Murano Project Documentation

Murano Project Documentation

v0.1

Publication date 2013-04-04

Abstract

This document is intended for individuals who wish to configure and use our product or intend to contribute.

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview

Welcome to Murano Project.

Intended Audience

This guide is intended to individuals who want to contribute to our project.

Document Change History

This version of the Murano Manual replaces and obsoletes all previous versions. The most recent changes are described in the table below:

Revision Date	Summary of Changes
April. 4, 2013	• Initial document creation.

Additional Resources

- Mirantis - Cloud Software [http://www.mirantis.com]

Chapter 2. Blueprint

Project Background

Enterprise customers frequently use Windows-based environments for their internal and external products. Configuration of the Windows environment is a complex task which usually requires a lot of effort from administrators. Windows setup consists of numerous services which might be tightly coupled to each other. While the automated installation of Windows services can be fairly straightforward, service configuration can be hard to automate because it requires a well-designed Windows architecture and deep knowledge of Windows services configuration.

Currently several open source solutions exists that can help to partially solve automation of Windows environment provisioning. In the world of OpenStack there is the Heat project, which is similar to Amazon Cloud Formation. Heat is an excellent tool for managing OpenStack cloud resources such as VM instances, security groups, and so on. It allows you to define all cloud resources in a single JSON template, then later maintain all of those resources by editing that template. Although the declarative template approach is well suited to OpenStack resources, it quickly becomes complex when it comes to application management.

Another option is a tool such as Chef or Puppet. These tools are flexible, but require you to have a deep knowledge of scripting and require a significant amount of effort to manually script or modify cookbooks for your specific environment configuration. This is manageable in a stable environment, but it becomes time-consuming and involves manual script coding when one needs to deploy various environments with rapidly changing configurations. Also Chef and Puppet require additional infrastructure to support them.

The biggest problem for both approaches above is in supporting multi-step configuration of services with circular dependencies required for correct configuration of Windows services. This can be solved by using external orchestration.

Another potential problem is the lack of UI functionality enabling creation and configuration of an environment without writing a script.

Proposal

Mirantis proposes to introduce a new service which will allow a non-experienced user to deploy reliable Windows based environments in a “push-the-button” manner. The key goal is to provide a UI and API enabling the deployment and operation of Windows Environments at the Windows Services abstraction level. The service should be able to orchestrate complex circular dependent cases in order to set up a complex Windows Environment with multiple dependant services.

The service will address following use cases:

- Self-provisioning of predefined Windows services with their dependencies
- Automation of administrative tasks during data center roll-out
- Custom windows application as a windows service

The solution will provide higher level of abstraction for manipulation Windows Environments. Key concepts are:

- Windows Service - a service such as Active Directory, MSSQL, or IIS, which usually consists of multiple virtual machines and has multiple dependencies.
- Windows Environment - a logical unit for all Services and represents a classical Windows Datacenter
- Windows VM instance - a VM which hosts a Windows Service. A Windows Service might be deployed

The Key Features of the Service are the following:

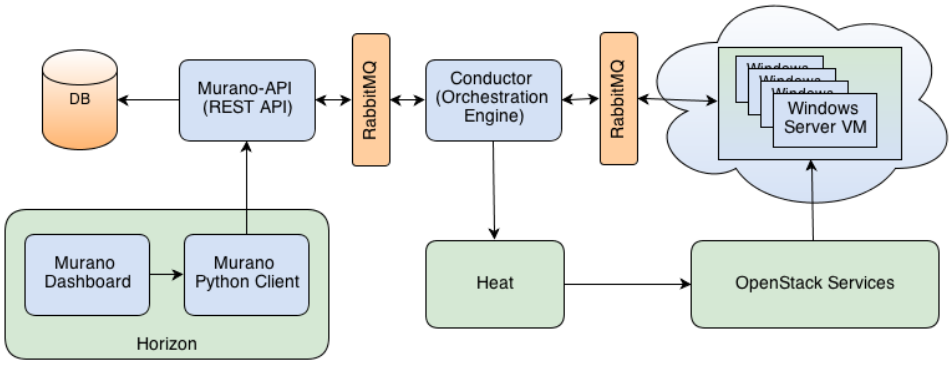
1. Native to OpenStack
2. Introduces abstraction level for Windows Environments
3. Supports Availability Zones and Disaster Recovery scenarios
4. Uses native Windows features for HA solutions

Architecture

The Murano Service communicates with the following OpenStack components:

- Horizon - provides a GUI with ability to use all Murano features;
- Keystone - authenticates users and provides the security token that is used to work with OpenStack, hence limiting the user abilities in Murano based on OpenStack privileges;
- Heat - is used to provision VMs and other OpenStack resources for Windows Environments;
- Glance - stores Windows Server VM images, with each image containing an installed OS and a set of scripts
- Quantum - provides the network configuration API
- Agent - provides agent functionality to communicate with the Orchestration Engine and executes tasks on VMs

Figure 2.1. Architecture



REST API

Murano exposes a service endpoint for communication with a client. It exposes API functions to manipulate objects such as environment and service.

This component is responsible for translating API function parameters to Object Model attributes and propagating the deployment status from the Orchestration Engine.

Object Model

An internal representation of Windows Services and Environments. All attributes and entities are described in the API specification.

Orchestration Engine

This is the core component which evaluates Object Model changes and creates a plan for implementing these changes on the instances or in the cloud. This component will support extensions via plug-ins. Plugins can add new services and extend existing services for integration. Currently there are two services which are already implemented as plugins. They are Active Directory and IIS Service.

Integration with Heat

Heat is a cloud resource management engine that allows you to manipulate resources that represent OpenStack entities (Security Groups, Instances, Floating IPs, Volumes, etc.) and some entities such as AutoScaling groups from a single point of control.

OpenStack resource provisioning is one of the steps required for environment deployment and Heat will be used for that purpose. Heat allows you to define all OpenStack resources in a single document that will be easy to maintain and will not require resorting to multiple OpenStack APIs while keeping the software configuration separate.

Windows on OpenStack

Windows works on KVM pretty smoothly, and with the RedHat-created open-source VirtIO drivers for Windows, it’s possible to work efficiently with KVM exposed devices.

In OpenStack’s Grizzly release, Microsoft’s hypervisor Hyper-V will be supported. The Hyper-V virtual switch will be also supported as a Quantum plug-in. From the performance viewpoint, Hyper-V Server 2012 compares very favorably with bare metal, processing just over 6% fewer transactions per second compared to the same workload running on a similarly configured physical server.

Also, unlike the current OpenStack, Hyper-V also natively supports Windows Clusters.

