

Amenesik

Manifest Editor

User Guide Version 1.0a

Iain James Marshall
13/05/2016

Table des matières

Introduction	2
Getting Started.....	2
Manifest Editor Operation	2
Manifest Editor Layout.....	3
Manifest Editor	3
The list of manifests	3
The selected manifest.....	3
The list of nodes of the selected manifest.....	3
The selected node.....	3
The list of known nodes	4
Console Output	4
Configuration Actions	5
The List of Configuration Actions.....	5
The List of Release Actions.....	5
The Action Expression Definition	5
Manifest Editor Operations	7
Manifest Creation	7
Manifest Inspection	7
Manifest Configuration	7
Manifest Modification	8
Manifest Deletion	9
Examples	11
Example 1: Linux Server	11
Example 2: Simple Apache Web Server with Embedded MYSQL Server	12
Example 3: Compound Apache Web Server with Embedded MYSQL Server	13
Example 4: Load-balanced Scalable Apache Web Server with Embedded MYSQL Server	15
References	18
OCCI.....	18
TOSCA.....	18
CIMI	18
CORDS	18
AMENESIK	18

Introduction

This document describes the operations and behaviour of the new Amenesik Manifest Editor designed, developed and distributed by Amenesik, for use with the Amenesik Cloud Engine, an industrialised version of the Amenesik Cloud Engine Cloud Brokerage and Cloud Provisioning software initially developed under Apache 2 license during the CompatibleOne project.

A Manifest Document is an XML formatted document that is used for the description and subsequent introduction of the technical details of cloud system configurations for use by the Cloud Provisioning mechanisms of the Amenesik Cloud Engine. The required structure and contents of this XML document is defined by the accompanying XSD schema document that can be retrieved from the Amenesik website using the following link: <http://www.amenesik.Com/schemes/manifest.xsd>.

Prior to the advent of the Amenesik Manifest Editor, these documents were prepared by hand using a standard text editor, a process which requires a solid working knowledge of the manifest schema and its adjacent properties. Each document would reference the XSD schema to ensure validation of the contents and structure during the processing of the document by the Amenesik Cloud Engine document parser function of the Amenesik Remote Command Service.

The Amenesik Manifest Editor provides an integrated and interactive manifest development tool which alleviates the process of preparation of manifest documents by masking the underlying XML syntax and allowing the document designer to concentrate on the actual components of the cloud system, their properties and their configuration.

Getting Started

The Amenesik Manifest Editor tool is part of the Amenesik Enterprise Cloud Application Management tool set that is installed when an Amenesik Cloud Engine installation is deployed on dedicated or virtual infrastructure or hardware.

You must first connect to the “Amenesik Cloud Access”. Here you must provide valid authorisation credentials before being transferred to the “Amenesik Service Dashboard” where the Manifest Editor Tool can be activated by pressing the “M” button on the “Selected Contract” frame.

Manifest Editor Operation

The Manifest Editor is a Web Application that is to be accessed through a standard Web Browser and is connected to the SQL database and other storage systems of the underlying “Amenesik Cloud Engine” having direct access to all tables and components. Certain operations will be performed by direct calls to the OCCl interface of the corresponding component of the management model. In other cases, operations involving the execution of CORDSCRIPT programs will be performed through the Amenesik Remote Command Service.

Manifest Editor Layout

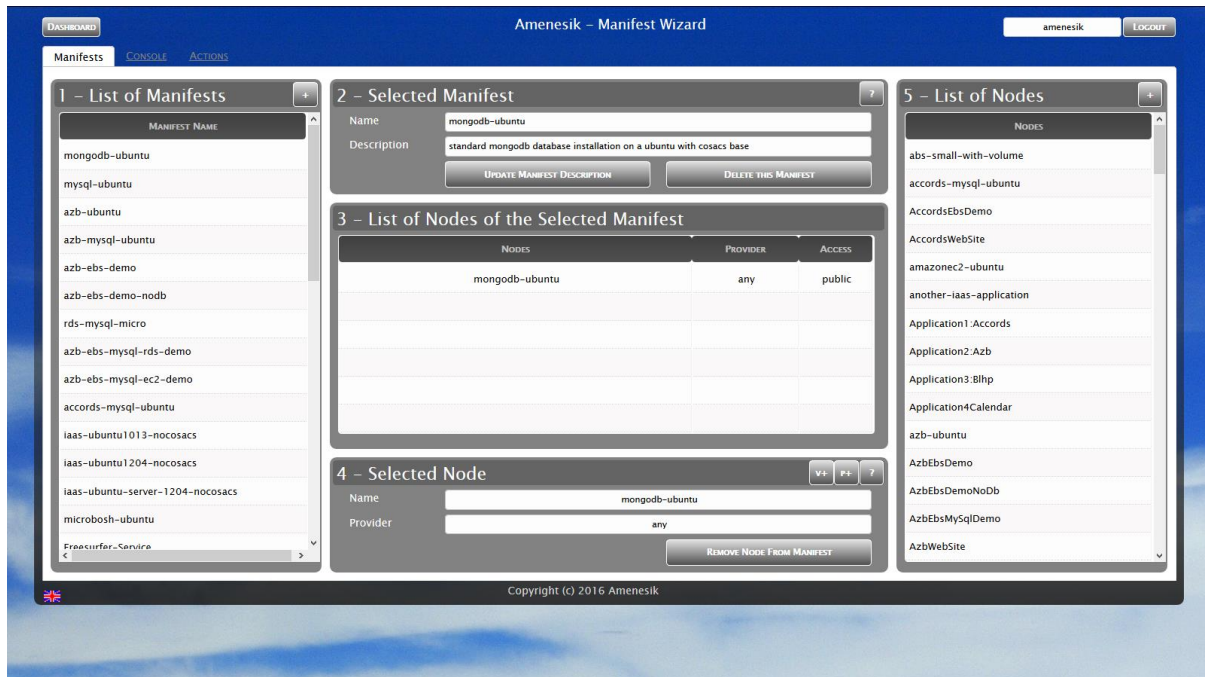
The Manifest Editor tool comprises a collection of “Tab Pages”. In the top left hand corner, the “Dashboard” button allows return to the “Amenesik Service Dashboard”.

