# Open Source Virtualization Implementation

by

Jordan Nedderman & Troy Elsen

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology



Jordan Nedderman Troy Elsen — 21 APR 2014

Jordan Nedderman, Troy Elsen — Date

Mark Stockman, Faculty Advisor — Date 4/22/14

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2014

# Acknowledgements

We would like to acknowledge our technical advisor Mark Stockman for giving us valuable advice on our project so that it would be successful. We also appreciated the help both Dr. Patrick Kumpf and Professor James Scott gave to our project and the encouragement they gave us along the way. We are grateful for Brian Verkamp and Jason Gerst, for resolving our many issues with UC's vCloud. In addition, we would like to thank Dave Burns from CBTS for helping us narrow down our scope and all of the professional advice he gave to the project. We would have not been as successful without the help from these technology professionals and we extend our gratitude of thanks for assisting us and our project to completion.

# Table of Contents

# Abstract

In the IT industry, the need to keep up with new technologies while continuing to be innovative within your business is greatly desired, but at what cost? With the need to stay at the forefront of the IT industry, companies need to be able to test configurations of services and discover how these innovations affect infrastructure and/or other services within the network. The use of a service provider that provides Infrastructure as a Service (IaaS) could be the perfect fit. With today's technology, it has become extremely easy to build a multitude of systems at a rapid pace on top of these cloud infrastructures. A company implementing an OpenStack environment could provision a complete small organization's business needs within a matter of a few hours. This open source virtualization implementation is a great way to access the "cloud" and help organizations become more efficient and reach their full technological potential.

# Introduction

## Problem

As many companies big and small grow, they begin to need an infrastructure that is able to provide high availability and increase stability. The need for these high performance services have caused a great strain on internal technology departments and have driven many companies to search for ways of moving these services to a hosted cloud infrastructure. As these companies go to consulting companies and datacenters to find the services they need, these service companies are hard-pressed to create these infrastructures for each company and maintain each system at the guarantee of their Service License Agreement. The other issue that they are continuing to face is the increasing need for standardization of these systems and how they integrate with one another. What is needed is some infrastructure that can house these many systems and automation to create systems that are preconfigured to integrate with each other. The automation process will allow IT professionals at these datacenters to take less time creating each system and spend more time supported and optimizing systems that are in place.

## Solution

In a growing technological world the use of newer technologies integrated together is becoming necessary as infrastructures and system configurations become more complex. Organizations are always looking to improve their computing systems and everywhere you turn these days you keep hearing about cloud solutions and how they will help your business. IT upgrades are usually expensive and take some time to implement, so these plans can be put on the back burner. Throughout the course work at the University of Cincinnati, networking/systems students get to work directly with virtualization software to complete projects and assignments. There are many benefits when it comes to virtualization, but knowing the different variables of virtualization can be most beneficial for an organization. A working demonstration of a virtualization implementation will be able to display these benefits to an organization thinking about moving to a cloud solution.

## Project Goals/Brief Methodology

The project concept was based from previous course work and an element of the IT industry today. The design objectives helped break down the project to address specific details and help develop some of the goals that were completed and some that had to be abandoned. The next step was to create the methodology for this project to include the requirements, procedures, and the needed testing. The budget was kept minimal due to the lack of funding and the schedule was simple to keep on track with the project assignments. The problems that occurred and how they were solved are discussed alongside the recommendations for the future and fro improving the process.

## Overview

The IT industry has been transitioning into the “cloud” over the past few years. There are many benefits to moving an organization to the cloud but there are still some gray areas one

being who owns the data, privacy, security and cost. These are the concerns of some startups and small organizations. To solve some of these concerns, the use of a service provider that provides infrastructure as a service (IaaS) could be the perfect fit. With technology today, it has become extremely easy to build out a multitude of systems at a rapid pace on top of these cloud infrastructures. With a company implementing an OpenStack setup, they could provision a complete small organization's business needs within a matter of a few hours. This implementation would then allow a company to build infrastructures for these small businesses and then give them the keys to their whole infrastructure. This virtualization implementation will allow the organization to keep ownership of their data along with security and low cost. Open source virtualization implementation is a great way to access the “cloud” and help the organization to more efficiently utilize their full technological potential.

# Discussion

## Project Concept

Our project brings the capability and scalability of the cloud to business through an easy to use application through which they are able to provision development infrastructure to test out different configurations and develop solutions. This product will allow any user to create and configure machines to utilize during their develop process before implementing any changes into a production environment.

We wanted to harness the power and features that cloud technologies have to offer through the use of open source solutions so that any company can take advantage of the features that cloud has to offer. This led us on search for an infrastructure configuration that would a simplified application to interfaces with the robust features it had to offer. OpenStack is that solution and we found that it gives us even more capabilities than we originally expected.

## Design Objectives

The main four project goals are broken down below and briefly detailed.

1. Configure a virtualized environment
    a. OpenStack Dashboard

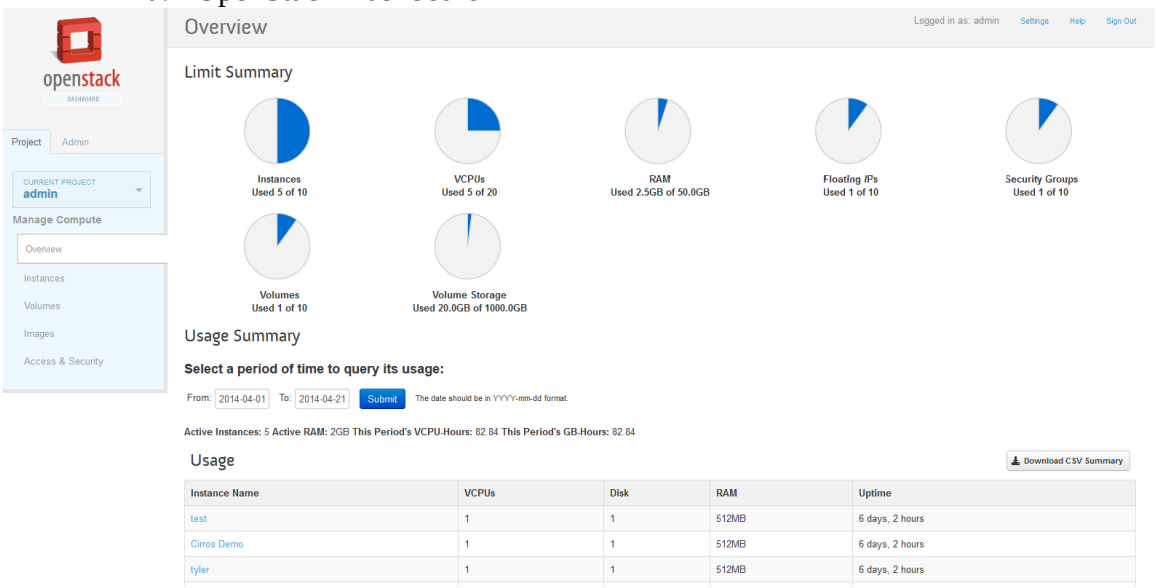

Image 1. OpenStack Dashboard

2. Utilize all open source technologies
    a. Ubuntu Server, OpenStack, KVM hypervisor
3. Create an application to interface with the system
    a. OSVI (Windows application)