Chapter 3. Roadmap

Phase 1. Initial Version

(Release date: May 30th)

- Core Services: REST API, Orchestration Engine
- Horizon dashboard extension as plugin
- Integration with Heat
- Support single Data Center (no Disaster Recovery)
- Support the following Windows Services:
  - Active Directory - Single Domain with multiple domain controllers
  - IIS Server - single instance of IIS Server
  - IIS Cluster - multiple IIS instances with Load Balancing
  - ASP.NET Application Service - ASP.NET application installed on top of IIS

Phase 2. Stable Release

(3 month)

- Stabilize Core Services (bug fixing)
- Support API, UI extensibility
- UI design
- Workflow and recipes repository
- Data transfer between service instances
- Basic Service monitoring
- Additional Services:
  - MS SQL - single instance of Microsoft SQL Server or Pair of SQL Servers with DB mirroring

Phase 3

- Add more services
- Services Dependencies support
- Extended health monitoring
- Agent extensibility (allow 3rd party configuration tools)
- Basic Self-healing (actions on monitoring events)
- Additional Services

Phase 4

- Availability Zones support
- Auto-scaling for Windows services
- Security improvements

Chapter 4. API Specification

Revision Date	Summary of Changes
February 4, 2013	• Initial document creation
February 22, 2013	• Enhance API with latest architecture changes
March 06, 2013	• Fix specification according to remarks from Dmitry Teselkin
Jun 06, 2013	• ASP.NET Application, Web Server Farm and ASP.NET Application Farm Services Added, uri/address/endpoint corrections, hostname assignment section added

Introduction

Murano Service API is a programmatic interface used for interaction with Murano. Other interaction mechanisms like Murano Dashboard or Murano CLI should use API as underlying protocol for interaction.

Glossary

For detailed information about entities and terms used in this document, please refer first to the section called “Architecture”.

Environment	Environment is a set of logically related Services managed by a single tenant. Environment defines Windows environment boundaries. Services within single Environment may comprise some complex configuration while Services in different Environments are always independent from one another. Each Environment is associated with single OpenStack project (tenant).
Service	Service is building block of Windows environment. Service is a set of one or more Virtual Machines sharing a common purpose and configured together. Each service belongs to a single Environment and single Service Type. Services are comprised from one or more Service Units.
Service Type	Service type is definition for describing set of features exposed by service.
Service Unit	Service Units are the actual Windows Server VMs instantiated by OpenStack and then configured according to its Service Type (this may also correspond to one of predefined Windows Server roles).
Service Metadata	Service Metadata is a JSON-encoded definition of Environment, its Services and their Service Units along with all their attributes. Service Metadata may describe both current and the intended state of the Environment.
Session	All changes to environment done in scope of Session. After all changes to Environment state are accumulated, changes actually are applied only after session is deployed.

Return codes and errors

All REST API calls return the natural HTTP response codes for the operations, e.g. a successful GET returns a HTTP 200, a successful PUT returns a HTTP 201, a GET for a non-existent entity returns HTTP 404, unauthorized operations return HTTP 401 or HTTP 403, internal errors return HTTP 500.

Response of POSTs and PUTs

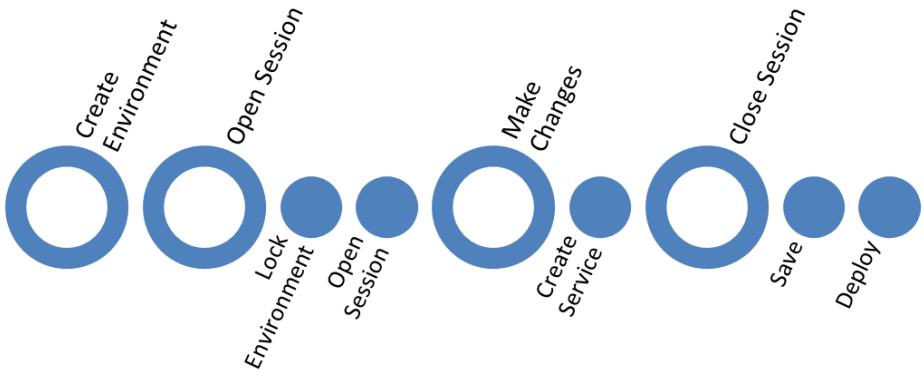
All POST and PUT requests by convention should return the created object (in the case of POST, with a generated ID) as if it was requested by GET.

Authentication

All requests include a Keystone authentication token header (X-Auth-Token). Clients must authenticate with Keystone before interacting with the Murano service.

Workflow

Figure 4.1. Sample Workflow



Let’s review a sample workflow (series of API calls) for creating new Environment with Active Directory Service deployment:

1. POST /environments/ - Creating new Environment
2. POST /environments/id/configure – Creating new configuration session for Environment
3. POST /environments/id/activeDirectory – Creating new ActiveDirectory service
4. POST /environments/id/sessions/session_id/deploy – Saving and deploying changes

Hostname assignment

Each Service Object definition may have an attribute "unitNamingPattern" that is used to control hostnames that will be assigned to spawned VM instances of the service.

Hostname pattern has the form of "name#" where "#" character is replaced with sequential number of unit within the service (starting with 1) and all other characters remain intact. For example Service with unitNamingPatter equal to "ad#-loc" will have units with hostnames "ad1-loc", "ad2-loc" etc.

"unitNamingPattern" attribute is optional. If it is omitted then a unique random hostname would be assigned.

API

Environment API

This section describes API calls for Environment management.

Get a List of existing Environments

Table 4.1. Environment Object

Attribute	Type	Description
id	string	Unique ID
name	string	User-friendly name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
tenant_id	string	OpenStack tenant ID
version	int	Current version
status	string	Deployment status: ready, pending, deploying

Call

Table 4.2. GET /environments Call

Method	URI	Description
GET	/environments	Get a list of existing Environments

Payload

None

Returns

This call returns list of environments. Only the basic properties are returned. For details see "Get Environment Detailed Information":

```
{
   "environments": [
       {
           "id": "0ce373a477f211e187a55404a662f968",
           "name": "dc1",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "tenant_id": "0849006f7ce94961b3aab4e46d6f229a",
```

```
            "version": 1,
            "status": "ready"
        },
        {
            "id": "c697bd2429304820a928d145aa42af59",
            "name": "dc2",
            "created": "2010-11-30T03:23:42Z",
            "updated": "2010-11-30T03:23:44Z",
            "tenant_id": "0849006f7ce94961b3aab4e46d6f229a",
            "version": 2,
            "status": "deploying"
        }
    ]
}
```


Create Environment instance

Table 4.3. Environment Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name

Call

Table 4.4. POST /environments Call

Method	URI	Description	
POST	/environments	Create new Environment	

Payload


```
{
  "name": "env1"
}
```


Returns

This call returns created environment:


```
{
    "id": "ce373a477f211e187a55404a662f968",
    "name": "env1",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
```

```
    "tenant_id": "0849006f7ce94961b3aab4e46d6f229a",
    "version": 0
}
```

Update Environment Instance

Table 4.5. Environment Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name

Call

Table 4.6. PUT /environments/<id> Call

Method	URI	Description
PUT	/environments/<id>	Update properties of Environment instance

Table 4.7. Error Response Codes

Code	Description
401	User is not authorized to access this tenant resources

Payload

```
{
  "name": "env1-changed"
}
```

Returns

This call returns modified environment object:

```
{
    "id": "ce373a477f211e187a55404a662f968",
    "name": "env1-changed",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
    "tenant_id": "0849006f7ce94961b3aab4e46d6f229a",
    "version": 0
}
```

Get Environment Instance Detailed Information

Call

Table 4.8. GET /environments/<id> Call

Method	URI	Description
GET	/environments/<id>	Returns detailed information about Environment including child entities

