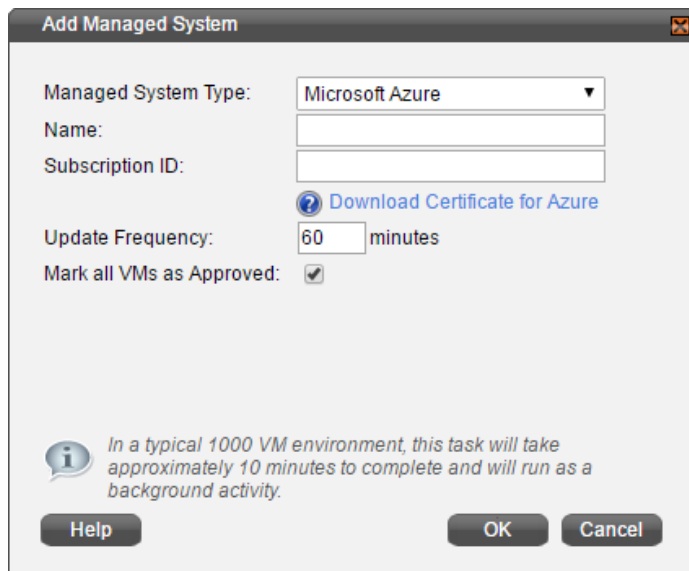
## Brokering Microsoft Azure Classic

This section covers the configuration needed to allow Service Portal users to add their own Microsoft Azure Classic managed system to vCommander.

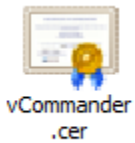
### Downloading the vCommander Azure Certificate

The approval workflow for this solution requires that you provide the vCommander certificate to the Service Portal user, so that they may upload it to the ASM portal to confirm the identity of the vCommander application server. Download the certificate by following the procedure below. You only need do so once, because the certificate does not change.

1. Login to vCommander using a **superuser** account.
2. Under the **Views** menu, choose **Operational**.
3. Right-click the top level of the tree and choose **Add Managed System**.
4. Select **Microsoft Azure** as the **Managed System Type**.
5. Click **Download Certificate for Azure**.



6. Save the **vCommander.cer** file to a known location, where vCommander admins approving the requests will be able to access it.



### Creating the vCommander Custom Attributes

This solution uses custom attributes to capture the Subscription Name and Subscription ID. Follow the steps below to create these.

1. Under the **Configuration** menu, choose **Custom Attributes**.
2. Click **Add**.
3. Enter *Subscription Name* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form

**Edit in Service Portal:** Enabled

The screenshot shows the 'Configure Attribute' dialog box with the 'Define Attribute' tab selected. The 'Name' field contains 'Subscription Name' and the 'Description' field contains 'Captures public cloud subscription name.' The 'Type' dropdown is set to 'Text' and the 'Applies To' dropdown is set to 'Form'. The 'Edit in Service Portal' checkbox is checked, with a note: 'Service Portal users can set custom attributes if their role has permission to set custom attributes.' At the bottom, there are 'Help', 'Back', 'Next', and 'Cancel' buttons.

4. Choose **Free Form** and click **Finish**.

The screenshot shows the 'Configure Attribute' dialog box with the 'Configure Attribute' tab selected. The 'Free Form' radio button is selected, with the text 'User input will not be validated' below it. The 'Specific Format' radio button is unselected, with the text 'Validate user input using Regular Expression below' below it. At the bottom, there are 'Help', 'Back', 'Finish', and 'Cancel' buttons.

5. Back on the **Custom Attributes** page, click **Add**.
6. Enter *Subscription ID* as the **Name** and optionally add a meaningful **Description**. Configure as follows, then click **Next**:

**Type:** Text

**Applies To:** Form  
**Edit in Service Portal:** Enabled

**Configure Attribute**

**Define Attribute**  
Define the attribute name, description, and type.

**Define Attribute**  
Configure Attribute

Name: Subscription ID

Description: Subscription ID for Azure managed systems.

Type: Text

Applies To: Form

Edit in Service Portal: ☒ Service Portal users can set custom attributes if their role has permission to set custom attributes.

Help Back Next Cancel

7. Choose **Free Form** and click **Finish**.



## Creating the Service Catalog Entry

Next, create the service catalog entry so that your users have access to the forms they need to request that their Azure subscription be added.