In the top right hand corner the active user login account name is displayed and the “Logout” button allows return to the “Amenesik Cloud Access” page.

Manifest Editor

This is the active tab page when you first arrive in the Manifest Editor and gives access to all operations required for the creation, consultation and modification of manifests and their component nodes.

The following screenshot shows the main Manifest Editor tab page layout.



The “Manifest” tab page is divided into 5 distinct regions or frames that will be described in the following sections.

The list of manifests

Here you can navigate the list using the scroll bar and select a manifest as the active manifest by clicking on the manifest name in the list. The “+” button allows a new manifest to be created.

The selected manifest

Here the name and description of the selected manifest are displayed for consultation or modification along with two push buttons allowing update of modifications or deletion of the manifest. The “?” button allows visualisation of the manifest as an XML document in the Console Output page.

The list of nodes of the selected manifest.

You can navigate this list using the scroll bar and select a node as the active node by clicking on a node name in the list.

The selected node

This frame shows the node name and provider type and offers a push buttons allowing nodes to be added to or removed from the manifest. The “P+” button allows the addition of ports to the selected

node, the “V+” button allows the addition of a new storage volume and the “?” button allows examination and modification of the nodes properties.

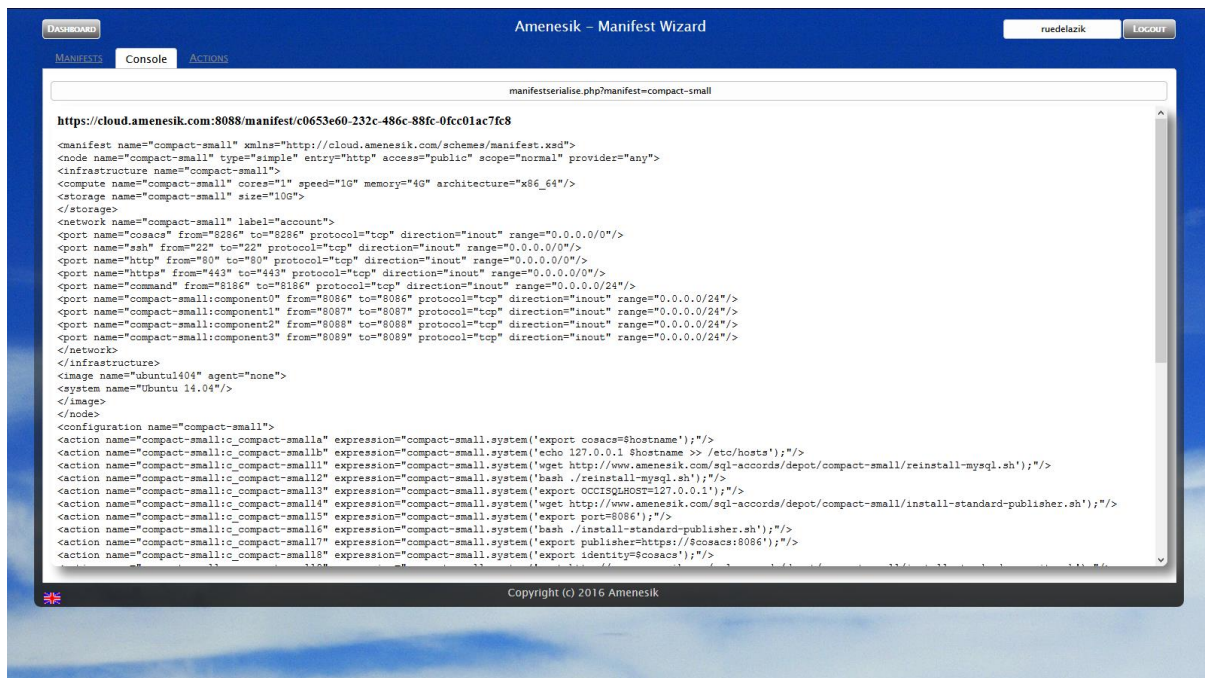
The list of known nodes

This list can be navigated using the scroll bar and a node can be selected by clicking on its name. The node will become the selected node allowing inspection and addition to the manifest. The “+” button allows creation of a new node.

Console Output

This tab page will be activated for the display of the results of any operations launched from the main editor page through the OCCI interface on the associated Amenesik Cloud Engine. You will be prompted to accept the launch of the operation and then the command will be sent to the Amenesik Remote Command Server for execution. Any resulting output returned will be displayed here for your inspection.