Table 4.9. Error Response Codes

Code	Description
401	User is not authorized to access this tenant resources

Payload

None

Returns

This call returns environment object with underlying services:

```
{
    "environments": [{
         "id": "0ce373a477f211e187a55404a662f968",
         "name": "dc1",
         "created": "2010-11-30T03:23:42Z",
         "updated": "2010-11-30T03:23:44Z",
         "tenant_id": "0849006f7ce94961b3aab4e46d6f229a",
         "version": 1,
         "status": "deploying",
         "services": [
             "activeDirectories": [{
                 "id": "96365940588b479294fe8e6dc073db04",
                 "name": "acme.dc",
                 "created": "2010-11-30T03:23:42Z",
                 "updated": "2010-11-30T03:23:44Z",
                 "status": "deploying",
                 "units": [{
                     "id": "d08887df15b94178b244904b506fe85b",
                     "isMaster": true,
                     "location": "west-dc"
                 }, {
                     "id": "dcf0de317e7046bea555539f19b8ea84",
                     "isMaster": false,
                     "location": "west-dc"
                 }]
             }]
         ]
```

```
    }]
}
```

Remove Environment

Call

Table 4.10. DELETE /environments/<id> Call

Method	URI	Description
DELETE	/environments/<id>	Remove specified Environment.

Table 4.11. Error Response Codes

Code	Description
401	User is not authorized to access this tenant resources

Payload

None

Returns

None

Environment Configuration API

Multiple sessions could be opened for one environment simultaneously, but only one session going to be deployed. First session that starts deploying is going to be deployed; other ones become invalid and could not be deployed at all. User could not open new session for environment that in `deploying` state (that's why we call it "almost lock free" model).

Table 4.12. Configuration Session Object

Attribute	Type	Description
id	string	Session unique ID
environment_id	string	Environment that going to be modified during this session
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
user_id	string	Session owner ID
version	int	Environment version for which configuration session is opened
state	string	Session state. Could be: open, deploying, deployed

Configure Environment / Open session

During this call new working session is created, and session ID should be sent in header (X-Configuration-Session) to all next API calls.

Call

Table 4.13. POST /environments/<id>/configure Call

Method	URI	Description
POST	/environments/<id>/configure	Creating new configuration session

Table 4.14. Error Response Codes

Code	Description
403	Could not open session for environment, environment has deploying status

Payload

None

Returns

This call returns created session:

```
{
    "id": "4aecdc2178b9430cbbb8db44fb7ac384",
    "environment_id": "4dc8a2e8986fa8fa5bf24dc8a2e8986fa8",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:54Z",
    "user_id": "d7b501094caf4daab08469663a9e1a2b",
    "version": 12,
    "state": "open"
}
```

Deploy changes from Session

Call

Table 4.15. POST /environments/<id>/sessions/<sessionId>/deploy Call

Method	URI	Description
POST	/environments/<id>/sessions/<sessionId>/deploy	Deploying changes made in session with specified <sessionId>

Table 4.16. Error Response Codes

Code	Description
403	Session is invalid
403	Session is already deployed or deployment is in progress

Payload

None

Returns

None

Get session information

Call

Table 4.17. GET /environments/<id>/sessions/<sessionId> Call

Method	URI	Description
GET	/environments/<id>/sessions/<sessionId>	Getting details about session with specified <sessionId>

Table 4.18. Error Response Codes

Code	Description
401	User is not authorized to access this session
403	Session is invalid

Payload

None

Returns

This call returns session information:

```
{
    "id": "4aecdc2178b9430cbbb8db44fb7ac384",
    "environment_id": "4dc8a2e8986fa8fa5bf24dc8a2e8986fa8",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:54Z",
    "user_id": "d7b501094caf4daab08469663a9e1a2b",
    "version": 0,
    "state": "deploying"
}
```

Delete Session

Call

Table 4.19. DELETE /environments/<id>/sessions/<sessionId> Call

Method	URI	Description
DELETE	/environments/<id>/sessions/<sessionId>	Delete session with specified <sessionId>

Table 4.20. Error Response Codes

Code	Description
401	User is not authorized to access this session
403	Session is in deploying state and could not be deleted

Payload

None

Returns

None

Active Directory API

This section describes API calls for Active Directory service management.

Get a List of existing Active Directory instances

Table 4.21. Active Directory Object

Attribute	Type	Description
id	string	Unique ID
name	string	Domain name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
domain	string	Domain name
uri	string	URI of the Service
units	object	Active Directory Unit object

Table 4.22. Active Directory Unit Object

Attribute	Type	Description
id	string	Unique ID
isMaster	boolean	true for primary domain controller, false otherwise
address	string	Unit address
location	string	AvailabilityZone or specific physical datacenter.

Call

Table 4.23. GET /environments/<id>/activeDirectories Call

Method	URI	Description
GET	/environments/<id>/activeDirectories	Get a list of Active Directory instances

Table 4.24. Headers

Name	Type	Required	Description
X-Configuration-Session	string	no	ID of valid configuration session

Payload

None

Returns

This call returns list of Active Directory instances:

```
{
    "activeDirectories": [{
        "id": "96365940588b479294fe8e6dc073db04",
        "name": "acme.dc",
        "created": "2010-11-30T03:23:42Z",
        "updated": "2010-11-30T03:23:44Z",
        "uri": "10.0.0.2",
        "units": [{
            "id": "d08887df15b94178b244904b506fe85b",
            "isMaster": true,
            "address": "10.0.0.2",
            "location": "west-dc"
        },
        {
            "id": "dcf0de317e7046bea555539f19b8ea84",
            "isMaster": false,
            "address": "10.0.0.3",
            "location": "west-dc"
        }]
    }]
}
```

Create Active Directory instance

Table 4.25. Active Directory Object

Attribute	Type	Required	Description
name	string	yes	Domain name
adminPassword	string	yes	Password from domain administrator account
domain	string	yes	Domain name
units	object	yes	Active Directory Unit object

Table 4.26. Active Directory Unit Object

Attribute	Type	Required	Description
isMaster	boolean	yes	true for primary domain controller, false otherwise
recoveryPassword	string	yes	Recovery password
location	string	yes	AvailabilityZone or specific physical datacenter.

Call

Table 4.27. POST /environments/<id>/activeDirectories Call

Method	URI	Description
POST	/environments/<id>/activeDirectories	Create new Active Directory

Table 4.28. Headers

Name	Type	Required	Description
X-Configuration-Session	string	yes	ID of valid configuration session

Payload

```
{
    "name": "acme.dc",
    "adminPassword": "password",
    "domain": "acme.dc",
    "units": [
        {
            "isMaster": true,
            "recoveryPassword": "password",
            "location": "west-dc"
        },
        {
            "isMaster": false,
            "recoveryPassword": "password",
            "location": "west-dc"
        }
    ]
}
```

Returns

This call returns created active directory domain:

```
{
    "id": "96365940588b479294fe8e6dc073db04",
```

```
    "name": "acme.dc",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
    "domain": "acme.dc",
    "units": [{
        "id": "d08887df15b94178b244904b506fe85b",
        "isMaster": true,
        "location": "west-dc"
    }, {
        "id": "dcf0de317e7046bea555539f19b8ea84",
        "isMaster": false,
        "location": "west-dc"
    }]
}
```

Web Server API

This section describes API calls for managing Windows web-server software – IIS.

