



THE UNIVERSITY OF  

---

MELBOURNE

# Aneka Web Portal

MEDC Research Project 2011

**RESEARCH & DEVELOPMENT BY: MOHAMMED S. ALROKAYAN**  
**STUDENT NUMBER: 337557**

November 21, 2011  
SUPERVISOR: DR. RAJKUMAR BUYYA

# Abstract

---

This project is a result of 600 hours of research, design and developments during the period from 18 July 2011 to 20 November 2011. Over 50 widgets have been designed and implemented, and over than 8,500 line of codes been written.

Aneka Web Portal aims to design and develop a web application for Aneka administrators to manage all the nodes in the system. Also, it aims to give the non-technical executives in any organization an easy to use web application to view a summary of the current system statues.

Aneka Web Portal is not just for the system administrators and top managements, but also for the Portal developers. One of the main goals in this project is to develop an easy to understand code structure and a very high flexible infrastructure for any future changes to the portal.

## Acknowledgment

I would like to thank Dileban Karunamoorthy for helping me designing, debugging and testing the project since the beginning; also he helped a lot steering the project to the right direction.

# This Report Has Six Chapters:

---

- **Chapter One:** Basic concepts
- **Chapter Two:** Project Design
- **Chapter Three:** Project Development
- **Chapter Four:** Aneka Web Portal Installation
- **Chapter Five:** Conclusion & Future Work
- **Chapter Six:** References & Appendixes

## Table of Contents

<b>Chapter One: Basic Concepts</b> .....	<b>8</b>
Introduction .....	8
Aneka .....	8
ASP.NET MVC 3, RAZOR Syntax .....	9
MVC .....	9
ASP.NET MVC 3 .....	10
Razor .....	11
Project Goal .....	11
Project Scope .....	11
Functional Requirements .....	11
Non-Functional Requirements .....	18
<b>Chapter Two: Project Design</b> .....	<b>20</b>
Aneka Web Portal Architecture .....	22
Database Design .....	23
The Concept of Widgets in the Dashboard .....	26
The Concept of the Issue and Activity Centres .....	27
Issues Centre .....	27
Activity Centre .....	28
The Concept of the In-page Popup Dialog Box .....	28
<b>Chapter Three: Project Development [10]</b> .....	<b>30</b>
Widgets .....	30
Widgets Summary .....	30
Widgets Details .....	32
Applications List Widget .....	32
Historical CPU Utilization Widget .....	33
Live CPU Utilization Widget .....	34
Summary of Jobs Status Widget .....	35
Create Cloud Widget .....	36

Delete Cloud Widget .....	37
Cloud's Details Widget .....	37
Edit a Cloud Widget .....	38
List of Clouds Widget .....	38
Create Cloud User Widget .....	39
Delete a Cloud User Widget.....	39
Cloud User's Details Widget.....	39
Edit a Cloud User Widget .....	40
List of Cloud Users Widget .....	40
Create a Dashboard Widget Widget .....	40
Create a Widget to add it To the Dashboard Widget .....	41
Delete a Dashboard Widget Widget .....	41
Dashboard Widget's Details Widget .....	42
Edit a Dashboard Widget Widget.....	42
Create Machine Widget .....	42
Delete a Machine Widget .....	43
Machine's Details Widget .....	43
Edit a Machine Widget.....	44
Create Machines Login Credential Widget .....	44
Delete a Machines Login Credential Widget.....	44
Machines Login Credential's Details Widget .....	45
Edit a Machines Login Credential Widget .....	45
List all Machines Login Credentials Widget .....	46
Create Portal User Widget .....	46
Delete a Portal User Widget .....	47
Portal User's Details Widget .....	47
Edit a Portal User Widget.....	47
List all Portal Users Widget .....	48
Create Resource Pool Widget .....	48
Delete a Resource Pool Widget .....	49
Resource Pool's Details Widget .....	49
Edit a Resource Pool Widget.....	50
List all Resource Pools Widget .....	50
Create Service Widget.....	51

Delete a Service Widget.....	52
Service’s Details Widget.....	52
Edit a Service Widget .....	52
List all Services Widget.....	53
Create Software Appliance Widget.....	53
Delete a Software Appliance Widget.....	54
Software Appliance’s Details Widget.....	54
Edit a Software Appliance Widget .....	54
List all Software Appliances Widget.....	55
Create Worker Widget .....	55
Delete a Worker Widget .....	56
Worker’s Details Widget .....	56
Edit a Worker Widget.....	56
List all Workers Widget .....	57
Pages .....	57
/CloudManagement/BrowseClouds .....	57
/CloudManagement/BrowseCloudUsers.....	58
/CloudManagement/CloudDetails/1 .....	58
/Home/Dashboard .....	59
/Home/LogOn .....	59
/Infrastructure/ManageMachinesandResourcePools.....	60
/Infrastructure/ManageMachinesLoginCredentials .....	60
/Services/BrowseServicesCatalog .....	60
/SoftwareAppliances/BrowseSoftwareAppliances .....	61
/WebPortalUsersManagement/UsersManagement.....	61
Partial Views (Other Than the Widgets): .....	61
/Home/DashboardContent .....	62
/Shared/_ActivitiesList .....	62
/Shares/_IssuesList .....	62
To-do list For a Developer to Create a New Widget Controller: .....	63

<b>Chapter Four: Aneka Web Portal Installation .....</b>	<b>64</b>
System requirements.....	64
Supported Operating Systems.....	64
Supported Architectures:.....	64
Hardware Requirements:.....	64
Prerequisites: .....	64
Installation .....	64
<b>Chapter Five: Conclusion &amp; Future Work.....</b>	<b>66</b>
Conclusion: .....	66
Future work: .....	66
Tasks in the Logic .....	66
Tasks in the User Interface .....	66
<b>Chapter Six: References &amp; Appendixes.....</b>	<b>68</b>
References .....	68
Appendix.....	71
The Constellation Admin Skin [9] .....	71
Highchart [16] .....	71
Fugue Icon Set [18] .....	72
FineFiles Icon Set [19] .....	72
Flags Icon Set [20] .....	72
Web Application Icon Set.....	72
jQuery hashChange plugin [21] .....	72
SyntaxHighlighter .....	72
DataTables [22] .....	72
jQuery DatePicker [23] .....	72
JSMin.php .....	72

## Table of Figures

Figure 1: Aneka PaaS [1] .....	9
Figure 2: MVC design pattern [2] .....	10
Figure 3: Aneka Web Portal System Design.....	20
Figure 4: Aneka Web Portal Architecture.....	23
Figure 5: Database Design Diagram 1 of 2.....	24
Figure 6: Database Design Diagram 2 of 2.....	25
Figure 7: Issues Centre.....	27
Figure 8: Activity Centre.....	28
Figure 9: The Concept of the In-page Popup Dialog Box .....	28
Figure 10: _ApplicationsReport Controller - ApplicationsList Action .....	32
Figure 11: _Charts Controller - CPU_Utilization_Range Action - Summary option .....	33
Figure 12: _Charts Controller - CPU_Utilization_Range Action - Detailed option .....	33
Figure 13: _Charts Controller - CPU_Utilization_Live Action .....	34
Figure 14: _Charts Controller - JobsStatusSummary Action .....	35
Figure 15: _Clouds Controller - Create Action .....	36
Figure 16: _Clouds Controller - Details Action.....	37
Figure 17: _Clouds Controller - List Action .....	38
Figure 18: _MachinesLoginCredentials Controller - List Action .....	46
Figure 19: _PortalUsers Controller - List Action .....	48
Figure 20: _ResourcePool Controller - Details Action.....	49
Figure 21: _ResourcePool Controller - List Action .....	51
Figure 22: _Services Controller - List Action .....	53
Figure 23: _SoftwareAppliances Controller - List Action .....	55
Figure 24: List all Workers Widget .....	57
Figure 25:Aneka Login Box.....	60
Figure 26: Activities List .....	62
Figure 27: Issues List .....	62



# Chapter One: Basic Concepts

---

## Introduction

Recently lots of researches and projects are focused on cloud computing, and most of the technology-based companies are investing millions of dollars in cloud computing area. Cloud computing technology aims to convert the idea of trading products to virtual trading, to trade services. As a result, in the near future instead of buying a computer the users will buy services and pay according to their usage, this kind of computing is also called Utility Computing.

This project, Aneka Web Portal, will use the concept of cloud/utility computing by giving the users an easy to use graphical user interface web application system to manage a huge number of distributed computers very easily. This project doesn't just give the user a wonderful GUI, but also it builds a very strong solid infrastructure for any future development or changes on the portal.

The project is very huge and it has hundreds of features. The most important features have been implemented and they are in a stable version without any known bugs or errors.

## Aneka

Aneka is a cloud computing Platform as a Service (PaaS) application that supports different programming models, like: Task Programming, Thread Programming and MapReduce Programming.

*One of the notable characteristics of Aneka PaaS is to support provisioning of private cloud resources ranging from desktops, clusters to virtual datacenters using VMWare, Citrix Zen server and public cloud resources such as Windows Azure, Amazon EC2, and GoGrid Cloud Service. [1]*

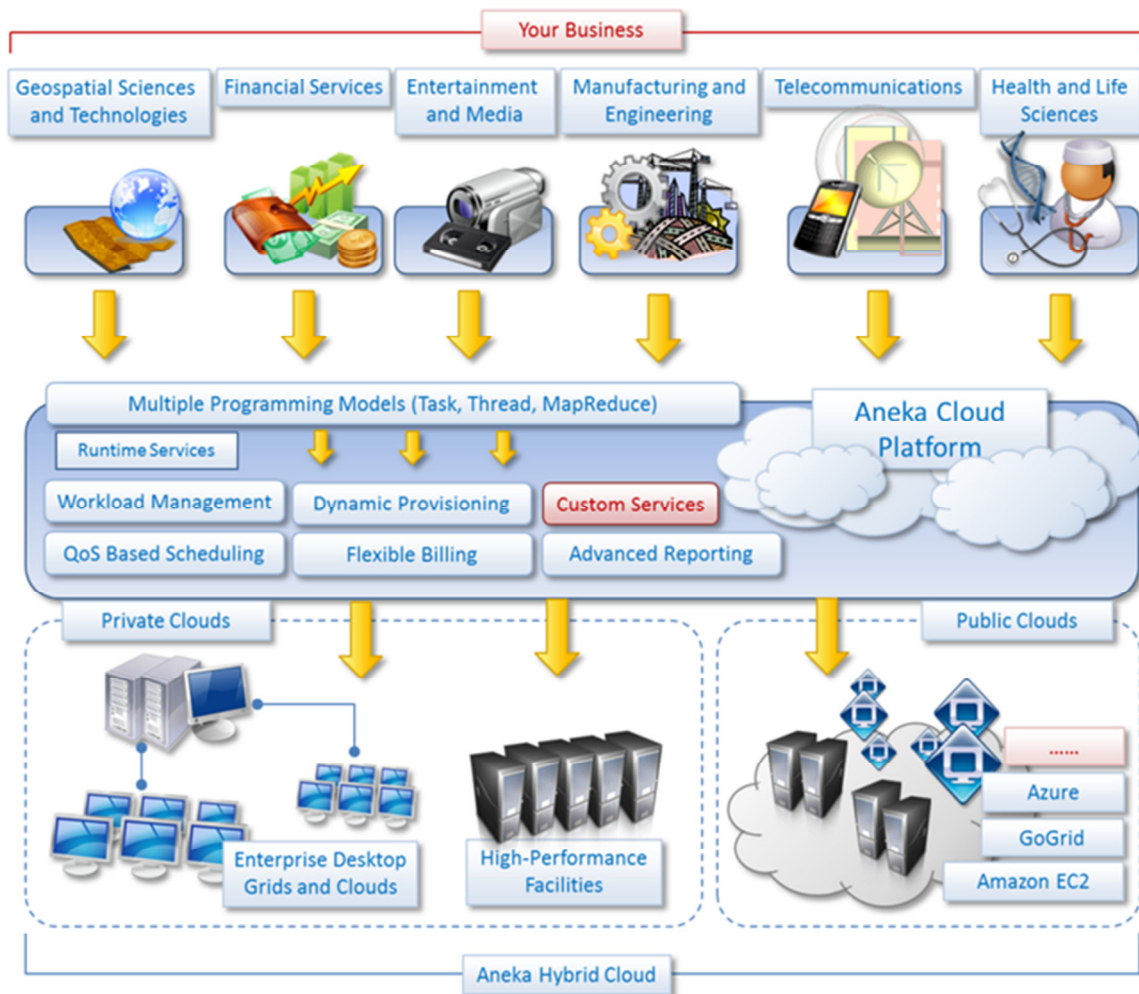


Figure 1: Aneka PaaS [1]

## ASP.NET MVC 3, RAZOR Syntax

The project has been implemented using the latest Microsoft web technology, ASP.NET MVC 4, and using the most recent ASP.NET syntax, which is Razor.

### MVC

The Model-View-Controller (MVC) design layout separates a Web app into three primary elements, as following:

- **Models:** Model components are the application's data layer. Often, this component interacts with the database to retrieve and store model state.
- **Views:** Views are the components that display the application's graphical user interface to the browser. Typically, this GUI is associated with a

model, so a single view operates on a single model; however this is not always the case.

- **Controllers:** This components works together with the model, and eventually choose a view to display UI to the user via a browser. The main use of this component is to handle user's functions, and this is what the user's call when they request a specific URL. In an MVC web app the view only shows information while the controller handles and responds to user requests and interactions.

The ASP.NET MVC framework is a compact, extremely “testable” presentation framework, which offers a different approach to the ASP.NET Web Forms pattern. The MVC design help the developers to create Web applications that isolate the different parts of the Web app (input logic, GUI logic, and business logic), as well as giving a loose coupling among these components. The loose coupling among the three main components of an MVC Web app provides parallel development, flexibility, easy future changes, and fast debugging. [2]

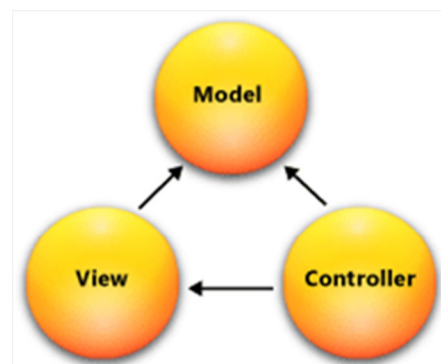


Figure 2: MVC design pattern [2]

### ASP.NET MVC 3

*“ASP.NET MVC 3 is a framework for building scalable, standards-based web applications using well-established design patterns and the power of ASP.NET and the .NET Framework.” [3]*

The third version of ASP.NET MVC brings very high development flexibility and very high web application performance. ASP.NET MVC 3 aims to give the developers a quicker method to develop web applications by using what Microsoft called it: Extensible Scaffolding with MvcScaffold integration. Using ASP.NET MVC 3 in Aneka Web Portal opens more opportunity for the Aneka Web Portal developers to edit any exist features and add almost any desired feature very easily and in a more professional way. Because ASP.NET MVC uses the latest ASP.NET framework, version 4, it gives the portal a higher

performance and utilizes the server resources in a more efficient way. [4] [5] [6] [3]

## Razor

The main idea of using the MVC is to separate the different logical functions parts of the application into three main parts or components, View, Model and Controllers. So because view is a separate components the developers can use any view-engine they like, and we are using Razor view-engine in Aneka Web Portal.