1. Under the **Configuration** menu, choose **Service Request Configuration**.
2. Switch to the **Service Catalog** tab.
3. Click **Add Service**.
4. Configure the options on the **Service Description** page as follows, then click **Next**:

**Name:** Add Azure Classic Public Cloud

**Description:** Adds a managed system based on the specified Azure subscription ID.

**Icon:** Choose as appropriate.

**Categories:** Choose as appropriate.

The screenshot shows the 'Add Service: Add Azure Public Cloud' form. The 'Service Description' tab is selected in the left sidebar. The form contains the following fields and options:

- Name:** Add Azure Public Cloud
- Description:** Adds a managed system based on the specified Microsoft Azure subscription ID.
- Icon:** A selection of icons including a database, MySQL, SAP, a cloud with a plus sign (selected), and a share icon.
- Categories:** A grid of checkboxes for Windows, Linux, Database, Production, Development, Multi Tier, Public (checked), and Private.

At the bottom, there are buttons for 'Help', 'Back', 'Next', 'Finish', and 'Cancel'.

5. On the **Components** page, click **Add > New Component Type**.
6. Configure the new component type as follows, then click **Add to Service**:

**Name:** Add Azure Classic Subscription

**Description:** Custom component for user-added Azure subscription.

**Annual Cost:** Set as appropriate (you must use 0.00 to not apply a cost.)

The screenshot shows the 'Create New Component Type' dialog box. It contains the following fields and buttons:

- Name:** Add Azure Subscription
- Description:** Custom component for user-added Azure subscription.
- Annual Cost:** 0.00

At the bottom, there are buttons for 'Help', 'Add to Service', and 'Cancel'.

7. Click **Next** and then switch to the **Attributes** tab.

8. Click **Add Attributes** and select **Subscription Name** and **Subscription ID**. Click **OK**.

Infrastructure Attributes **Form**

Assign metadata with custom attributes and groups. On the Form tab, you can allow requesters to set values for custom attributes.

Attribute Name	Default Value
Subscription ID:	<input type="text"/>
Subscription Name:	<input type="text"/>

+ Add Attributes Manage Attributes

Group Name	Default Value

+ Add Groups Manage Groups

Back Next Finish Cancel

9. Switch to the **Form** tab.
10. Under the **Toolbox** section, click **Subscription Name** and **Subscription ID** to add them to the form. Mark them as **Required**.
11. Add a **Header** with the text *Important Note*.
12. Add a **Text** element with the following content: *Once your request is approved, you will be emailed a certificate which you must upload to Azure, using the legacy portal. Once you confirm that you have done so, the process will complete.*

Infrastructure Attributes **Form**

Allow requesters to modify the default component settings configured on the other tabs.

↑  
↓  
↕  
↕  
↓  
↓

(Custom Attribute)  
Subscription Name

(Custom Attribute)  
Subscription ID

H (Header)  
Important Note

Aa (Text)  
Once your request is approved, you will be emailed a certificate which you must upload to Azure. Once you confirm that you have done so, the process will complete.

Edit Delete  
Edit Delete  
Edit Delete  
Edit Delete

Back Next Finish Cancel

13. Sort the form elements using the arrow controls and click **Next**.
14. On the **Deployment** page, choose **None** for the **Completion Workflow**. Click **Next**.

15. On the **Visibility** page, choose to publish globally or to specific organizations, groups and users. Click **Next**.
16. Click **Finish**.

## Creating the Approval Workflow

Next, you need to make sure you have configured two extra steps in the approval workflows that will cover some manual actions which must be performed:

1. A vCommander administrator downloading an Azure certificate and delivering the certificate to the requester.
2. The requester acknowledging that they have uploaded the Azure certificate.

Within the relevant approval workflow(s):

1. Click **Add > Send Approval Email**.
2. Configure the step as follows, then click **Next**:

**Step Name:** vCommander Admin Sends Cert

**Step Execution:** Execute when conditions are met:

```
#{request.services[1].components[1].settings.componentName} -eq "Add Azure"
```

**Address List:** Enter an email address for one or more vCommander administrators.

**Email Subject:** Service request #{request.id}: Azure Cert Needed

**Email Body:**

#{request.requester.name} has asked to add an Azure managed system to vCommander.