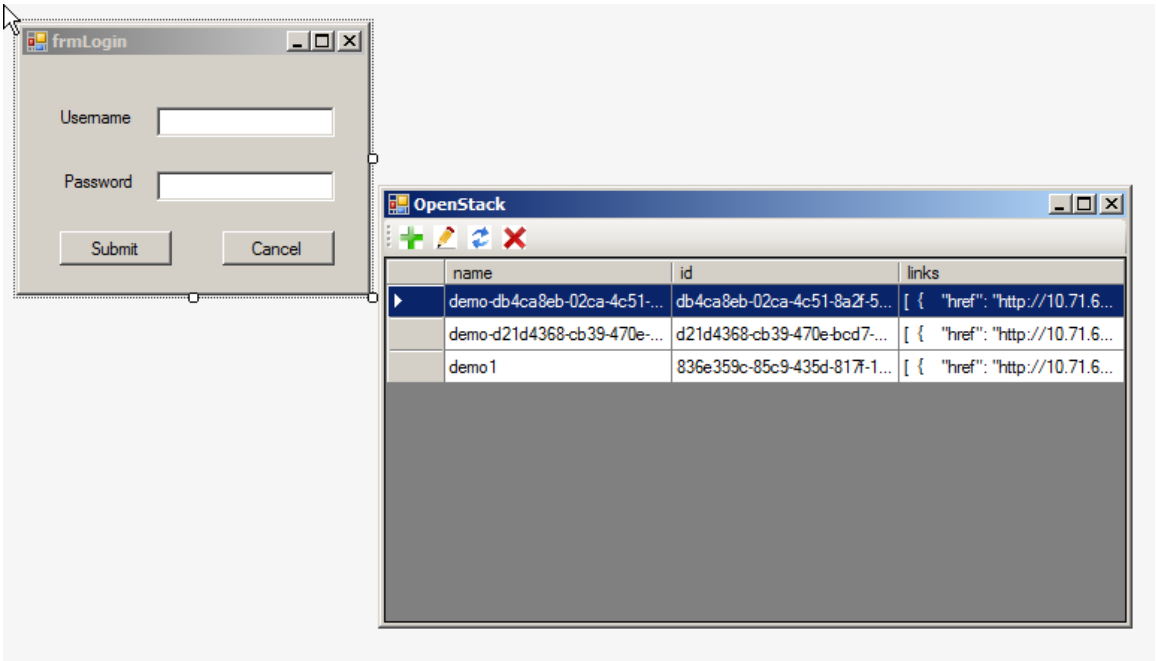

Image 2. OSVI Application

4. Document the project process
    a. Setup/configuration, worked/didn't work, recommendation

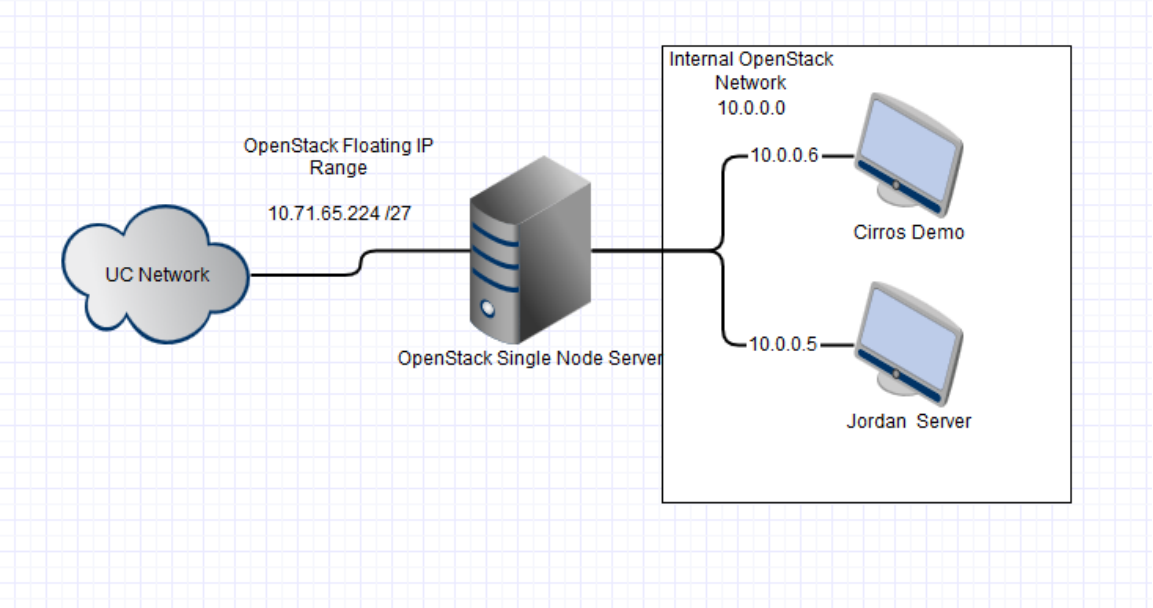

Image 3. Network Diagram

The following includes a list of the initial goals that had been abandoned in order to create the full featured application in which interfaced with the virtualized environment, utilizing the documented open source technologies.

1. Test multiple hypervisors
2. Test out node setup

    a. Single node
    b. Multi node
    c. Use juju to deploy OpenStack nodes
3. Write an application
    a. Web form
    b. Windows form
4. Utilization of MaaS (Metal as a Service)

**Methodology/Technical Approach**

There are many companies that want to use the benefits of virtualization and move their data into the cloud. This project will be a setup using Ubuntu Server and OpenStack to allow for the creation of an infrastructure that can help build highly scalable cloud computing environments. Our project consisted of developing an IaaS on top of the Ubuntu Cloud. The use of this open source software will keep the cost of implementation low. The plan was to document the process and have a working solution for an open source IaaS. In addition, this project could be presented to some small tech savvy organization to see if this solution would be something they would be interested in implementing. In the end the use of automation to scale the infrastructure will be tested and introduced into the final working demonstration.

1. The system will use OpenStack for base infrastructure.
    a. All components of OpenStack need to be installed.
    b. All components must be configured and started.
2. The system will have a repository of images.
    a. Images must be created and/or imported into the repository
        i. Ubuntu
        ii. CentOS
        iii. Fedora
        iv. RHEL
        v. Windows Server
3. The system needs to have the necessary flavors for each image.
    a. Preconfigured flavors must be tested and approved based on images
    b. More flavors need to be configured when necessary
4. Collection of scripts built out for necessary tasks against the system.
    a. Scripts need to be created to do the following tasks
        i. Build a machine
        ii. Delete a machine
        iii. Update machine configuration
5. The usability of scripts to interface with system API.

Setup scripts to run in sequence with a user interface possibly

| Req. No. | Item No. | Test Case No. | Input | Expected Output | Actual Output | Pass / Fail | Reason for Failure/ Success |
|---|---|---|---|---|---|---|---|
| 1 & 2 | 1.a. | 1 | sudo git clone git://github.com/openst | Installed all components | Installation of | P | One line command |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | ack-dev/devstack.git; cd devstack; ./stack.sh | of OpenStack | OpenStack Completed | | installed all components of OpenStack |
| | 1.b. | 2 | Log into OpenStack dashboard | Utilize all components of OpenStack | Did not initially have correct permissions and configuratio ns | F | Not components were configured and/or started |
| | 2.a.i. | 3 | Download Ubuntu image from source and upload to OpenStack dashboard | Image available for use | Correctly uploaded image and functioning | P | Downloade d image from source and upladed to OpenStack dashboard |
| | 2.a.ii . | 4 | Download CentOS image from source and upload to OpenStack dashboard | Image available for use | Correctly uploaded image and functioning | P | Downloade d image from source and upladed to OpenStack dashboard |
| | 2.a.iii. | 5 | Download Fedora image from source and upload to OpenStack dashboard | Image available for use | Correctly uploaded image and functioning | P | Downloade d image from source and upladed to OpenStack dashboard |
| | 2.a.iv. | 6 | Download RHEL image from source and upload to OpenStack dashboard | Image available for use | Correctly uploaded image and functioning | P | Downloade d image from source and upladed to OpenStack dashboard |
| | 2.a.v | 7 | Download Windows | Image | Correctly | P | Downloade |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | . | | Server image from source and upload to OpenStack dashboard | available for use | uploaded image and functioning | | d image from source and upladed to OpenStack dashboard |
| 3 | 3.a. | 8 | Pair image with flavor | Utilize system resources to function correctly | Error for not allocating enough system resources | F | Images were configured with the incorrect flavors initially |
| | 3.b. | 9 | Create flavor to work with image | Flavor configuration has enough resources for an image | Paired flavor and image functioning correctly | P | Additional flavors were configured for approved images |
| 4 | 4.a.i. | 10 | Run script to build virtual machine | Virtual machine was created | Virtual machine was successfully created | P | Template used to write working script for building a machine |
| | 4.a.ii. | 11 | Run script to delete virtual machine | Virtual machine was deleted | Virtual machine was successfully deleted | P | Template used to write working script for deleting a machine |
| | 4.a.iii. | 12 | Run script to update machine configuration | Virtual machine configuration updated | Error for not correctly writing script | F | Template used to write working script for updating machine |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | configuration |
| 5 | 5.a | 13 | Run script with correct sequence | Scripts run correct sequence | Scripts successfully ran in sequencial order | P | Initial scripts successfully run in sequence working towards UI |

Table A. Testing Report – Functional Requirements

**Budget**

The proposed budget for this project is free with the given open source software and the virtual sandbox with CECH. We were looking to work with CBTS and run our project on some of their hardware pending our last meeting. Additionally we installed the software for the project on a physical server on campus to run additional test for our project. The table below displays components we are going to utilize for our project. These components have either been donated, supplied free of charge, leased until the completion of our project. In addition these components should be sufficient enough to complete our project. This budget would also be recommended to companies because the project utilizes already existing hardware.

| Component | Price |
|---|---|
| Existing Desktop/Laptop | $0 |
| UC vCloud | $0 |
| Ubuntu Server/OpenStack | $0 |
| KVM Hypervisor | $0 |

Table B. Proposed project budget

**Gantt Chart**

This timeline was planned out for two semesters for the academic calendar year 2013-2014 in order to complete the assignments for senior design. The timeline had been updated to include any changes to keep on task for implementing the open source virtualization project. One time constraint was waiting for the use of additional hardware in the process of being supplied by CBTS which occurred in the first semester and a second time constraint was waiting for UC's vCloud issues to be resolved in the second semester. According to the timeline the project was kept on track completing each of the requirements for senior design this academic year. The Gantt chart below list the detailed senior design assignments that were needed for completion.