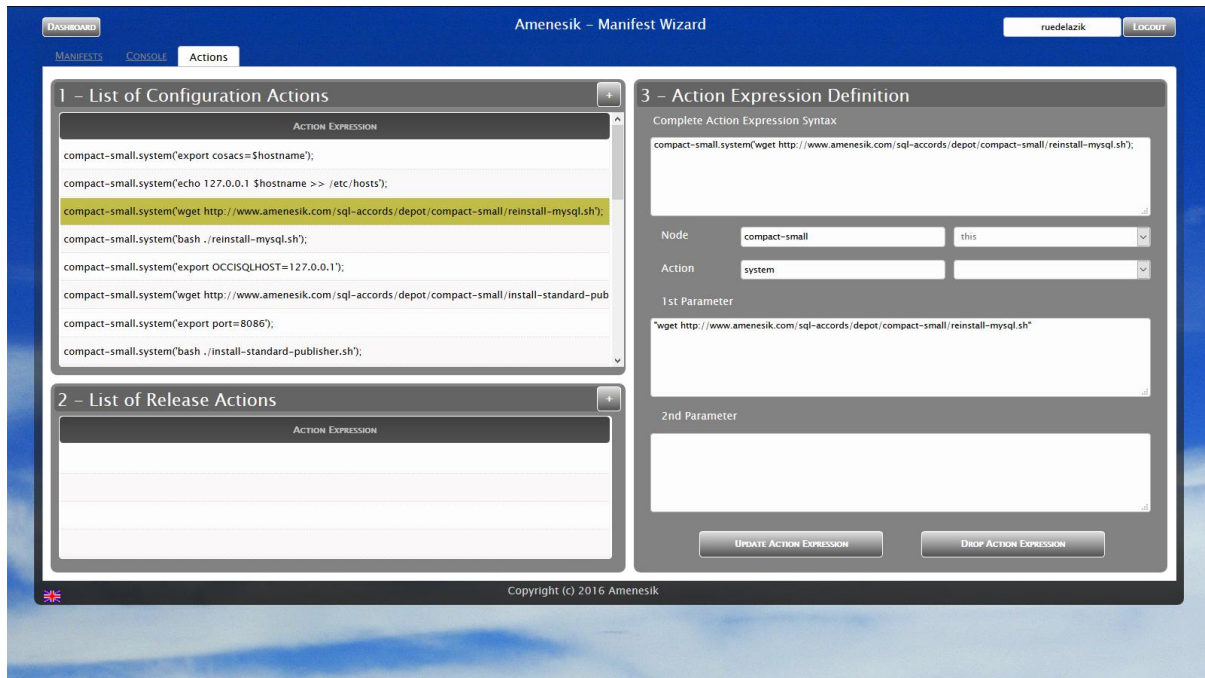
The following screen shot shows the Console Output tab page layout for the manifest inspection operation.



Configuration Actions

This tab page allows definition and management of configuration actions of the currently selected manifest description.

The following screenshot shows the Configuration Actions tab page layout.



The “Actions” is divided into the regions or frames described in the following sections.

The List of Configuration Actions

This list presents the ordered list of action expressions currently associated with the destruction phase of the selected manifest. The list can be navigated using the scroll bar and an action may be selected for consultation and modification. The “+” button allows a new configuration action to be created and added to the end of the list.

The List of Release Actions

This list presents the ordered list of action expressions currently associated with the destruction phase of the selected manifest. The list can be navigated using the scroll bar and an action may be selected for consultation and modification. The “+” button allows a new release action to be created and added to the end of the list.

The Action Expression Definition

This frame presents an exploded view of the different components of the selected action expression for consultation and modification. The push buttons allow modifications to be update or an action expression to be deleted.

The action expression is composed of the following elements:

Node Name

This element defines the name of the node for which the action is to be performed. The node can be selected from the list of nodes presented in the drop down list provided to this effect.

Action Name

The action name defines the action to be performed on the selected node. This may be selected from the drop down list provided to this effect.

1st Parameter

Most actions only require one parameter value. This may be either a constant string or the name of a node property. When a constant string value is provided it should be specified as a double quoted string which will be processed during the UPDATE operation replacing double quote characters as required by the single quote characters used by the internal action expression syntax.

2nd Parameter



Some actions require a second parameter which must conform to the composition rules specified for the 1st parameter value.

Manifest Editor Operations


This section of the document provides a description of the various high level operations that can be performed using the manifest editor.

Manifest Creation

To create a new manifest please complete the following steps:


1. A new manifest is to be created by pressing on the  button in the top right hand corner of the list of manifests on the left hand section of the main “Manifest Editor” page. The resulting manifest will be loaded into the editor as the current manifest. The name of the manifest will be set to “new-manifest” and the description will be set to “new-manifest-description”. The provisioning plan associated with the manifest will be created at the same time that the manifest element is created.
2. Modify the name of the manifest and its description as required to represent and describe the concept that it contains.
3. Save your modifications by pressing the “Update Manifest” button and the corresponding provisioning plan will be updated accordingly.
4. A new node can be created using the  button in the top right hand corner of the list of nodes in the right hand section of the main “Manifest Editor” page.
5. Fill in the information required by the Node definition dialog box and in particular set the name of the node. When you have finished then save the node by clicking on the “Update Node” button. The firewall element associated with the node will be created during the node creation operation. The application VM element associated with the image description of the new node will be created when the new node is created.
6. Select the node in the list of nodes to make it the currently select node in the central frame and then click on the “Add Node to Manifest” button.
7. This simple manifest, comprising one single node, is now complete and may be used for the as the service description of a service level agreement for provisioning of the associated cloud infrastructure.