Please send the vCommander certificate to the following email address before approving this request:

#{request.requester.email}

3. Configure the step as follows, then click **Next**:

**Step Name:** Requester Confirms Cert

**Step Execution:** Execute when conditions are met:

```
#{request.services[1].components[1].settings.componentName} -eq "Add Azure"
```

**Address List:** #{request.requester.email}

**Email Subject:** Confirm Azure Certificate Upload

**Email Body:**

```
#{request.requester.name},
```

Please click the link below to approve this request only after you have uploaded the certificate provided to you by your vCommander administrators to Microsoft Azure's legacy portal.

Refer to the documentation available here:

[http://docs.embotics.com/index.html?adding\\_a\\_managed\\_system.htm#add\\_azure](http://docs.embotics.com/index.html?adding_a_managed_system.htm#add_azure)

Approval and Pre-Provisioning Workflow Configuration

**Steps**  
Enter a name and the details for each step. The Approval and Pre-Provisioning Workflow will execute the steps in the listed order.

**Name & Type**  
Assignment  
**Steps**  
Automation Options  
Summary

**Step Order**  
1. vCommander Admin Send  
2. Requester Confirms Cert  
Add Delete

**Send Approval Email Step Details**  
Step Name: Requester Confirms Cert  
Step Execution: Execute when conditions are met Edit  
Address List: #{request.requester.email}  
Email Subject: Confirm Azure Certificate Upload  
Email Body: #{request.requester.name},  
Please click the link below to approve this request only after you have uploaded the certificate provided to you by your vCommander administrators to Microsoft Azure's legacy portal.  
You can pass arguments into the address list, email subject and body.  
[Click here for more details.](#)

Help Back Next Cancel

4. Complete the wizard making any other necessary changes as appropriate, and save.

## Creating the Completion Workflow

Finally, create a completion workflow to link to the service and perform automation.

1. Under the **Configuration** menu, click **Service Request Configuration**.
2. Switch to the **Completion Workflow** tab.
3. Click **Add**.
4. Set the **Name** as *Add Azure Managed System* and choose to **Apply this workflow: after a Custom Component is deployed**. Click **Next**.

Completion Workflow Configuration

**Name**  
Provide a name for this workflow.

**Name**  
Steps  
Assigned Components  
Summary

Name: Add Azure Managed System

Apply this workflow: after a Custom Component is deployed

Allows you to specify actions to be carried out after an custom component is deployed.

Help Back Next Cancel

5. Click **Add > Execute Script** and configure the step as follows, then click **Next**:

**Name:** Execute API Call

**Step Execution:** Always Execute

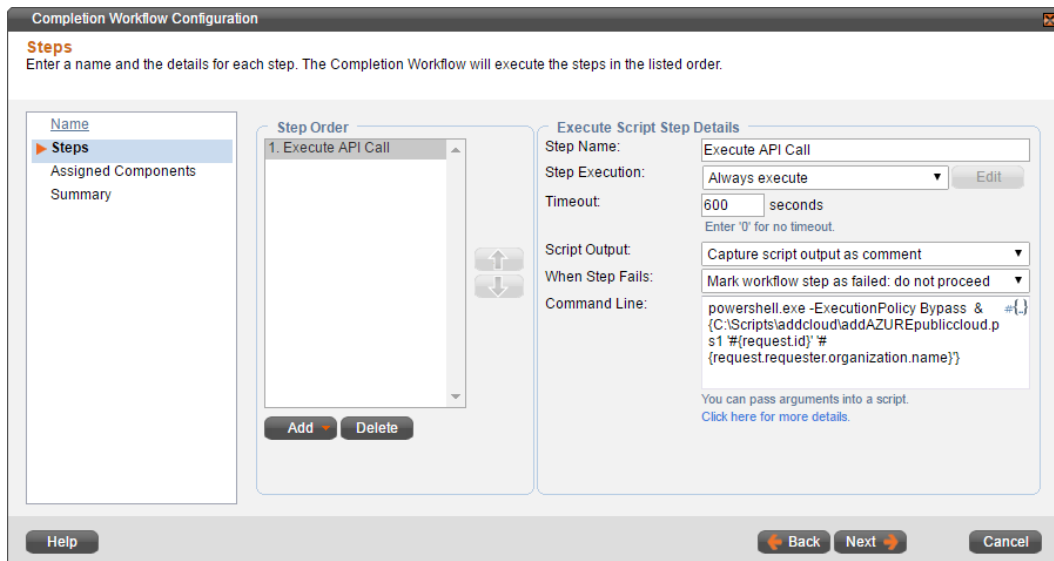
**Timeout:** 600 seconds

**Script Output:** Capture script output as comment

**When Step Fails:** Mark workflow step as failed: do not proceed

**Command Line:**

```
powershell.exe -ExecutionPolicy Bypass
&{C:\Scripts\addcloud\addAZUREpubliccloud.ps1 '#{request.id}'
 '#{request.requester.organization.name}' }
```



**Completion Workflow Configuration**

**Steps**  
Enter a name and the details for each step. The Completion Workflow will execute the steps in the listed order.