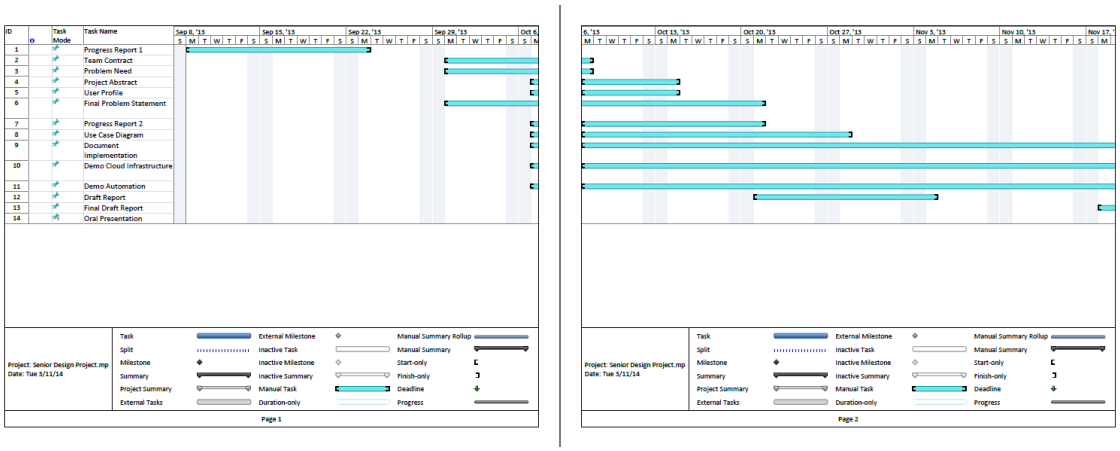


Figure 1. First semester Gantt chart

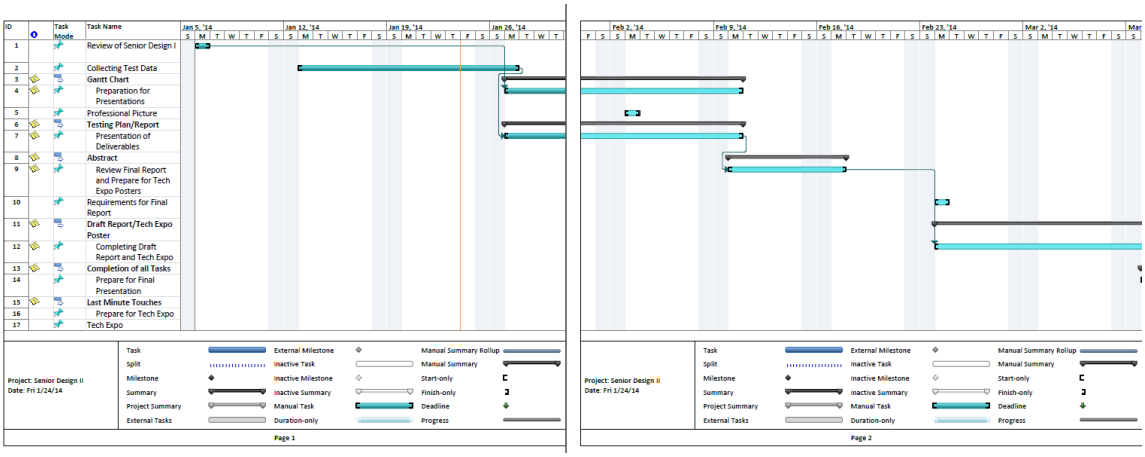


Figure 2. Second semester Gantt chart

**Problems Encountered and Analysis of Problems Solved**

Problems encountered during this project were numerous because of the nature of working with open source technologies. Having some Linux desktop/server experience helped but in the end fell short of being productive consistently throughout the project. Finding information on open source private clouds proved to be challenging and took plenty of hours of researching, combing through online manuals and forums. The setup and configuration of OpenStack also took some patients and time to get functioning correctly. Lastly, working with the OpenStack API to interface with the system was a learning curve which helped create a sound working application for the project. The typical communication and scheduling can be thrown into the mix that occurs with any group project being completed.

Each of these problems was solved with great effort and time researching until each part of the project was functioning. Asking for help from advisors and other technical professionals helped to figure out tough spots within the project. The communication and scheduling was solved by practicing patients and having the willingness for things to change. This project proved to help one grow, not only in the field of expertise but also professionally as a whole.

**Future Recommendations/Recommendations for Improvement**

Future recommendations for this project would be to be more versed in the open source communities and better understand the cloud technologies you work with. Knowing where to look for known issue can save a lot of time researching for the answer. When it comes to cloud technologies, it would have been better to understand each of the components of OpenStack rather than just figuring out how to configure it along the way. In addition, having more realistic time frames for certain tasks for the project would also be helpful. Another recommendation for improvement could be to mandate set meeting times rather than rushing assignments at the last minute. Lastly, having reliable hardware/software would also cut down on time spent waiting for certain issues to be resolved.

# Conclusion

## Lessons Learned

With a lot of research, configuring and hard work, this project has provided a knowledgeable baseline among the open source cloud technologies. Having the knowledge to provide a valuable skill set to help an organization save money with open source software and already existing hardware will be beneficial for future success. It can be extremely challenging working with open source technologies but the payoff in the end will be great. Knowing that there is not a whole lot in the technology industry with open source automation makes one feel confident that this project could help make a change for a brighter future. The time and effort it took together to produce a good application for open source cloud automation is a good lesson learned.

## Abilities/Skills Developed or Enhanced with Completion of Project

Abilities/skills that were developed with the completion of the project would be the following: setup and configuration of OpenStack with Ubuntu Server, knowledgebase of the OpenStack API, and writing scripts for an application.

# List of Illustrations

# References


1. Dornan, Andy. "Linux Virtualization Finds Some Rich Uncles." *InformationWeek* 13 June 2011: 21. *Expanded Academic ASAP*. Web. 4 Nov. 2013.
2. "Linux Expected to Thrive as Economy Recovers; Virtualization, cloud computing, and adoption by startups will make Linux the fastest-growing operating system, post economic downturn, report says." *InformationWeek* 9 Apr. 2009. *Expanded Academic ASAP*. Web. 4 Nov. 2013.
3. "Ubuntu Documentation." *Ubuntu Cloud*. N.p., n.d. Web. 4 Nov. 2013. <https://help.ubuntu.com/12.10/serverguide/ubuntucloud.html>.
4. "Ubuntu Documentation." *UbuntuCloudInfrastructure*. N.p., n.d. Web. 4 Nov. 2013. <https://help.ubuntu.com/community/UbuntuCloudInfrastructure>.
5. Musthaler, Linda. "From physical to virtual to cloud and everywhere in between." *Network World* 1 May 2013. *Expanded Academic ASAP*. Web. 4 Nov. 2013.
6. "Eucalyptus Offers Open Source VMware Based Cloud Platform 100923." *eWeek* 9 Sept. 2009. *Expanded Academic ASAP*. Web. 4 Nov. 2013.

# Appendix A.
# Project Documentation

**Ubuntu Cloud Setup with OpenStack**

The project had some success through the research that was accomplished and the Ubuntu Cloud documentation (Ubuntu Cloud). This helped with the installation of the open source orchestrator, OpenStack which ran on top of Ubuntu Sever. The additional links went more into detail about the configuration of OpenStack.

**Ubuntu Cloud Setup with MAAS and Juju**

The documentation for using the Ubuntu cloud setup with MAAS and Juju was thrown out when the project scope was narrowed down (UbuntuCloudInfrastructure). This documentation did provide some insight into the configuration of OpenStack which became beneficial later with the project.

