CLOUD APPLICATIONS AND
CA APPLOGIC

CA AppLogic® application-centric cloud platform simplifies application management

OVERVIEW

CA AppLogic is an effective application-centric cloud solution for deploying, managing, and maintaining network-based distributed applications and services. A key problem for many IT departments is increasing the speed of service provisioning. Managed service providers (MSPs) have stepped up to the plate and shown how they can provision application services more efficiently for their clients than their clients could do for themselves. CA AppLogic streamlines the administration and optimal use of application and hardware resources and enables MSPs to better align with the business needs of their clients and react to changes in these needs with greater speed, agility, and efficiency.

In Principled Technologies tests in our labs, we found that CA AppLogic greatly simplified application management while significantly reducing the time it takes to deploy, resource, or replicate an application or service, thanks to the unique management and configuration capabilities embedded in its modeling and run-time environments and in its cloud fabric.

FEBRUARY 2011
A PRINCIPLED TECHNOLOGIES WHITE PAPER
Commissioned by CA Technologies

The CA AppLogic cloud technology manages cloud hardware resources and application components in a way that allows applications to be modeled, created, provisioned, managed, and copied without concern about the underlying hardware. This greatly speeds and simplifies management of the cloud. This paper focuses on the application management aspects CA AppLogic.

We tested the CA AppLogic application management features for version 2.9.9 on three dual-processor Dell™ PowerEdge™ 2950 servers. We did some of the testing in a two-server configuration, and then added a third server to test the ease with which you could add a server to the cloud. We configured these servers with directly attached storage drives, and connected them with dual front-end and backend switches and an Internet accessible input/output security layer.

We examined several aspects of the CA AppLogic cloud application-management infrastructure concerning application configuration, resourcing, replication, and metering. Specifically, we evaluated the ease of the following: (1) importing an existing business service into the cloud, (2) resourcing an application in response to hardware changes, (3) replicating services as an alternative to multi-client applications, and (4) metering the resource consumption for each application.

INTRODUCTION

Clouds and modern applications

Clouds are all about applications. The agility and responsiveness a cloud can bring for provisioning, deploying, and scaling applications is what makes it such an attractive technological solution to align business needs with IT infrastructure.

Modern application architectures differ significantly from the monolithic architecture of early computing applications. The advent of the Internet and its associated Web computing models has introduced complex, multi-layered, and distributed application architectures, where each layer consists of many components deployed and executed on network fabrics and interconnects linking potentially dissimilar servers and data storage facilities. The sheer intricacy of configuring, testing, and deploying such component layers can challenge even highly skilled users. The need to deploy applications manually further compounds these issues, increasing both operational costs and time-to-market for new applications.

Packaging, resourcing, scaling, and metering cloud applications

Modern applications consist of stacks of software components. Configuring the components and the connections between the components, and provisioning hardware infrastructure to these components are complex and error-prone tasks. CA AppLogic allows you to package an entire distributed application as a single entity that you can start, stop, manage, move, and replicate as a whole. The entity can be instantiated on any grid with sufficient resources, regardless of the underlying hardware.

CA AppLogic makes it much easier to reprovision applications based on changes in the cloud hardware infrastructure. AppLogic detects many changes automatically and the cloud adjusts resources available to the applications accordingly. Being able to manage the entire cloud from one central console simplifies any manual operations or monitoring of the applications. Applications can scale by being given more or less resources as needed or by replication. The ease with which you can replicate applications eliminates the need for multi-client applications; you get the advantages of multi-client applications without introducing the problems of correct data partitioning, data security, and non-interference.

The CA AppLogic metering and monitoring system provides metrics for each component, network connection, or the application as a whole. You can adjust the granularity of the reporting to suit your needs, from performance monitoring to implementing a sophisticated billing system. The metering and monitoring system provides a rich set of preconfigured counters, and lets you define counters that exactly meet your needs.

CLOUD APPLICATION MANAGEMENT IN CA APPLOGIC

Cloud applications in CA AppLogic

CA AppLogic uses modeling as its primary cloud interface. To this effect, it provides a diagrammatic, Microsoft® Visio®-like interface to model applications as hierarchical compositions of virtualized appliances (i.e., application software components) and their network interconnections. AppLogic visually represents applications as diagrams, where diagram boxes denote virtualized appliances and links between boxes represent links between appliances (see, for example, the application diagram at the beginning of this paper). In CA AppLogic, a cloud is configured from server, storage, and networking components in a manner opaque to the end user. A cloud controller module takes care of provisioning and managing all networking and hardware infrastructure on behalf of cloud applications. The end user need not know the specifics of the cloud hardware and networking infrastructure to define, deploy, and redeploy cloud applications.

Each box in the diagram represents a software component in the application stack. You can create your own custom component, or use one of the prepackaged components that CA AppLogic provides. These include simple components, such as an input component for managing network connections, to higher-level components such as databases or Web servers. You can customize the prepackaged components to suit your particular needs.

Each component provides a set of input and output pads. These are logical connections, abstracted from the cloud's interconnect fabric. The input and output pads implement different possible incoming or outgoing connections from the appliance, and only the connection types appropriate for a given component type are available. Functions designate inputs and outputs. Some functions implemented in CA AppLogic include *http* for http links, *db* for database links, *fs* for file system links, and *nfs* and *cifs* for NFS and CIFS file system protocols, among others.

Connections in these application diagrams are logical. The CA AppLogic cloud controller implements these connections on the cloud's physical infrastructure. By ensuring that all component connections are logical, CA AppLogic has control over all application connections, and can ensure that each connection is locally firewalled, a feat that is nearly impossible on traditional deployments. This has important implications for network security of CA AppLogic clouds (see the Other advantages of CA AppLogic section below to learn about other white papers in this series that address this issue).

Once you have built an application, you can clone it to use for different customers. This offers many of the advantages of multi-client apps without the drawbacks. In particular, the structure of the application and the CA AppLogic inherent zero-trust security model essentially eliminate concerns over data security and noninterference.

Cloud application management advantages of CA AppLogic

CA AppLogic makes it easy to model, monitor, reuse, and share cloud applications. The ability to deal with the entire application stack as a single entity makes it easy to clone applications, which largely eliminates the need for multi-client applications. The CA AppLogic metering and monitoring system allows for sophisticated monitoring of hotspots and makes true "pay for what you use" billing possible.

The logical abstraction of all resources and interconnections greatly simplifies deployment and provisioning. CA AppLogic automatically discovers the grid's topology, the available servers and their resources, all available attached storage, and all network connections, allowing resources of the entire grid to be managed as a whole. CA AppLogic automatically detects the addition or deletion of servers or VSAN storage, changes to individual servers, wiring changes, failures of various sorts, and many other management conditions. CA AppLogic can handle many of these automatically, and the single point of management makes manual interventions easy. The cloud's management console receives notification of all failures and repair actions.

CA AppLogic provides a rich set of prepackaged components, all of which are available in 32- and 64-bit versions. These include the following:

- MySQL and PostgreSQL database components
- Tomcat application server components
- Gateway components that control access to your application, including SSL
- A monitoring component that lets you monitor the health and resource usage of your application
- Infrastructure components, such as load balancing switches and NAS
- Web server components
- SQUID proxy components

These prepackaged components allow you to build a distributed Web application by simply dragging the components on the modeling interface, filling in a few parameters, and drawing the connections between them. As we mentioned earlier, each connection is of a specific type, and connections are only allowed between input and output pads of the same type. The components allow you to specify resource requirements, such as number of CPUs, amount of memory, and amount of storage. CA AppLogic fills the resource requirements without your needing to consider the underlying hardware resources.

If the predefined components do not meet your particular needs, you can create your own custom components. These components allow you all the control and hardware independence of the prebuilt components, but you can tailor them to your particular environment. Once these custom components are built, you can add them to your catalog for future use. Any future applications you build and deploy can use the custom components as easily as the prepackaged components.

Other advantages of CA AppLogic

CA AppLogic provides far more advantages than we can discuss in this white paper. PT has produced three companion white papers that discuss some of the advantages not covered here:

- CLOUD NETWORKING WITH CA APPLOGIC[1] examines the ease of configuring and scaling the physical network infrastructure to use in the cloud and robustness of cloud application deployments when faced with network failures or reconfigurations.

[1] http://principledtechnologies.com/clients/reports/CA/AppLogic_networking.pdf

- VIRTUALIZED STORAGE IN CA APPLOGIC[2] examines how easy it is to configure and change a VSAN as part of a cloud deployment and how resilient cloud applications and the underlying storage infrastructure are in the face of unexpected failures of storage devices and servers.
- CLOUD SECURITY IN CA APPLOGIC[3] examines how the CA AppLogic zero-trust security model effectively protects cloud applications from external attacks, and how its architecture and application models ensure robust and secure operation in a changing environment in a flexible manner.