Manifest Inspection

The  button in the top right hand corner of the Selected Manifest upper central frame allows inspection of the currently selected manifest in standard XML document format. The request will be sent for processing by the underlying Amenesik Cloud Engine and the resulting output will be displayed in the Console Output page.

Manifest Configuration

The manifest configuration section describes the list of actions to be performed for the various nodes as they are provisioned during the construction of a cloud system instance. The details of the manifest configuration can be specified through the “Configuration Actions” page of the Manifest Editor tool.

A new configuration action can be added by clicking on the  button in the top right hand corner of the configuration actions list to the left of the Configuration Actions page. The default details of the created configuration action are presented in the right hand section of the screen for modification and adaptation.

1. The concerned node can be selected from the drop down list of nodes.
2. The required action type can be select from the drop down list of action types.

3. The first and second parameters may be specified as required in the corresponding data entry fields.
4. The modifications can be saved by clicking on the “Update Action” button.

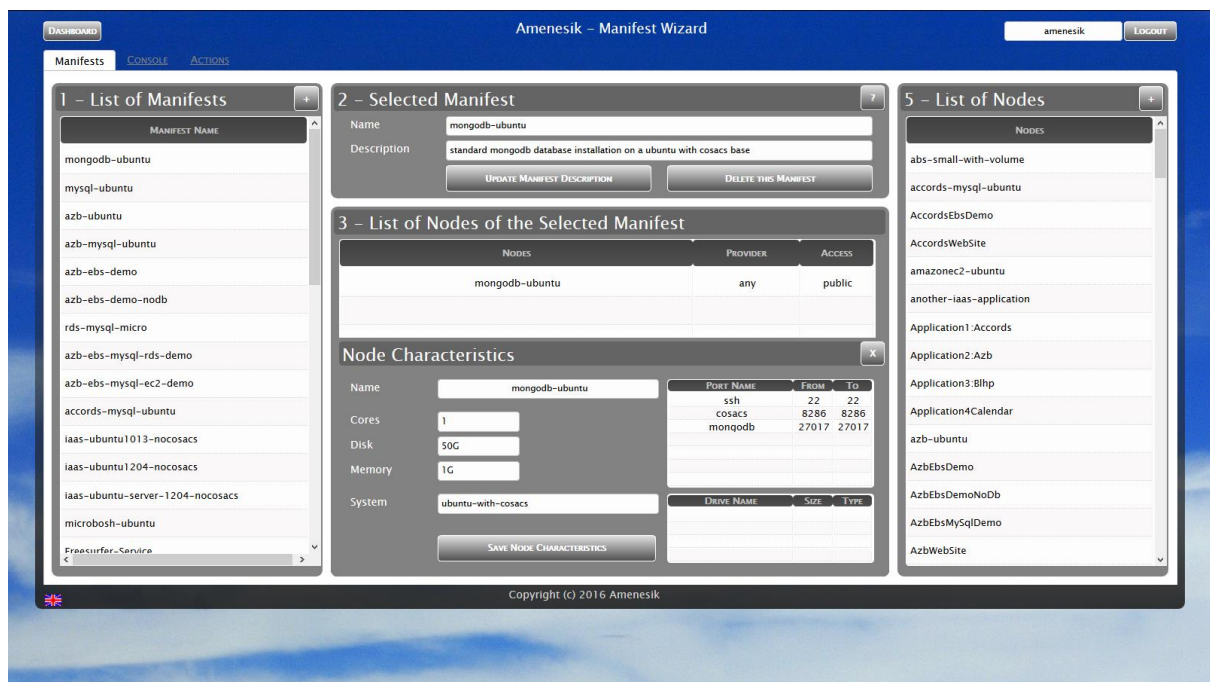
Expressions for either of the two parameter values, which require quoting, must be performed using the double quote character. This will be transformed to single quotes as required by the CORDS action expression syntax.

Manifest Modification

To modify an existing manifest please complete the following steps:

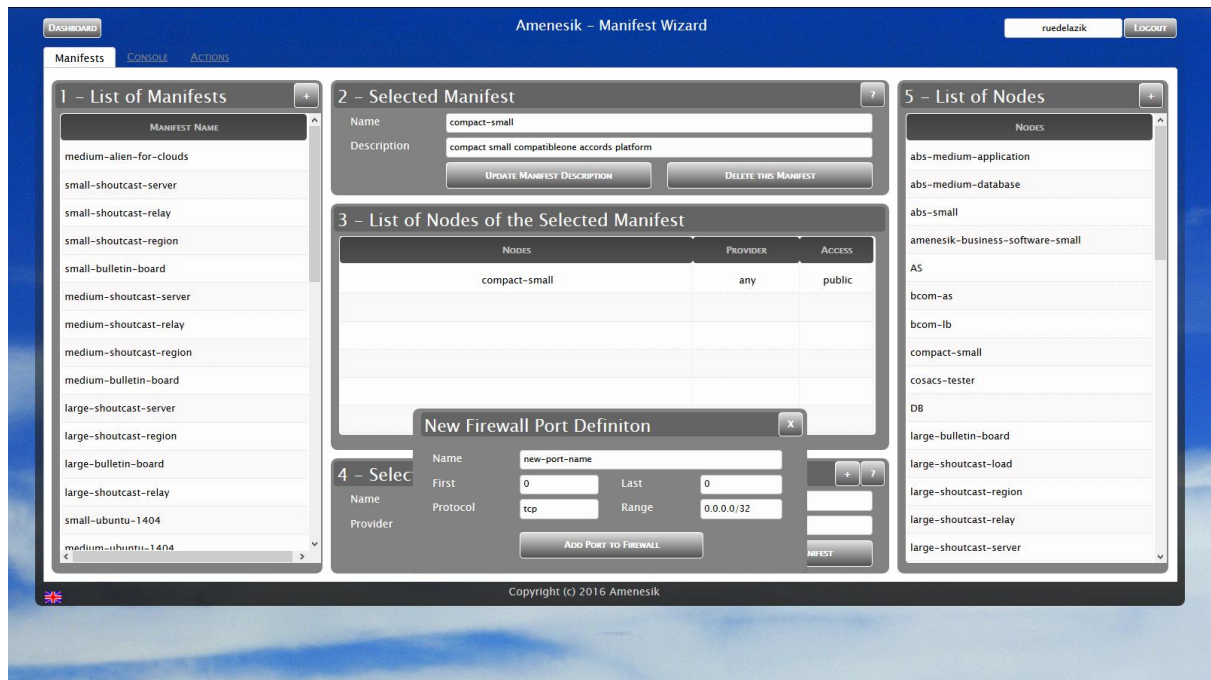
1. Select the required manifest from the list of manifests on the left hand section of the main “Manifest Editor” page.
2. Select or create new nodes and add them to the manifest as described during the manifest creation section. The properties of a manifest node may be made visible for inspection and subsequent modification by clicking the “?” button in the top left hand corner of the Selected Node central frame. The node properties dialog box will be displayed allowing the node name, compute cores, compute memory, disk size and operating system properties to be modified as required.

By clicking on the “Save Node Characteristics” button the modifications will be save to the database.

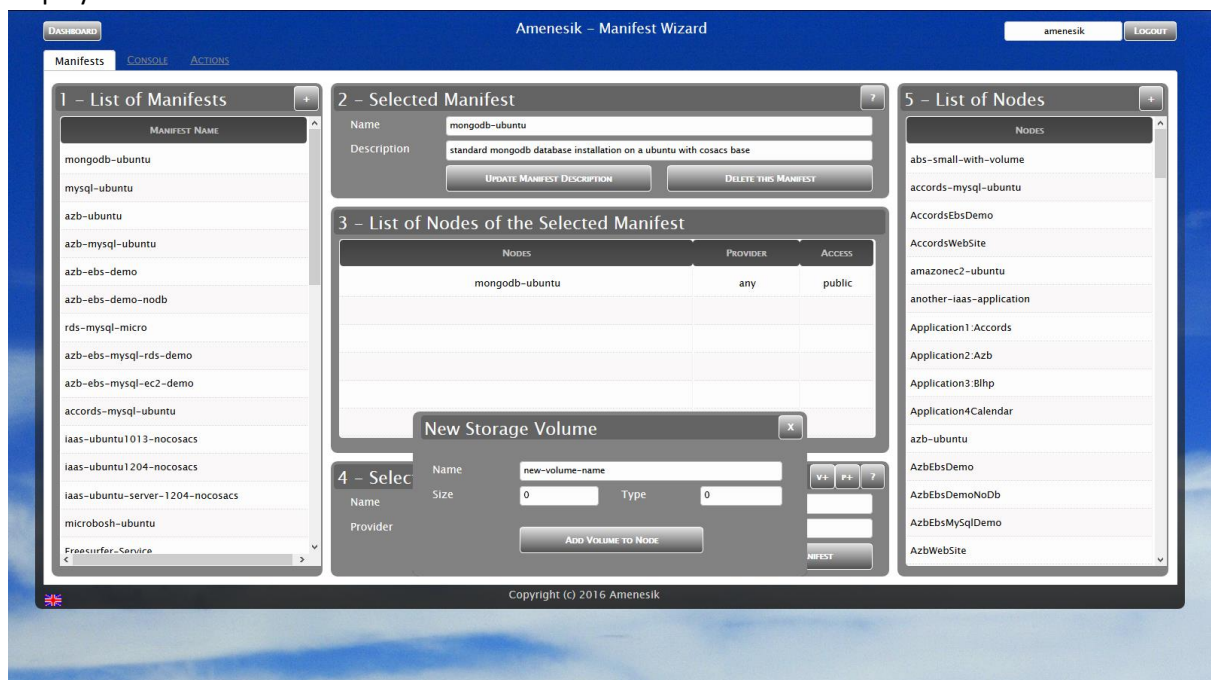


The node properties dialog also shows the list of firewall ports and the list of storage volumes currently defined for the currently selected node of the current manifest.

New port definitions may be added by clicking on the “P+” button in the top right hand corner of the “Selected Node” central frame and the New Firewall Port definition dialog box will be displayed as can be seen below.