Get a List of existing Web Servers

Table 4.29. Web Server Object

Attribute	Type	Description
id	string	Unique ID
name	string	User-friendly name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
uri	string	URI of the Service
domain	string	Domain name. This attribute may be empty/null/omitted if machine is not a domain member
units	object	Web Server Unit object

Table 4.30. Web Server Unit Object

Attribute	Type	Description
id	string	Unique ID
address	string	Unit address
location	string	AvailabilityZone or specific physical datacenter.

Call

Table 4.31. GET /environments/<id>/webServers Call

Method	URI	Description
GET	/environments/<id>/webServers	Get a list of existing Web Servers

Table 4.32. Headers

Name	Type	Required	Description
X-Configuration-Session	string	no	ID of valid configuration session

Payload

None

Returns

This call returns list of web servers:

```
{
  "webServers":
      [
         {
           "id": "0ce373a477f211e187a55404a662f968",
           "name": "frontend",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "domain": "ACME",
           "uri": "http://10.0.0.2",
           "units": [{
               "id": "1bf3491c409b4541b6f18ea5988a6437",
               "address": "10.0.0.2",
               "location": "west-dc"
           }]
         },
         {
           "id": "c697bd2429304820a928d145aa42af59",
           "name": "backend",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "domain": "ACME",
           "uri": "http://10.0.0.3",
           "units": [{
               "id": "eb32f97866d24001baa430cb34e4049f",
               "address": "10.0.0.3",
               "location": "west-dc"
           }]
         }
      ]
}
```

Create Web Server instance

Table 4.33. Web Server Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name
domain	string	no	Domain name
units	object	yes	Web Server Unit object

Table 4.34. Web Server Unit Object

Attribute	Type	Required	Description
location	string	yes	AvailabilityZone or specific physical datacenter.

Call

Table 4.35. POST /environments/<id>/webServers Call

Method	URI	Description
POST	/environments/<id>/webServers	Create new Web Server

Table 4.36. Headers

Name	Type	Required	Description
X-Configuration-Session	string	yes	ID of valid configuration session

Payload

```
{
  "name": "frontend",
  "adminPassword": "password",
  "domain": "acme.dc",
  "units": [{
    "location": "west-dc"
  }]
}
```

Returns

This call returns created web server:

```
{
    "id": "ce373a477f211e187a55404a662f968",
    "name": "frontend",
    "created": "2010-11-30T03:23:42Z",
```

```
    "updated": "2010-11-30T03:23:44Z",
    "domain": "ACME",
    "units": [{
        "id": "1bf3491c409b4541b6f18ea5988a6437",
        "location": "west-dc"
    }]
}
```

ASP.NET Application API

This section describes API calls for managing ASP.NET Applications

Get a List of existing ASP.NET Applications

Table 4.37. ASP.NET Application Object

Attribute	Type	Description
id	string	Unique ID
name	string	User-friendly name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
repository	string	URL of git repository containing the application source files
uri	string	URI of the Service
domain	string	Domain name. This attribute may be empty/null/omitted if machine is not a domain member
units	object	ASP.NET Application Unit object

Table 4.38. ASP.NET Application Unit Object

Attribute	Type	Description
id	string	Unique ID
address	string	Unit address
location	string	AvailabilityZone or specific physical datacenter.

Call

Table 4.39. GET /environments/<id>/aspNetApps Call

Method	URI	Description
GET	/environments/<id>/aspNetApps	Get a list of existing ASP.NET Applications

Table 4.40. Headers

Name	Type	Required	Description
X-Configuration-Session	string	no	ID of valid configuration session

Payload

None

Returns

This call returns list of ASP.NET Applications:

```
{
  "aspNetApps":
      [
         {
           "id": "88f6ed99ff3645bcb84e1e37ab9ece3d",
           "name": "frontend",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "domain": "ACME",
           "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
           "uri": "http://10.0.0.2",
           "units": [{
               "id": "59255829f0574297acc1cd3a18ff6fd7",
               "address": "10.0.0.2",
               "location": "west-dc"
           }]
         },
         {
           "id": "aa49dcaff9914b8abb26855f0799b0e0",
           "name": "backend",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
           "uri": "http://10.0.0.3",
           "domain": "ACME2" ,
           "units": [{
               "id": "274b54f6bbe6493690e7107aa947e112",
               "address": "10.0.0.3",
               "location": "west-dc"
           }]
         }
      ]
}
```

Create ASP.NET Application instance

Table 4.41. ASP.NET Application Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name
repository	string	yes	URL of git repository containing the application source files

Attribute	Type	Required	Description
domain	string	no	Domain name
units	object	yes	ASP.NET Application Unit object

Table 4.42. ASP.NET Application Unit Object

Attribute	Type	Required	Description
location	string	yes	AvailabilityZone or specific physical datacenter.

Call

Table 4.43. POST /environments/<id>/aspNetApps Call

Method	URI	Description
POST	/environments/<id>/aspNetApps	Create new ASP.NET Application

Table 4.44. Headers

Name	Type	Required	Description
X-Configuration-Session	string	yes	ID of valid configuration session

Payload

```
{
  "name": "frontend",
  "adminPassword": "password",
  "domain": "acme.dc",
  "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
  "units": [{
    "location": "west-dc"
  }]
}
```

Returns

This call returns created ASP.NET Application:

```
{
    "id": "5374bd535e26452e8acb6ae3ee87c596",
    "name": "frontend",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
    "domain": "ACME",
    "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
    "units": [{
        "id": "7bb3f29bcc9449f7b91f7a4d4668499a",
```

```
        "location": "west-dc"
    }]
}
```

Web Server Farm API

This section describes API calls for managing Web Server (IIS) Web Farm services

Get a List of existing Web Server Farms

Table 4.45. Web Server Farm Object

Attribute	Type	Description
id	string	Unique ID
name	string	User-friendly name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
uri	string	URI of the Service
loadBalancerPort	integer	Port number of the Farm
domain	string	Domain name. This attribute may be empty/null/omitted if machine is not a domain member
units	object	Web Server Farm Unit object

Table 4.46. Web Server Farm Unit Object

Attribute	Type	Description
id	string	Unique ID
location	string	AvailabilityZone or specific physical datacenter.

Call

Table 4.47. GET /environments/<id>/webServerFarms Call

Method	URI	Description
GET	/environments/<id>/webServerFarms	Get a list of existing Web Server Farms

Table 4.48. Headers

Name	Type	Required	Description
X-Configuration-Session	string	no	ID of valid configuration session

Payload

None

Returns

This call returns list of Web Server Farms:

```
{
  "webServerFarms":
      [
         {
           "id": "01fa4412ab4849acb27394aaf307ca88",
           "name": "frontend",
           "created": "2010-11-30T03:23:42Z",
           "updated": "2010-11-30T03:23:44Z",
           "domain": "ACME",
           "loadBalancerPort": 80,
           "uri": "http://192.168.1.1:80",
           "units": [
               {
                   "id": "a34992c8634b482798187d3c0e1c999a",
                   "address": "10.0.0.2",
                   "location": "west-dc"
               },
               {
                   "id": "fcd60488bb6f4acf97ccdb8f8fc6bc9a",
                   "address": "10.0.0.3",
                   "location": "west-dc"
               }
           ]
         }
      ]
}
```

Create Web Server Farm instance

Table 4.49. Web Server Farm Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name
loadBalancerPort	integer	yes	Port number for the Farm
domain	string	no	Domain name
units	object	yes	Web Server Farm Unit object

Table 4.50. Web Server Farm Unit Object

Attribute	Type	Required	Description
location	string	yes	AvailabilityZone or specific physical datacenter.