WHAT WE FOUND

To test the application management features of CA AppLogic, we chose Openbravo® ERP, a complex multi-component enterprise application. We installed Openbravo ERP 2.40 (Openbravo) on a standalone server. We then set up a test cloud and defined a model for Openbravo on it, instantiated the application on the cloud, and migrated the data from our single server installation to the cloud application. Then, to determine the manageability of a CA AppLogic application, we evaluated reprovisioning the application in response to hardware changes, used the CA AppLogic built-in metering to look at application resource usage, and replicated the application. To learn more about Openbravo ERP, visit http://www.openbravo.com/.

Initial cloud setup

We began our testing of CA AppLogic by setting up an operational cloud. The most challenging part of the process was to ensure that our physical hardware setup satisfied all constraints imposed by the CA AppLogic cloud controller, which involved our studying in depth the available documentation and performing several experiments. Once we achieved this, deploying the cloud was simple and involved only a minimal amount of work. We expect that for most users, a successful first CA AppLogic cloud deployment will make subsequent deployments much simpler as users will better understand the physical hardware requirements of their clouds.

We decided to build our cloud using commodity Linux servers. We selected one of these as our deployment server, called the *Aldo Distribution Server* in the CA AppLogic documentation. We downloaded the CA AppLogic software onto it and set up the software. Next, we defined a cloud configuration file, which included a number of required attributes at the cloud boundary, including the name of the cloud, the IP address range for external IPs, a list of IP addresses of all servers to use for deployment of the cloud, an SSH certificate path, and a user name associated with the certificate. Defining the configuration file was

[2] http://principledtechnologies.com/clients/reports/CA/AppLogic_VSAN.pdf
[3] http://principledtechnologies.com/clients/reports/CA/AppLogic_security.pdf

straightforward. The SSH certificate needed to be set up from the outset as it serves the role of a license key—it is authenticated by the CA AppLogic servers, verifies that an AppLogic cloud deployment is valid, and is required to download updates from the CA AppLogic Web site.

Before you can install CA AppLogic on a cloud server, it must be running a Linux distribution. CA has verified that AppLogic works with Red Hat®, CentOS, and Fedora™ distributions. We had CA AppLogic install 32- and 64-bit versions of CentOS initially, and later we had it add a Red Hat Enterprise Linux® 5.5 server to our cloud.

Once we had defined the configuration file and our deployment and cloud servers were in place, we ran a CA AppLogic command to create the cloud from the configuration file. This installed a bare-minimum Linux configuration on each of the cloud servers and created two partitions in each cloud server with the CA AppLogic cloud software, one active and one backup. This feature allows you to roll back to a previous version of CA AppLogic if a software upgrade does not work as expected.

At this point, CA AppLogic built a VSAN accessible from all directly attached storage in the cloud servers. During this process, CA AppLogic selected one of the cloud servers as the primary server for the cloud controller and one or more secondary servers as backups should the primary server fail. Once CA AppLogic finished deploying the cloud, we verified that the cloud was operational and that we could log in and start defining our test application.

Application definition and deployment

For this paper, we created a composite business service inside the CA AppLogic framework. We were able to encapsulate the infrastructure, software, and data as a single entity that we could configure, resource, instantiate, and copy as a single unit.

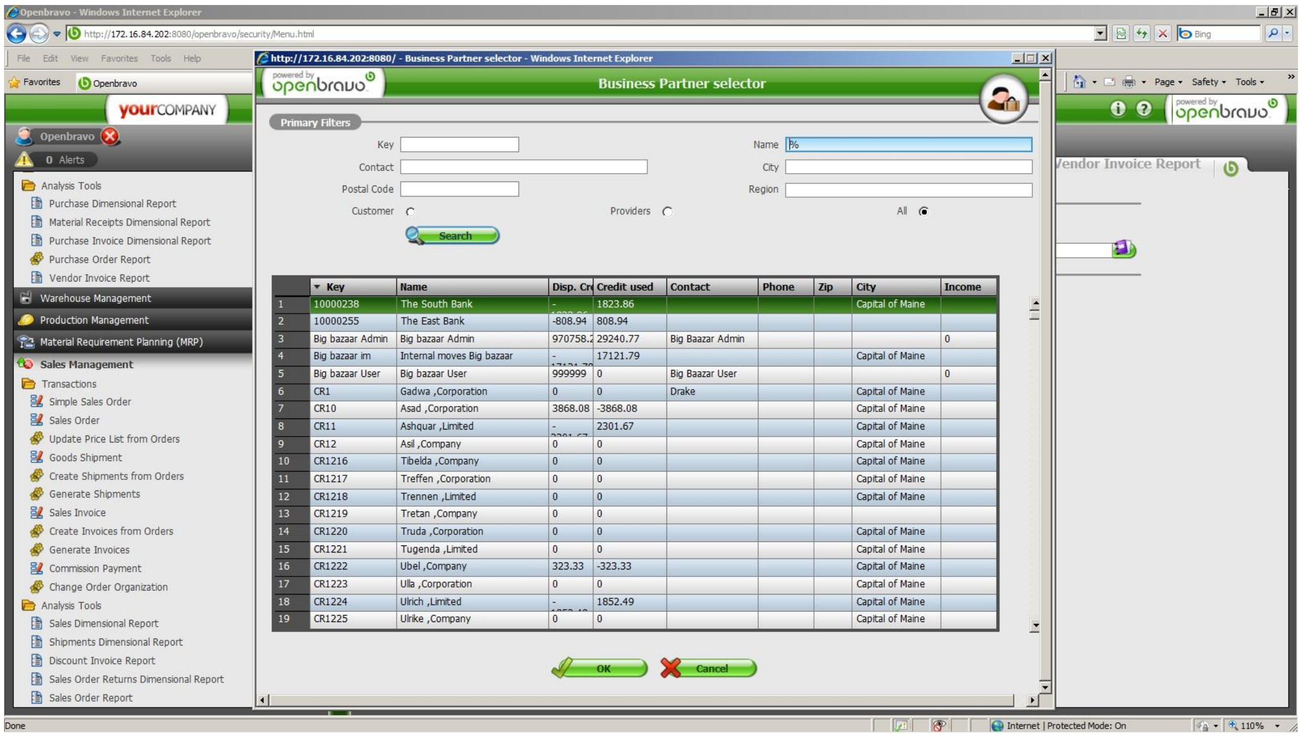
For this purpose, we imported a major business function, enterprise resource planning (ERP) into CA AppLogic. Such a function is an important part of any composite business service. We selected the Openbravo ERP.

Openbravo ERP is an open source, Web-based, enterprise resource planning system for small-and medium-scale businesses. Openbravo ERP has been recognized as best-in-breed software by IBM® Corporation, LinuxWorld, Red Herring, and the Open Source Business Foundation, and is a three-time winner of the InfoWorld® Bossie awards for the Best of Open Source Enterprise Software.

For our installation of Openbravo ERP, we used CA AppLogic predefined open-source components such as the Linux operating system and a PostgreSQL database. We also easily created a custom Apache Tomcat™ Java servlet container.

The Openbravo ERP installation allows the database server function and the Java servlet function to be on separate servers. We took advantage of this capability as we imported the ERP function into the cloud.

We created the application template using the Microsoft Visio-style diagrammatic user interface that includes a catalog of appliance icons. The user drags and drops these onto a drawing frame, and connects the appliances by dragging links from output pads to input pads. Note that we were able to manipulate the custom Tomcat component we created in the same way we could manipulate the predefined components. Because all appliances are running on their own virtual machines, by drawing the application graph, the user is effectively drawing the network topology. Figure 1 shows the application we built running inside CA AppLogic.