New volume definitions may be added by clicking on the “V+” button in the top right hand corner of the “Selected Node” central frame and the New Storage Volume definition dialog box will be displayed as can be seen below.



3. Modify existing node properties.
4. Modify configuration actions as described in the configuration actions section.

Manifest Deletion

To delete an existing manifest please complete the following steps:

5. Select the required manifest from the list of manifests on the left hand section of the main “Manifest Editor” page.
6. Carefully check that this is the precise manifest that you wish to delete.

7. Click on the “Delete Manifest” button.

The manifest and all its associated sub components will be deleted along with all subordinate provisioning plans, node firewalls and the application VM elements of node image elements.

Examples

In this section of the document examples will be given showing how to use the manifest editor to create cloud system configurations. In the examples, the values of the parameters, shown in ***bold italic*** text are to be copied exactly as shown including eventual double quote characters. The term \$DEPOT should be replaced wherever it occurs by the name of the depot server from which the installation packages are to be retrieved. This may be ***<http://www.amenesik.com/blobstore/deployable>***.

Example 1: Linux Server

This example shows a manifest with a single node definition of a small LINUX Server with 1G of RAM, 10G of disk and running the UBUNTU 14.04 operating system. The node is accessible through the standard SSH port.



This example can be created by using the following workflow:

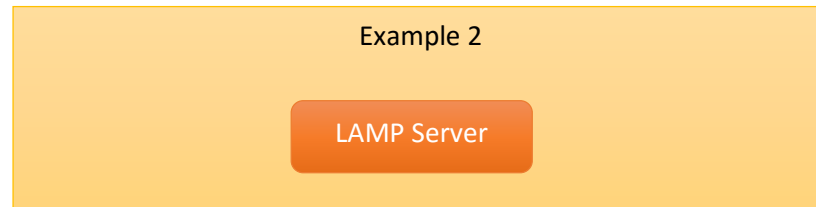
1. Activate the Manifest Editor
2. Create a new manifest
3. Set the manifest name to "example1" and the description to "simple linux server"
4. Save the name and description
5. Create a new node
6. Set the name to "example-linux-server"
7. Save the node characteristics
8. Add the TCP ports SSH=22, COSACS=8286
9. Choose the node
10. Add the node to the manifest.

Inspect the XML format of the contents of your manifest which should look like:

```
<manifest name="example1" xmlns="http://cloud.amenesik.com/schemes/manifest.xsd">
<node name="example-linux-server" type="simple" access="public" scope="normal" provider="any">
<infrastructure name="example-linux-server">
<compute name="example-linux-server" cores="1" speed="1G" memory="1G" architecture="x86_64"/>
<storage name="example-linux-server" size="10G">
</storage>
<network name="example-linux-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0/32"/>
</network>
</infrastructure>
<image name="example-linux-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<configuration name="example1"/>
<release name="example1"/>
<interface name="example1"/>
<account name="example"/>
</manifest>
```

Example 2: Simple Apache Web Server with Embedded MYSQL Server

This example shows a manifest with a single node definition of a Web Application running on a medium sized LINUX Server with 4G of RAM, 40G of disk and running the UBUNTU 14.04 operating system. In addition to the usual HTTP socket, the node is accessible through the standard SSH port.



This second example is similar to the first and can be created by using the following workflow:

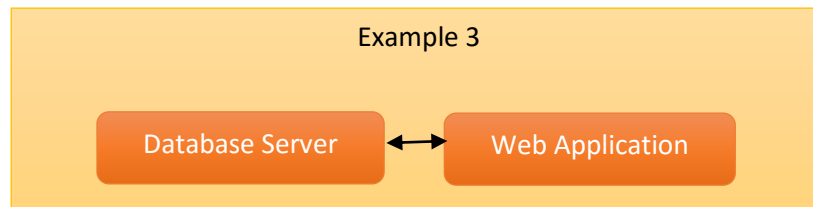
1. Activate the Manifest Editor
2. Create a new manifest
3. Set the manifest name to "example2" and the description to "example of lamp server"
4. Save the name and description
5. Create a new node
6. Set the name to "example-lamp -server"
7. Set the memory and disk properties
8. Save the node characteristics
9. Add the TCP ports SSH=22, COSACS=8286 and HTTP=80
10. Choose the node
11. Add the node to the manifest.
12. Activate the Actions Tab page
13. Add a new configuration item
14. Select the node name "example-lamp-server"
15. Select the action "system"
16. Specify the parameter 1 as ***"wget \$(DEPOT)install-apache.sh; bash ./install-apach.sh"***
17. Save the configuration action.

Inspect the XML format of the contents of your manifest which should look like:

```
<manifest name="example2" xmlns="http://cloud.amenesik.com/schemes/manifest.xsd">
<node name="example-lamp-server" type="simple" access="public" scope="normal" provider="any">
<infrastructure name="example-lamp-server">
<compute name="example-lamp-server" cores="1" speed="1G" memory="4G" architecture="x86_64"/>
<storage name="example-lamp-server" size="40G">
</storage>
<network name="example-lamp-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="http" from="80" to="80" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-lamp-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<configuration name="example2">
<action expression="example-lamp-server.system('wget
http://www.amenesik.com/blobstore/deployable/install-apache.sh; bash ./install-apach.sh');"/>
</configuration>
<release name="example2"/>
<interface name="example2"/>
<account name=" example"/>
</manifest>
```

Example 3: Compound Apache Web Server with Embedded MYSQL Server

This example shows a manifest of a Web Application and MYSQL Database each running on a medium sized LINUX Server with 4G of RAM, 40G of disk with the UBUNTU 14.04 operating system. In addition to the usual HTTP and HTTPS sockets, the application node is accessible through the standard SSH port. The database node is available on the MYSQL socket as well as the SSH port.