# Appendix B.
# Application Scripts

### AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
[assembly: AssemblyTitle("OpenStackScripting")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("OpenStackScripting")]
[assembly: AssemblyCopyright("Copyright ©  2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]
[assembly: Guid("edc1e152-0855-4e21-a8ba-b54a54ed6e7f")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

### Resources.Designer.cs

```
namespace OpenStackScripting.Properties {
    using System;
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyT
ypedResourceBuilder", "4.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources {
        private static global::System.Resources.ResourceManager resourceMan;
        private static global::System.Globalization.CultureInfo resourceCulture;
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance"
, "CA1811:AvoidUncalledPrivateCode")]
        internal Resources() {
        }
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Edi
torBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager {
            get {
                if (object.ReferenceEquals(resourceMan, null)) {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("OpenStackScripting.Properties.Resources",
typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Edi
torBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
```

```
            }
        }
        internal static System.Drawing.Bitmap add {
            get {
                object obj = ResourceManager.GetObject("add", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
        internal static System.Drawing.Bitmap delete {
            get {
                object obj = ResourceManager.GetObject("delete", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
        internal static System.Drawing.Bitmap edit {
            get {
                object obj = ResourceManager.GetObject("edit", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
        internal static System.Drawing.Bitmap password {
            get {
                object obj = ResourceManager.GetObject("password", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
        internal static System.Drawing.Bitmap reboot {
            get {
                object obj = ResourceManager.GetObject("reboot", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
        internal static System.Drawing.Bitmap refresh {
            get {
                object obj = ResourceManager.GetObject("refresh", resourceCulture);
                return ((System.Drawing.Bitmap)(obj));
            }
        }
    }
}
```

**Settings.Desinger.cs**

```
namespace OpenStackScripting.Properties
{
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.S
ettingsDesigner.SettingsSingleFileGenerator", "11.0.0.0")]
    internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
    {
        private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings())));

        public static Settings Default
        {
            get
```

```
            {
                return defaultInstance;
            }
        }
    }
}
```

## Connect.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Net;
using System.IO;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
namespace OpenStackScripting
{
    public class Connect
    {
        private const string IP = "10.71.66.71";
        private const string authPort = ":5000";
        private const string novaPort = ":8774";
        private const string glancePort = ":9292";
        private const string identityPort = ":35357";
        public static string result;
        public static string URL;
        public static string adminID;
        public static string imageID;
        public static string serverID;
        private static string tokenStr = "";
        private static string authString = "";
        public static string userName = "";
        public JArray images;
        public JArray flavors;
        public JArray servers;
        public JObject serverDetail;
        public Connect(string user, string password)
        {
            authString = "{\"auth\": {\"tenantName\": \"admin\", \"passwordCredentials\":
{\"username\": \"" + user + "\", \"password\": \"" + password + "\"}}}";
            userName = user;
        }
        public void getToken()
        {
            URL = "http://" + IP + authPort + "/v2.0/tokens";
            Uri authUri = new Uri(URL);
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
            request.Method = "POST";
            request.ContentType = "application/json";
            request.Accept = "application/json";
            byte[] formData = UTF8Encoding.UTF8.GetBytes(authString);
            request.ContentLength = formData.Length;
            using (Stream post = request.GetRequestStream())
            {
```

```
            post.Write(formData, 0, formData.Length);
        }
        using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
        {
            StreamReader reader =
                new StreamReader(resp.GetResponseStream());
            result = reader.ReadToEnd();
            JObject json = new JObject();
            json = JObject.Parse(result);
            tokenStr = json["access"]["token"]["id"].ToString();
        }

    }
    public void getUserID()
    {
        URL = "http://" + IP + identityPort + "/v2.0/tenants";
        Uri authUri = new Uri(URL);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
        request.Method = "GET";
        request.UserAgent = "python-keystoneclient";
        request.Headers.Add("X-Auth-Token: " + tokenStr);
        string result = "";
        using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
        {
            StreamReader reader =
                new StreamReader(resp.GetResponseStream());
            result = reader.ReadToEnd();
            JObject json = new JObject();
            json = JObject.Parse(result);
            JArray array = JArray.Parse(json["tenants"].ToString());
            foreach (JObject o in array.Children<JObject>())
            {
                if (o.GetValue("name").ToString() == userName)
                {
                    adminID = o.GetValue("id").ToString();
                }
            }
        }
    }
    public void getImage()
    {
        URL = "http://" + IP + novaPort + "/v2/" + adminID + "/images";
        Uri authUri = new Uri(URL);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
        request.Method = "GET";
        request.UserAgent = "python-keystoneclient";
        request.Headers.Add("X-Auth-Token: " + tokenStr);
        string result = "";
        using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
        {
            StreamReader reader =
                new StreamReader(resp.GetResponseStream());
            result = reader.ReadToEnd();
            JObject json = new JObject();
            json = JObject.Parse(result);
```

```
            images = JArray.Parse(json["images"].ToString());
        }
    }
    public void getFlavors()
    {
        URL = "http://" + IP + novaPort + "/v2/" + adminID + "/flavors/detail";
        Uri authUri = new Uri(URL);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
        request.Method = "GET";
        request.UserAgent = "python-keystoneclient";
        request.Headers.Add("X-Auth-Token: " + tokenStr);
        request.ContentType = "application/json";
        request.Accept = "application/json";
        string result = "";
        using (HttpWebResponse resp = request.GetResponse()
                                      as HttpWebResponse)
        {
            StreamReader reader =
                new StreamReader(resp.GetResponseStream());
            result = reader.ReadToEnd();
            JObject json = new JObject();
            json = JObject.Parse(result);
            flavors = JArray.Parse(json["flavors"].ToString());
        }
    }
    public void getServers()
    {
        URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers";
        Uri authUri = new Uri(URL);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
        request.Method = "GET";
        request.ContentType = "application/json";
        request.Accept = "application/json";
        request.UserAgent = "python-keystoneclient";
        request.Headers.Add("X-Auth-Token: " + tokenStr);
        string result = "";
        using (HttpWebResponse resp = request.GetResponse()
                                      as HttpWebResponse)
        {
            StreamReader reader =
                new StreamReader(resp.GetResponseStream());
            result = reader.ReadToEnd();
            JObject json = new JObject();
            json = JObject.Parse(result);
            servers = JArray.Parse(json["servers"].ToString());
        }
    }
    public void createServer(string name, string flavor, string image)
    {
        URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers";
        Uri authUri = new Uri(URL);
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
        request.Method = "POST";
        request.ContentType = "application/json";
        request.Accept = "application/json";
        request.UserAgent = "python-keystoneclient";
        request.Headers.Add("X-Auth-Token: " + tokenStr);
```

```
            byte[] formData = UTF8Encoding.UTF8.GetBytes("{\"server\": {\"flavorRef\":
\"" + flavor + "\",\"imageRef\": \"" + image + "\",\"metadata\": {\"My Server Name\": \""
+ name + "\"},\"name\": \"" + name + "\"}}");
            request.ContentLength = formData.Length;
            using (Stream post = request.GetRequestStream())
            {
                post.Write(formData, 0, formData.Length);
            }
            string result = "";
            using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
            {
                StreamReader reader =
                    new StreamReader(resp.GetResponseStream());
                result = reader.ReadToEnd();
                JObject json = new JObject();
                json = JObject.Parse(result);
                Console.Write(result);
                Console.Read();
            }
        }
        public void getServerDetail(string id)
        {
            URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers/" + id;
            Uri authUri = new Uri(URL);
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
            request.Method = "GET";
            request.ContentType = "application/json";
            request.Accept = "application/json";
            request.UserAgent = "python-keystoneclient";
            request.Headers.Add("X-Auth-Token: " + tokenStr);
            string result = "";
            using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
            {
                StreamReader reader =
                    new StreamReader(resp.GetResponseStream());
                result = reader.ReadToEnd();
                JObject json = new JObject();
                json = JObject.Parse(result);
                serverDetail = JObject.Parse(json.GetValue("server").ToString());
            }
        }
        public void deleteServer(string serverID)
        {
            URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers/" + serverID;
            Uri authUri = new Uri(URL);
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
            request.Method = "DELETE";
            request.ContentType = "application/json";
            request.Accept = "application/json";
            request.UserAgent = "python-keystoneclient";
            request.Headers.Add("X-Auth-Token: " + tokenStr);
            request.GetResponse();
        }
        public void serverAction(string serverID, string command)
        {
```