Openbravo - Windows Internet Explorer
http://172.16.84.202:8080/openbravo/security/Menu.html
http://172.16.84.202:8080/ - Business Partner selector - Windows Internet Explorer
Business Partner selector
Primary Filters
yourCOMPANY
Openbravo
0 Alerts
Analysis Tools
Purchase Dimensional Report
Material Receipts Dimensional Report
Purchase Invoice Dimensional Report
Purchase Order Report
Vendor Invoice Report
Warehouse Management
Production Management
Material Requirement Planning (MRP)
Sales Management
Transactions
Simple Sales Order
Sales Order
Update Price List from Orders
Goods Shipment
Create Shipments from Orders
Generate Shipments
Sales Invoice
Create Invoices from Orders
Generate Invoices
Commission Payment
Change Order Organization
Analysis Tools
Sales Dimensional Report
Shipments Dimensional Report
Discount Invoice Report
Sales Order Returns Dimensional Report
Sales Order Report
Search
OK
Cancel
Done
Internet | Protected Mode: On
110%

Key

Contact

Postal Code

Name %

City

Region

Customer ○ Providers ○ All ●

	Key	Name	Disp. Cr	Credit used	Contact	Phone	Zip	City	Income
1	10000238	The South Bank	-	1823.86				Capital of Maine	
2	10000255	The East Bank	-808.94	808.94					
3	Big bazaar Admin	Big bazaar Admin	970758.2	29240.77	Big Baazar Admin				0
4	Big bazaar im	Internal moves Big bazaar	-	17121.79				Capital of Maine	
5	Big bazaar User	Big bazaar User	999999	0	Big Baazar User				0
6	CR1	Gadwa ,Corporation	0	0	Drake			Capital of Maine	
7	CR10	Asad ,Corporation	3868.08	-3868.08				Capital of Maine	
8	CR11	Ashquar ,Limited	-	2301.67				Capital of Maine	
9	CR12	Asil ,Company	0	0				Capital of Maine	
10	CR1216	Tibelda ,Company	0	0				Capital of Maine	
11	CR1217	Treffen ,Corporation	0	0				Capital of Maine	
12	CR1218	Trennen ,Limited	0	0				Capital of Maine	
13	CR1219	Tretan ,Company	0	0					
14	CR1220	Truda ,Corporation	0	0				Capital of Maine	
15	CR1221	Tugenda ,Limited	0	0				Capital of Maine	
16	CR1222	Ubel ,Company	323.33	-323.33				Capital of Maine	
17	CR1223	Ulla ,Corporation	0	0				Capital of Maine	
18	CR1224	Ulrich ,Limited	-	1852.49				Capital of Maine	
19	CR1225	Ulrike ,Company	0	0				Capital of Maine	

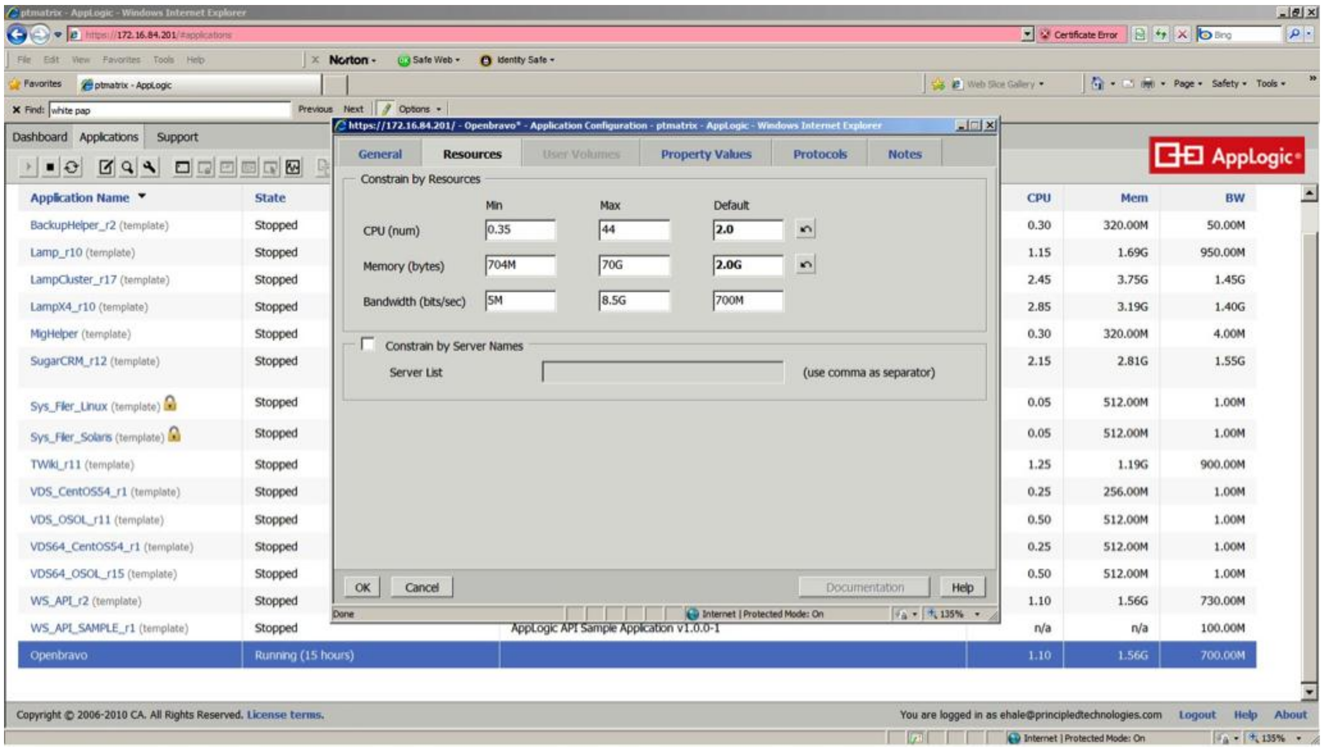
Figure 1: Openbravo running inside CA AppLogic.

Aside from the simplicity of the visual interface for defining an application, the CA AppLogic feature we found most useful for this stage was the self-documenting nature of each AppLogic component. By right-clicking a component and selecting Class Documentation, the user goes directly to detailed documentation of the component. This documentation includes a description of the component, its properties, its attributes, and—most valuably—many examples of actual application deployment situations the user can alter or use as desired. CA AppLogic self-documentation is invaluable for application definition.

Another feature we found invaluable was the ability to set global parameters for the entire application. CA AppLogic applies these parameters to every component in the application, which greatly simplifies copying business services and reconfiguring them for new clients. It allows a user to quickly and easily provide each client with an isolated environment, avoiding the problems of interference that can arise in multi-client applications.

Change application resources

Changing application resources was very easy. After selecting the Applications tab in the dashboard, we right-clicked the Openbravo application. The application had defaulted to 1.1 CPUs and 1.562 GB of RAM. We raised both of those to 2 CPUs and 2.0 GB of RAM (see Figure 2). Note that the changed values are in bold and that buttons to restore original values appear beside the changed values. After we restarted the application, the new values were in place. The resources available to the Grid reflected the change automatically.


ptmatrix - AppLogic - Windows Internet Explorer
https://172.16.84.201/#applications
Find: white pap
Dashboard
Applications
Support
https://172.16.84.201/ - Openbravo* - Application Configuration - ptmatrix - AppLogic - Windows Internet Explorer
General
Resources
User Volumes
Property Values
Protocols
Notes
OK
Cancel
Documentation
Help
AppLogic

Constrain by Resources

	Min	Max	Default
CPU (num)	0.35	44	**2.0**
Memory (bytes)	704M	70G	**2.0G**
Bandwidth (bits/sec)	5M	8.5G	700M

☐ Constrain by Server Names

Server List (use comma as separator)

Application Name	State		CPU	Mem	BW
BackupHelper_r2 (template)	Stopped		0.30	320.00M	50.00M
Lamp_r10 (template)	Stopped		1.15	1.69G	950.00M
LampCluster_r17 (template)	Stopped		2.45	3.75G	1.45G
LampX4_r10 (template)	Stopped		2.85	3.19G	1.40G
MigHelper (template)	Stopped		0.30	320.00M	4.00M
SugarCRM_r12 (template)	Stopped		2.15	2.81G	1.55G
Sys_Filer_Linux (template)	Stopped		0.05	512.00M	1.00M
Sys_Filer_Solaris (template)	Stopped		0.05	512.00M	1.00M
TWiki_r11 (template)	Stopped		1.25	1.19G	900.00M
VDS_CentOS54_r1 (template)	Stopped		0.25	256.00M	1.00M
VDS_OSOL_r11 (template)	Stopped		0.50	512.00M	1.00M
VDS64_CentOS54_r1 (template)	Stopped		0.25	512.00M	1.00M
VDS64_OSOL_r15 (template)	Stopped		0.50	512.00M	1.00M
WS_API_r2 (template)	Stopped		1.10	1.56G	730.00M
WS_API_SAMPLE_r1 (template)	Stopped	AppLogic API Sample Application v1.0.0-1	n/a	n/a	100.00M
Openbravo	Running (15 hours)		1.10	1.56G	700.00M