This third example builds on the principals of second and can be created by using the following workflow:

1. Activate the Manifest Editor
2. Create a new manifest
3. Set the manifest name to "example3" and the description to "example lamp server with database"
4. Save the name and description
5. Create a new node
6. Set the name to "example-database -server"
7. Set the memory and disk properties
8. Save the node characteristics
9. Add the TCP ports SSH=22, COSACS=8286 and MYSQL=3306
10. Choose the node
11. Add the node to the manifest.
12. Choose the node named "example-lamp-server"
13. Add the node to the manifest
14. Activate the Actions Tab page
15. Add a new configuration item
16. Select the node name "example-database-server"
17. Select the action "system"
18. Specify the parameter 1 as ***"wget \$(DEPOT)install-mysql.sh; bash ./install-mysql.sh"***
19. Save the configuration action.
20. Add a new configuration item
21. Select the node name "example-lamp-server"
22. Select the action "system"
23. Set the value of parameter 1 as ***"wget \$(DEPOT)install-apache.sh; bash ./install-apach.sh"***
24. Save the configuration item

Inspect the XML format of the contents of your manifest which should look like:

```

<manifest name="example3" xmlns="http://cloud.amenesik.com/schemes/manifest.xsd">
<node name="example-database-server" type="simple" access="public" scope="normal"
provider="any">
<infrastructure name="example-database-server">
<compute name="example-database-server" cores="1" speed="1G" memory="4G"
architecture="x86_64"/>
<storage name="example-database-server" size="40G">
</storage>
  
```

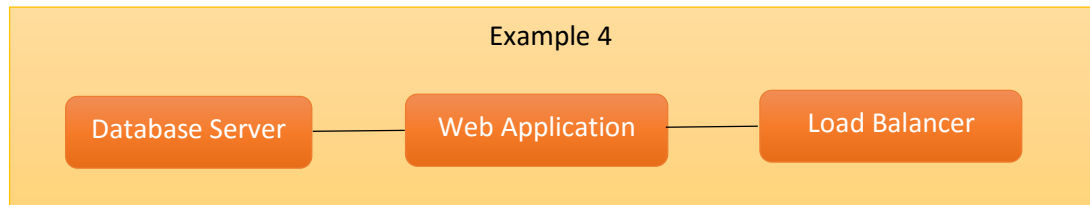
```

<network name="example-database-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="mysql" from="3306" to="3306" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-database-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<node name="example-lamp-server" type="simple" access="public" scope="normal" provider="any">
<infrastructure name="example-lamp-server">
<compute name="example-lamp-server" cores="1" speed="1G" memory="4G" architecture="x86_64"/>
<storage name="example-lamp-server" size="40G">
</storage>
<network name="example-lamp-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="http" from="80" to="80" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-lamp-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<configuration name="example3">
<action expression="example-database-server.system('wget
http://www.amenesik.com/blobstore/deployable/install-mysql.sh; bash ./install-mysql.sh');"/>
<action expression="example-lamp-server.system('wget
http://www.amenesik.com/blobstore/deployable/install-apache.sh; bash ./install-apache.sh');"/>
</configuration>
<release name="example3">
</release>
<interface name="example3">
</interface>
<account name=" example">
</account>
</manifest>

```

Example 4: Load-balanced Scalable Apache Web Server with Embedded MYSQL Server

This example shows a manifest of a load balanced and scalable Web Application with MYSQL Database each running on a medium sized LINUX Server with 4G of RAM, 40G of disk with the UBUNTU 14.04 operating system. In addition to the usual HTTP and HTTPS sockets, the application node is accessible through the standard SSH port. The database node is available on the MYSQL socket as well as the SSH port. An addition node is required for the load balancer and scalability engine and will be the client facing endpoint of the system.



This fourth example builds on the elements of the second and third examples and can be created by using the following workflow:

1. Activate the Manifest Editor
2. Create a new manifest
3. Set the manifest name to "example4" and the description to "example load balanced web application with database"
4. Save the name and description
5. Choose the node named "example-database-server"
6. Add the node to the manifest
7. Choose the node named "example-lamp-server"
8. Add the node to the manifest
9. Create a new node
10. Set the name to "example-load-balancer"
11. Set the memory and disk properties
12. Save the node characteristics
13. Choose the node
14. Add the TCP ports SSH=22, COSACS=8286, COOL=8386 and HTTP=80
15. Add the node to the manifest.
16. Activate the Actions Tab page
17. Add a new configuration item
18. Select the node name "example-database-server"
19. Select the action "system"
20. Specify the parameter 1 as ***"wget \$(DEPOT)install-mysql.sh; bash ./install-mysql.sh"***
21. Save the configuration action.
22. Add a new configuration item
23. Select the node name "example-lamp-server"
24. Select the action "system"
25. Specify the parameter 1 as ***"wget \$(DEPOT)install-apache.sh; bash ./install-apach.sh"***
26. Save the configuration item.
27. Add a new configuration item
28. Select the node name "example-load-balancer"
29. Select the action "system"