I choose Razer over the “default” ASP.NET Web Forms view-engine that uses the .aspx/.ascx/.master file templates because it uses code-focused templating approach, and that means the developers do not have to deal with the designer user interface in Visual Studio, we just have to define all GUI valuables in the CSS and write a very simple straightforward HTML mixed with Razor code in the views. In Aneka Web Portal those views have .cshtml file extension because we are using C# language. [7]

## Project Goal

Aneka Web Portal aims to design and develop a web application for Aneka administrators to manage all the nodes in the system. Also, it aims to give the non-technical executives in any organization an easy to use web application to view a summary of the current system statuses.

Aneka Web Portal is not just for the system administrators and non-technical executives, but also for the Portal developers. One of the main goals is to develop a very high flexible infrastructure for any future changes to the portal. Because we are using the MVC, this change could be occurred in the graphical user interface, which is the views, or in the business logic, which is the models, or in the data process, which are the controllers.

## Project Scope

The scope of Aneka Web Portal project is to accomplish the following functional and non-functional requirements:

### Functional Requirements

- Web Portal Login Credential:

- The ability to execute the following functions:
  - Create new portal user
  - Delete a portal user
  - Show a detailed information about a portal user
  - Edit a portal user
  - List all portal users
- “Web Portal Login Credential” database entity using Entity Framework that has the following attributes:
  - Username as String
  - Password as String
  - Widgets as a list of *Widget* entities
- Widget:
  - The ability to execute the following functions:
    - Create new Dashboard Widget
    - Delete a Dashboard Widget
    - Show a detailed information about a Dashboard Widget
    - Edit a Dashboard Widget
  - “Widget” database entity using Entity Framework that has the following attributes:
    - Controller Name as String
    - Action Name as String
    - Action Id Number (like user ID or cloud ID) as Integer
    - Widget Width as Integer
- Machine Login Credential:
  - The ability to execute the following functions:
    - Create new Machine Login Credential
    - Delete a Machine Login Credential
    - Show a detailed information about a Machine Login Credential
    - Edit a Machine Login Credential
    - List all Machine Login Credentials
  - “Machine Login Credential” database entity using Entity Framework that has the following attributes:
    - Username as String
    - Password as String

- Associated Machines as a list of *Machine* entities
- Service:
  - The ability to execute the following functions:
    - Create new Service
    - Delete a Service
    - Show a detailed information about a Service
    - Edit a Service
    - List all Services
  - “Service” database entity using Entity Framework that has the following attributes:
    - Service name as String
    - Master only Boolean as Boolean
    - Configuration Spring XML Segment as String
    - Associated Workers as a list of *Worker* entities
    - Associated Masters as a list of *Master* entities
- Cloud User Account:
  - The ability to execute the following functions:
    - Create new Cloud User Account
    - Delete a Cloud User Account
    - Show a detailed information about a Cloud User Account
    - Edit a Cloud User Account
    - List all Cloud User Accounts
  - “Cloud User Account” database entity using Entity Framework that has the following attributes:
    - Username as String
    - Password as String
    - Allow this account to be used for reporting Boolean as Boolean
    - Associated Clouds as a list of *Cloud* entities
- Software Appliance:
  - The ability to execute the following functions:
    - Create new Software Appliance
    - Delete a Software Appliance
    - Show a detailed information about a Software Appliance
    - Edit a Software Appliance

- List all Software Appliances
- “Software Appliance” database entity using Entity Framework that has the following attributes:
  - Software appliance name as String
  - Software appliance vendor as String
  - Software appliance version number as String
  - Associated Machines as a list of *Machine* entities
- Machine and Daemon:
  - The ability to execute the following functions:
    - Create new Machine and the ability
    - Install Aneka Daemon on a machine
    - Delete a Machine
    - Show a detailed information about a Machine and it’s Daemon
    - Edit a Machine and it’s Daemon information
    - Uninstall Daemon
    - Stop Aneka Daemon on a machine
    - Start Aneka Daemon on a machine
    - Restart Aneka Daemon on a machine
  - “Machine” database entity using Entity Framework that has the following attributes:
    - Machine Display Name as String
    - IP address as String
    - Platform/OS as *MachinePlatform* entity
    - Machine type as *MachineType* entity
    - Daemon as *Daemon* entity
    - Associated Workers as a list of *Worker* entities
    - Associated Masters as a list of *Masters* entities
    - Daemon Status as enumerable (stored in database as integer), which has the following values:
      - Unknown = 0
      - InvalidServicePort = 1
      - ServiceAccessDenied = 2
      - NetworkNotReachable = 3
      - BadCredentials = 4

- ServiceStarted = 5
  - ServiceStopped = 6
  - ServiceNotInstalled = 7
  - Is machine in progress Boolean as Boolean
  - Progress Message as String
  - Associated Software Appliances as a list of *SoftwareAppliance* entities
- “Daemon” database entity using Entity Framework that has the following attributes:
  - Daemon port number as Integer
  - Daemon installation directory as String
- Machine Type:
  - “Machine Type” database entity using Entity Framework that has the following attributes:
    - Type as String
- Machine Platform:
  - “Machine Platform” database entity using Entity Framework that has the following attributes:
    - Platform as String
- Resource Pool:
  - The ability to execute the following functions:
    - Create new Resource Pool
    - Delete a Resource Pool
    - Show a detailed information about a Resource Pool
    - Edit a Resource Pool
    - List of all Resource Pools
  - “Resource Pool” database entity using Entity Framework that has the following attributes:
    - Resource Pool Display Name as String
    - Associated Machines as a list of Machine entities
- Worker:
  - The ability to execute the following functions:
    - Create new Worker
    - Delete a Worker
    - Show a detailed information about a Worker



- Edit a Worker
  - List of all Workers
- “Worker” database entity using Entity Framework that has the following attributes:
  - Worker display name as String
  - Worker port number as Integer
  - Cost as Integer
  - Aneka Container ID as String
  - Is this worker container installed Boolean as Boolean
  - Status as enumerable (stored in database as integer), which has the following values:
    - BadAddress = 1
    - NetworkUnreachable = 2
    - ServiceActive = 4
    - ServiceUnknown = 8
    - ServiceUnactive = 16
    - Error = 32
  - Is this worker container in progress Boolean as Boolean
  - Progress Message as String
  - Is this worker quarantined Boolean as Boolean
  - Associated Services as a list of *Service* entities
- Master and Cloud:
  - The ability to execute the following functions:
    - Create new Cloud (and Master as a result)
    - Delete a Cloud (and Master as a result)
    - Show a detailed information about a Cloud (and Master as a result)
    - Edit a Cloud (and Master as a result)
    - List all Clouds
  - “Master” database entity using Entity Framework that has the following attributes:
    - Master display name as String
    - Master port number as Integer
    - Cost as Integer
    - Master Failover Backup URI as String

- Aneka Container ID as String
- Is this worker container installed Boolean as Boolean
- Status as enumerable (stored in database as integer), which has the following values:
  - BadAddress = 1
  - NetworkUnreachable = 2
  - ServiceActive = 4
  - ServiceUnknown = 8
  - ServiceUnactive = 16
  - Error = 32
- Is this master container in progress Boolean as Boolean
- Progress Message as String
- Associated Services as a list of *Service* entities
- “Cloud” database entity using Entity Framework that has the following attributes:
  - Cloud Name as String
  - DB Connection String as String
  - Security Shared Key as String
  - Master as *Master* entity
  - Associated Workers as a list of *Worker* entities
  - Associated User Accounts as a list of *CloudUserAccount* entities
- Applications Report:
  - The ability to execute the following functions:
    - Filter the list of application by date and time
    - List all applications
    - List only active application
    - List only applications with errors
    - Force close an active application
- Live CPU Utilization Chart:
  - The ability see a summary of a live CPU utilization chart of a specific cloud, the displayed CPU will be for all the nodes in the cloud, includes the master and all the workers
- Historical CPU Utilization Chart:
  - The ability to filter the data by data and time

- The ability to see a summary of a historical CPU utilization chart of a specific cloud. The displayed CPU will be for all the nodes in the cloud, includes the master and all the workers.
- The ability to see a detailed historical CPU utilization chart of a specific cloud. Detailed means each node will be displayed in a separate chart line.
- Jobs Status Summary:
  - The ability to see a summary of a job status pie chart of a specific cloud. The displayed pie chart will be for all the nodes in the cloud including the master and all the workers, and for all jobs.

### Non-Functional Requirements

- **Accessibility:** Aneka Web Portal is a web-based application, and that makes it accessible from any device (computers, tablets, or mobile devices). However, the Aneka Web Portal can be configured to be accessible over the Internet, intranet or just locally.
- **Availability:** Because Aneka Web Portal users ASP.NET 4 and SQL Server 2008 R2 database, the Portal can be installed into more than one web servers and configuring the database for a live backup/duplication, then the organization's load balance or DNS server can point to the backup web server if the main one is down. All the users' states are stored in the database, so as long as the database kept duplicated the users will not notice any hiccup in the system even if the main web server is down.
- **Scalability:** The limitation of the Aneka Web Portal is the limitation of the web server machine that hosts the Portal. We recommend installing the Portal in a machine with the recommended specifications (see the installation chapter).
- **Security:** Aneka Web Portal has five level of security:
  - **Browser Level:** all the forms in the views are validated using JQuery JavaScript fields validation.
  - **Action Level:** all the Get and Post requests are authenticated, not just the current page/view but also every widget on that page/view and every AJAX request require a valid ASP.NET authentication cookie.
  - **Controller Level:** every Post requests are checked manually inside the controllers to make sure that the submitted data is valid.

- **Model Level:** Every database entity has its own validation check to make sure that the query matches the database data types, foreign-key, and entity-relationship before sending the query to the database.
- **Aneka.Security Level:** Aneka framework has its own implementation of security and verification.

All the critical information is stored in the Master, not in the Aneka Web Portal. So, if the Aneka Web Portal got comprised, all what the system administrator needs to do is to change the Security Shared Key manually from the master and all the workers.

- **Usability:** The system infrastructure has been designed and implemented for any non-expert users to use the Portal very easily. The graphical user interface and the concept of widgets and dashboard make it even easier to use. The error messages are shown on the top right corner of the Portal, which is called *Issues Centre*, gives the Portal users a summary and a quick overview of what are the current errors and warnings, and as soon as the user clicked on any error or warning it pops up a dialog so the user can solve the problem very easily. In addition to the *Issues Centre*, Aneka Web Portal has the *Activity Centre* that gives the user a summary and a quick overview on the current running tasks. Read more about *Issues Centre* and *Activity Centre* in the Project Design chapter.

# Chapter Two: Project Design

The project has been designed to achieve a very high standard design and implementation quality. The concept of widgets gives the application the flexibility to adjust the graphical user interface and to modify the business logic very easily.

We will go through the Aneka Web Portal design showing the steps of data flow between all the components.

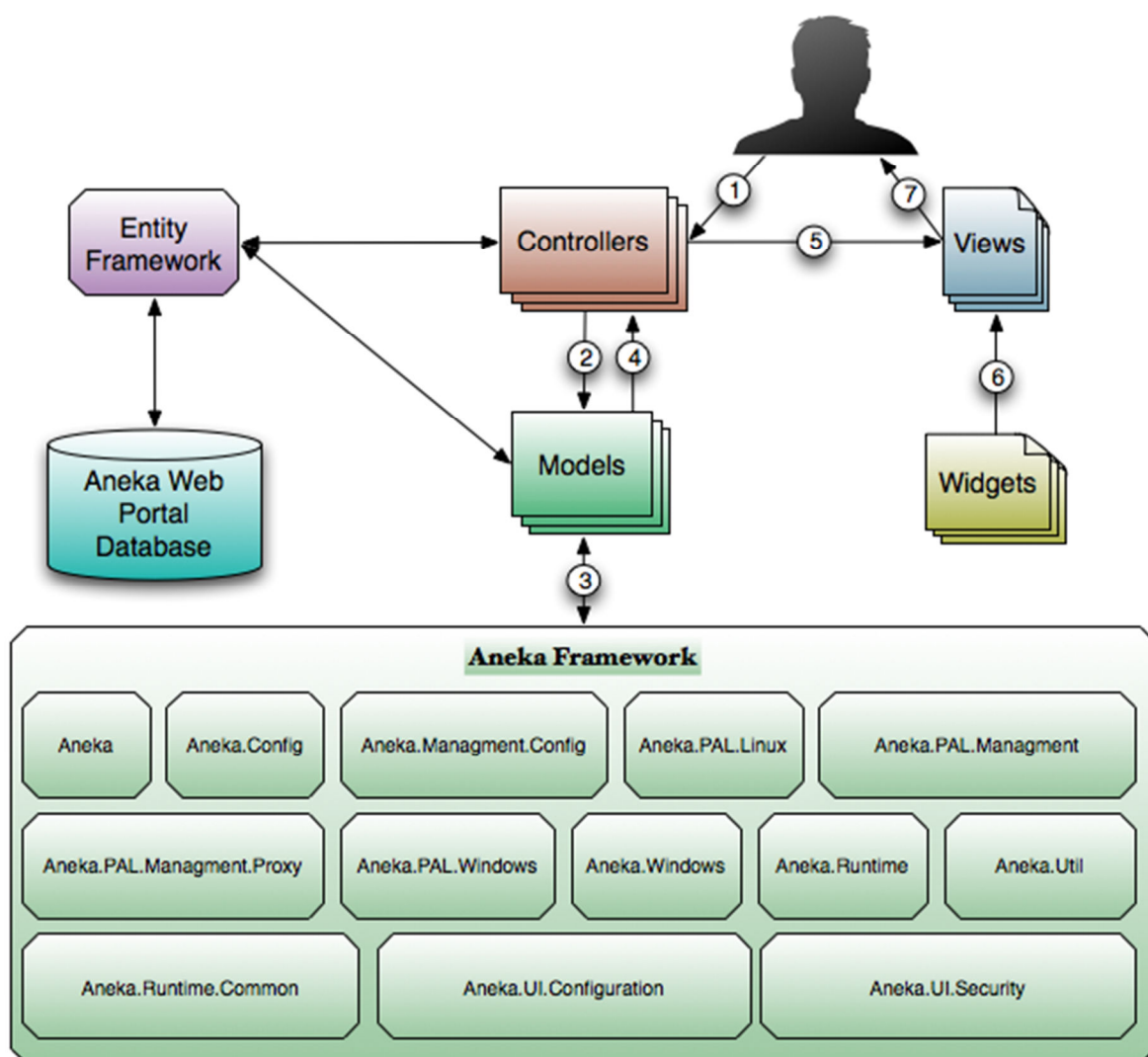


Figure 3: Aneka Web Portal System Design

From the above diagram, Aneka Web Portal has six components:

1. **Controller:** This component is what the Portal users call when they navigate through the Portal. Inside each controller there are several actions and the user will ask for a specific action to be executed and then returns a view (an HTML page) with the results. For Example: if the user entered this URL: "<http://Manjrasoft.com/AnekaWebPortal/CloudManagement/BrowseClouds>" that means they have called *BrowseClouds* action inside *CloudManagement* controller. Some controllers require enquiring the database to get, update or delete data via the Entity Framework.
2. **Entity Framework and Database:** What makes Aneka Web Portal very stable is that every step is registered in the database, so if there is any error in the data flow it will be easily recovered with very minimal data lost. For example instead of passing the data between the Controllers and Models the Controllers will store the data in the database and ask from the Models to retrieve it, and when the Models finishes executing jobs in the Aneka Framework it will do the same and asks from the Controllers to retrieve the data from the database. This will ensure that we don't have a very long series of actions executing them at once, and that will reduce the single point of failure issue. Because the series of action are decoupled the users will be aware of all those steps and what is the current status for any action from the Activity Centre at the top right corner of the Portal. The Portal is a multithreaded web application so the user will not wait for any action to finish, they can continue doing anything else and monitoring the current status for all any actions from the Activity Centre. All the queries and database communication is done via the Entity Framework.
3. **Models:** this component is the logical and the most complicated part of the project, it takes actions from the Controllers and do some computations then execute that action in the Aneka Framework. Most of the time the Controllers will not pass data, the Model has to ask for the data from the database via the Entity Framework. Usually the Models are executed inside a thread so the user does not have to wait. After getting the results from the Aneka Framework it will be stored in the database or passed to the Controllers if the controller is Ajax call. When

the user refreshes the page or the Ajax call for an update they will fetch the new updates from the database, not form the Models.