Call

Table 4.51. POST /environments/<id>/webServerFarms Call

Method	URI	Description	
POST	/environments/<id>/webServerFarms	Create new Web Server Farm	

Table 4.52. Headers

Name	Type	Required	Description
X-Configuration-Session	string	yes	ID of valid configuration session

Payload

```
{
  "name": "frontend",
  "adminPassword": "password",
  "domain": "acme.dc",
  "loadBalancerPort": 80,
  "units": [{
    "location": "west-dc"
  }]
}
```

Returns

This call returns created Web Server Farm:

```
{
    "id": "fa723794e1b7454f967deb65b30d2c22",
    "name": "frontend",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
    "domain": "ACME",
    "loadBalancerPort": 80,
    "units": [{
        "id": "54213d8fb87d41f7a1be2f81dbba0d69",
        "location": "west-dc"
    }]
}
```

ASP.NET Application Farm API

This section describes API calls for managing ASP.NET Web Farm Application Services

Get a List of existing ASP.NET Application Farms

Table 4.53. ASP.NET Application Farm Object

Attribute	Type	Description
id	string	Unique ID

Attribute	Type	Description
name	string	User-friendly name
created	datetime	Creation date and time in ISO format
updated	datetime	Modification date and time in ISO format
uri	string	URI of the Service
repository	string	URL of git repository containing the application source files
loadBalancerPort	integer	Port number of the Farm
domain	string	Domain name. This attribute may be empty/null/omitted if machine is not a domain member
units	object	ASP.NET Application Farm Unit object

Table 4.54. ASP.NET Application Farm Unit Object

Attribute	Type	Description
id	string	Unique ID
address	string	Unit address
location	string	AvailabilityZone or specific physical datacenter.

Call

Table 4.55. GET /environments/<id>/aspNetAppFarms Call

Method	URI	Description
GET	/environments/<id>/aspNetAppFarms	Get a list of existing ASP.NET Application Farms

Table 4.56. Headers

Name	Type	Required	Description
X-Configuration-Session	string	no	ID of valid configuration session

Payload

None

Returns

This call returns list of ASP.NET Application Farms:

```
{
  "aspNetAppFarms":
      [
         {
           "id": "01fa4412ab4849acb27394aaf307ca88",
           "name": "frontend",
           "created": "2010-11-30T03:23:42Z",
```

```
            "updated": "2010-11-30T03:23:44Z",
            "domain": "ACME",
            "loadBalancerPort": 80,
            "uri": "http://192.168.1.1:80",
            "units": [
                {
                    "id": "3374f4eb850e4b27bf734649d7004cc0",
                    "address": "10.0.0.2",
                    "location": "west-dc"
                },
                {
                    "id": "fcd60488bb6f4acf97ccdb8f8fc6bc9a",
                    "address": "10.0.0.3",
                    "location": "west-dc"
                }
            ]
        }
    ]
}
```

Create ASP.NET Application Farm instance

Table 4.57. ASP.NET Application Farm Object

Attribute	Type	Required	Description
name	string	yes	User-friendly name
repository	string	yes	URL of git repository containing the application source files
loadBalancerPort	integer	yes	Port number for the Farm
domain	string	no	Domain name
units	object	yes	ASP.NET Application Farm Unit object

Table 4.58. ASP.NET Application Farm Unit Object

Attribute	Type	Required	Description
location	string	yes	AvailabilityZone or specific physical datacenter.

Call

Table 4.59. POST /environments/<id>/aspNetAppFarms Call

Method	URI	Description
POST	/environments/<id>/aspNetAppFarms	Create new ASP.NET Application Farm

Table 4.60. Headers

Name	Type	Required	Description
X-Configuration-Session	string	yes	ID of valid configuration session

Payload

```
{
  "name": "frontend",
  "adminPassword": "password",
  "domain": "acme.dc",
  "loadBalancerPort": 80,
  "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
  "units": [{
    "location": "west-dc"
  }]
}
```

Returns

This call returns created ASP.NET Application Farm:

```
{
    "id": "1eb6cd5fc3f24d60a9a4c431512d3708",
    "name": "frontend",
    "created": "2010-11-30T03:23:42Z",
    "updated": "2010-11-30T03:23:44Z",
    "domain": "ACME",
    "loadBalancerPort": 80,
    "repository": "https://github.com/Mirantis/murano-mvc-demo.git",
    "units": [{
        "id": "54213d8fb87d41f7a1be2f81dbba0d69",
        "location": "west-dc"
    }]
}
```

Chapter 5. Installation Guide

This chapter is about installation and configuration Murano services.

Note that all Murano modules can be downloaded from our page [https://launchpad.net/murano/] on launchpad.

Common Pre-Requirements

Operation system:

1. Ubuntu
2. RHEL/CentOS

Packages:

1. python-dev
2. libxml2-dev
3. libxslt-dev

Murano API Service

Murano API provides access to the Murano orchestration engine via API.

This chapter describes Murano API for contributors of the project, and assumes that you are already familiar with Murano API from an end-user perspective.

Install

- Project source can be checked out as git repository (see below) or downloaded from here [http://tarballs.openstack.org/murano-api/]

```
user@work:~/$ git clone https://github.com/stackforge/murano-api.git
```

- Switch to just created directory

```
user@work:~/$ cd murano-api
```

- And them perform installation:

```
user@work:~/murano-api$ chmod +x setup.sh ; sudo ./setup.sh install
```

Configure

- First configure rabbitMQ by adding vhost and user with administrator rights:

```
sudo rabbitmqctl add_user murano murano
sudo rabbitmqctl set_user_tags murano administrator
sudo rabbitmqctl add_vhost murano
sudo rabbitmqctl set_permissions -p murano murano ".*" ".*" ".*"
```

- Edit configuration file:

```
user@work:~/murano-api$ nano ./etc/murano-api.conf
```

- Configure it according to your environment:

  [DEFAULT] section sets up logging.

  In *[reports]* section you can set names for new rabbitMQ queues.

  [rabbitmq] section sets up host configuration where rabbitMQ with just created user and vhost is running.