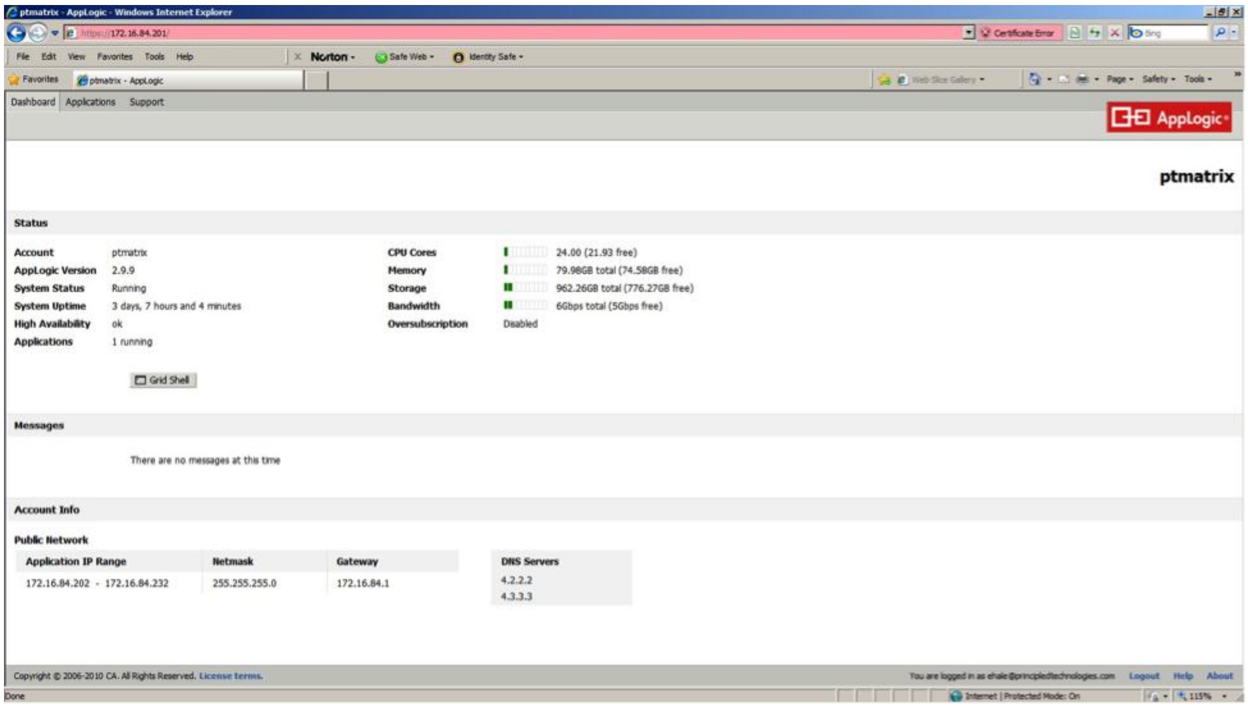
Copyright © 2006-2010 CA. All Rights Reserved. License terms.

You are logged in as ehale@principledtechnologies.com Logout Help About

Figure 2: Changing the resources for the Openbravo application.

Change cloud resources

Adding a server to the cloud was also very easy. We physically cabled the new server into the network, and installed CentOS 5.3 on it, using the default options. Once the installation was complete, we verified that we could connect via SSH to the new server, then let the Aldo distribution server configure it. Afterward, the dashboard showed the increased resources available to the cloud. Figure 3 shows the dashboard displaying the increased number of CPUs, as well as increased memory, storage, and bandwidth.



ptmatrix

Status

Account	ptmatrix	CPU Cores	24.00 (21.93 free)
AppLogic Version	2.9.9	Memory	79.98GB total (74.58GB free)
System Status	Running	Storage	962.26GB total (776.27GB free)
System Uptime	3 days, 7 hours and 4 minutes	Bandwidth	6Gbps total (5Gbps free)
High Availability	ok	Oversubscription	Disabled
Applications	1 running		

Messages

There are no messages at this time

Account Info

Public Network

Application IP Range	Netmask	Gateway	DNS Servers
172.16.84.202 - 172.16.84.232	255.255.255.0	172.16.84.1	4.2.2.2 4.3.3.3

Copyright © 2006-2010 CA. All Rights Reserved. License terms.

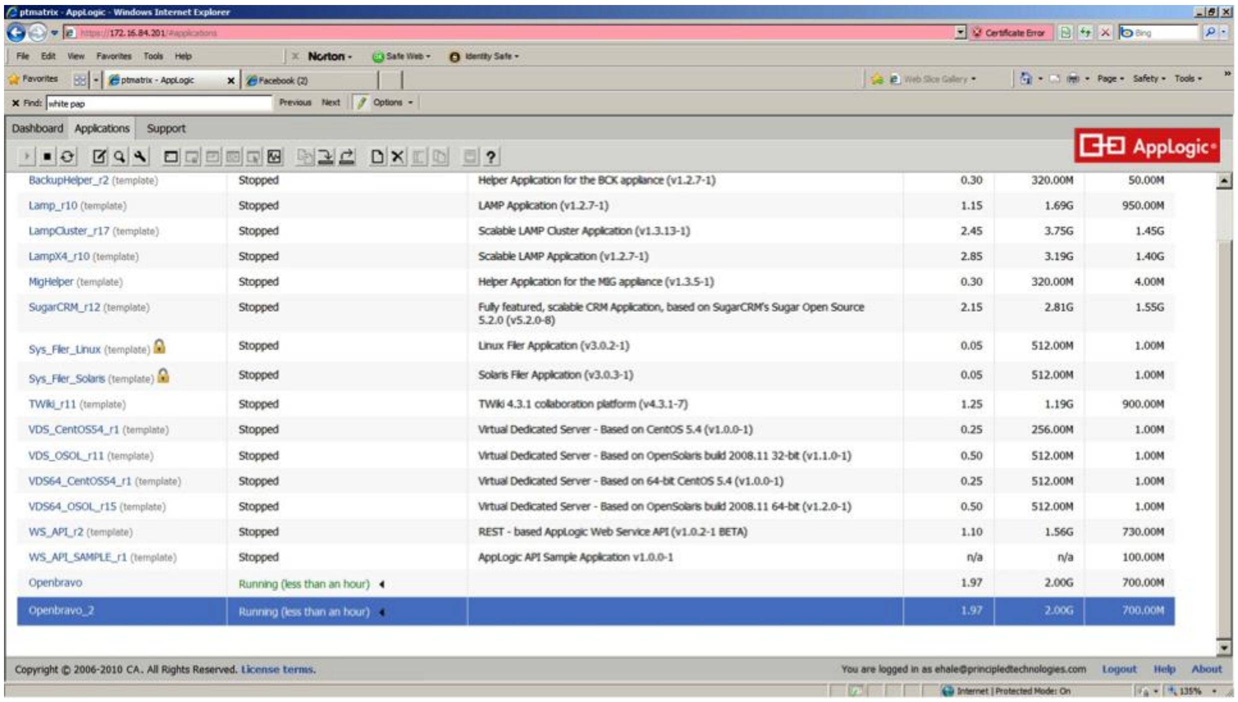
You are logged in as ehale@principledtechnologies.com

Figure 3: The dashboard showing the increased resources available to the cloud.

How CA AppLogic eliminates the need for multi-client applications

CA AppLogic makes it very easy to instantiate multiple copies of an application. Each copy uses its own set of virtual machines and dedicated storage, and is completely isolated from any other copies of that application. This avoids the problems with data interference that can plague multi-client applications.

To duplicate the Openbravo application, all we had to do was stop it, select Copy from a menu, and give a name to the new application. Once the copy was complete, we gave the new application its own set of IP addresses and started both applications. The entire process, from the beginning until both copies of the application were running, took about 10 minutes. Figure 4 shows the dashboard with both copies of the application running.



BackupHelper_r2 (template)	Stopped	Helper Application for the BCK appliance (v1.2.7-1)	0.30	320.00M	50.00M
Lamp_r10 (template)	Stopped	LAMP Application (v1.2.7-1)	1.15	1.69G	950.00M
LampCluster_r17 (template)	Stopped	Scalable LAMP Cluster Application (v1.3.13-1)	2.45	3.75G	1.45G
LampX4_r10 (template)	Stopped	Scalable LAMP Application (v1.2.7-1)	2.85	3.19G	1.40G
MigHelper (template)	Stopped	Helper Application for the MIG appliance (v1.3.5-1)	0.30	320.00M	4.00M
SugarCRM_r12 (template)	Stopped	Fully featured, scalable CRM Application, based on SugarCRM's Sugar Open Source 5.2.0 (v5.2.0-8)	2.15	2.81G	1.55G
Sys_Filer_Linux (template)	Stopped	Linux Filer Application (v3.0.2-1)	0.05	512.00M	1.00M
Sys_Filer_Solaris (template)	Stopped	Solaris Filer Application (v3.0.3-1)	0.05	512.00M	1.00M
TWiki_r11 (template)	Stopped	TWiki 4.3.1 collaboration platform (v4.3.1-7)	1.25	1.19G	900.00M
VDS_CentOS54_r1 (template)	Stopped	Virtual Dedicated Server - Based on CentOS 5.4 (v1.0.0-1)	0.25	256.00M	1.00M
VDS_OSOL_r11 (template)	Stopped	Virtual Dedicated Server - Based on OpenSolaris build 2008.11 32-bit (v1.1.0-1)	0.50	512.00M	1.00M
VDS64_CentOS54_r1 (template)	Stopped	Virtual Dedicated Server - Based on 64-bit CentOS 5.4 (v1.0.0-1)	0.25	512.00M	1.00M
VDS64_OSOL_r15 (template)	Stopped	Virtual Dedicated Server - Based on OpenSolaris build 2008.11 64-bit (v1.2.0-1)	0.50	512.00M	1.00M
WS_API_r2 (template)	Stopped	REST - based AppLogic Web Service API (v1.0.2-1 BETA)	1.10	1.56G	730.00M
WS_API_SAMPLE_r1 (template)	Stopped	AppLogic API Sample Application v1.0.0-1	n/a	n/a	100.00M
Openbravo	Running (less than an hour)		1.97	2.00G	700.00M
Openbravo_2	Running (less than an hour)		1.97	2.00G	700.00M

Copyright © 2006-2010 CA. All Rights Reserved. License terms.