4. **Aneka Framework:** Aneka is a PaaS, so the Portal uses the following libraries as a reference:

- Aneka.dll
- Aneka.Config.dll
- Aneka.Management.Config.dll
- Aneka.PAL.Management.dll
- Aneka.PAL.Management.Proxy.dll
- Aneka.PAL.Windows.dll
- Aneka.PAL.Linux.dll
- Aneka.Runtime.dll
- Aneka.Runtime.Common.dll
- Aneka.UI.Configuration.dll
- Aneka.UI.Security.exe
- Aneka.Util.dll

5. **Views:** views are the HTML pages mixed with C# razor view-engine. Razor view-engine is very powerful and easy to use, it is code-focused so with Razor we do not need any graphical designer, and the Portal developers have to write all the UI specifications in the CSS, and focus on the logic, which already have been done.

6. **Widgets:** Widgets have been designed to be reused anywhere with a single line of call. They are very highly customized and can be called inside a page or as a popup dialog (more information in the Project Development section). Each widget is security protected as extra level of protection. Widgets UI CSS uses the Constellation complete admin skin [8] for the graphical representation. More details on how to use the widgets, and how they are implemented in the Project Development chapter.

## Aneka Web Portal Architecture

The following diagram illustrates the architecture of Aneka Web Portal and the different layers of the system.

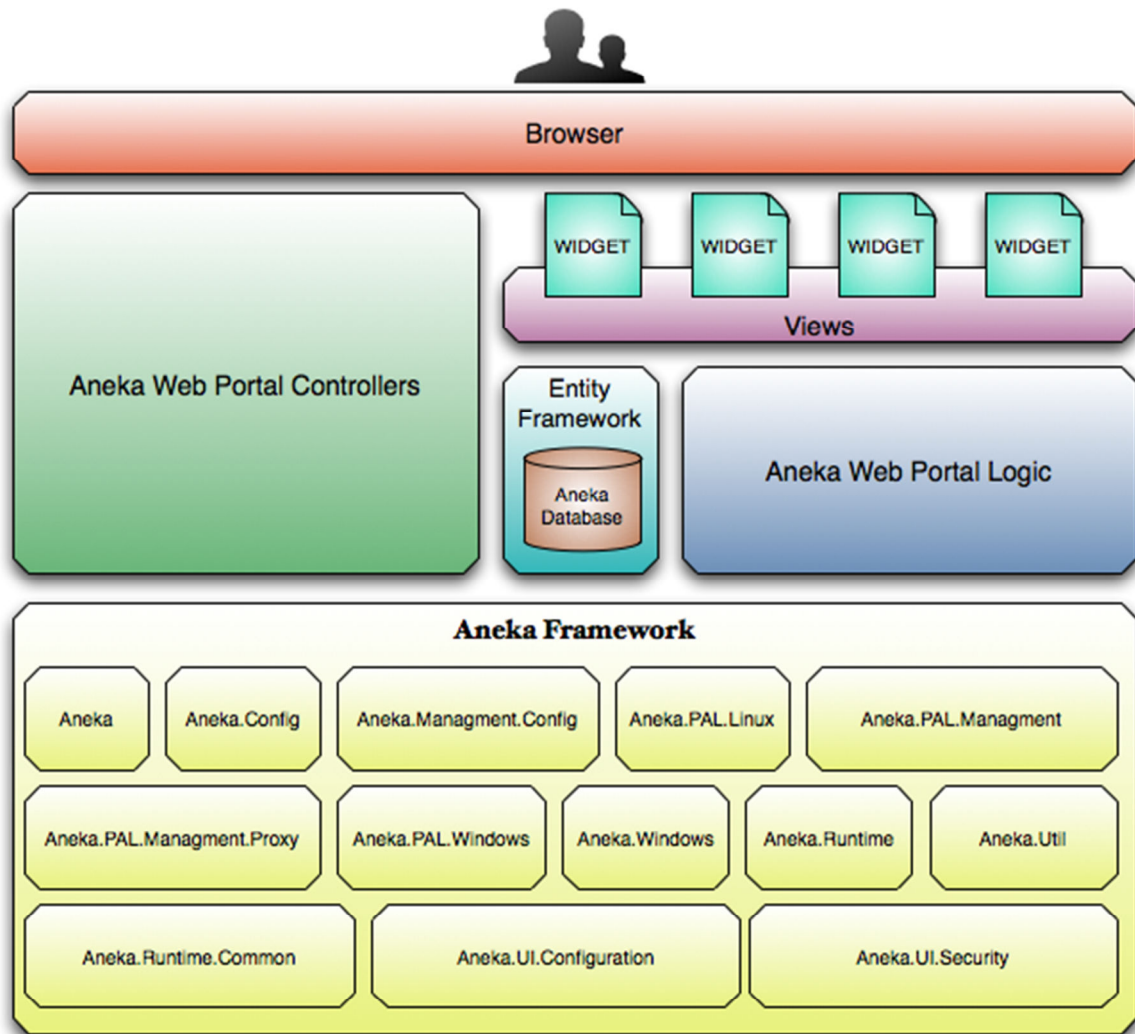


Figure 4: Aneka Web Portal Architecture

All the components of the Aneka Web Portal’s architecture been described in the previous section.

### Database Design

Entity Framework has been used to build the database using the code-first approach, so the database will be created according to the c# classes that we implement using the Entity Framework. The following is the database diagram of the Aneka Web Portal entitles:



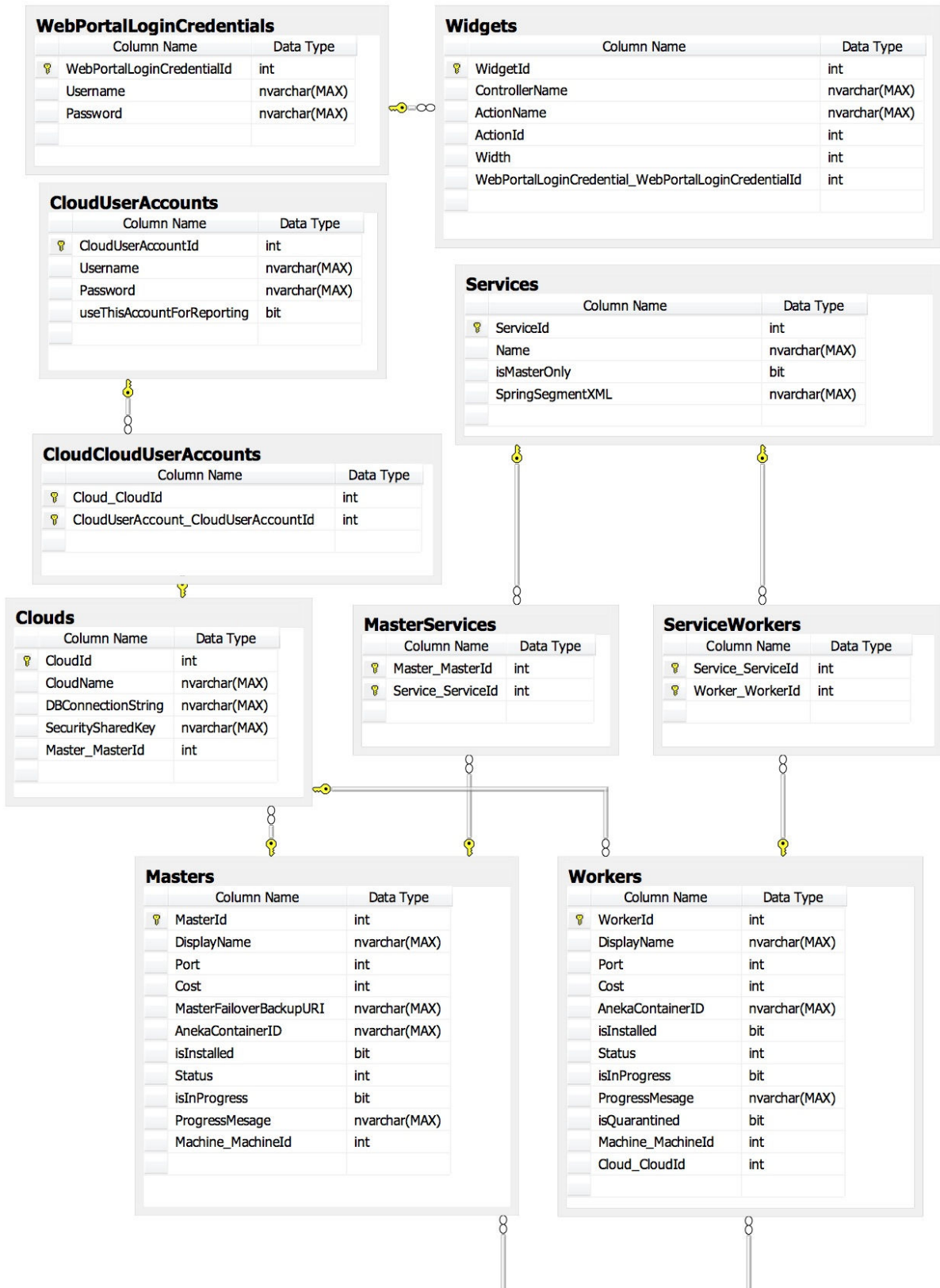


Figure 5: Database Design Diagram 1 of 2

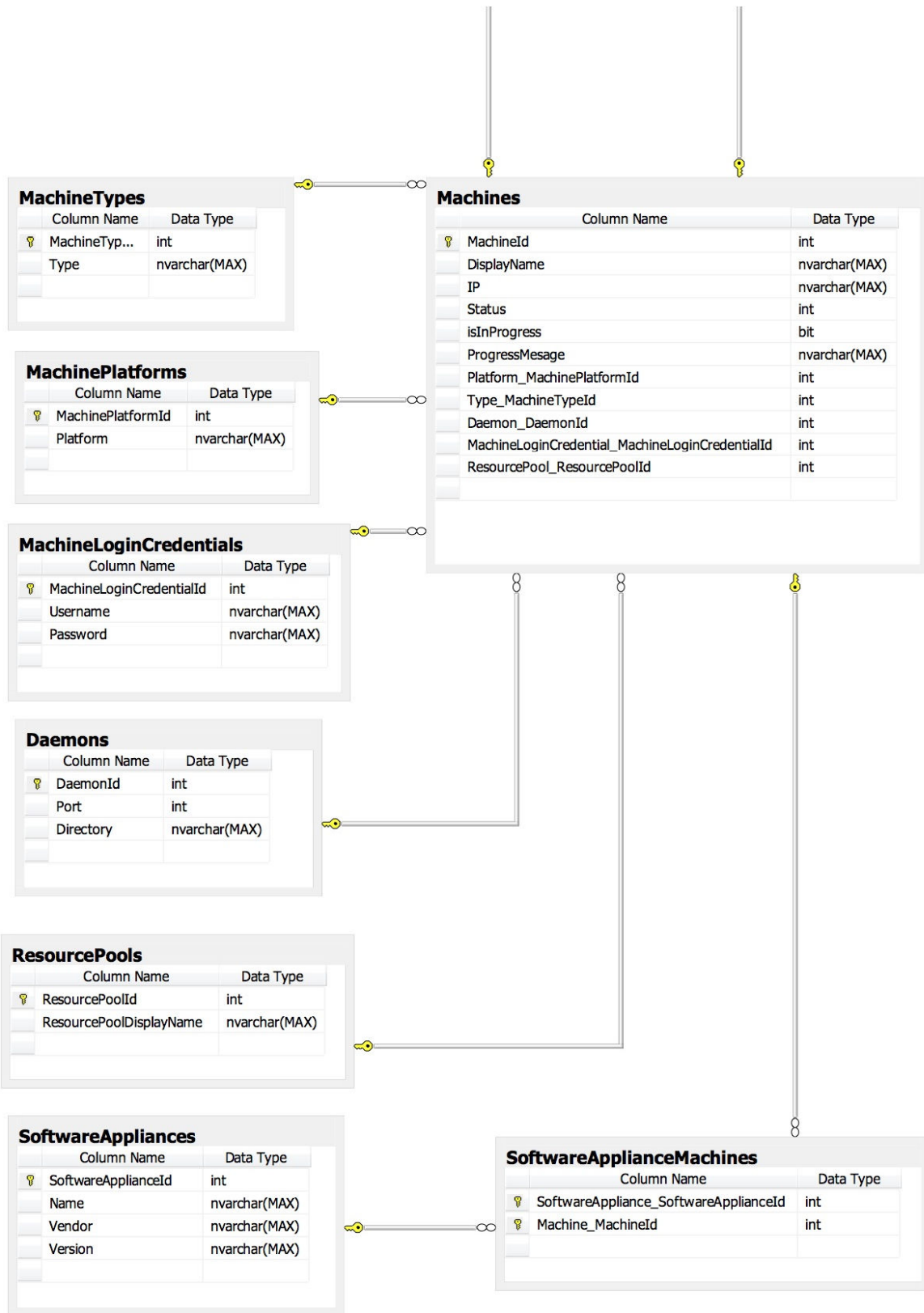


Figure 6: Database Design Diagram 2 of 2

Please note that the middle tables that have been created instead of the many-to-many entity-relationship is a result of using the Entity Framework, so we do not have control over the table naming on those tables.

## The Concept of Widgets in the Dashboard

All widgets in Aneka Web Portal are ASP.NET MVC 3 partial views, which do not use the `_layout` view (aka Master Page).

Each widget designed to be wrapped-up with `<article>` and `<section>` HTML tags; those wrapped-up tags specify the widget's width and create a beautiful box of information or form. The width is between 1 and 12, so 12 is the full width.

For Example:

```
<article class="container_12">
  <section class="grid_12">
    <div class="block-border">
      @Html.Action("List", "_Clouds")
    </div>
  </section>
</article>
```

The example above called the `"_Clouds"` controller and `"List"` Action in a full width widget, which is 12.

Each widget must follow a specific HTML format to match the CSS, for example:

```
@{
    Layout = null;
}
<div class="block-content">
  <h1>Delete Confirmation</h1>
  <div class="infos">
    <!-- WIDGET HTML RAZOR CONTENT -->
  </div>
</div>
```

Referee to the Constellation complete admin skin [9] for more information, and to know how to use the pre-designed CSS classes, and to learn more about the different type of UIs.