```
            URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers/" + serverID +
"/action";
            Uri authUri = new Uri(URL);
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
            request.Method = "POST";
            request.ContentType = "application/json";
            request.Accept = "application/json";
            request.UserAgent = "python-keystoneclient";
            request.Headers.Add("X-Auth-Token: " + tokenStr);
            byte[] formData = UTF8Encoding.UTF8.GetBytes(command);
            request.ContentLength = formData.Length;
            using (Stream post = request.GetRequestStream())
            {
                post.Write(formData, 0, formData.Length);
            }
            string result = "";
            using (HttpWebResponse resp = request.GetResponse()
                                        as HttpWebResponse)
            {
                StreamReader reader =
                    new StreamReader(resp.GetResponseStream());
                result = reader.ReadToEnd();
                JObject json = new JObject();
                try
                {
                    json = JObject.Parse(result);
                }
                catch { }
            }
        }
        public void imageServer(string serverID, string command)
        {
            URL = "http://" + IP + novaPort + "/v2/" + adminID + "/servers/" + serverID +
"/action";
            Uri authUri = new Uri(URL);
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(authUri);
            request.Method = "POST";
            request.ContentType = "application/json";
            request.Accept = "application/json";
            request.UserAgent = "python-keystoneclient";
            request.Headers.Add("X-Auth-Token: " + tokenStr);
            byte[] formData = UTF8Encoding.UTF8.GetBytes(command);
            request.ContentLength = formData.Length;
            using (Stream post = request.GetRequestStream())
            {
                post.Write(formData, 0, formData.Length);
            }
            request.GetResponse();
        }
    }
}
```

**Form1.Designer.cs**

```
namespace OpenStackScripting
{
    partial class Form1
    {
```

```
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.toolStrip1 = new System.Windows.Forms.ToolStrip();
            this.btnAdd = new System.Windows.Forms.ToolStripButton();
            this.btnEdit = new System.Windows.Forms.ToolStripButton();
            this.btnRefresh = new System.Windows.Forms.ToolStripButton();
            this.btnDelete = new System.Windows.Forms.ToolStripButton();
            this.toolStripContainer1 = new System.Windows.Forms.ToolStripContainer();
            this.gridMachines = new System.Windows.Forms.DataGridView();
            this.toolStrip1.SuspendLayout();
            this.toolStripContainer1.ContentPanel.SuspendLayout();
            this.toolStripContainer1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.gridMachines)).BeginInit();
            this.SuspendLayout();
            //
            // toolStrip1
            //
            this.toolStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.btnAdd,
            this.btnEdit,
            this.btnRefresh,
            this.btnDelete});
            this.toolStrip1.Location = new System.Drawing.Point(0, 0);
            this.toolStrip1.Name = "toolStrip1";
            this.toolStrip1.Size = new System.Drawing.Size(474, 25);
            this.toolStrip1.TabIndex = 3;
            this.toolStrip1.Text = "toolStrip1";
            //
            // btnAdd
            //
            this.btnAdd.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.btnAdd.Image = global::OpenStackScripting.Properties.Resources.add;
            this.btnAdd.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnAdd.Name = "btnAdd";
            this.btnAdd.Size = new System.Drawing.Size(23, 22);
            this.btnAdd.Text = "Add";
            this.btnAdd.Click += new System.EventHandler(this.btnAdd_Click);
            //
            // btnEdit
            //
            this.btnEdit.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.btnEdit.Image = global::OpenStackScripting.Properties.Resources.edit;
            this.btnEdit.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnEdit.Name = "btnEdit";
            this.btnEdit.Size = new System.Drawing.Size(23, 22);
            this.btnEdit.Text = "Edit";
```

```
            this.btnEdit.Click += new System.EventHandler(this.btnEdit_Click);
            //
            // btnRefresh
            //
            this.btnRefresh.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.btnRefresh.Image =
global::OpenStackScripting.Properties.Resources.refresh;
            this.btnRefresh.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnRefresh.Name = "btnRefresh";
            this.btnRefresh.Size = new System.Drawing.Size(23, 22);
            this.btnRefresh.Text = "Refresh";
            this.btnRefresh.Click += new System.EventHandler(this.btnRefresh_Click);
            //
            // btnDelete
            //
            this.btnDelete.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Image;
            this.btnDelete.Image =
global::OpenStackScripting.Properties.Resources.delete;
            this.btnDelete.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnDelete.Name = "btnDelete";
            this.btnDelete.Size = new System.Drawing.Size(23, 22);
            this.btnDelete.Text = "Delete";
            this.btnDelete.Click += new System.EventHandler(this.btnDelete_Click);
            //
            // toolStripContainer1
            //
            //
            // toolStripContainer1.ContentPanel
            //
            this.toolStripContainer1.ContentPanel.Controls.Add(this.gridMachines);
            this.toolStripContainer1.ContentPanel.Size = new System.Drawing.Size(474,
220);
            this.toolStripContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.toolStripContainer1.Location = new System.Drawing.Point(0, 25);
            this.toolStripContainer1.Name = "toolStripContainer1";
            this.toolStripContainer1.Size = new System.Drawing.Size(474, 245);
            this.toolStripContainer1.TabIndex = 4;
            this.toolStripContainer1.Text = "toolStripContainer1";
            //
            // gridMachines
            //
            this.gridMachines.AllowUserToAddRows = false;
            this.gridMachines.AllowUserToDeleteRows = false;
            this.gridMachines.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumnsMode.Fill;
            this.gridMachines.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.gridMachines.Dock = System.Windows.Forms.DockStyle.Fill;
            this.gridMachines.Location = new System.Drawing.Point(0, 0);
            this.gridMachines.MultiSelect = false;
            this.gridMachines.Name = "gridMachines";
            this.gridMachines.RowHeadersWidth = 40;
            this.gridMachines.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect;
            this.gridMachines.Size = new System.Drawing.Size(474, 220);
            this.gridMachines.TabIndex = 3;
```

```
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(474, 270);
            this.Controls.Add(this.toolStripContainer1);
            this.Controls.Add(this.toolStrip1);
            this.MaximumSize = new System.Drawing.Size(482, 297);
            this.MinimumSize = new System.Drawing.Size(482, 297);
            this.Name = "Form1";
            this.Text = "OpenStack";
            this.toolStrip1.ResumeLayout(false);
            this.toolStrip1.PerformLayout();
            this.toolStripContainer1.ContentPanel.ResumeLayout(false);
            this.toolStripContainer1.ResumeLayout(false);
            this.toolStripContainer1.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.gridMachines)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.ToolStrip toolStrip1;
        private System.Windows.Forms.ToolStripButton btnAdd;
        private System.Windows.Forms.ToolStripButton btnEdit;
        private System.Windows.Forms.ToolStripButton btnDelete;
        private System.Windows.Forms.ToolStripContainer toolStripContainer1;
        private System.Windows.Forms.DataGridView gridMachines;
        private System.Windows.Forms.ToolStripButton btnRefresh;
    }
}
```

**Form1**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    public partial class Form1 : Form
    {
        public string user;
        public string pass;
        public Connect conn;
        public Form1()
        {
            InitializeComponent();
            try
            {
                auth();
                gridMachines.DataSource = conn.servers;
                if (conn.servers.HasValues == true)
                {
```

```
                gridMachines.Columns.Remove("id");
                gridMachines.Columns.Remove("links");
            }
        }
        catch
        {
            Application.Exit();
        }
    }
    public void auth()
    {
        using (frmLogin login = new frmLogin())
        {
            var dialog = login.ShowDialog();

            if (dialog == System.Windows.Forms.DialogResult.OK)
            {
                pass = login.passWord;
                user = login.userName;
                conn = new Connect(user, pass);
                try
                {
                    conn.getToken();
                }
                catch
                {
                    System.Windows.Forms.MessageBox.Show("Unable to authenticate!!
Try again");
                    auth();
                }
                conn.getUserID();
                conn.getServers();
                conn.getImage();
            }
            else
            {

            }
        }
    }
    private void btnDelete_Click(object sender, EventArgs e)
    {
        DialogResult dialogResult = MessageBox.Show("Are you sure you want to delete
this machine?", "Delete Server", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            try
            {
                string servID = gridMachines.CurrentRow.Cells["id"].Value.ToString();
                conn.deleteServer(servID);
            }
            catch
            {
                System.Windows.Forms.MessageBox.Show("Machine Not Found");
                refreshGrid();
            }
        }
        refreshGrid();
```

```
        }
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            refreshGrid();
        }
        public void refreshGrid()
        {
            conn.getServers();
            gridMachines.DataSource = conn.servers;
            if (conn.servers.HasValues == true && gridMachines.RowCount <= 0)
            {
                gridMachines.Columns.Remove("id");
                gridMachines.Columns.Remove("links");
            }
        }
        private void btnAdd_Click(object sender, EventArgs e)
        {
            using (frmBuild build = new frmBuild(conn))
            {
                var dialogAdd = build.ShowDialog();
            }
        }
        private void btnEdit_Click(object sender, EventArgs e)
        {
            string servID = gridMachines.CurrentRow.Cells["id"].Value.ToString();
            using (frmEdit edit = new frmEdit(conn, servID))
            {
                var dialogEdit = edit.ShowDialog();
            }
        }
    }
}
```