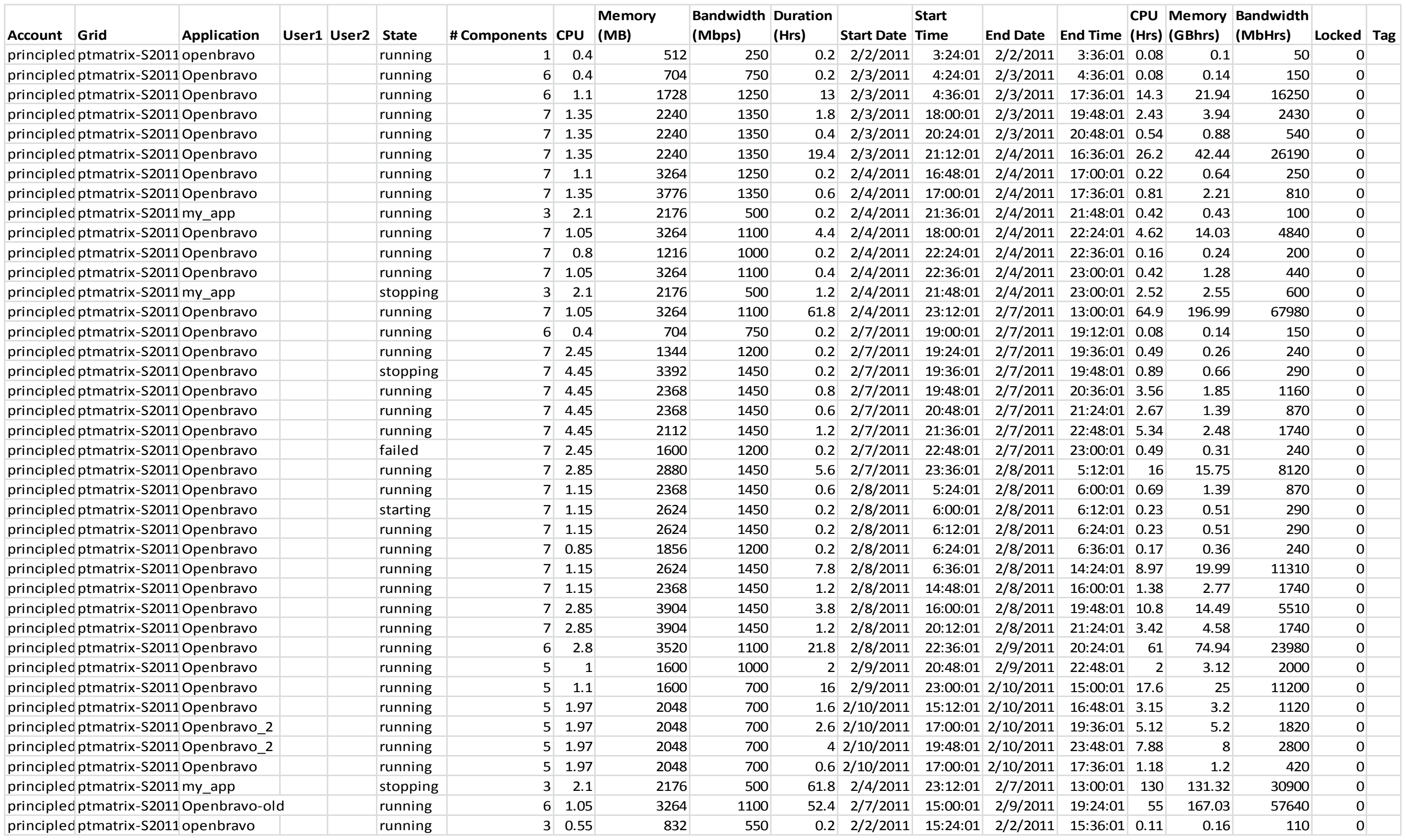
You are logged in as ehale@principledtechnologies.com Logout Help About

Figure 4: The dashboard showing both copies of the application running.

Application metering in CA AppLogic

CA AppLogic includes built-in metering; the Basic Metering Client (BMC) is part of the standard installation. The BMC reports the metering information for a grid to the Basic Metering Server (BMS) once a day. The CA AppLogic master grid maintains the BMS, so we were not required to complete a setup; all the metering needs is a connection to the Internet. Metering a simple grid requires no further steps – the grid automatically reports to the BMS once daily. By default, each grid reports separately to the metering server. To enable a collection of grids to report as a group, you can configure the Metering Gateway (MGT) to provide a single gateway for all the grids in a datacenter.

Collecting the information is as simple as issuing an SSH command to grm.ca.net, exporting the data in CSV format, and importing it into the tool of your choice. To gather the information we present in Figure 5, we ran only one command. For more information about the metering system, see the Metering System Companion at https://support.ca.com/phpdocs/0/common/impcd/r11/Cloud/doc/AppLogic_Metering_System_012811.pdf.