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True
# Show debugging output in logs (sets DEBUG log level output)
debug = True
# Address to bind the server to
bind_host = 0.0.0.0
# Port the bind the server to
bind_port = 8082
# Log to this file. Make sure the user running skeleton-api has
# permissions to write to this file!
log_file = /tmp/murano-api.log
#A valid SQLAlchemy connection string for the metadata database
sql_connection = sqlite:///murano.sqlite

[reports]
results_exchange = task-results
results_queue = task-results
reports_exchange = task-reports
reports_queue = task-reports
```

```
[rabbitmq]
host = localhost
port = 5672
virtual_host = murano
login = murano
password = murano
```

- Edit one more configuration file:

```
user@work:~/murano-api$ nano ./etc/murano-api-paste.ini
```

- Configure keystone auth_token in *[filter:authtoken]* section. For more information see Auth-Token Middleware with Username and Password [http://docs.openstack.org/developer/keystone/configuringservices.html]

```
[pipeline:murano-api]
pipeline = authtoken context apiv1app
[app:apiv1app]
paste.app_factory = muranoapi.api.v1.router:API.factory
[filter:context]
paste.filter_factory = muranoapi.api.middleware.context:ContextMiddleware.factory

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = localhost
auth_port = 35357
auth_protocol = http
admin_tenant_name = admin
admin_user = admin
admin_password = password
signing_dir = /tmp/keystone-signing-muranoapi
```

- Register murano-api service in Openstack (note: you need to be authorized in Openstack to run this commands)

```
user@work:~/$ keystone service-create --name muranoapi --type murano --descriptio
```

```
user@work:~/$ keystone endpoint-create

    --region RegionOne
    --service-id The ID field returned by the keystone service-create
```

```
--publicurl http://x.x.x.x:8082 (where x.x.x.x - host ip where murano-api ins
--internalurl the same as publicurl
--adminurl the same as publicurl
```

Run

Run Murano API and supply valid configuration file:

```
user@work:~/$ murano-api --config-file=./murano/api/etc/murano-api.conf
```

Conductor Service

Conductor is a Murano orchestration engine that transforms object model sent by REST API service into a series of Heat and Murano-Agent commands.

This document describes Conductor for contributors of the project.

Install

- Project source code can be checked out from git repository (see below) or downloaded from here. [http://tarballs.openstack.org/murano-conductor/]

```
user@work:~/$ git clone https://github.com/stackforge/murano-conductor.git
```

- Switch to just created directory

```
user@work:~/cd murano-conductor
```

- And install Conductor Service to the system:

```
user@work:~/murano-conductor$ chmod +x setup.sh ; sudo ./setup.sh install
```

Configure

- Edit configuration file:

```
user@work:~/murano-conductor$ nano ./etc/conductor.conf
```

- Change it according to your environment.

  [DEFAULT] section is responsible for logging.

  [heat] points where heat is running.

  [rabbitmq] section points where your rabbitMQ installed and configured.

```
[DEFAULT]
log_file = logs/conductor.log
debug=True
verbose=True

[heat]
auth_url = http://localhost:5000/v2.0

[rabbitmq]
host = localhost
port = 5672
virtual_host = murano
login = murano
password = murano
```

Run

Run Conductor and supply valid configuration file:

```
user@work:~/murano-conductor$ conductor --config-file=./etc/conductor.conf
```

Murano Dashboard

Murano Dashboard provides Web UI for Murano Project.

Pre-Requirements

- To setup Murano Dashboard on a host with Openstack Dashbord already installed you just need to install *the python-muranoclient*. You can download it from here. [http://tarballs.openstack.org/python-muranoclient/]
  And then perform installation with pip:

```
user@work:~/$ sudo pip install  just_downloaded.tar.gz
```

- If there is no OpenStack Dashboard (horizon) you'll need to install it. See here [http://docs.openstack.org/trunk/openstack-compute/install/yum/content/ch_install-dashboard.html] how to do that.

Install

- Project source code can be checked out from git repository (see below) or downloaded from here. [http://tarballs.openstack.org/murano-dashboard/]

```
user@work:~/$ git clone https://github.com/stackforge/murano-dashboard.git
```

- Switch to just created directory

```
user@work:~/$ cd murano-dashboard
```

- And perform installation

```
user@work:~/murano-dashboard$ sudo python setup.py install
```

Configure

- Open Django configuration file:

```
user@work:~/$ cd <Horizon Installation Dir> && nano settings.py
```

Please, make sure that no local/local_settings.py file exists.

- Add to import section

```
from muranoclient.common import exceptions as muranoclient
```

- And this so muranoclient exceptions can be safely handle by horizon:

```
RECOVERABLE_EXC = (muranoclient.HTTPException,
                   muranoclient.CommunicationError,
                   muranoclient.Forbidden)
EXTENDED_RECOVERABLE_EXCEPTIONS = tuple(
    exceptions.RECOVERABLE + RECOVERABLE_EXC)

NOT_FOUND_EXC = (muranoclient.HTTPNotFound, muranoclient.EndpointNotFound)
EXTENDED_NOT_FOUND_EXCEPTIONS = tuple(exceptions.NOT_FOUND + NOT_FOUND_EXC)

UNAUTHORIZED_EXC = (muranoclient.HTTPUnauthorized, )
EXTENDED_UNAUTHORIZED_EXCEPTIONS = tuple(
    exceptions.UNAUTHORIZED + UNAUTHORIZED_EXC)
```

- And finally edit HORIZON_CONFIG and INSTALLED_APPS sections

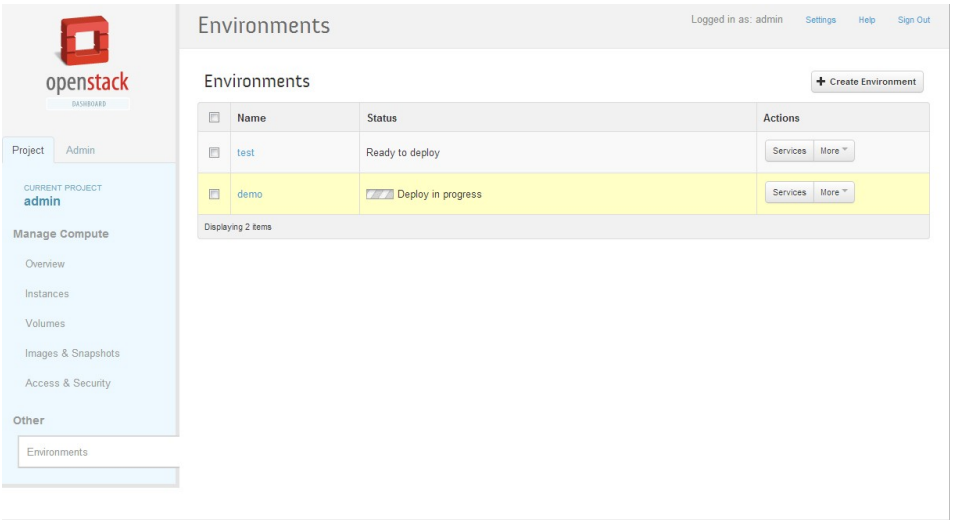
```
HORIZON_CONFIG = {
...
    'exceptions': {'recoverable': EXTENDED_RECOVERABLE_EXCEPTIONS,
                   'not_found': EXTENDED_NOT_FOUND_EXCEPTIONS,
                   'unauthorized': EXTENDED_UNAUTHORIZED_EXCEPTIONS},
    'customization_module': 'muranodashboard.panel.overrides'