## The Concept of the Issue and Activity Centres

The issues and activity centres have been designed to check the following entities in the database:

- Machines
- Masters
- Workers

Each one of those entities has the following attribute:

- Status
- Is It In Progress?
- Progress Message

### Issues Centre

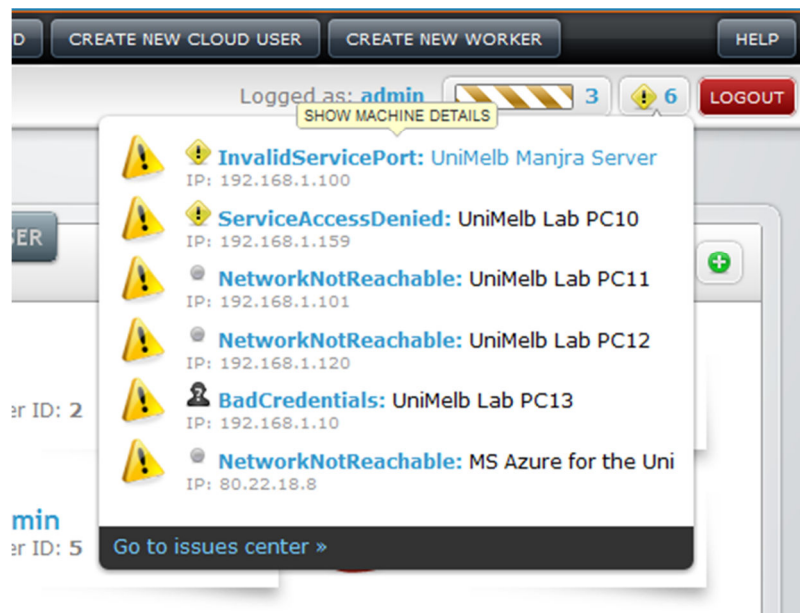


Figure 7: Issues Centre

The issues centre check if the statuses is not active or unsuccessful and display an error or warning message according to the status type and message, and each status is clickable to popup a dialog box to solve the clicked issue.

## Activity Centre

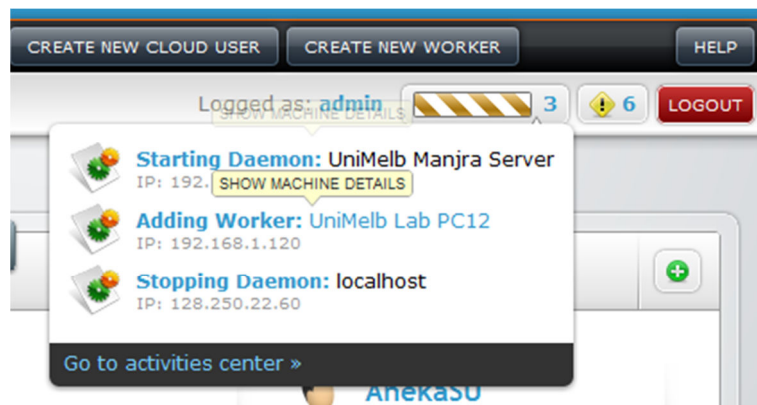


Figure 8: Activity Centre

The activity centre checks if the entity is in progress, and if it does it will show a progress bar with a progress message for each active job. Each one of those jobs is clickable to open a popup dialog box to show the user more information about the clicked job.

## The Concept of the In-page Popup Dialog Box

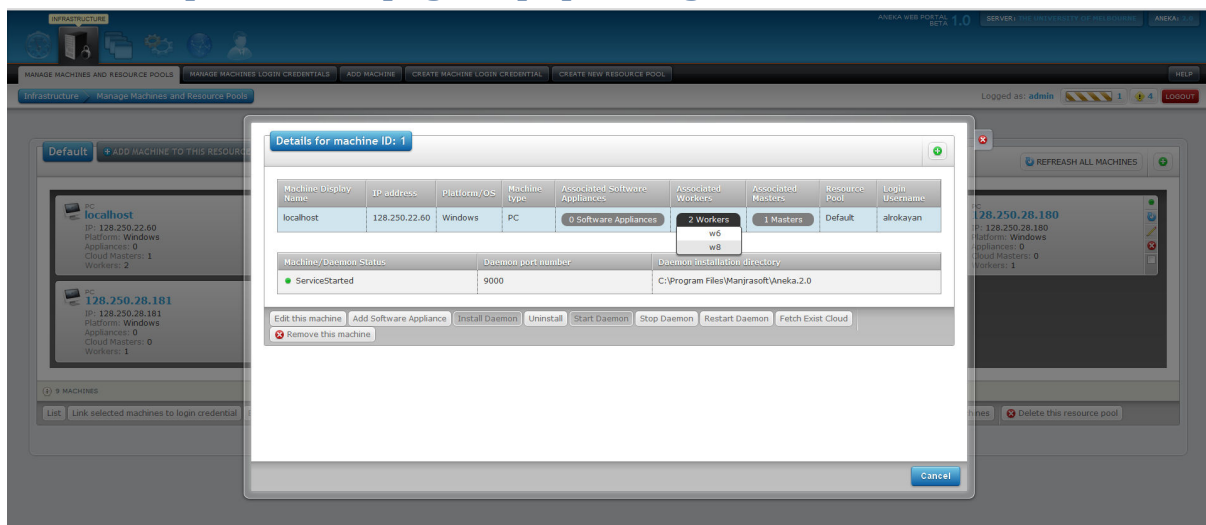


Figure 9: The Concept of the In-page Popup Dialog Box

All widgets been designed to be included in any view (page) or as a popup dialog box. The benefit of the popup dialog box is that the user will not lose the current page navigation, and they can read what's in the current page along with the new dialog.

To call this dialog you have to call this Java Script function:

```
openModalWindow(null, 'Create', '_Clouds', 'Create New Cloud', 6)
```

This Java Script function will popup a dialog box having the *\_Clouds* controller and *Create* action widget with the title: “Create New Cloud” and 6 points width (the full browser width is 12, so this dialog box will fill half of the browser width, centred). This function has been implemented in the *\_layout* view.

# Chapter Three: Project Development

## [10]

---

### Widgets

All widgets are partial views, so the `_Layout` view (aka Master page) does not apply to them.

### Widgets Summary

The following table shows a summary of the most important widgets in Aneka Web Portal:

Controller Name	Action Name	ID
<b>_ApplicationsReport</b>	ApplicationsList	Cloud ID
<b>_Charts</b>	CPU_Utilization_Range	Cloud ID
	CPU_Utilization_Live	Cloud ID
	JobsStatusSummary	Cloud ID
<b>_Clouds</b>	Create	N/A
	Delete	Cloud ID
	Details	Cloud ID
	Edit	Cloud ID
	List	N/A
<b>_CloudUsers</b>	Create	N/A
	Delete	CloudUserAccount ID
	Details	CloudUserAccounts ID
	Edit	CloudUserAccount ID
	List	N/A
<b>_DashboardWidgets</b>	Create	WebPortalLoginCredential ID
	CreateForLoggedInUser	MANY PARAMETERS
	Delete	Widget ID
	Details	Widget ID
	Edit	Widget ID
<b>_Machines</b>	Create	ResourcePool ID
	Delete	Machine ID

	Details	Machine ID
	Edit	Machine ID
<b>_MachinesLoginCredentials</b>	Create	N/A
	Delete	MachineLoginCredential ID
	Details	MachineLoginCredential ID
	Edit	MachineLoginCredential ID
	List	N/A
<b>_PortalUsers</b>	Create	N/A
	Delete	WebPortalLoginCredential ID
	Details	WebPortalLoginCredential ID
	Edit	WebPortalLoginCredential ID
	List	N/A
<b>_ResourcePools</b>	Create	N/A
	Delete	ResourcePool ID
	Details	ResourcePool ID
	Edit	ResourcePool ID
	List	N/A
<b>_Services</b>	Create	N/A
	Delete	Service ID
	Details	Service ID
	Edit	Service ID
	List	N/A
<b>_SoftwareAppliances</b>	Create	N/A
	Delete	SoftwareAppliance ID
	Details	SoftwareAppliance ID
	Edit	SoftwareAppliance ID
	List	N/A
<b>_Workers</b>	Create	Cloud ID
	Delete	Worker ID
	Details	Worker ID
	Edit	Worker ID
	List	Cloud ID

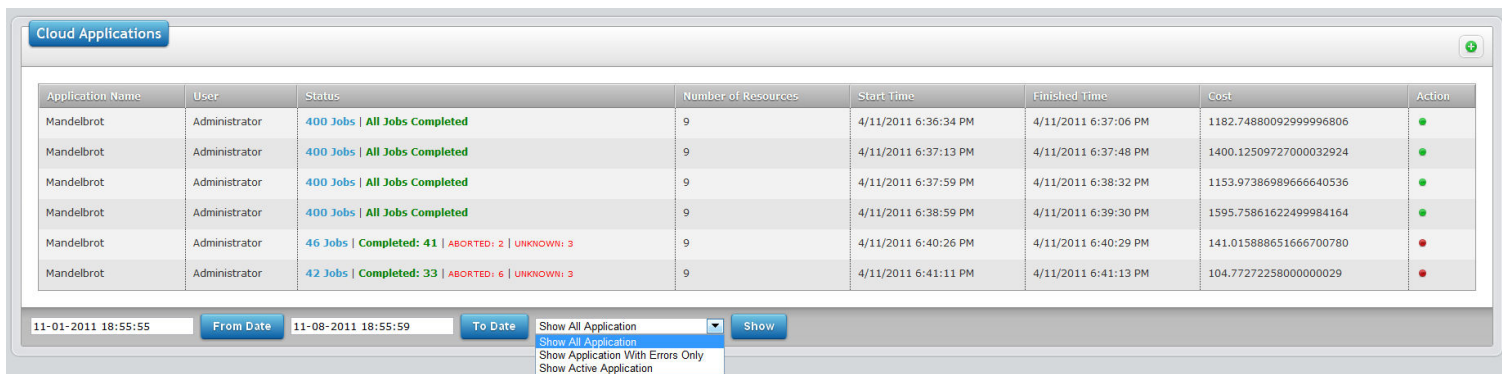


## Widgets Details

The following is the details for most of the implemented widgets in Aneka Web Portal:

### Applications List Widget

Controller Name	Action Name	ID
_ApplicationsReport	ApplicationsList	Cloud ID



The screenshot shows a web interface titled "Cloud Applications". It contains a table with the following columns: Application Name, User, Status, Number of Resources, Start Time, Finished Time, Cost, and Action. Below the table are filters for "From Date" and "To Date", and a dropdown menu for filtering options.

Application Name	User	Status	Number of Resources	Start Time	Finished Time	Cost	Action
Mandelbrot	Administrator	400 Jobs   All Jobs Completed	9	4/11/2011 6:36:34 PM	4/11/2011 6:37:06 PM	1182.74880092999996806	●
Mandelbrot	Administrator	400 Jobs   All Jobs Completed	9	4/11/2011 6:37:13 PM	4/11/2011 6:37:48 PM	1400.12509727000032924	●
Mandelbrot	Administrator	400 Jobs   All Jobs Completed	9	4/11/2011 6:37:59 PM	4/11/2011 6:38:32 PM	1153.97386989666640536	●
Mandelbrot	Administrator	400 Jobs   All Jobs Completed	9	4/11/2011 6:38:59 PM	4/11/2011 6:39:30 PM	1595.75861622499984164	●
Mandelbrot	Administrator	46 Jobs   Completed: 41   ABORTED: 2   UNKNOWN: 3	9	4/11/2011 6:40:26 PM	4/11/2011 6:40:29 PM	141.015888651666700780	●
Mandelbrot	Administrator	42 Jobs   Completed: 33   ABORTED: 6   UNKNOWN: 3	9	4/11/2011 6:41:11 PM	4/11/2011 6:41:13 PM	104.77272258000000029	●

Filters: From Date: 11-01-2011 18:55:55, To Date: 11-08-2011 18:55:59. Filter options: Show All Application, Show Application With Errors Only, Show Active Application.

Figure 10: \_ApplicationsReport Controller - ApplicationsList Action

### Description

Get a list of application from Aneke, this list can be filtered by a range of time and date. On the top of the time and date filter this widget can be filtered by choosing on of the following options:

- *Only active application*, so the user can force close them or application with errors.
- *Applications with Errors only*, so the user can have a summary of unsuccessful applications.
- *All application*.

### In code

1. Get a cloud and a master machine entities from the database.
2. Pass the master machine IP and the master port number to the view.

## Historical CPU Utilization Widget

Controller Name	Action Name	ID
_Charts	CPU_Utilization_Range	Cloud ID

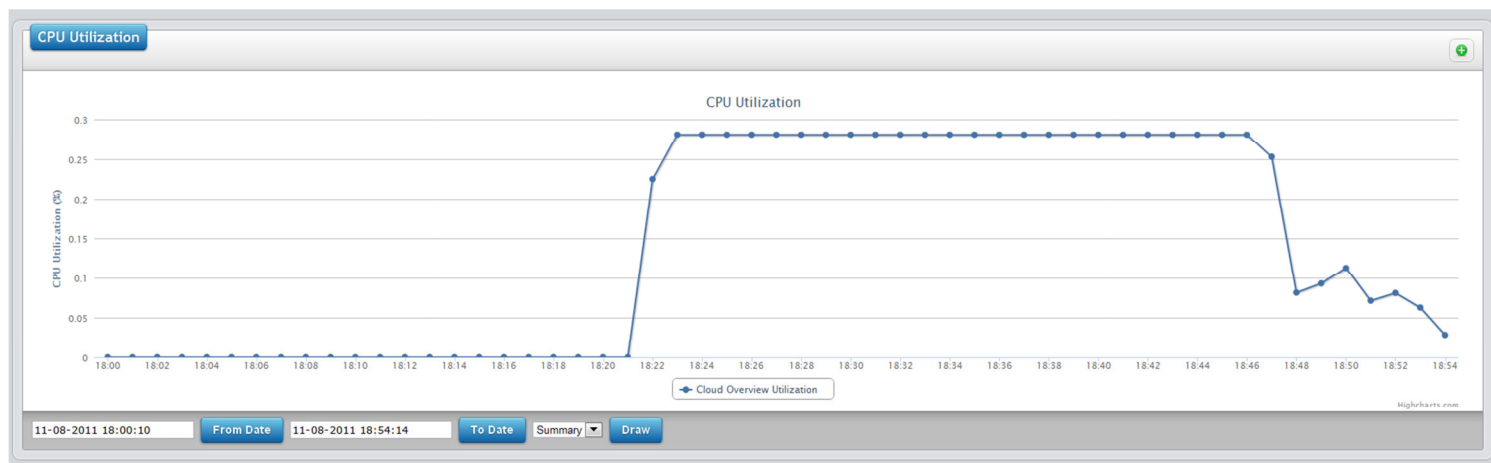


Figure 11: \_Charts Controller - CPU\_Utilization\_Range Action - Summary option

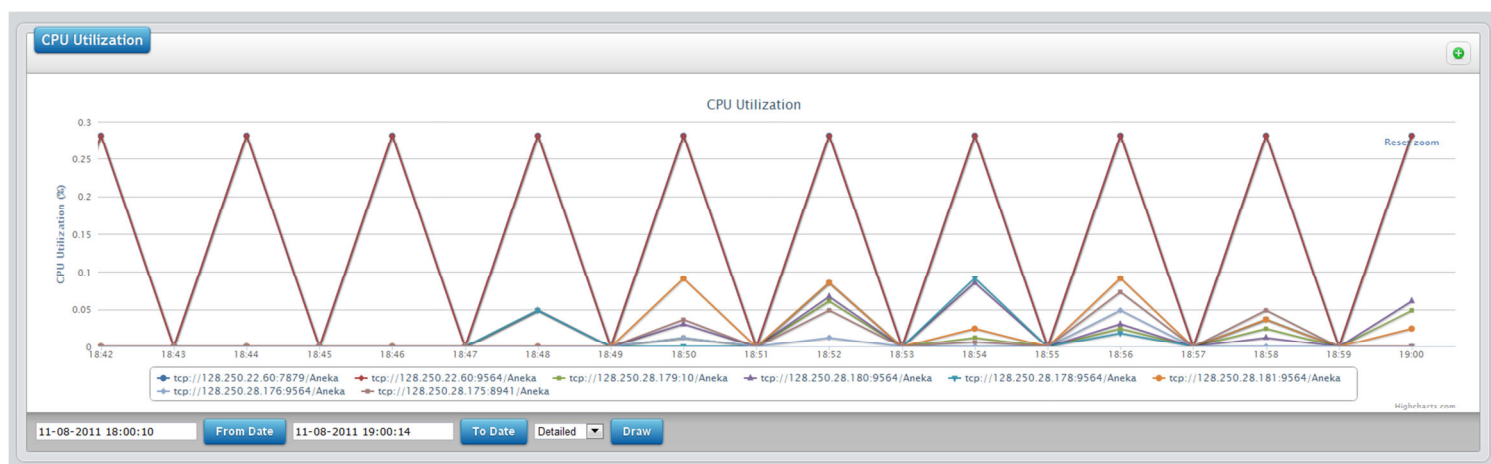


Figure 12: \_Charts Controller - CPU\_Utilization\_Range Action - Detailed option

### Description

Display a historical CPU utilization by selecting a range of time and date. Also, you can display a summary of the CPU utilization or display each node in a separate line.