Account	Grid	Application	User1	User2	State	# Components	CPU	Memory (MB)	Bandwidth (Mbps)	Duration (Hrs)	Start Date	Start Time	End Date	End Time	CPU (Hrs)	Memory (GBhrs)	Bandwidth (MbHrs)	Locked	Tag
principled	ptmatrix-S2011	openbravo			running	1	0.4	512	250	0.2	2/2/2011	3:24:01	2/2/2011	3:36:01	0.08	0.1	50	0	
principled	ptmatrix-S2011	Openbravo			running	6	0.4	704	750	0.2	2/3/2011	4:24:01	2/3/2011	4:36:01	0.08	0.14	150	0	
principled	ptmatrix-S2011	Openbravo			running	6	1.1	1728	1250	13	2/3/2011	4:36:01	2/3/2011	17:36:01	14.3	21.94	16250	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.35	2240	1350	1.8	2/3/2011	18:00:01	2/3/2011	19:48:01	2.43	3.94	2430	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.35	2240	1350	0.4	2/3/2011	20:24:01	2/3/2011	20:48:01	0.54	0.88	540	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.35	2240	1350	19.4	2/3/2011	21:12:01	2/4/2011	16:36:01	26.2	42.44	26190	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.1	3264	1250	0.2	2/4/2011	16:48:01	2/4/2011	17:00:01	0.22	0.64	250	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.35	3776	1350	0.6	2/4/2011	17:00:01	2/4/2011	17:36:01	0.81	2.21	810	0	
principled	ptmatrix-S2011	my_app			running	3	2.1	2176	500	0.2	2/4/2011	21:36:01	2/4/2011	21:48:01	0.42	0.43	100	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.05	3264	1100	4.4	2/4/2011	18:00:01	2/4/2011	22:24:01	4.62	14.03	4840	0	
principled	ptmatrix-S2011	Openbravo			running	7	0.8	1216	1000	0.2	2/4/2011	22:24:01	2/4/2011	22:36:01	0.16	0.24	200	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.05	3264	1100	0.4	2/4/2011	22:36:01	2/4/2011	23:00:01	0.42	1.28	440	0	
principled	ptmatrix-S2011	my_app			stopping	3	2.1	2176	500	1.2	2/4/2011	21:48:01	2/4/2011	23:00:01	2.52	2.55	600	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.05	3264	1100	61.8	2/4/2011	23:12:01	2/7/2011	13:00:01	64.9	196.99	67980	0	
principled	ptmatrix-S2011	Openbravo			running	6	0.4	704	750	0.2	2/7/2011	19:00:01	2/7/2011	19:12:01	0.08	0.14	150	0	
principled	ptmatrix-S2011	Openbravo			running	7	2.45	1344	1200	0.2	2/7/2011	19:24:01	2/7/2011	19:36:01	0.49	0.26	240	0	
principled	ptmatrix-S2011	Openbravo			stopping	7	4.45	3392	1450	0.2	2/7/2011	19:36:01	2/7/2011	19:48:01	0.89	0.66	290	0	
principled	ptmatrix-S2011	Openbravo			running	7	4.45	2368	1450	0.8	2/7/2011	19:48:01	2/7/2011	20:36:01	3.56	1.85	1160	0	
principled	ptmatrix-S2011	Openbravo			running	7	4.45	2368	1450	0.6	2/7/2011	20:48:01	2/7/2011	21:24:01	2.67	1.39	870	0	
principled	ptmatrix-S2011	Openbravo			running	7	4.45	2112	1450	1.2	2/7/2011	21:36:01	2/7/2011	22:48:01	5.34	2.48	1740	0	
principled	ptmatrix-S2011	Openbravo			failed	7	2.45	1600	1200	0.2	2/7/2011	22:48:01	2/7/2011	23:00:01	0.49	0.31	240	0	
principled	ptmatrix-S2011	Openbravo			running	7	2.85	2880	1450	5.6	2/7/2011	23:36:01	2/8/2011	5:12:01	16	15.75	8120	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.15	2368	1450	0.6	2/8/2011	5:24:01	2/8/2011	6:00:01	0.69	1.39	870	0	
principled	ptmatrix-S2011	Openbravo			starting	7	1.15	2624	1450	0.2	2/8/2011	6:00:01	2/8/2011	6:12:01	0.23	0.51	290	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.15	2624	1450	0.2	2/8/2011	6:12:01	2/8/2011	6:24:01	0.23	0.51	290	0	
principled	ptmatrix-S2011	Openbravo			running	7	0.85	1856	1200	0.2	2/8/2011	6:24:01	2/8/2011	6:36:01	0.17	0.36	240	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.15	2624	1450	7.8	2/8/2011	6:36:01	2/8/2011	14:24:01	8.97	19.99	11310	0	
principled	ptmatrix-S2011	Openbravo			running	7	1.15	2368	1450	1.2	2/8/2011	14:48:01	2/8/2011	16:00:01	1.38	2.77	1740	0	
principled	ptmatrix-S2011	Openbravo			running	7	2.85	3904	1450	3.8	2/8/2011	16:00:01	2/8/2011	19:48:01	10.8	14.49	5510	0	
principled	ptmatrix-S2011	Openbravo			running	7	2.85	3904	1450	1.2	2/8/2011	20:12:01	2/8/2011	21:24:01	3.42	4.58	1740	0	
principled	ptmatrix-S2011	Openbravo			running	6	2.8	3520	1100	21.8	2/8/2011	22:36:01	2/9/2011	20:24:01	61	74.94	23980	0	
principled	ptmatrix-S2011	Openbravo			running	5	1	1600	1000	2	2/9/2011	20:48:01	2/9/2011	22:48:01	2	3.12	2000	0	
principled	ptmatrix-S2011	Openbravo			running	5	1.1	1600	700	16	2/9/2011	23:00:01	2/10/2011	15:00:01	17.6	25	11200	0	
principled	ptmatrix-S2011	Openbravo			running	5	1.97	2048	700	1.6	2/10/2011	15:12:01	2/10/2011	16:48:01	3.15	3.2	1120	0	
principled	ptmatrix-S2011	Openbravo_2			running	5	1.97	2048	700	2.6	2/10/2011	17:00:01	2/10/2011	19:36:01	5.12	5.2	1820	0	
principled	ptmatrix-S2011	Openbravo_2			running	5	1.97	2048	700	4	2/10/2011	19:48:01	2/10/2011	23:48:01	7.88	8	2800	0	
principled	ptmatrix-S2011	Openbravo			running	5	1.97	2048	700	0.6	2/10/2011	17:00:01	2/10/2011	17:36:01	1.18	1.2	420	0	
principled	ptmatrix-S2011	my_app			stopping	3	2.1	2176	500	61.8	2/4/2011	23:12:01	2/7/2011	13:00:01	130	131.32	30900	0	
principled	ptmatrix-S2011	Openbravo-old			running	6	1.05	3264	1100	52.4	2/7/2011	15:00:01	2/9/2011	19:24:01	55	167.03	57640	0	
principled	ptmatrix-S2011	openbravo			running	3	0.55	832	550	0.2	2/2/2011	15:24:01	2/2/2011	15:36:01	0.11	0.16	110	0	

Figure 5: Metering information for one of the grids we used.

HOW WE TESTED

One advantage of using CA AppLogic is that the user does not need identical server configurations for his or her cloud infrastructure. For testing, we used three Dell PowerEdge 2950 servers with different hardware configurations. See Appendix A for detailed configuration information for the test servers.

Configuring the Openbravo application

Figure 6 shows the Openbravo application we used for MSP testing. We dragged all components from the left panes to the application editor main window. We connected all components as we outline below.

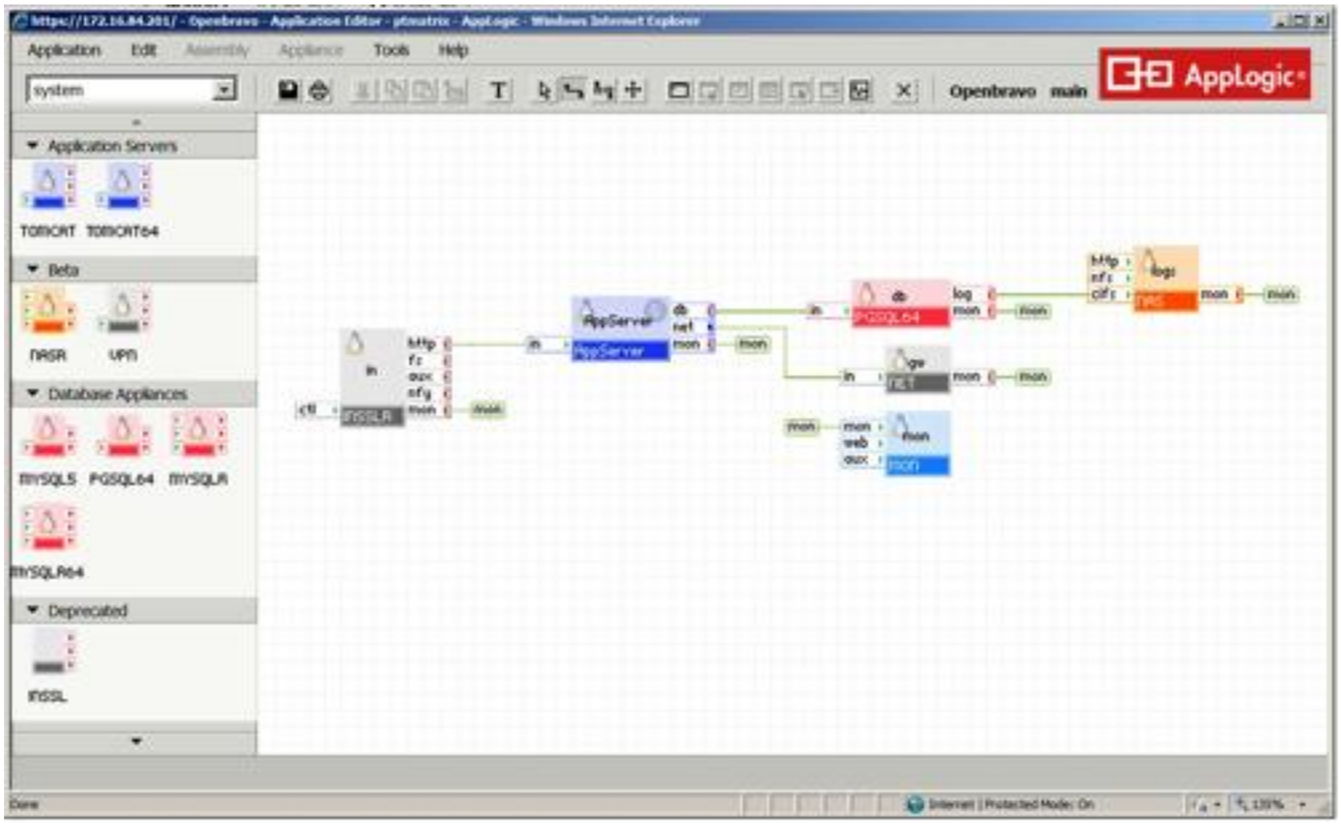


Figure 6: The Openbravo application we used for our evaluation.