**Step Order**

Name
1. Execute API Call

**Execute Script Step Details**

Step Name: Execute API Call

Step Execution: Always execute

Timeout: 600 seconds

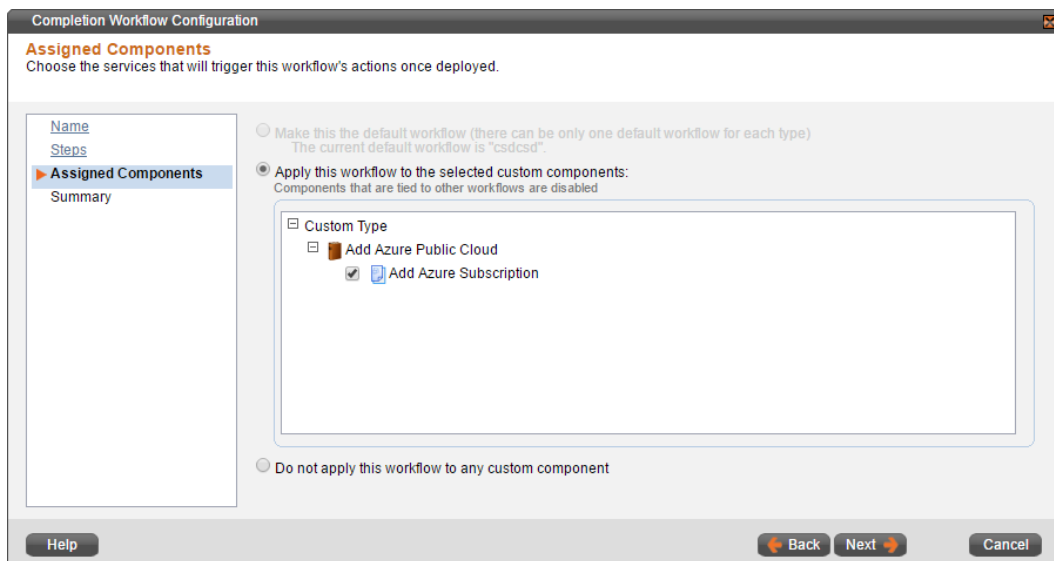
Script Output: Capture script output as comment

When Step Fails: Mark workflow step as failed: do not proceed

Command Line: powershell.exe -ExecutionPolicy Bypass & {C:\Scripts\addcloud\addAZUREpubliccloud.ps1 \$(request.id) \$(request.requester.organization.name)}

**Buttons:** Add, Delete, Back, Next, Cancel, Help

6. Check **Apply this workflow to the selected custom components** and enable **Add Azure Subscription**. Click **Next**.



**Completion Workflow Configuration**

**Assigned Components**  
Choose the services that will trigger this workflow's actions once deployed.

**Custom Type**

- ☐ Add Azure Public Cloud
- ☒ Add Azure Subscription

**Buttons:** Back, Next, Cancel, Help

7. Click **Finish**.

## Requesting the Service

When users request the service, they will use the forms as designed in the previous steps.

**Add Azure Classic Public Cloud**

### New Service Request

Quantity: 1

#### Business Information

Primary Owner: \*

Username: superuser

Full Name:

Email:

Phone:

#### Add Azure Classic Subscription

Subscription Name: \* Development - Azure

Subscription ID: \* 8-84f2-4c3d-a234-0d66ff37

#### Important Note

Once your request is approved, you will be emailed a certificate which you must upload to Azure, using the legacy portal. Once you confirm that you have done so, the process will complete.

The **Azure Subscription Name** will be used as the label for the managed system wherever it appears in vCommander. Typically, Service Portal users will not often see this label. The **Subscription ID** is used to connect vCommander to Azure.

Once the completion workflow finishes, the managed system appears in vCommander. All VMs in the managed system are assigned to the requester's organization. This means that any user who does not have the Service Portal permission **Show All Organization Services** will not be able to view the VMs, as users are not assigned individual ownership. If individual ownership is required, it must be assigned afterwards.