### In code

1. Get a cloud and a master machine entities from the database.
2. Pass the master machine IP and the master port number to the view.

## Live CPU Utilization Widget

Controller Name	Action Name	ID
_Charts	CPU_Utilization_Live	Cloud ID

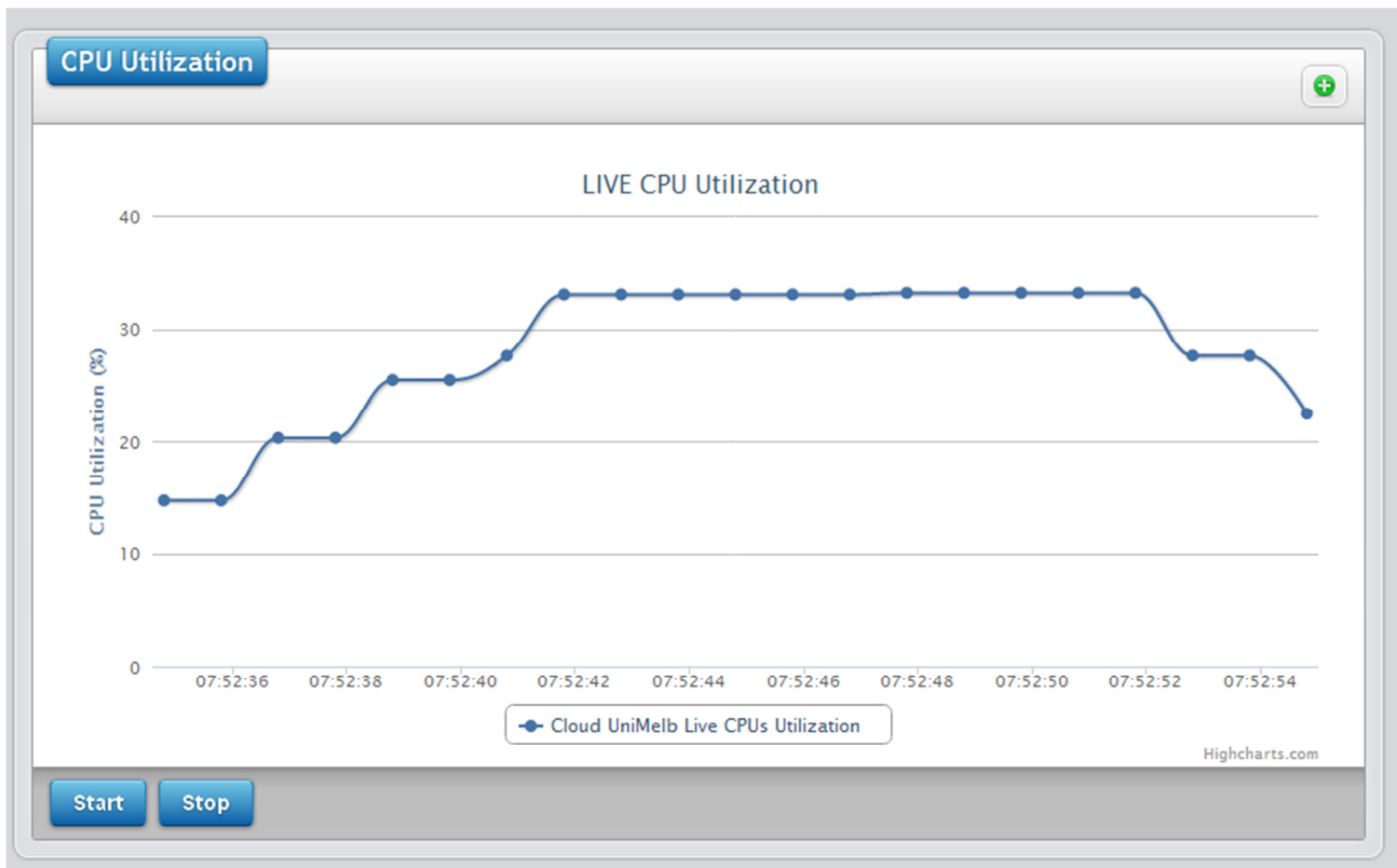


Figure 13: \_Charts Controller - CPU\_Utilization\_Live Action

### Description

Display a live chart of the current cloud (master and all workers) CPU utilization.

### In code

1. Get a cloud and a master machine entities from the database.
2. Pass the master machine IP and the master port number to the view.

## Summary of Jobs Status Widget

Controller Name	Action Name	ID
_Charts	JobsStatusSummary	Cloud ID

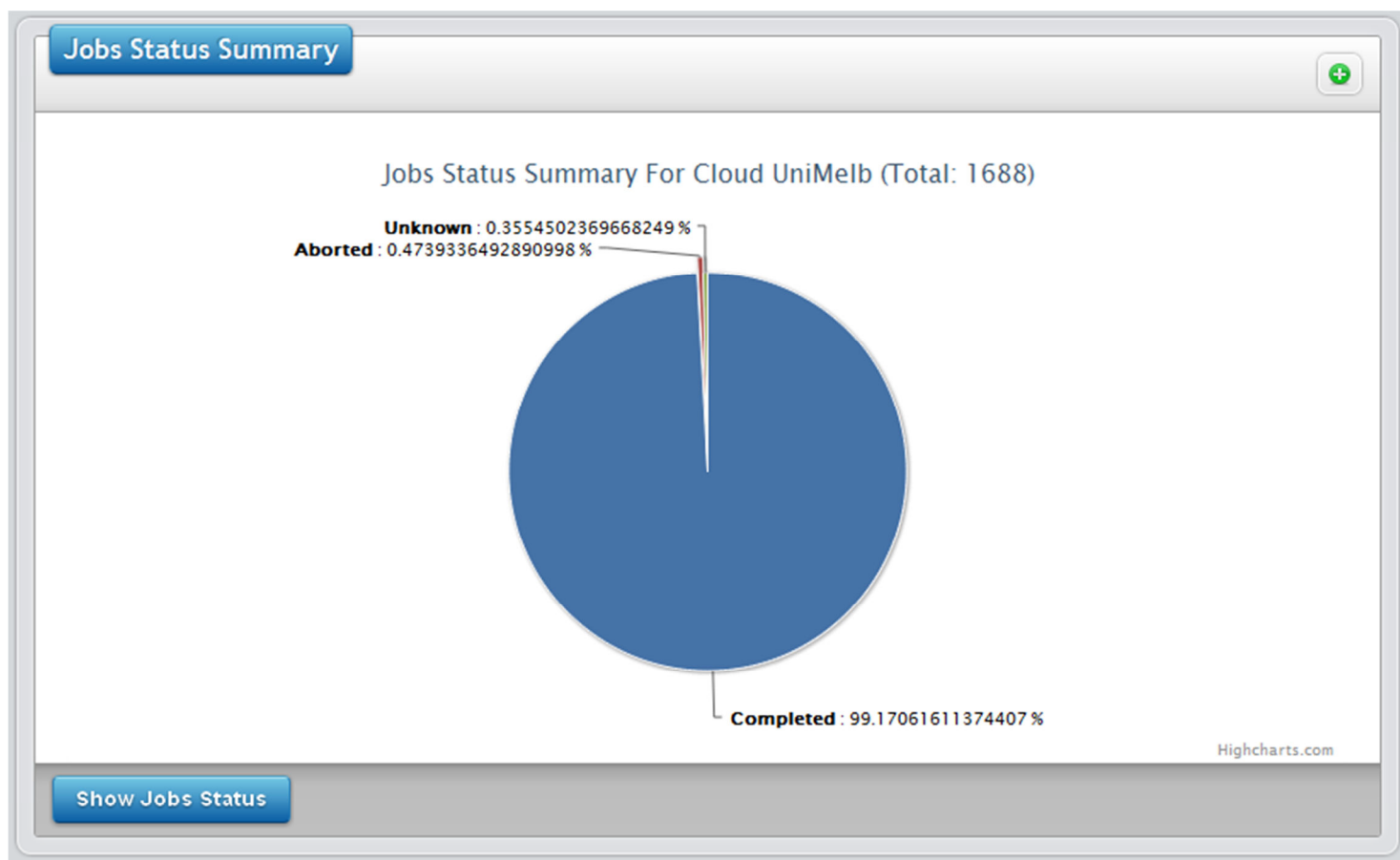


Figure 14: \_Charts Controller - JobsStatusSummary Action

### Description

This will be used to display a pie chart showing how many failed, succeeded, aborted jobs. There are many jobs status, not just three.

### In code

1. Get a cloud and a master machine entities from the database.
2. Pass the master machine IP and the master port number to the view.

## Create Cloud Widget

Controller Name	Action Name	ID
_Clouds	Create	N/A

**Create New Cloud**

**Cloud Info**

Cloud Name

Database Connection String (Keep it blank to store in memory)

EXAMPLE1:  
UID=ADMIN;PWD=PASSWORD;SERVER=128.250.35.190\SQLEXPRESS;DATABASE=DBNAME

EXAMPLE2:  
DATA SOURCE=128.250.35.190;INITIAL CATALOG=DBNAME;PERSIST SECURITY  
INFO=TRUE;USER\_ID=ADMIN;PASSWORD=PASSWORD

**Master Container Info**

Master display name

Master port number

Cost

Select Master Machine

- localhost
- 128.250.28.175
- 128.250.28.179
- 128.250.28.180
- 128.250.28.181
- 128.250.28.178
- 128.250.28.177
- 128.250.28.176

Select Services (must select at least one)

- Logging Service
- Monitoring Service
- Discovery Service
- Task Execution Service

Cancel

Figure 15: \_Clouds Controller - Create Action

### Description

A form to create a new cloud. Note: the user will input both the cloud and the master information in one form.

### In code

1. Get a list of active machines, and pass them to the view.
2. Get a list of all services, and pass them to the view.

## Delete Cloud Widget

Controller Name	Action Name	ID
_Clouds	Delete	Cloud ID

### Description

A confirmation message to delete a cloud

### In code

Get and pass the cloud

## Cloud's Details Widget

Controller Name	Action Name	ID
_Clouds	Details	Cloud ID

The screenshot shows a web interface for managing cloud workers and details. The top section, titled 'Cloud UniMeB Cloud Workers', features a '+ ADD NEW WORKER' button and 'REFRESH ALL WORKERS' and 'REFRESH MASTER' buttons. Below this, there are ten worker cards labeled M1 through w10, each displaying a port number and a machine name. The bottom section, titled 'Cloud UniMeB Cloud Details', contains a table with various fields and their values, along with several action buttons at the bottom.

DB Connection String	SecuritySharedKey	Associated Workers	Associated User Accounts	Master display name	Associated Services	Master port number	Cost	Master Failover Backup URI	Master Status
Show Connection String	Show Security Shared Key	10 Workers	0 Users	M1	12 Services	7879	10		Restarting Master Container

Buttons at the bottom: Edit this cloud and its master, Create New Worker For This Cloud, Create New Service User For This Cloud, Create New Cloud User For This Cloud, Start Container, Stop Container, Restart Container, Install Container, Uninstall Container, Remove this cloud.

Figure 16: \_Clouds Controller - Details Action

### Description

This widget is the largest, and the most complicated one. It will show all the cloud, master, and workers information.

### In code

1. Get the cloud entity from the database.
2. Pass the master machine entity to the view.
3. Create and pass a Dictionary to the view, the key is the worker ID, and the value is the machine entity.
4. Pass the cloud entity to the view.

## Edit a Cloud Widget

Controller Name	Action Name	ID
_Clouds	Edit	Cloud ID

### Description

A form to edit a selected cloud.

### In code

1. Get the master machine entity, and pass it to the view as dropdown list.
2. Get all associated services to this cloud, and pass it to the view as a multi select list.

## List of Clouds Widget

Controller Name	Action Name	ID
_Clouds	List	N/A

The screenshot displays the 'Aneka Clouds' management interface. At the top, there is a navigation bar with 'CLOUD MANAGEMENT' and 'ANEKA WEB PORTAL BETA 1.0'. Below this, a menu includes 'BROWSE CLOUDS', 'BROWSE CLOUD USERS', 'CREATE A CLOUD', 'CREATE NEW CLOUD USER', and 'CREATE NEW WORKER'. The user is logged in as 'admin' and has 3 warnings and 6 alerts. The main content area shows two cloud entries:

- UniMelb**: Master: UniMelb Manjra Master. Metrics: # of Workers: 3, # of Users: 1, CPU Load: 66%, Workers Load: 90%, Disk Space: 250GB/1024GB.
- Hyper Cloud EC2 and Local - Test Phase**: Master: Lab12 PC1 Master. Metrics: # of Workers: 2, # of Users: 1, CPU Load: 66%, Workers Load: 90%, Disk Space: 250GB/1024GB.

A summary bar at the bottom indicates 'NUMBER OF CLOUDS: 2'.

Figure 17: \_Clouds Controller - List Action

### *Description*

A list of all created clouds. With a summary for each one, this summary includes a small list of all associated nodes, and some hardware statistics like: CPU Load, Workers Load, and available Disc Space.

### *In code*

Get all clouds from the database, and pass them to the view.

### Create Cloud User Widget

Controller Name	Action Name	ID
_CloudUsers	Create	N/A

### *Description*

A form to create a new CloudUserAccount.

### *In code*

Pass a list of clouds as a multi select list to the view.

### Delete a Cloud User Widget

Controller Name	Action Name	ID
_CloudUsers	Delete	CloudUserAccount ID

### *Description*

A confirmation message to delete a CloudUserAccount.