}
...
INSTALLED_APPS = (
...
    'muranodashboard',
...
)
```

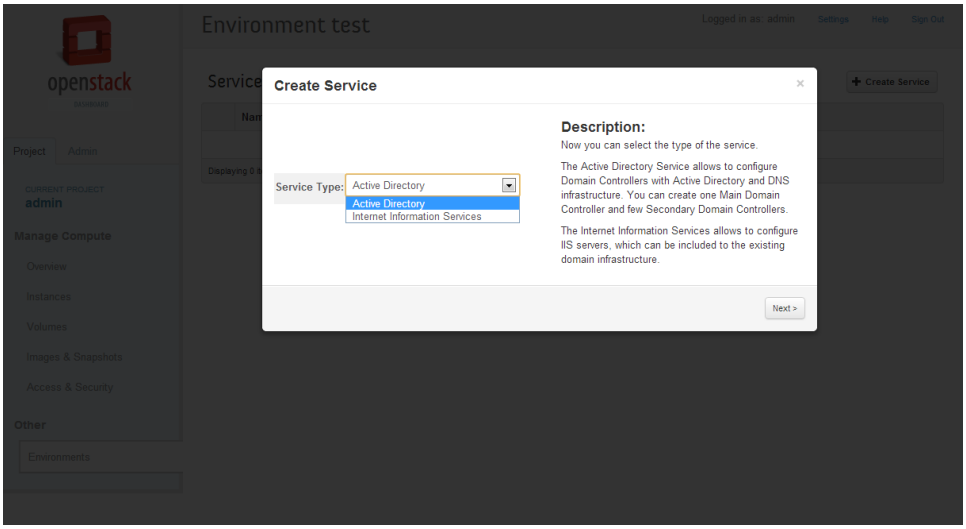
Run

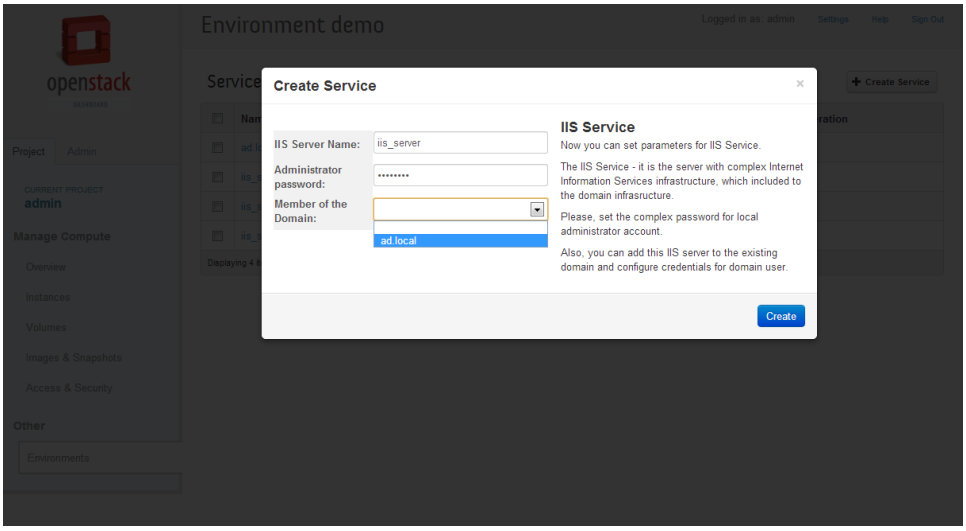
Horizon usually running on apache2 server, and can be restarted to apply changes by:

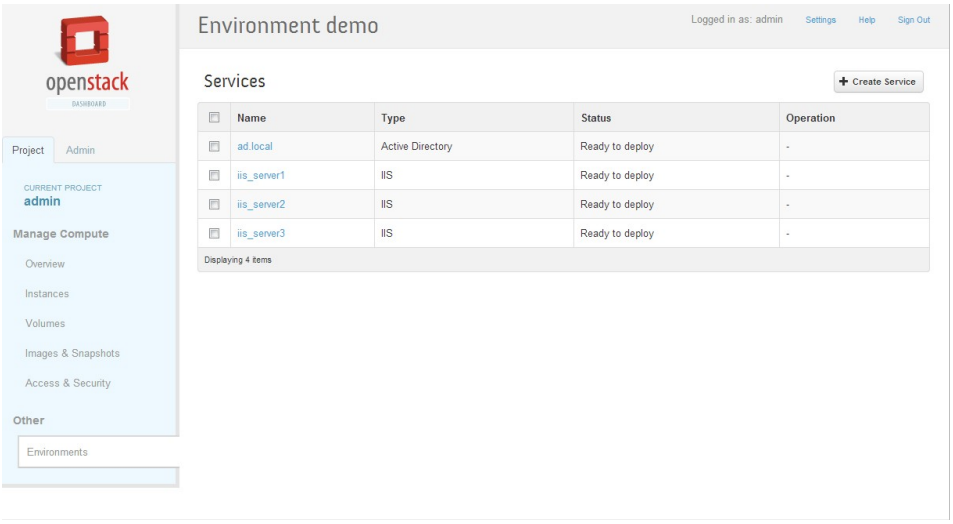
```
user@work:~/$ sudo service apache2 restart
```

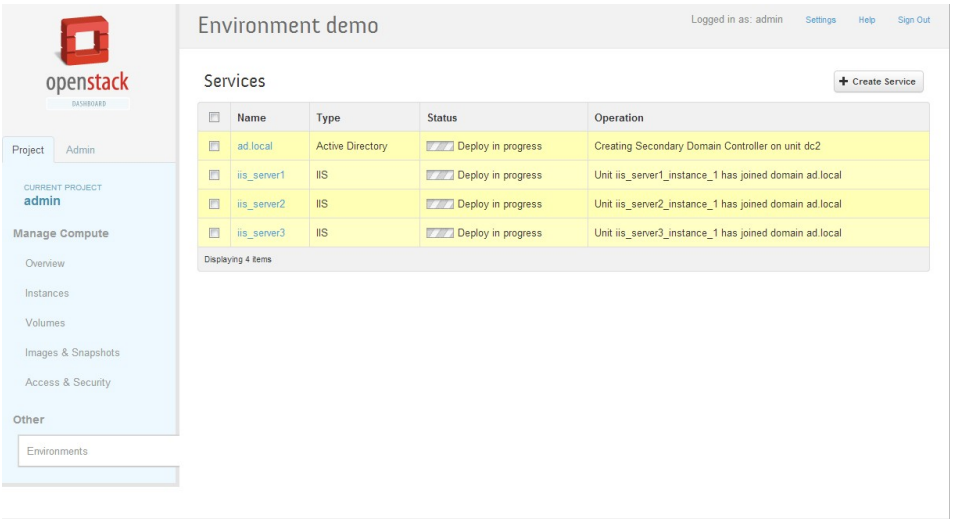
Chapter 6. ScreenShots

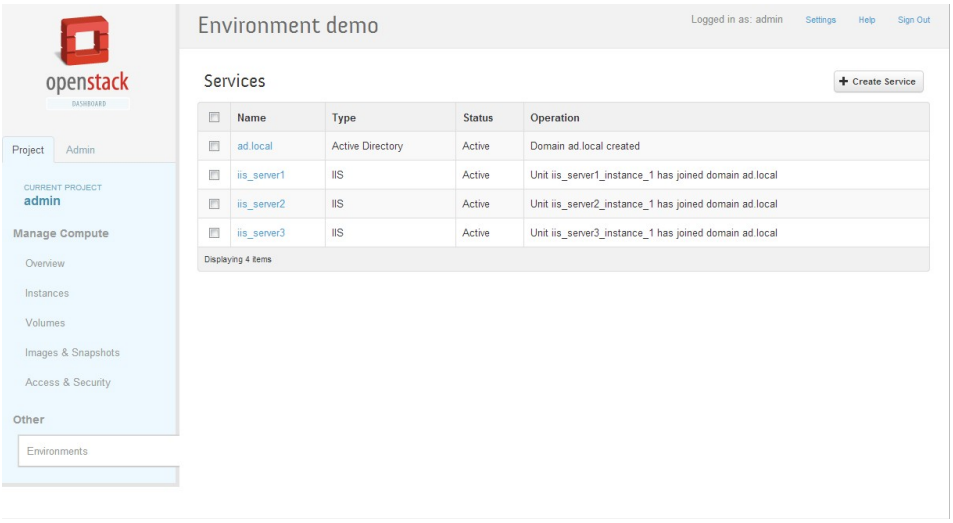


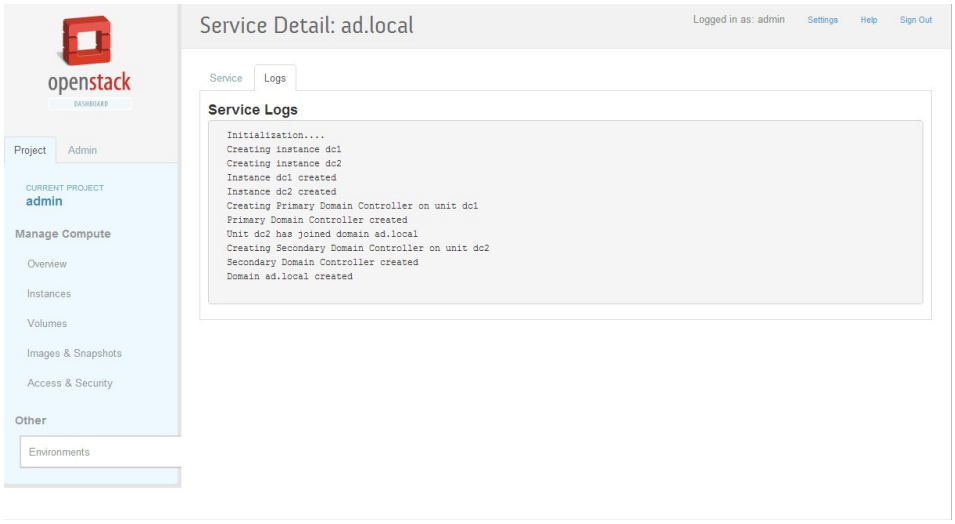


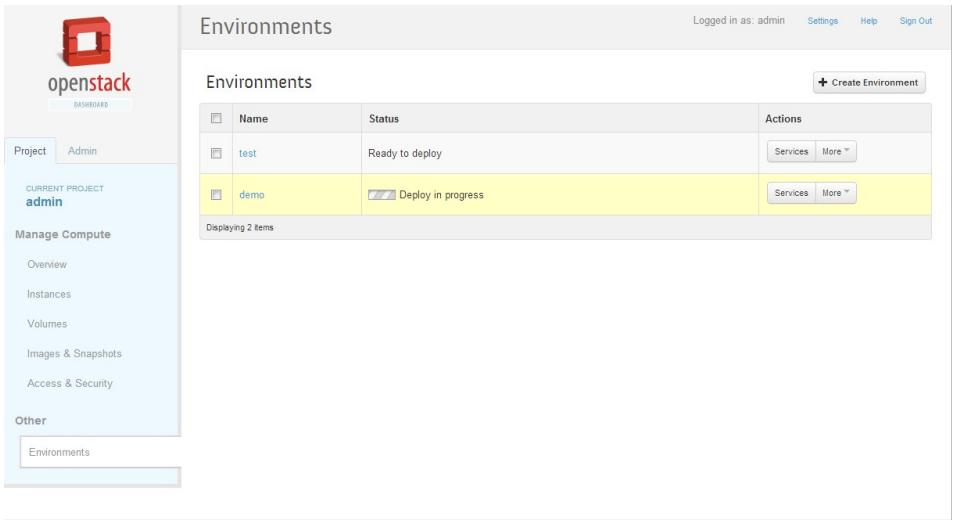



openstack
DASHBOARD
Project
Admin
CURRENT PROJECT
admin
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Other
Environments
Environment demo
Logged in as: admin
Settings
Help
Sign Out
Services
+ Create Service
Name
Type
Status
Operation
ad.local
Active Directory
Ready to deploy
-
iis_server1
IIS
Ready to deploy
-
iis_server2
IIS
Ready to deploy
-
iis_server3
IIS
Ready to deploy
-
Displaying 4 items


openstack
DASHBOARD
Project
Admin
CURRENT PROJECT
admin
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Other
Environments
Environment demo
Logged in as: admin
Settings
Help
Sign Out
Services
+ Create Service
Name
Type
Status
Operation
ad.local
Active Directory
Deploy in progress
Creating Secondary Domain Controller on unit dc2
iis_server1
IIS
Deploy in progress
Unit iis_server1_instance_1 has joined domain ad.local
iis_server2
IIS
Deploy in progress
Unit iis_server2_instance_1 has joined domain ad.local
iis_server3
IIS
Deploy in progress
Unit iis_server3_instance_1 has joined domain ad.local
Displaying 4 items


openstack
DASHBOARD
Project
Admin
CURRENT PROJECT
admin
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Other
Environments
Environment demo
Logged in as: admin
Settings
Help
Sign Out
Services
+ Create Service
Name
Type
Status
Operation
ad.local
Active Directory
Active
Domain ad.local created
iis_server1
IIS
Active
Unit iis_server1_instance_1 has joined domain ad.local