**frmBuild.Designer.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    public partial class Form1 : Form
    {
        public string user;
        public string pass;
        public Connect conn;
        public Form1()
        {
            InitializeComponent();
            try
            {
                auth();
                gridMachines.DataSource = conn.servers;
```

```
                if (conn.servers.HasValues == true)
                {
                    gridMachines.Columns.Remove("id");
                    gridMachines.Columns.Remove("links");
                }
            }
            catch
            {
                Application.Exit();
            }
        }
        public void auth()
        {
            using (frmLogin login = new frmLogin())
            {
                var dialog = login.ShowDialog();
                if (dialog == System.Windows.Forms.DialogResult.OK)
                {
                    pass = login.passWord;
                    user = login.userName;
                    conn = new Connect(user, pass);
                    try
                    {
                        conn.getToken();
                    }
                    catch
                    {
                        System.Windows.Forms.MessageBox.Show("Unable to authenticate!!
Try again");
                        auth();
                    }
                    conn.getUserID();
                    conn.getServers();
                    conn.getImage();
                }
                else
                {
                }
            }
        }
        private void btnDelete_Click(object sender, EventArgs e)
        {
            DialogResult dialogResult = MessageBox.Show("Are you sure you want to delete
this machine?", "Delete Server", MessageBoxButtons.YesNo);
            if (dialogResult == DialogResult.Yes)
            {
                try
                {
                    string servID = gridMachines.CurrentRow.Cells["id"].Value.ToString();
                    conn.deleteServer(servID);
                }
                catch
                {
                    System.Windows.Forms.MessageBox.Show("Machine Not Found");
                    refreshGrid();
                }
            }
            refreshGrid();
```

```
        }
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            refreshGrid();
        }
        public void refreshGrid()
        {
            conn.getServers();
            gridMachines.DataSource = conn.servers;
            if (conn.servers.HasValues == true && gridMachines.RowCount <= 0)
            {
                gridMachines.Columns.Remove("id");
                gridMachines.Columns.Remove("links");
            }
        }
        private void btnAdd_Click(object sender, EventArgs e)
        {
            using (frmBuild build = new frmBuild(conn))
            {
                var dialogAdd = build.ShowDialog();
            }
        }
        private void btnEdit_Click(object sender, EventArgs e)
        {
            string servID = gridMachines.CurrentRow.Cells["id"].Value.ToString();
            using (frmEdit edit = new frmEdit(conn, servID))
            {
                var dialogEdit = edit.ShowDialog();
            }
        }
    }
}
```

## frmBuild

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
namespace OpenStackScripting
{
    public partial class frmBuild : Form
    {
        public Connect conn;
        public frmBuild(Connect connect)
        {
            InitializeComponent();
            conn = connect;
            conn.getImage();
            conn.getFlavors();
            gridImg.DataSource = conn.images;
```

```
            gridImg.Columns.Remove("id");
            gridImg.Columns.Remove("links");
            gridFlavor.DataSource = conn.flavors;
            gridFlavor.Columns.Remove("id");
            gridFlavor.Columns.Remove("links");
        }
        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult = System.Windows.Forms.DialogResult.Cancel;
        }
        private void btnSubmit_Click(object sender, EventArgs e)
        {
            try
            {
                string imgID = gridImg.CurrentRow.Cells["id"].Value.ToString();
                string flavorID = gridFlavor.CurrentRow.Cells["id"].Value.ToString();
                if (txtName.Text != "" && txtName.Text != null && flavorID != "" &&
flavorID != null && imgID != "" && imgID != null)
                {
                    conn.createServer(txtName.Text, flavorID, imgID);
                    DialogResult = System.Windows.Forms.DialogResult.OK;
                }
                else
                {
                    MessageBox.Show("Please verify that you have made a selection in each
section.");
                }
            }
            catch
            {
                MessageBox.Show("There was an error in the creation of this machine.
Please try again.");
            }
        }
    }
}
```

## frmEdit.Designer.cs

```
namespace OpenStackScripting
{
    partial class frmEdit
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(frmEdit));
            this.txtName = new System.Windows.Forms.TextBox();
```

```
this.txtStatus = new System.Windows.Forms.TextBox();
this.txtIntIP = new System.Windows.Forms.TextBox();
this.txtExtIP = new System.Windows.Forms.TextBox();
this.lblName = new System.Windows.Forms.Label();
this.lblStatus = new System.Windows.Forms.Label();
this.lblIntIP = new System.Windows.Forms.Label();
this.lblExtIP = new System.Windows.Forms.Label();
this.toolStrip1 = new System.Windows.Forms.ToolStrip();
this.btnRefresh = new System.Windows.Forms.ToolStripButton();
this.btnReboot = new System.Windows.Forms.ToolStripButton();
this.btnPassword = new System.Windows.Forms.ToolStripButton();
this.btnImage = new System.Windows.Forms.ToolStripButton();
this.toolStrip1.SuspendLayout();
this.SuspendLayout();
//
// txtName
//
this.txtName.Location = new System.Drawing.Point(20, 64);
this.txtName.Name = "txtName";
this.txtName.ReadOnly = true;
this.txtName.Size = new System.Drawing.Size(150, 20);
this.txtName.TabIndex = 0;
//
// txtStatus
//
this.txtStatus.Location = new System.Drawing.Point(211, 65);
this.txtStatus.Name = "txtStatus";
this.txtStatus.ReadOnly = true;
this.txtStatus.Size = new System.Drawing.Size(150, 20);
this.txtStatus.TabIndex = 1;
//
// txtIntIP
//
this.txtIntIP.Location = new System.Drawing.Point(20, 122);
this.txtIntIP.Name = "txtIntIP";
this.txtIntIP.ReadOnly = true;
this.txtIntIP.Size = new System.Drawing.Size(150, 20);
this.txtIntIP.TabIndex = 2;
//
// txtExtIP
//
this.txtExtIP.Location = new System.Drawing.Point(211, 123);
this.txtExtIP.Name = "txtExtIP";
this.txtExtIP.ReadOnly = true;
this.txtExtIP.Size = new System.Drawing.Size(150, 20);
this.txtExtIP.TabIndex = 4;
//
// lblName
//
this.lblName.AutoSize = true;
this.lblName.Location = new System.Drawing.Point(20, 45);
this.lblName.Name = "lblName";
this.lblName.Size = new System.Drawing.Size(69, 13);
this.lblName.TabIndex = 5;
this.lblName.Text = "Server Name";
//
// lblStatus
//
```

```
            this.lblStatus.AutoSize = true;
            this.lblStatus.Location = new System.Drawing.Point(211, 45);
            this.lblStatus.Name = "lblStatus";
            this.lblStatus.Size = new System.Drawing.Size(37, 13);
            this.lblStatus.TabIndex = 6;
            this.lblStatus.Text = "Status";
            //
            // lblIntIP
            //
            this.lblIntIP.AutoSize = true;
            this.lblIntIP.Location = new System.Drawing.Point(20, 103);
            this.lblIntIP.Name = "lblIntIP";
            this.lblIntIP.Size = new System.Drawing.Size(55, 13);
            this.lblIntIP.TabIndex = 7;
            this.lblIntIP.Text = "Internal IP";
            //
            // lblExtIP
            //
            this.lblExtIP.AutoSize = true;
            this.lblExtIP.Location = new System.Drawing.Point(213, 104);
            this.lblExtIP.Name = "lblExtIP";
            this.lblExtIP.Size = new System.Drawing.Size(58, 13);
            this.lblExtIP.TabIndex = 8;
            this.lblExtIP.Text = "External IP";
            //
            // toolStrip1
            //
            this.toolStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.btnRefresh,
            this.btnReboot,
            this.btnPassword,
            this.btnImage});
            this.toolStrip1.Location = new System.Drawing.Point(0, 0);
            this.toolStrip1.Name = "toolStrip1";
            this.toolStrip1.Size = new System.Drawing.Size(382, 25);
            this.toolStrip1.TabIndex = 9;
            this.toolStrip1.Text = "toolStrip1";
            //
            // btnRefresh
            //
            this.btnRefresh.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Text;
            this.btnRefresh.Image =
((System.Drawing.Image)(resources.GetObject("btnRefresh.Image")));
            this.btnRefresh.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnRefresh.Name = "btnRefresh";
            this.btnRefresh.Size = new System.Drawing.Size(49, 22);
            this.btnRefresh.Text = "Refresh";
            this.btnRefresh.Click += new System.EventHandler(this.btnRefresh_Click);
            //
            // btnReboot
            //
            this.btnReboot.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Text;
            this.btnReboot.Image =
((System.Drawing.Image)(resources.GetObject("btnReboot.Image")));
            this.btnReboot.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnReboot.Name = "btnReboot";
```

```
            this.btnReboot.Size = new System.Drawing.Size(46, 22);
            this.btnReboot.Text = "Reboot";
            this.btnReboot.Click += new System.EventHandler(this.btnReboot_Click);
            //
            // btnPassword
            //
            this.btnPassword.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Text;
            this.btnPassword.Image =
((System.Drawing.Image)(resources.GetObject("btnPassword.Image")));
            this.btnPassword.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnPassword.Name = "btnPassword";
            this.btnPassword.Size = new System.Drawing.Size(88, 22);
            this.btnPassword.Text = "Reset Password";
            this.btnPassword.Visible = false;
            this.btnPassword.Click += new System.EventHandler(this.btnPassword_Click);
            //
            // btnImage
            //
            this.btnImage.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.Text;
            this.btnImage.Image =
((System.Drawing.Image)(resources.GetObject("btnImage.Image")));
            this.btnImage.ImageTransparentColor = System.Drawing.Color.Magenta;
            this.btnImage.Name = "btnImage";
            this.btnImage.Size = new System.Drawing.Size(83, 22);
            this.btnImage.Text = "Image Machine";
            this.btnImage.Click += new System.EventHandler(this.btnImage_Click);
            //
            // frmEdit
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(382, 183);
            this.Controls.Add(this.toolStrip1);
            this.Controls.Add(this.lblExtIP);
            this.Controls.Add(this.lblIntIP);
            this.Controls.Add(this.lblStatus);
            this.Controls.Add(this.lblName);
            this.Controls.Add(this.txtExtIP);
            this.Controls.Add(this.txtIntIP);
            this.Controls.Add(this.txtStatus);
            this.Controls.Add(this.txtName);
            this.MaximizeBox = false;
            this.MaximumSize = new System.Drawing.Size(390, 210);
            this.MinimizeBox = false;
            this.MinimumSize = new System.Drawing.Size(390, 210);
            this.Name = "frmEdit";
            this.Text = "Machine Info";
            this.toolStrip1.ResumeLayout(false);
            this.toolStrip1.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.TextBox txtName;
        private System.Windows.Forms.TextBox txtStatus;
        private System.Windows.Forms.TextBox txtIntIP;
```

```
        private System.Windows.Forms.TextBox txtExtIP;
        private System.Windows.Forms.Label lblName;
        private System.Windows.Forms.Label lblStatus;
        private System.Windows.Forms.Label lblIntIP;
        private System.Windows.Forms.Label lblExtIP;
        private System.Windows.Forms.ToolStrip toolStrip1;
        private System.Windows.Forms.ToolStripButton btnReboot;
        private System.Windows.Forms.ToolStripButton btnPassword;
        private System.Windows.Forms.ToolStripButton btnImage;
        private System.Windows.Forms.ToolStripButton btnRefresh;

    }
}
```