### *In code*

1. Get the CloudUserAccount entity from the database using the given ID
2. Pass this entity to the view

### Cloud User's Details Widget

Controller Name	Action Name	ID
_CloudUsers	Details	CloudUserAccount ID

### *Description*

Detailed information about a specific CloudUserAccount.



### *In code*

1. Get the CloudUserAccount entity from the database using the given ID.
2. Pass this entity to the view.

### **Edit a Cloud User Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_CloudUsers	Edit	CloudUserAccount ID

### *Description*

Edit a specific CloudUserAccount by getting its ID.

### *In code*

1. Get the CloudUserAccount entity from the database.
2. Send a list of selected clouds as a multi select list to the view.
3. Pass the whole entity to the view so the user can edit it.

### **List of Cloud Users Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_CloudUsers	List	N/A

### *Description*

A simple List of all Cloud User Accounts.

### *In code*

Get all the CloudUserAccounts and pass them to the view.

### **Create a Dashboard Widget Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_DashboardWidgets	Create	WebPortalLoginCredential ID

### *Description*

A form to create a new Widget. Note: WebPortalLoginCredential ID is an optional parameter to make a specific web portal user the default one in the user's dropdown list

### *In code*

Pass a dropdown list of portal users to assign this new widget to. The default user will be according the past WebPortalLoginCredential ID parameter.

### Create a Widget to add it To the Dashboard Widget

Controller Name	Action Name	ID
_DashboardWidgets	CreateForLoggedInUser	MANY PARAMETERS

### *Description*

This is a form to create a widget to the logged in user, and filling the form with Action Name and Controller Name of the selected widget. This widget will be called when the user clicked on the green plus button on the top right corner of each widget.

### *In code*

1. Get the WebPortalLoginCredential entity by looking up for the user using its username.
2. Pass a dropdown list of users to the view.
3. Pass the Action Name to the view.
4. Pass the Controller Name to the View.
5. Pass the width to the View.
6. Pass the Action ID to the View.

### Delete a Dashboard Widget Widget

Controller Name	Action Name	ID
_DashboardWidgets	Delete	Widget ID

### *Description*

A confirmation message to delete a Widget.

### *In code*

1. Get the Widget entity from the database using the given ID.
2. Pass this entity to the view.

## Dashboard Widget's Details Widget

Controller Name	Action Name	ID
_DashboardWidgets	Details	Widget ID

### *Description*

Detailed information about a specific Widget.

### *In code*

1. Get the Widget entity from the database using the given ID.
2. Pass this entity to the view.

## Edit a Dashboard Widget Widget

Controller Name	Action Name	ID
_DashboardWidgets	Edit	Widget ID

### *Description*

Edit a specific Widget by getting its ID.

### *In code*

1. Get the Widget entity from the database.
2. Pass the whole entity to the view so the user can edit it.

## Create Machine Widget

Controller Name	Action Name	ID
_Machines	Create	ResourcePool ID

### *Description*

A form to create a new Machine. Note: ResourcePool ID is optional; it is just to select the default resource pool while creating a machine.

### *In code*

1. Pass to the view a dropdown list of Machine Platforms.
2. Pass to the view a dropdown list of Machine Types.

3. Pass to the view a dropdown list of Resource Pools, if we passed the id it will be selected as the default one.
4. Pass to the view a dropdown list of Login Credentials.
5. Pass to the view a Multi-Select list of Software Appliances.

### Delete a Machine Widget

Controller Name	Action Name	ID
_Machines	Delete	Machine ID

#### *Description*

A confirmation message to delete a Machine.

#### *In code*

1. Get the Machine entity from the database using the given ID
2. Pass this entity to the view

### Machine's Details Widget

Controller Name	Action Name	ID
_Machines	Details	Machine ID

#### *Description*

Detailed information about a specific Machine.

#### *In code*

1. Get the Resource Pool name associated with this machine, and then pass it to the view.
2. Get the Machine Login Credential associated with this machine, and then pass it to the view.
3. Get the Machine entity from the database using the given ID.
4. Pass this entity to the view.

### Edit a Machine Widget

Controller Name	Action Name	ID
_Machines	Edit	Machine ID

#### *Description*

Edit a specific Machine by getting its ID.

#### *In code*

1. Get the Machine entity from the database.
2. Pass to the view a dropdown list of Machine Platforms.
3. Pass to the view a dropdown list of Machine Types.
4. Pass to the view a dropdown list of Resource Pools, if we passed the id it will be selected as the default one.
5. Pass to the view a dropdown list of Login Credentials.
6. Pass to the view a Multi-Select list of Software Appliances.
7. Pass the whole entity to the view so the user can edit it.

### Create Machines Login Credential Widget

Controller Name	Action Name	ID
_MachinesLoginCredentials	Create	N/A

#### *Description*

A form to create a new MachineLoginCredential.

#### *In code*

Pass to the view a dropdown list of Machines.

### Delete a Machines Login Credential Widget

Controller Name	Action Name	ID
_MachinesLoginCredentials	Delete	MachineLoginCredential ID

### *Description*

A confirmation message to delete a MachineLoginCredential.

### *In code*

1. Get the MachineLoginCredential entity from the database using the given ID.
2. Pass this entity to the view.

### **Machines Login Credential's Details Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_MachinesLoginCredentials	Details	MachineLoginCredential ID

### *Description*

Detailed information about a specific MachineLoginCredential

### *In code*

1. Get the MachineLoginCredential entity from the database using the given ID.
2. Pass this entity to the view.

### **Edit a Machines Login Credential Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_MachinesLoginCredentials	Edit	MachineLoginCredential ID

### *Description*

Edit a specific MachineLoginCredential by getting it's ID.

### *In code*

1. Get the MachineLoginCredential entity from the database.
2. Pass to the view a dropdown list of selected Machines.
3. Pass the whole entity to the view so the user can edit it.

## List all Machines Login Credentials Widget

Controller Name	Action Name	ID
_MachinesLoginCredentials	List	N/A

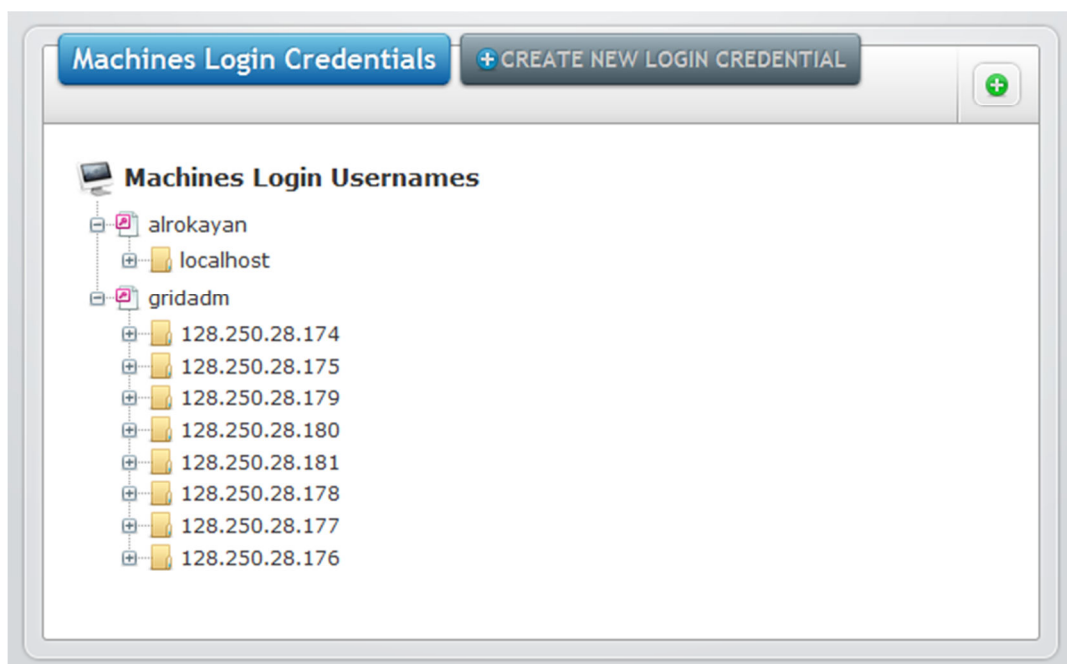


Figure 18: \_MachinesLoginCredentials Controller - List Action

### Description

List all Machine Login Credentials in table.

### In code

Get all MachineLoginCredentials entities and pass them to the view.

## Create Portal User Widget

Controller Name	Action Name	ID
_PortalUsers	Create	N/A

### Description

A form to create a new WebPortalLoginCredential.

## Delete a Portal User Widget

Controller Name	Action Name	ID
_PortalUsers	Delete	WebPortalLoginCredential ID

### *Description*

A confirmation message to delete a WebPortalLoginCredential.

### *In code*

1. Get the WebPortalLoginCredential entity from the database using the given ID.
2. Pass this entity to the view.

## Portal User's Details Widget

Controller Name	Action Name	ID
_PortalUsers	Details	WebPortalLoginCredential ID

### *Description*

Detailed information about a specific WebPortalLoginCredential.

### *In code*

1. Get the WebPortalLoginCredential entity from the database using the given ID
2. Pass this entity to the view

## Edit a Portal User Widget

Controller Name	Action Name	ID
_PortalUsers	Edit	WebPortalLoginCredential ID

### *Description*

Edit a specific WebPortalLoginCredential by getting it's ID.



### *In code*

- 1- Get the WebPortalLoginCredential entity from the database.
- 2- Pass the whole entity to the view so the user can edit it.

### List all Portal Users Widget

Controller Name	Action Name	ID
_PortalUsers	List	N/A

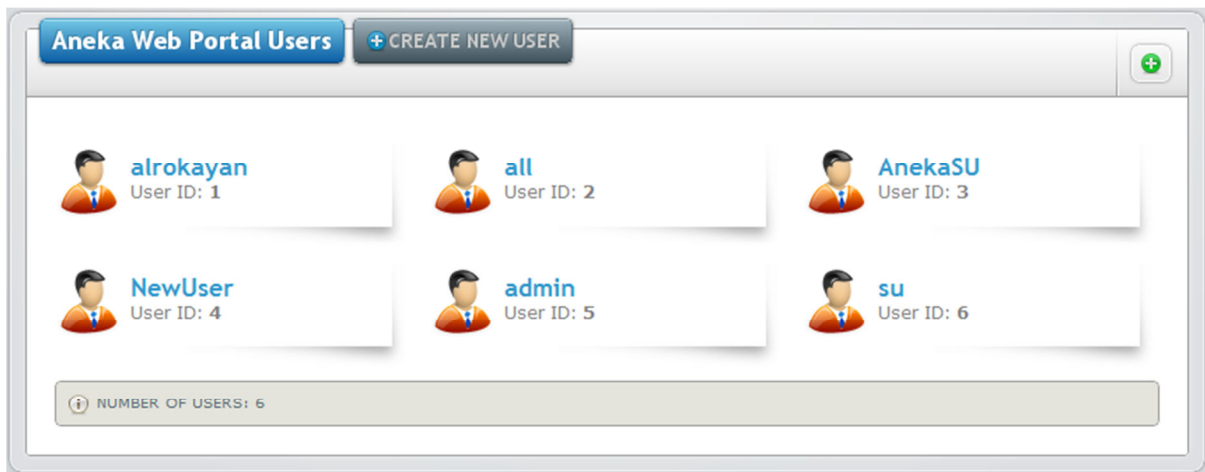


Figure 19: \_PortalUsers Controller - List Action

### *Description*

A form to create a WebPortalLoginCredential.

### *In code*

Get all the WebPortalLoginCredential entities from the database, and then pass them to the view.

### Create Resource Pool Widget

Controller Name	Action Name	ID
_ResourcePools	Create	N/A

## Description

A form to create a new ResourcePool. You will see a disabled option in this widget, which is creating a new resource pool from a cloud provider; this option is not implemented yet. Currently the form is just to create an empty resource pool.

## Delete a Resource Pool Widget

Controller Name	Action Name	ID
_ResourcePools	Delete	ResourcePool ID

## Description

A confirmation message to delete a ResourcePool.

## In code

1. Get the ResourcePool entity from the database using the given ID.
2. Pass this entity to the view.

## Resource Pool's Details Widget

Controller Name	Action Name	ID
_ResourcePools	Details	ResourcePool ID

Resource Pool: ManjraSoft Lab12

REFRESH ALL MACHINES

DisplayName	IP	Platform	Type	Software Appliances	Workers	Masters	Machine Status	Login Username	Actions
UniMelb Manjra Server	192.168.1.100	Windows	PC	1 Appliances	2 Workers	2 Masters	Starting Daemon	Admin	[Edit] [Delete] [Refresh]
UniMelb Lab PC10	192.168.1.159	Linux	Virtual Machine	1 Appliances	0 Workers	0 Masters	ServiceAccessDenied	Admin	[Edit] [Delete] [Refresh]
UniMelb Lab PC11	192.168.1.101	Windows	Virtual Machine	1 Appliances	1 Workers	1 Masters	NetworkNotReachable	Admin	[Edit] [Delete] [Refresh]
UniMelb Lab PC12	192.168.1.120	Windows	PC	1 Appliances	0 Workers	0 Masters	Adding Worker	Admin	[Edit] [Delete] [Refresh]
UniMelb Lab PC13	192.168.1.10	Windows	Virtual Machine	1 Appliances	0 Workers	0 Masters	BadCredentials	Admin	[Edit] [Delete] [Refresh]
UniMelb Lab PC14	192.168.1.18	Linux	Virtual Machine	1 Appliances	0 Workers	0 Masters	ServiceStarted	Admin	[Edit] [Delete] [Refresh]
localhost	128.250.22.60	Windows	PC	0 Appliances	0 Workers	0 Masters	Stopping Daemon	alrokayan	[Edit] [Delete] [Refresh]

Edit this resource pool | Add new machine to this resource pool | Delete this resource pool

Figure 20: \_ResourcePool Controller - Details Action

### *Description*

Detailed information about a specific ResourcePool.

### *In code*

1. Pass to the view a list of Machine Login Credentials.
2. Get the ResourcePool entity from the database using the given ID.
3. Pass this entity to the view.

### **Edit a Resource Pool Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_ResourcePools	Edit	ResourcePool ID

### *Description*

Edit a specific ResourcePool by getting its ID.