Configuring Openbravo application global properties

To configure the Openbravo global properties, we performed the following steps:

1. Open the Application Editor for the Openbravo appliance (see Figure 6).
2. Click Application→Modify Boundary.
3. At the main – Class Definition (Assembly) window, click the Properties tab, and add the following mandatory properties:

   ```
   in_ip
   net_ip
   ```

   Add the following default properties:

   - `netmask: 255.255.255.0`
   - `gateway: 172.16.84.1`
   - `dns1: 4.2.2.2`
   - `dns2: 4.3.3.3`

4. Click OK.
5. Click Application→Configure.
6. Click the Property Values tab, and set the following values:

        - in_ip: `172.16.84.210`
        - net_ip: `172.16.84.211`

7. Click OK.

Configuring the gateway

We made the following configuration changes to the *in* gateway:

- ip_addr: redirect to `in_ip`
- netmask: redirect to `netmask`
- gateway: redirect to `gateway`
- l7_accept: `http`

Configuring the *net*

We made the following configuration changes to the *net* appliance:

- ip_addr: redirect to `net_ip`
- netmask: redirect to `netmask`
- gateway: redirect to `gateway`
- dns1: redirect to `dns1`
- dns2: redirect to `dns2`

Configuring the LINUX64 appliance

1. Right-click the LINUX64 appliance, and click Branch Class.
2. A window opens, and tells you when it is finished.
3. Right-click the LINUX64 appliance, and click Modify Boundary.
4. On the General tab, change the name field to `ObERP` to rename the appliance, and click OK.
5. At the pop-up window, click OK.
6. Right-click the ObERP appliance, and click Modify Boundary.
7. On the Class Definition window, click the Interfaces tab.
8. Rename the `out` port to `db`, and set the protocol to `pgsql`
9. Click the Resources tab, and change the default CPU (num) value to `0.5` and the default Memory (bytes) value to `640M`

Additional application configuration

We connected to the cloud controller with a SSH connection using PuTTY. Once connected to the cloud controller, we used the `3t` command to start a grid shell inside the SSH session.

Type the following commands:

```
ca Openbravo
vol create mon fs=ext3 size=50M
vol create pgsql fs=ext3 size=2G
vol resize AppServer.boot size=5G
vol list Openbravo -all
app start
```

Configuring the custom AppServer appliance

We connected to the cloud controller with a SSH connection using PuTTY. Once connected to the cloud controller, we used the `3t` command to start a grid shell inside the SSH session. Once the session started, we used the `SSH Openbravo:main.AppServer` command, where `Openbravo` is the name of the application and `main.AppServer` is the name of the appliance you are connecting to.

Installing the Java SDK

To install the Java SDK, run the following commands:

```
yum -y erase postgresql*
yum -y install postgresql84
wget http://...<URL>.../jdk-6u23-linux-x64.bin
su -
sh jdk-6u23-linux-x64.bin
mv jdk1.6.0_23 /usr/local/sun-jdk
echo 'export JAVA_HOME=/usr/local/sun-jdk' > /etc/profile.d/sun-
jdk.sh
echo 'export PATH=$JAVA_HOME/bin/:$PATH' >> /etc/profile.d/sun-jdk.sh
chmod +x /etc/profile.d/sun-jdk.sh
exit
su -
echo $JAVA_HOME
java -version
exit
```

Installing Apache Tomcat

To install Apache Tomcat, run the following commands:

```
wget http://archive.apache.org/dist/tomcat/tomcat-
6/v6.0.32/bin/apache-tomcat-6.0.32.tar.gz
su -
tar zxf apache-tomcat-6.0.32.tar.gz
mv apache-tomcat-6.0.32/ /usr/local/tomcat
echo 'export CATALINA_HOME=/usr/local/tomcat' >
/etc/profile.d/tomcat.sh
echo 'export CATALINA_BASE=/usr/local/tomcat' >>
/etc/profile.d/tomcat.sh
echo 'export CATALINA_OPTS="-server -Xms128M -Xmx512M -
XX:MaxPermSize=256M"' >> /etc/profile.d/tomcat.sh
chmod +x /etc/profile.d/tomcat.sh
groupadd -g 505 openbravo
useradd -g 505 -u 505 openbravo
chown -R openbravo:openbravo /usr/local/tomcat
wget http://openbravo.pastebin.com/download.php?i=8mrFtJ6H -O
/etc/init.d/tomcat
dos2unix /etc/init.d/tomcat
```

```
chmod +x /etc/init.d/tomcat
chkconfig tomcat on
exit
```

Installing Apache Ant™

To install Apache Ant, run the following commands:

```
wget http://archive.apache.org/dist/ant/binaries/apache-ant-1.7.1-
bin.tar.gz
su –
tar zxf apache-ant-1.7.1-bin.tar.gz
mv apache-ant-1.7.1/ /usr/local/ant
echo 'export ANT_HOME=/usr/local/ant' > /etc/profile.d/ant.sh
echo 'export PATH=$ANT_HOME/bin/:$PATH' >> /etc/profile.d/ant.sh
echo 'export ANT_OPTS="-Xmx512M -XX:MaxPermSize=128M"' >>
/etc/profile.d/ant.sh
chmod +x /etc/profile.d/ant.sh
exit
su –
ant –version
exit
```

Installing Openbravo ERP

1. Run the following commands:
   - `wget http://downloads.sourceforge.net/project/openbravo/09-openbravo-old-releases/Installers/2.40/OpenbravoERP_2.40-linux-x64-installer.bin`
   - `./OpenbravoERP_2.40-linux-x64-installer.bin`
2. Press Enter until the application asks Do you accept this license? [y/n]. Type `y` and press Enter.
3. Accept the defaults until you reach the Database: server parameters screen.
4. Accept the default database port, and press Enter.
5. Enter the database administrator password as follows:
   - Password: `SYSTEM`
   - Retype password: `SYSTEM`
6. At the Openbravo Database: parameters screen, accept the default database name, the default username, and enter a password for the new database user:
   - Password: `tad`
   - Retype password: `tad`
7. Accept the defaults until you reach the demo data screen.
8. When the application asks Should the installer populate the database with demo data?, type `Y`, and press Enter.
9. When the application asks if you want to continue, type `Y` and press Enter.

Finalizing Openbravo settings

To finalize Openbravo settings, run the following commands:

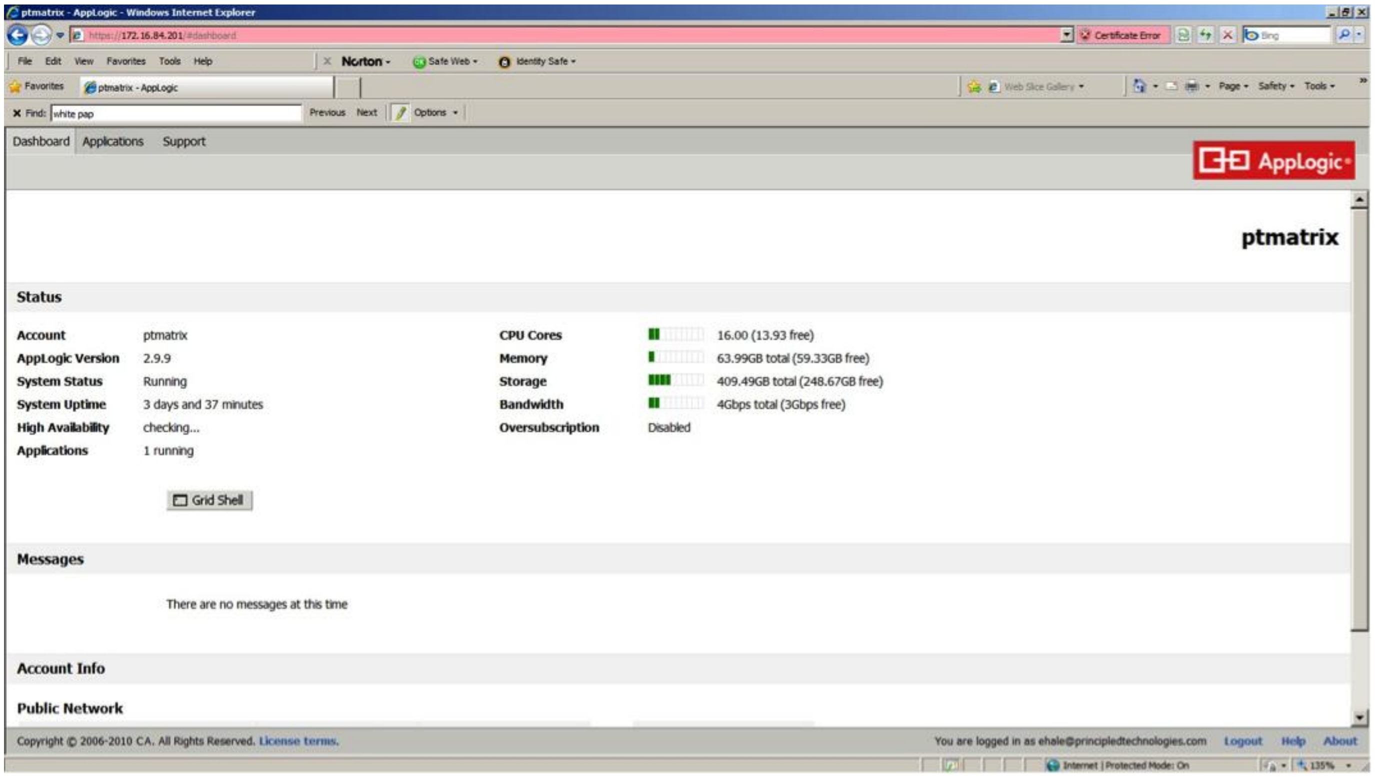
```
service tomcat start
```

```
iptables -t nat -I PREROUTING --source 0/0 --destination 0/0 -p tcp -
-dport 80 -j REDIRECT --to-ports 8080
iptables -t nat -I PREROUTING --source 0/0 --destination 0/0 -p tcp -
-dport 443 -j REDIRECT --to-ports 8443
service iptables save
```