**frmEdit**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
namespace OpenStackScripting
{
    public partial class frmEdit : Form
    {
        public Connect conn;
        public string externalIP;
        public string internalIP;
        public string servID;
        public frmEdit(Connect connect, string serverID)
        {
            InitializeComponent();
            servID = serverID;
            conn = connect;
            this.refresh();
        }
        private void btnReboot_Click(object sender, EventArgs e)
        {
            try
            {
                conn.serverAction(servID, "{\"reboot\": {\"type\": \"SOFT\"}}");
            }
            catch
            {
                MessageBox.Show("Machine could not be rebooted at this time.");
            }
        }
        private void btnPassword_Click(object sender, EventArgs e)
        {
            using (frmPassword pass = new frmPassword())
            {
                var dialogPass = pass.ShowDialog();
```

```
                if (dialogPass == DialogResult.OK)
                {
                    string newPass = pass.password;
                    conn.serverAction(servID, "{\"changePassword\": {\"adminPass\": \"" +
newPass + "\"}}");
                }
            }
        }
        private void btnImage_Click(object sender, EventArgs e)
        {
            try
            {
                using (frmImage img = new frmImage())
                {
                    var dialogPass = img.ShowDialog();
                    if (dialogPass == DialogResult.OK)
                    {
                        conn.imageServer(servID, "{\"createImage\": {\"name\": \"" +
img.imageName + "\"}}");
                    }
                }
            }
            catch
            {
                MessageBox.Show("Machine could not be imaged at this time.");
            }
        }
        private void refresh()
        {
            conn.getServerDetail(servID);
            txtName.Text = conn.serverDetail.GetValue("name").ToString();
            txtStatus.Text = conn.serverDetail.GetValue("status").ToString();
            try
            {
                JArray IPs =
JArray.Parse(JObject.Parse(conn.serverDetail.GetValue("addresses").ToString()).GetValue("
private").ToString());
                foreach (JObject o in IPs.Children<JObject>())
                {
                    if (o.GetValue("OS-EXT-IPS:type").ToString() == "floating")
                    {
                        externalIP = o.GetValue("addr").ToString();
                    }
                    else if (o.GetValue("OS-EXT-IPS:type").ToString() == "fixed")
                    {
                        internalIP = o.GetValue("addr").ToString();
                    }
                }
                txtIntIP.Text = internalIP;
                txtExtIP.Text = externalIP.ToString();
            }
            catch { }
        }
        private void btnRefresh_Click(object sender, EventArgs e)
        {
            this.refresh();
        }
    }
```

```
}
```

**frmImage.Designer.cs**

```
namespace OpenStackScripting
{
    partial class frmImage
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.lblName = new System.Windows.Forms.Label();
            this.txtName = new System.Windows.Forms.TextBox();
            this.btnSubmit = new System.Windows.Forms.Button();
            this.btnCancel = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // lblName
            //
            this.lblName.AutoSize = true;
            this.lblName.Location = new System.Drawing.Point(18, 34);
            this.lblName.Name = "lblName";
            this.lblName.Size = new System.Drawing.Size(67, 13);
            this.lblName.TabIndex = 0;
            this.lblName.Text = "Image Name";
            //
            // txtName
            //
            this.txtName.Location = new System.Drawing.Point(21, 50);
            this.txtName.Name = "txtName";
            this.txtName.Size = new System.Drawing.Size(226, 20);
            this.txtName.TabIndex = 1;
            //
            // btnSubmit
            //
            this.btnSubmit.Location = new System.Drawing.Point(51, 85);
            this.btnSubmit.Name = "btnSubmit";
            this.btnSubmit.Size = new System.Drawing.Size(75, 23);
            this.btnSubmit.TabIndex = 2;
            this.btnSubmit.Text = "Submit";
            this.btnSubmit.UseVisualStyleBackColor = true;
            this.btnSubmit.Click += new System.EventHandler(this.btnSubmit_Click);
            //
            // btnCancel
            //
            this.btnCancel.Location = new System.Drawing.Point(141, 85);
            this.btnCancel.Name = "btnCancel";
            this.btnCancel.Size = new System.Drawing.Size(75, 23);
            this.btnCancel.TabIndex = 3;
```