### *In code*

1. Get the ResourcePool entity from the database.
2. Pass the whole entity to the view so the user can edit it.

### **List all Resource Pools Widget**

<b>Controller Name</b>	<b>Action Name</b>	<b>ID</b>
_ResourcePools	List	N/A

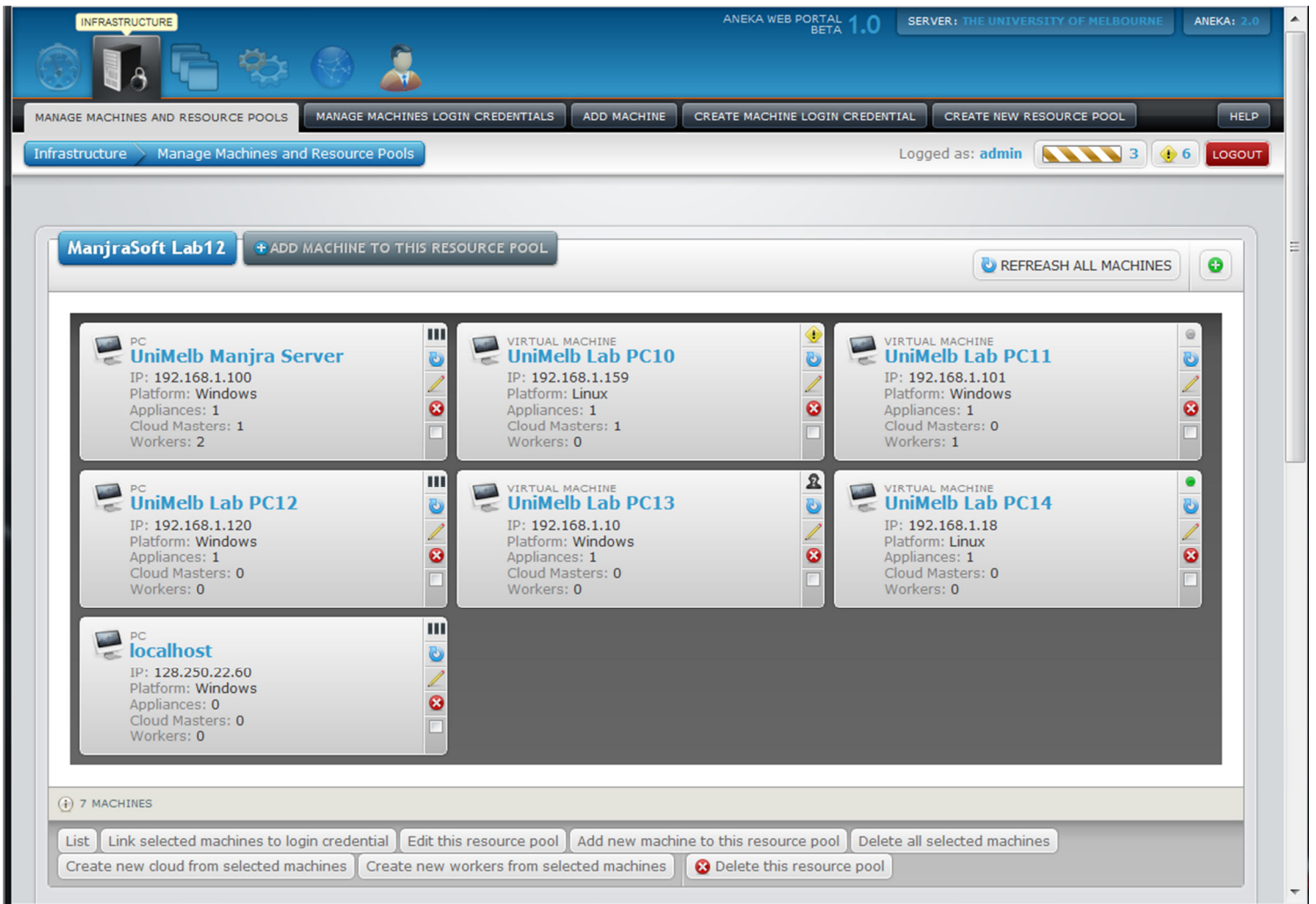


Figure 21: \_ResourcePool Controller - List Action

### Description

List all resource pools and its machines.

### In code

- 1- Get all the Resource Pools from the database.
- 2- Pass them to the view.

### Create Service Widget

Controller Name	Action Name	ID
_Services	Create	N/A

### *Description*

A form to create a new Service.

### *In code*

1. Pass to the view a dropdown list of worker.
2. Pass to the view a dropdown list of masters.

### Delete a Service Widget

Controller Name	Action Name	ID
_Services	Delete	Service ID

### *Description*

A confirmation message to delete a Service.

### *In code*

1. Get the Service entity from the database using the given ID.
2. Pass this entity to the view.

### Service's Details Widget

Controller Name	Action Name	ID
_Services	Details	Service ID

### *Description*

Detailed information about a specific Service.

### *In code*

1. Get the Service entity from the database using the given ID.
2. Pass this entity to the view.

### Edit a Service Widget

Controller Name	Action Name	ID
_Services	Edit	Service ID

### *Description*

Edit a specific Service by getting it's ID.

### *In code*

1. Get the Service entity from the database.
2. Pass to the view a dropdown list of selected worker.
3. Pass to the view a dropdown list of selected masters.
4. Pass the whole entity to the view so the user can edit it.

### List all Services Widget

Controller Name	Action Name	ID
_Services	List	N/A

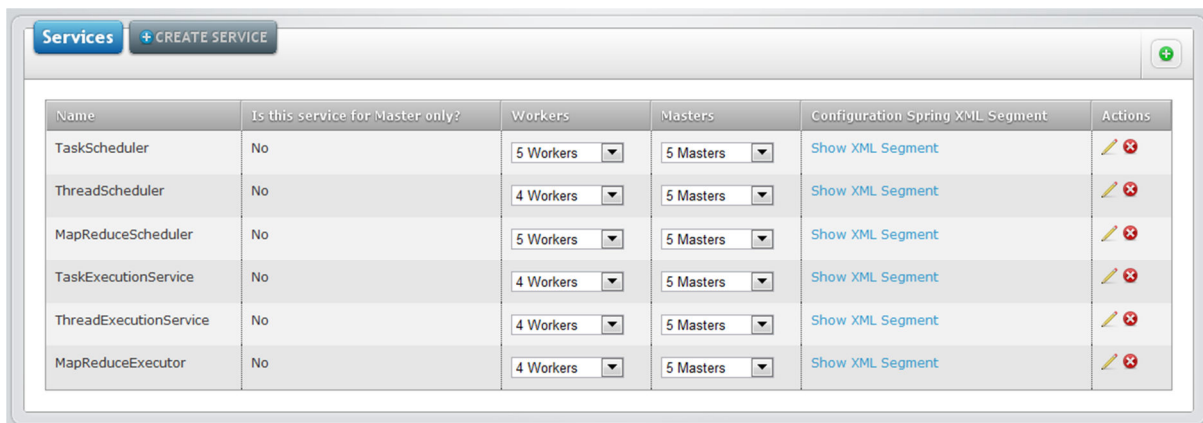


Figure 22: \_Services Controller - List Action

### *Description*

A table of services with the ability to edit or delete each Service.

### *In code*

Get and pass to the view all Services.

### Create Software Appliance Widget

Controller Name	Action Name	ID
_SoftwareAppliances	Create	N/A

### *Description*

A form to create a new SoftwareAppliance.

### *In code*

Pass to the view a dropdown list of Machines.

### Delete a Software Appliance Widget

Controller Name	Action Name	ID
_SoftwareAppliances	Delete	SoftwareAppliance ID

### *Description*

A confirmation message to delete a SoftwareAppliance.

### *In code*

1. Get the SoftwareAppliance entity from the database using the given ID.
2. Pass this entity to the view.

### Software Appliance's Details Widget

Controller Name	Action Name	ID
_SoftwareAppliances	Details	SoftwareAppliance ID

### *Description*

Detailed information about a specific SoftwareAppliance.

### *In code*

1. Get the SoftwareAppliance entity from the database using the given ID.
2. Pass this entity to the view.

### Edit a Software Appliance Widget

Controller Name	Action Name	ID
_SoftwareAppliances	Edit	SoftwareAppliance ID

### *Description*

Edit a specific SoftwareAppliance by getting it's ID.

### *In code*

1. Get the SoftwareAppliance entity from the database.
2. Pass to the view a dropdown list of selected Machines.

3. Pass the whole entity to the view so the user can edit it.

### List all Software Appliances Widget

Controller Name	Action Name	ID
_SoftwareAppliances	List	N/A



Figure 23: \_SoftwareAppliances Controller - List Action

### Description

A table of software appliances with the ability to edit or delete each Service.

### In code

Get and pass to the view all software appliances.

### Create Worker Widget

Controller Name	Action Name	ID
_Workers	Create	Cloud ID

### Description

A form to create a new Worker. The cloud ID is to link the cloud with the new worker.

### In code

1. Pass to the view a dropdown list of clouds.
2. Pass to the view a Multi-Select list of workers only Services.
3. Pass to the view dropdown list of Machines.



## Delete a Worker Widget

Controller Name	Action Name	ID
_Workers	Delete	Worker ID

### *Description*

A confirmation message to delete a Worker.

### *In code*

1. Gets the Worker entity from the database using the given ID.
2. Pass this entity to the view.

## Worker's Details Widget

Controller Name	Action Name	ID
_Workers	Details	Worker ID

### *Description*

Detailed information about a specific Worker.

### *In code*

1. Get the Worker entity from the database using the given ID.
2. Get and pass to the view the machine that has this worker.
3. Pass this entity to the view.

## Edit a Worker Widget

Controller Name	Action Name	ID
_Workers	Edit	Worker ID

### *Description*

Edit a specific Worker by getting it's ID.

### *In code*

1. Get the Worker entity from the database.
2. Pass to the view a dropdown list of selected clouds.
3. Pass to the view a Multi-Select list of workers only selected Services.

4. Pass the whole entity to the view so the user can edit it.

## List all Workers Widget

Controller Name	Action Name	ID
_Workers	List	Cloud ID

Worker Name	Port	Cost	Services	Is Quarantined?	Worker Status	Actions
w1 (128.250.28.174)	9799	1	6 Services	NO	Creating Worker Container	[Refresh] [Stop] [Delete]
w2 (128.250.28.175)	8941	2	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w3 (128.250.28.179)	10	9893	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w4 (128.250.28.180)	9564	10	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w5 (128.250.28.181)	9564	10	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w6 (localhost)	9564	20	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w7 (128.250.28.178)	9564	88	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]
w8 (localhost)	9564	8877	6 Services	NO	Creating Worker Container	[Refresh] [Stop] [Delete]
w9 (128.250.28.177)	9564	87	6 Services	NO	NetworkUnreachable	[Refresh] [Stop] [Delete]
w10 (128.250.28.176)	9564	8	6 Services	NO	ServiceActive	[Refresh] [Stop] [Delete]

Figure 24: List all Workers Widget

### Description

A simple list of workers for a specific cloud.

### In code

1. Get that specific cloud using the Cloud ID.
2. Create a Dictionary, the key is the worker ID and the value is the Machine entity.
3. Pass the Dictionary to the view.
4. Pass the cloud to the view.

### Pages

We should not use “Page” phrase because we are using MVC pattern. However, because we have widgets, I found it very useful to call “Page” on the views that are not partial and include the \_Layout view before loading the displayed view.

### /CloudManagement/BrowseClouds

This page has been implemented to call only one widget, which is: \_Clouds/List

```

<article class="container_12">
  <div id="ValidationSummaryDiv_CloudManagement_BrowseClouds">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_Clouds")
    </div>
  </section>
</article>

```

### [/CloudManagement/BrowseCloudUsers](#)

This page has been implemented to call only one widget, which is: `_CloudUsers/List`

```

<article class="container_12">
  <div id="ValidationSummaryDiv_CloudManagement_BrowseCloudUsers">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_CloudUsers")
    </div>
  </section>
</article>

```

### [/CloudManagement/CloudDetails/1](#)

Number "1" after the action name is just to illustrate that this view requires an ID, the ID here is Cloud ID. This page has been implemented to call the following six widgets:

1. `_Clouds/Details/1`
2. `_Workers/List/1`
3. `_Charts/CPU_Utilization_Range/1`
4. `_Charts/CPU_Utilization_Live/1`
5. `_Charts/JobsStatusSummary/1`
6. `_ApplicationsReport/ApplicationsList/1`

```

<article class="container_12">
  <div id="ValidationSummaryDiv_CloudManagement_CloudDetails">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("Details", "_Clouds", new { id = @Model.CloudId })
    </div>
  </section>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_Workers", new { id = @Model.CloudId })
    </div>
  </section>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("CPU_Utilization_Range", "_Charts", new { id = @Model.CloudId })
    </div>
  </section>
  <section class="grid_6"><br />
    <div class="block-border">
      @Html.Action("CPU_Utilization_Live", "_Charts", new { id = @Model.CloudId })
    </div>
  </section>
  <section class="grid_6"><br />
    <div class="block-border">
      @Html.Action("JobsStatusSummary", "_Charts", new { id = @Model.CloudId })
    </div>
  </section>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("ApplicationsList", "_ApplicationsReport", new { id = @Model.CloudId })
    </div>
  </section>
</article>

```

## /Home/Dashboard

This page is just a container to the partial view: Home/DashboardContent/

```

<article class="container_12">
  <div id="ValidationSummaryDiv_Home_Dashboard">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>

  @Html.Action("DashboardContent", "Home")
</article>

```

## /Home/LogOn

This page has the login page and the JQuery validation Java Script files.

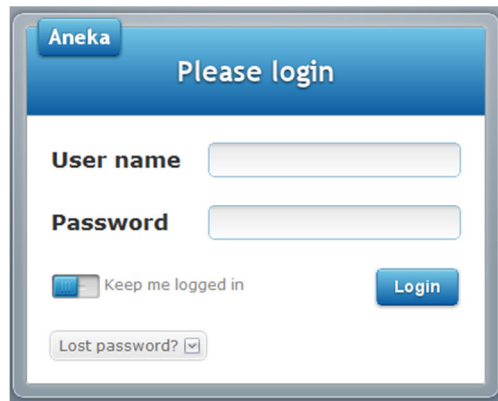


Figure 25:Aneka Login Box

### [/Infrastructure/ManageMachinesandResourcePools](#)

This page has been implemented to call only one widget, which is: `_ResourcePools/List`.

```
<article class="container_12">
  <div id="ValidationSummaryDiv_Infrastructure_ManageMachinesandResourcePools">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_ResourcePools")
    </div>
  </section>
</article>
```

### [/Infrastructure/ManageMachinesLoginCredentials](#)

This page has been implemented to call only one widget, which is: `_MachinesLoginCredentials/List`.

```
<article class="container_12">
  <div id="ValidationSummaryDiv_Infrastructure_ManageMachinesLoginCredentials">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_MachinesLoginCredentials")
    </div>
  </section>
</article>
```

### [/Services/BrowseServicesCatalog](#)

This page has been implemented to call only one widget, which is: `_Services/List`.

```

<article class="container_12">
  <div id="ValidationSummaryDiv_Services_BrowseServicesCatalog">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_Services")
    </div>
  </section>
</article>

```

### [/SoftwareAppliances/BrowseSoftwareAppliances](#)

This page has been implemented to call only one widget, which is: [\\_SoftwareAppliances/List](#).

```

<article class="container_12">
  <div id="ValidationSummaryDiv_SoftwareAppliances_BrowseSoftwareAppliances">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_SoftwareAppliances")
    </div>
  </section>
</article>

```

### [/WebPortalUsersManagement/UsersManagement](#)

This page has been implemented to call only one widget, which is: [\\_PortalUsers/List](#).

```

<article class="container_12">
  <div id="ValidationSummaryDiv_WebPortalUsersManagement_UsersManagement">
    @Html.ValidationSummary()
  </div>
  <div class="clear"></div>
  <section class="grid_12"><br />
    <div class="block-border">
      @Html.Action("List", "_PortalUsers")
    </div>
  </section>
</article>

```

### Partial Views (Other Than the Widgets):

Partial views do not use the `_layout` view (aka master page).

## /Home/DashboardContent

This view loops among all the added widgets to the logged-in user and display them on the dashboard.