iis_server2
IIS
Active
Unit iis_server2_instance_1 has joined domain ad.local
iis_server3
IIS
Active
Unit iis_server3_instance_1 has joined domain ad.local
Displaying 4 items


openstack
DASHBOARD
Project
Admin
CURRENT PROJECT
admin
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Other
Environments
Service Detail: ad.local
Logged in as: admin
Settings
Help
Sign Out
Service
Logs
Service Logs
Initialization....
Creating instance dc1
Creating instance dc2
Instance dc1 created
Instance dc2 created
Creating Primary Domain Controller on unit dc1
Primary Domain Controller created
Unit dc2 has joined domain ad.local
Creating Secondary Domain Controller on unit dc2
Secondary Domain Controller created
Domain ad.local created


openstack
DASHBOARD
Project
Admin
CURRENT PROJECT
admin
Manage Compute
Overview
Instances
Volumes
Images & Snapshots
Access & Security
Other
Environments
Environments
Logged in as: admin
Settings
Help
Sign Out
Environments
+ Create Environment
Name
Status
Actions
test
Ready to deploy
Services
More
demo
Deploy in progress
Services
More
Displaying 2 items

Chapter 7. How To Participate

If you would like to ask some questions or make proposals, feel free to reach us on #murano irc channel at FreeNode. Typically somebody from our team will be online at irc from 6:00 to 20:00 UTC. You can also contact Murano community directly by murano-all@lists.launchpad.net [mailto:murano-all@lists.launchpad.net] (please, note that your email address should be registered in launchpad, otherwise your mail will be ignored by mailing system).

We’re going to hold public weekly meetings on Mondays at 17:00 UTC on #openstack-meeting-alt irc channel.

If you want to contribute either to docs or to code, simply send us change request via review.openstack.org [http://review.openstack.org] (gerrit). You can file bugs and register blueprints at Muranolaunchpad page [https://launchpad.net/murano].