30. Specify the parameter 1 as ***"wget \$(DEPOT)install-cool.sh"***
31. Save the configuration action.
32. Add a new configuration item
33. Select the node name "example-load-balancer"
34. Select the action "configure"
35. Specify the parameter 1 as ***example-lamp-server.contract***
36. Save the configuration action.
37. Add a new configuration item
38. Select the node name "example-load-balancer"
39. Select the action "export"
40. Specify the parameter 1 as ***small_lamp_server_contract***
41. Specify the parameter 2 as ***elastic_contract***
42. Save the configuration action.
43. Add a new configuration item
44. Select the node name "example-load-balancer"
45. Select the action "system"
46. Specify the parameter 1 as ***"bash ./install-cool.sh"***
47. Save the configuration action.

Inspect the XML format of the contents of your manifest.

```
<manifest name="example4" xmlns="http://cloud.amenesik.com/schemes/manifest.xsd">
<node name="example-database-server" type="simple" access="public" scope="normal"
provider="any">
<infrastructure name="example-database-server">
<compute name="example-database-server" cores="1" speed="1G" memory="4G"
architecture="x86_64"/>
<storage name="example-database-server" size="40G">
</storage>
<network name="example-database-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="mysql" from="3306" to="3306" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-database-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<node name="example-lamp-server" type="simple" access="public" scope="normal" provider="any">
<infrastructure name="example-lamp-server">
<compute name="example-lamp-server" cores="1" speed="1G" memory="4G" architecture="x86_64"/>
<storage name="example-lamp-server" size="40G">
</storage>
<network name="example-lamp-server" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="http" from="80" to="80" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-lamp-server" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<node name="example-load-balancer" type="simple" access="public" scope="normal"
provider="any">
<infrastructure name="example-load-balancer">
<compute name="example-load-balancer" cores="1" speed="1G" memory="4G" architecture="x86_64"/>
<storage name="example-load-balancer" size="40G">
</storage>
<network name="example-load-balancer" label="account" vlan="100M">
<port name="ssh" from="22" to="22" protocol="tcp" range="0.0.0.0/32"/>
<port name="cosacs" from="8286" to="8286" protocol="tcp" range="0.0.0.0/32"/>
<port name="cool" from="8386" to="8386" protocol="tcp" range="0.0.0.0/32"/>
<port name="http" from="80" to="80" protocol="tcp" range="0.0.0.0/32"/>
</network>
</infrastructure>
<image name="example-load-balancer" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
```

```

</network>
</infrastructure>
<image name="example-load-balancer" agent="none">
<system name="Ubuntu 14.10"/>
</image>
</node>
<configuration name="example4">
<action expression="example-database-server.system('wget
http://www.amenesik.com/blobstore/deployable/install-mysql.sh; bash ./install-mysql.sh');"/>
<action expression="example-lamp-server.system('wget
http://www.amenesik.com/blobstore/deployable/install-apache.sh; bash ./install-apache.sh');"/>
<action expression="example-load-balancer.system('wget
http://www.amenesik.com/blobstore/deployable/install-cool.sh');"/>
<action expression="example-load-balancer.configure(example-lamp-server.contract);"/>
<action
expression="example-load-balancer.export(example_lamp_server_contract,"elastic_contract");"/>
</configuration>
<release name="example4"/>
<interface name="example4"/>
<account name="example"/>
</manifest>

```

References

This section of the document provides a collection of links to cloud standards documentation and Amenesik support documents.

OCCI

The following documents are available from the OGF web site:

- OCCI CORE Version 1.1:
<https://www.ogf.org/documents/GFD.183.pdf>
- OCCI INFRASTRUCTURE Version 1.1 :
<https://www.ogf.org/documents/GFD.184.pdf>
- OCCI http Version 1.1 :
<https://www.ogf.org/documents/GFD.185.pdf>

TOSCA

The following documents are available from the OASIS web site

- TOSCA Version 1.1:
<http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf>
- TOSCA Namespace:
<http://docs.oasis-open.org/tosca/ns/2011/12>

CIMI

The following documents are available from the DMTF web site

- CIMI Version 1.1 :
http://www.dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.1.pdf

CORDS

The following documents are available from the CompatibleOne community web site:

- CORDS Version 1.1 :
<http://www.compatibleone.com/community/wp-content/uploads/2014/05/CordsReferenceManualV2.15.pdf>

AMENESIK

The following documents are available from the AMENESIK web site:

- Amenesik Enterprise Cloud (AEC) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikCloud.pdf>
- Amenesik Cloud Engine (ACE) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikCloudEngine.pdf>
- Amenesik Manifest Editor (AME) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikManifestEditor.pdf>
- Amenesik Agreement Editor (ASE) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikServiceEditor.pdf>
- Amenesik Service Dashboard (ASD) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikServiceDashboard.pdf>
- Amenesik Cloud Operator (ACO) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikCloudOperator.pdf>

- Amenesik Platform Editor (APE) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikPlatformEditor.pdf>
- Amenesik Platform Service (APS) Version 1.1:
<http://www.amenesik.com/cloud/AmenesikPlatformService.pdf>