## /Shared/\_ActivitiesList

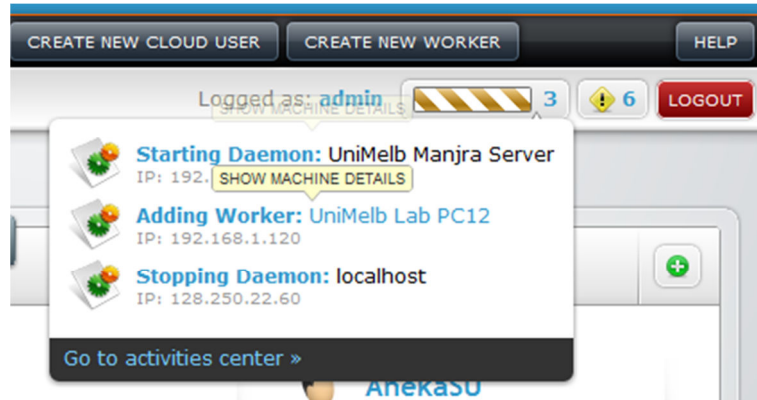


Figure 26: Activities List

This view loops among many entities to check if there are any jobs in progress to display them in the Activity Centre at the top right corner of the page. This view checks the following entities:

- Machines
- Clouds and Master
- Workers

## /Shares/\_IssuesList

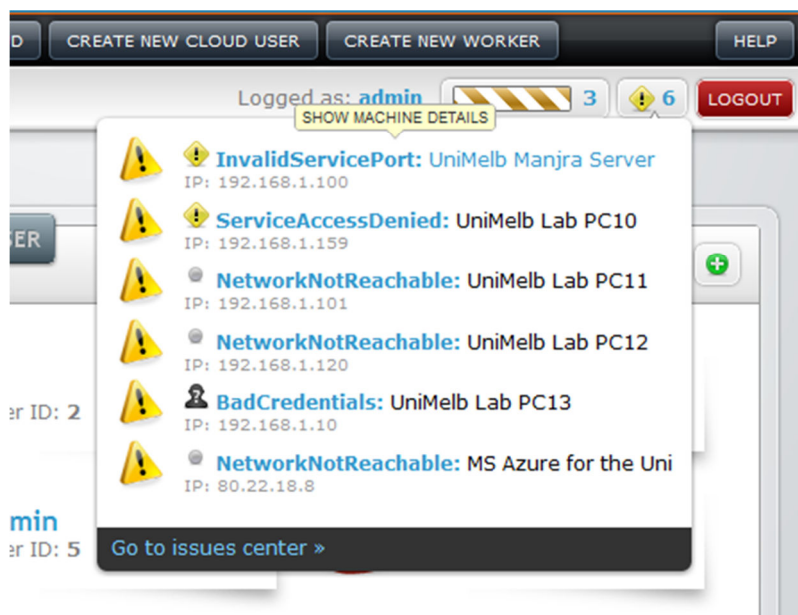


Figure 27: Issues List

This view loops among many entities to check if there are any errors or warnings to display them in the Issues Centre at the top right corner of the page. This view checks the following entities:

- Machines
- Clouds and Master
- Workers

### **To-do list For a Developer to Create a New Widget Controller:**

- 1- Rename the post actions to xxxxPost
- 2- All actions return *PartialView* except "CreatePost" and "EditPost" they  
"return View("~/Views/Home/Dashboard.cshtml",obj);"
- 3- Rename the *index* to *List*
- 4- Rename the *DeletePost* action parameter from "id" to  
"ControllerNameId"
- 5- Replace "return RedirectToAction("Index");" with "return  
RedirectToAction("Dashboard", "Home");"



# Chapter Four: Aneka Web Portal Installation

---

## System requirements

### Supported Operating Systems

- Windows XP SP3
- Windows Server 2003 SP2
- Windows Vista SP1 or later
- Windows Server 2008 (not supported on Server Core Role)
- Windows 7
- Windows Server 2008 R2 (not supported on Server Core Role)
- Windows 7 SP1
- Windows Server 2008 R2 SP1

### Supported Architectures:

- x86
- x64
- ia64 (some features are not supported on ia64 for example, WPF)

### Hardware Requirements:

- Recommended Minimum: Pentium 1 GHz or higher with 512 MB RAM or more
- Minimum disk space:
  - x86 – 850 MB
  - x64 – 2 GB

### Prerequisites:

- Windows Installer 3.1 or later [11]
- Internet Explorer 5.01 or later [12]
- Microsoft .NET Framework 4 [13]

## Installation

To Install Aneka Web Portal you must have the IIS 7.x or later installed in the server, IIS 7.x is already installed in the following operating systems:

- Windows Server 2008 R2
- Windows Server 2008
- Windows Vista (Home Premium, Business, Enterprise and Ultimate editions)
- Windows 7 (Home Premium, Professional, Enterprise and Ultimate editions)

If you want to install Aneka Web Portal on other windows operating systems you must install Microsoft WebMatrix [14] instead to IIS 7.x.

On the top of the IIS/WebMatrix you have to install and configure a database, we recommend: SQL Server 2008. SQL Express will work fine. You can install the database in the web server or on a remote server.

If you have the IIS/WebMatrix and SQL Server installed, now copy the Aneka Web Portal files to the web directory, or just create a virtual web directory.

Before going to the Aneka Web Portal web application home page you have to configure the database connection string, repository location, and repository access URL from the Web.Config. There are two Web.Config, you have to configure the one in the root folder. For example:

```
<connectionStrings>
  <add
    name="AnekaWebPortalDbContext"
    connectionString="
      Data Source=SERVER-WEB110;
      Initial Catalog=AnekaWebPortal;
      Integrated Security=True"
    providerName="System.Data.SqlClient"
  />
</connectionStrings>

<appSettings>
  <add key="webpages:Version" value="1.0.0.0"/>
  <add key="ClientValidationEnabled" value="true"/>
  <add key="UnobtrusiveJavaScriptEnabled" value="true"/>

  <add key="RepoLocalLocation" value="C:\VS Project\AnekaWebPortal\AnekaWebPortal\Repository"/>
  <add key="RepoAccessURL" value="\\128.250.22.60\C$\VS Project\AnekaWebPortal\AnekaWebPortal\Repository"/>
  <add key="RepoAccessUser" value="SERVER_USERNAME"/>
  <add key="RepoAccessPassword" value="SERVER_PASSWORD"/>
</appSettings>
```

# Chapter Five: Conclusion & Future Work

---

## **Conclusion:**

The infrastructure of Aneka Web Portal is very solid and well implemented. With this new Portal for Aneka the organizations' top management and executives can access the portal from anywhere around the world and have a summary of the current statuses for the whole system or just for a specific cloud. Aneka Web Portal is not just designed for the best user experience, but also for the developers to edit any exist function or add any new feature to the Portal.

## **Future work:**

The project successfully achieved the planned goal and all the requirements within time within scope. However, I would like to list some suggested future work tasks to be added to the web portal, some of these tasks are just for UI improvements, while the others are to add some more functionality.

### **Tasks in the Logic**

- Quarantine a worker
- Add custom services
- Probe ports and host while creating a machine
- Storage accounting and reporting
- Import machines using a Machine List text file
- Level of permissions (Access Control)
- Lost password handler
- Fetch exist cloud's workers (currently fiches only masters)
- Create resource pool from cloud provider
- Log every Action in every Controller

### **Tasks in the User Interface**

- Actions in controllers must redirect to the last opened page
- Sort widgets in the dashboard
- Refresh button with every widget (Ajax update) - with Interval time

- Auto update Activities and Issues List (aka Centre)
- In /\_ResourcePools/List.cshtml:
  - Link bulk of selected machines to a login credential
  - Delete bulk of selected machines
  - Create new cloud from a group of selected machines
  - Create new workers from a group selected machines
- In /Clouds/List.cshtml: Implements the bar figures (CPU Load, Workers Load and Disk Space)
- Filter Jobs based on users
- Display all actions in one place (as a list or tree view list)
- Resources from multiple domains
- List of resource names
- Export widget as a text or HTML file
- Large Aneka Web Portal title at the top
- Reduce number of widgets in a page
- Mobile view

# Chapter Six: References & Appendixes

---

## References

- [1] “Aneka: Enabling .NET-based Enterprise Grid and Cloud Computing,” Manjrasoft Pty. Ltd., [Online]. Available: <http://www.manjrasoft.com/products.html>. [Accessed 15 10 2011].
- [2] “ASP.NET MVC Overview,” 2011. [Online]. Available: [http://msdn.microsoft.com/en-us/library/dd381412\(VS.98\).aspx](http://msdn.microsoft.com/en-us/library/dd381412(VS.98).aspx). [Accessed 15 10 2011].
- [3] Microsoft, “ASP.NET MVC 3,” Microsoft, 2011. [Online]. Available: <http://www.asp.net/mvc/mvc3>. [Accessed 1 10 2011].
- [4] ScottGu, “Introducing ASP.NET MVC 3 (Preview 1),” ScottGu's Blog, 27 7 2010. [Online]. Available: <http://weblogs.asp.net/scottgu/archive/2010/07/27/introducing-asp-net-mvc-3-preview-1.aspx>. [Accessed 5 10 2011].
- [5] Microsoft, “ASP.NET MVC 3,” Microsoft, 5 October 2011. [Online]. Available: [http://msdn.microsoft.com/en-us/library/gg416514\(v=vs.98\).aspx](http://msdn.microsoft.com/en-us/library/gg416514(v=vs.98).aspx). [Accessed 7 10 2011].
- [6] Microsoft, “Intro to ASP.NET MVC 3,” Microsoft, [Online]. Available: <http://www.asp.net/mvc/tutorials/getting-started-with-mvc3-part1-cs>. [Accessed 1 9 2011].
- [7] ScottGu, “Introducing “Razor” – a new view engine for ASP.NET,” ScottGu's Blog, 2 7 2010. [Online]. Available: <http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx>. [Accessed 9 10 2011].
- [8] ThemeForest, “Constellation complete admin skin,” ThemeForest, [Online]. Available: <http://themeforest.net/item/constellation-complete-admin->

- skin/discussion/116461. [Accessed 1 7 2011].
- [9] Constellation, "Constellation complete admin skin," Constellation, [Online]. Available: <http://themeforest.net/item/constellation-complete-admin-skin/116461>. [Accessed 1 9 2011].
- [10] P. Haack, J. Galloway, B. Wilson and K. S. Allen, Professional ASP.NET MVC 3, Wrox, 2011.
- [11] Microsoft, "Windows Installer 3.1," Microsoft, [Online]. Available: <http://www.microsoft.com/downloads/details.aspx?familyid=889482fc-5f56-4a38-b838-de776fd4138c&displaylang=en>. [Accessed 1 10 2011].
- [12] Microsoft, "Internet Explorer 5.01," Microsoft, [Online]. Available: <http://www.microsoft.com/windows/downloads/ie/getitnow.mspx>. [Accessed 1 9 2011].
- [13] Microsoft, "Microsoft .NET Framework 4 (Web Installer)," Microsoft, [Online]. Available: <http://www.microsoft.com/download/en/details.aspx?id=17851>. [Accessed 1 9 2011].
- [14] Microsoft, "WebMatrix," Microsoft, [Online]. Available: <http://www.microsoft.com/web/webmatrix/>. [Accessed 1 9 2011].
- [15] Display-inline.fr, "CSS gradient generator," Display-inline.fr, [Online]. Available: <http://www.display-inline.fr/projects/css-gradient/>. [Accessed 1 9 2011].
- [16] "Highcharts," High Soft, [Online]. Available: <http://www.highcharts.com/>. [Accessed 2 9 2011].
- [17] CC, "Creative Commons Attribution 3.0," CC, [Online]. Available: <http://creativecommons.org/licenses/by/3.0/>. [Accessed 4 9 2011].
- [18] Y. Kamiyamane, "p.yusukekamiyamane icons," pinvoke.com, [Online]. Available: <http://www.pinvoke.com/>. [Accessed 12 10 2011].
- [19] P. Style, "Plain Style icons," Plain Style, [Online]. Available: [http://plainz.oh.land.to/download\\_icon.html](http://plainz.oh.land.to/download_icon.html). [Accessed 1 9 2011].

- [20] FamFamFam, "FamFamFam icons," FamFamFam, [Online]. Available: <http://www.famfamfam.com>. [Accessed 1 9 2011].
- [21] B. Alman, "Ben Alman license," Ben Alman , [Online]. Available: <http://benalman.com/about/license/>. [Accessed 1 10 2011].
- [22] a. jardine, "allan jardine ui developer," allan jardine, [Online]. Available: <http://www.sprymedia.co.uk/>. [Accessed 3 8 2011].
- [23] K. Wood, "Keith Wood," [Online]. Available: <http://keith-wood.name/index.html>. [Accessed 8 9 2011].
- [24] "Open Source Initiative OSI - The MIT License (MIT):Licensing," MIT License, [Online]. Available: <http://opensource.org/licenses/mit-license.php>.
- [25] Microsoft, "ASP.NET MVC 3 RTM Download Center," Microsoft, 12 1 2011. [Online]. Available: <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=4211>. [Accessed 1 10 2011].

## Appendix

### The Constellation Admin Skin [9]

This template is intended for large admins: it provides a wide range of CSS styles for displaying data and powerful JS/PHP functions. The attached documentation with this report in */constellation-complete-admin-skin/doc/index.htm* will guide you into using and customizing this skin, with detailed guides on how to use CSS classes and template functions. Also, for full API functions can found in the same documentation.

This template is not free; please visit this page for commercial licenses: <http://themeforest.net/item/constellation-complete-admin-skin/116461>

This template is built on HTML5 with CSS3 features. It is compatible with Internet Explorer 7+, Firefox 3+, Opera 9+, Safari 3+, Chrome 3+ and any modern browser. It uses CSS shadows and gradients [15] to reduce required images for backgrounds, thus reducing bandwidth use.

It is powered by jQuery, on a non-obtrusive way: plug-ins are written to work even with **jQuery.noConflict()** on.

The template provides a mobile version, which is compatible with most webkit-based mobile browsers (iPhone, Android...). All satellite pages (login, error, error documents...) are compatible with both standard and mobile templates, by using CSS media queries. However, the mobile view of Aneka Web Portal is not implemented and it is out of the project scope, I've added the mobile view in the future works section.

For any question, use the theme comment page [8] on ThemeForest.

### Highchart [16]

Highcharts is a charting library written in pure JavaScript, offering intuitive, interactive charts to Aneka Web Portal web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie and scatter chart types.

Highchart is licensed under a Creative Commons Attribution 3.0 [17] license for non-commercial use. Please visit this page for the commercial licenses: <http://www.highcharts.com/license>



### **Fugue Icon Set [18]**

Copyright © 2009 Yusuke Kamiyamane via WebAppers.com. All rights reserved. The icons are licensed under a Creative Commons Attribution 3.0 license.

### **FineFiles Icon Set [19]**

Copyright © Plain Style

### **Flags Icon Set [20]**

Copyright © FamFamFam

### **Web Application Icon Set**

Copyright © WebAppers.com. The icons are licensed under a Creative Commons Attribution 3.0 [17] license.

### **jQuery hashChange plugin [21]**

Copyright (c) 2010 "Cowboy" Ben Alman. Dual licensed under the MIT and GPL licenses.

### **SyntaxHighlighter**

Copyright © alexgorbatchev.com. SyntaxHighlighter is licenced under LGPL 3

### **DataTables [22]**

Copyright Allan Jardine © 2007-2010. DataTables is dual licensed under the GPL v2 license or a BSD (3-point) license.

### **jQuery DatePicker [23]**

Copyright © Keith Wood

### **JSMIn.php**

Copyright © 2002 Douglas Crockford. Licensed under MIT License [24]