```
            this.btnCancel.Text = "Cancel";
            this.btnCancel.UseVisualStyleBackColor = true;
            this.btnCancel.Click += new System.EventHandler(this.btnCancel_Click);
            //
            // frmImage
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(265, 123);
            this.Controls.Add(this.btnCancel);
            this.Controls.Add(this.btnSubmit);
            this.Controls.Add(this.txtName);
            this.Controls.Add(this.lblName);
            this.MaximizeBox = false;
            this.MaximumSize = new System.Drawing.Size(273, 150);
            this.MinimizeBox = false;
            this.MinimumSize = new System.Drawing.Size(273, 150);
            this.Name = "frmImage";
            this.Text = "Image Machine";
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Label lblName;
        private System.Windows.Forms.TextBox txtName;
        private System.Windows.Forms.Button btnSubmit;
        private System.Windows.Forms.Button btnCancel;
    }
}
```

**frmImage**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    public partial class frmImage : Form
    {
        public string imageName;
        public frmImage()
        {
            InitializeComponent();
        }
        private void btnSubmit_Click(object sender, EventArgs e)
        {
            if (txtName.Text != null && txtName.Text != "")
            {
                imageName = txtName.Text;
                DialogResult = System.Windows.Forms.DialogResult.OK;
            }
            else
```

```
            {
                MessageBox.Show("Please enter a name for the image!");
            }
        }
        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult = System.Windows.Forms.DialogResult.Cancel;
        }
    }
}
```

**frmLogin.Designer.cs**

```
namespace OpenStackScripting
{
    partial class frmLogin
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.label2 = new System.Windows.Forms.Label();
            this.txtUser = new System.Windows.Forms.TextBox();
            this.txtPass = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.btnSubmit = new System.Windows.Forms.Button();
            this.btnCancel = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(25, 77);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(53, 13);
            this.label2.TabIndex = 1;
            this.label2.Text = "Password";
            //
            // txtUser
            //
            this.txtUser.Location = new System.Drawing.Point(90, 34);
            this.txtUser.Name = "txtUser";
            this.txtUser.Size = new System.Drawing.Size(119, 20);
            this.txtUser.TabIndex = 1;
            //
            // txtPass
            //
            this.txtPass.Location = new System.Drawing.Point(90, 77);
            this.txtPass.Name = "txtPass";
            this.txtPass.PasswordChar = '*';
```

```
        this.txtPass.Size = new System.Drawing.Size(119, 20);
        this.txtPass.TabIndex = 2;
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(22, 34);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(55, 13);
        this.label1.TabIndex = 0;
        this.label1.Text = "Username";
        //
        // btnSubmit
        //
        this.btnSubmit.Location = new System.Drawing.Point(25, 117);
        this.btnSubmit.Name = "btnSubmit";
        this.btnSubmit.Size = new System.Drawing.Size(75, 23);
        this.btnSubmit.TabIndex = 3;
        this.btnSubmit.Text = "Submit";
        this.btnSubmit.UseVisualStyleBackColor = true;
        this.btnSubmit.Click += new System.EventHandler(this.btnSubmit_Click);
        //
        // btnCancel
        //
        this.btnCancel.Location = new System.Drawing.Point(134, 117);
        this.btnCancel.Name = "btnCancel";
        this.btnCancel.Size = new System.Drawing.Size(75, 23);
        this.btnCancel.TabIndex = 4;
        this.btnCancel.Text = "Cancel";
        this.btnCancel.UseVisualStyleBackColor = true;
        this.btnCancel.Click += new System.EventHandler(this.btnCancel_Click);
        //
        // frmLogin
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(229, 156);
        this.Controls.Add(this.btnCancel);
        this.Controls.Add(this.btnSubmit);
        this.Controls.Add(this.txtPass);
        this.Controls.Add(this.txtUser);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.MaximizeBox = false;
        this.MaximumSize = new System.Drawing.Size(237, 183);
        this.MinimizeBox = false;
        this.MinimumSize = new System.Drawing.Size(237, 183);
        this.Name = "frmLogin";
        this.Text = "Login";
        this.ResumeLayout(false);
        this.PerformLayout();
    }
    #endregion
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.TextBox txtUser;
    private System.Windows.Forms.TextBox txtPass;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Button btnSubmit;
```

```
        private System.Windows.Forms.Button btnCancel;
    }
}
```

## frmLogin

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    public partial class frmLogin : Form
    {
        public string userName;
        public string passWord;
        public frmLogin()
        {
            InitializeComponent();
        }
        private void btnSubmit_Click(object sender, EventArgs e)
        {
            if(txtPass.Text != "" && txtPass != null && txtUser.Text != "" && txtUser !=
null)
            {
                userName = txtUser.Text;
                passWord = txtPass.Text;
            }
            DialogResult = DialogResult.OK;
        }
        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.Cancel;
        }
    }
}
```

## frmPassword.Designer.cs

```
namespace OpenStackScripting
{
    partial class frmPassword
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        #region Windows Form Designer generated code
        private void InitializeComponent()
```

```
{
    this.lblPassword = new System.Windows.Forms.Label();
    this.lblConfirm = new System.Windows.Forms.Label();
    this.txtPassword = new System.Windows.Forms.TextBox();
    this.txtConfirm = new System.Windows.Forms.TextBox();
    this.btnSubmit = new System.Windows.Forms.Button();
    this.btnCancel = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // lblPassword
    //
    this.lblPassword.AutoSize = true;
    this.lblPassword.Location = new System.Drawing.Point(12, 35);
    this.lblPassword.Name = "lblPassword";
    this.lblPassword.Size = new System.Drawing.Size(81, 13);
    this.lblPassword.TabIndex = 0;
    this.lblPassword.Text = "New Password:";
    //
    // lblConfirm
    //
    this.lblConfirm.AutoSize = true;
    this.lblConfirm.Location = new System.Drawing.Point(12, 78);
    this.lblConfirm.Name = "lblConfirm";
    this.lblConfirm.Size = new System.Drawing.Size(94, 13);
    this.lblConfirm.TabIndex = 1;
    this.lblConfirm.Text = "Confirm Password:";
    //
    // txtPassword
    //
    this.txtPassword.Location = new System.Drawing.Point(112, 32);
    this.txtPassword.Name = "txtPassword";
    this.txtPassword.PasswordChar = '*';
    this.txtPassword.Size = new System.Drawing.Size(181, 20);
    this.txtPassword.TabIndex = 1;
    //
    // txtConfirm
    //
    this.txtConfirm.Location = new System.Drawing.Point(112, 75);
    this.txtConfirm.Name = "txtConfirm";
    this.txtConfirm.PasswordChar = '*';
    this.txtConfirm.Size = new System.Drawing.Size(181, 20);
    this.txtConfirm.TabIndex = 2;
    //
    // btnSubmit
    //
    this.btnSubmit.Location = new System.Drawing.Point(53, 124);
    this.btnSubmit.Name = "btnSubmit";
    this.btnSubmit.Size = new System.Drawing.Size(75, 23);
    this.btnSubmit.TabIndex = 3;
    this.btnSubmit.Text = "Submit";
    this.btnSubmit.UseVisualStyleBackColor = true;
    this.btnSubmit.Click += new System.EventHandler(this.btnSubmit_Click);
    //
    // btnCancel
    //
    this.btnCancel.Location = new System.Drawing.Point(188, 124);
    this.btnCancel.Name = "btnCancel";
    this.btnCancel.Size = new System.Drawing.Size(75, 23);
```

```
            this.btnCancel.TabIndex = 4;
            this.btnCancel.Text = "Cancel";
            this.btnCancel.UseVisualStyleBackColor = true;
            this.btnCancel.Click += new System.EventHandler(this.btnCancel_Click);
            //
            // frmPassword
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(312, 162);
            this.Controls.Add(this.btnCancel);
            this.Controls.Add(this.btnSubmit);
            this.Controls.Add(this.txtConfirm);
            this.Controls.Add(this.txtPassword);
            this.Controls.Add(this.lblConfirm);
            this.Controls.Add(this.lblPassword);
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "frmPassword";
            this.Text = "Password Reset";
            this.ResumeLayout(false);
            this.PerformLayout();
        }
        #endregion
        private System.Windows.Forms.Label lblPassword;
        private System.Windows.Forms.Label lblConfirm;
        private System.Windows.Forms.TextBox txtPassword;
        private System.Windows.Forms.TextBox txtConfirm;
        private System.Windows.Forms.Button btnSubmit;
        private System.Windows.Forms.Button btnCancel;
    }
}
```

**frmPassword**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    public partial class frmPassword : Form
    {
        public string password;
        public frmPassword()
        {
            InitializeComponent();
        }
        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult = System.Windows.Forms.DialogResult.Cancel;
        }
        private void btnSubmit_Click(object sender, EventArgs e)
```

```
        {
            if (txtPassword.Text == txtConfirm.Text)
            {
                password = txtPassword.Text;
                DialogResult = System.Windows.Forms.DialogResult.OK;
            }
            else
            {
                MessageBox.Show("Passwords do not match!");
            }
        }
    }
}
```

**Program.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace OpenStackScripting
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```