Reprovisioning the Openbravo-based composite business service

1. In the Dashboard, select the Applications tab.
2. Right-click the Openbravo application.
3. Click Configure→Resources.
4. Change default value of 1.1 CPUs to 2 CPUs.
5. Change default value of 1.562 GB of RAM to 2 GB of RAM.
6. Click OK.
7. Right-click the Openbravo application, and choose Restart. (Note: The application took 2 minutes and 10 seconds to reboot.)
8. Select the Dashboard tab to see the grid reflect the change in available resources.

Figure 7 shows the grid reflecting the changes in available resources.



Status

Account	ptmatrix	CPU Cores	16.00 (13.93 free)
AppLogic Version	2.9.9	Memory	63.99GB total (59.33GB free)
System Status	Running	Storage	409.49GB total (248.67GB free)
System Uptime	3 days and 37 minutes	Bandwidth	4Gbps total (3Gbps free)
High Availability	checking...	Oversubscription	Disabled
Applications	1 running		

Messages

There are no messages at this time

Account Info

Public Network

Copyright © 2006-2010 CA. All Rights Reserved. License terms.

You are logged in as ehale@principledtechnologies.com Logout Help About

Figure 7: The cloud showing the decrease in available resources after giving more CPU and RAM to Openbravo.

Adding a server to the cloud

1. Physically connect the server to the network.
2. Install CentOS on the server using the default options.

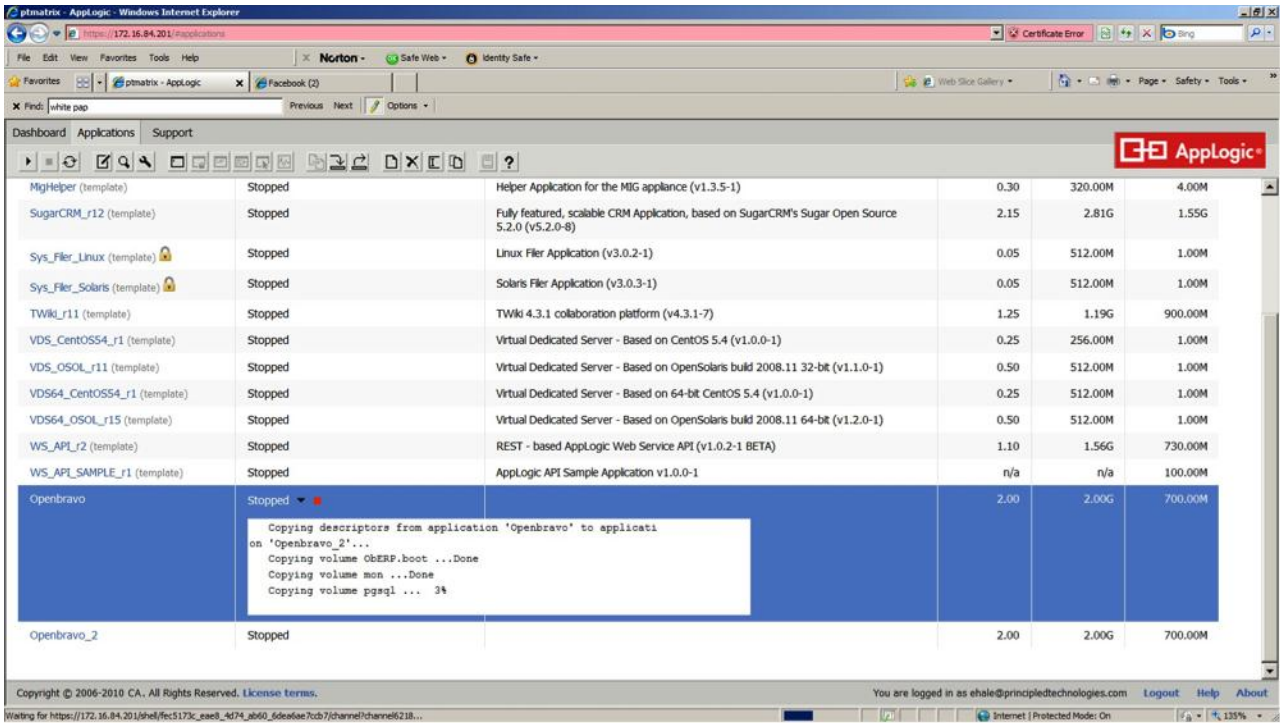
3. Configure the backbone and external network connections with IPs in the correct subnet.
4. Run the following commands:

```
ssh-copy-id -i /root/.ssh/id_rsa.pub 172.16.1.43
ssh 172.16.1.43
exit
cd applogic-2.9.9/
./ald-reimage 172.16.1.43
./aldo addsrv grid=ptmatrix servers=172.16.1.43
```

Replicating the application for new clients

1. Right-click the Openbravo application, and select Stop. (Less that 1 minute)
2. Right-click Openbravo, and select Copy.
3. Enter the name `Openbravo_2`
4. Click OK. (5 minutes)
5. Right-click Openbravo_2, and select Configure.
6. Select Property Values tab.
7. Change in_ip and net_ip addresses. The changed values appear in bold and Restore default value buttons appear.
8. Click OK.
9. Right-click each application, and select Start. (2 minutes 32 seconds for both to be copies of Openbravo to be running)

Figure 8 shows CA AppLogic copying the Openbravo application.



MigHelper (template)	Stopped	Helper Application for the MIG appliance (v1.3.5-1)	0.30	320.00M	4.00M
SugarCRM_r12 (template)	Stopped	Fully featured, scalable CRM Application, based on SugarCRM's Sugar Open Source 5.2.0 (v5.2.0-8)	2.15	2.81G	1.55G
Sys_Filer_Linux (template)	Stopped	Linux Filer Application (v3.0.2-1)	0.05	512.00M	1.00M
Sys_Filer_Solaris (template)	Stopped	Solaris Filer Application (v3.0.3-1)	0.05	512.00M	1.00M
TWiki_r11 (template)	Stopped	TWiki 4.3.1 collaboration platform (v4.3.1-7)	1.25	1.19G	900.00M
VDS_CentOS54_r1 (template)	Stopped	Virtual Dedicated Server - Based on CentOS 5.4 (v1.0.0-1)	0.25	256.00M	1.00M
VDS_OSOL_r11 (template)	Stopped	Virtual Dedicated Server - Based on OpenSolaris build 2008.11 32-bit (v1.1.0-1)	0.50	512.00M	1.00M
VDS64_CentOS54_r1 (template)	Stopped	Virtual Dedicated Server - Based on 64-bit CentOS 5.4 (v1.0.0-1)	0.25	512.00M	1.00M
VDS64_OSOL_r15 (template)	Stopped	Virtual Dedicated Server - Based on OpenSolaris build 2008.11 64-bit (v1.2.0-1)	0.50	512.00M	1.00M
WS_API_r2 (template)	Stopped	REST - based AppLogic Web Service API (v1.0.2-1 BETA)	1.10	1.56G	730.00M
WS_API_SAMPLE_r1 (template)	Stopped	AppLogic API Sample Application v1.0.0-1	n/a	n/a	100.00M
Openbravo	Stopped		2.00	2.00G	700.00M
Openbravo_2	Stopped		2.00	2.00G	700.00M

```
  Copying descriptors from application 'Openbravo' to applicati
on 'Openbravo_2'...
  Copying volume ObERP.boot ...Done
  Copying volume mon ...Done
  Copying volume pgsql ...  3%
```

Copyright © 2006-2010 CA. All Rights Reserved. License terms.
You are logged in as ehale@principledtechnologies.com Logout Help About
Waiting for https://172.16.84.201/shell/fec5173c_eae8_4d74_ab60_6dea6ae7ccb7/channel?channel6218...
Internet | Protected Mode: On
135%

Figure 8: CA AppLogic in the process of copying the Openbravo application.

Metering the applications

As we note above, CA AppLogic includes built-in metering and requires no special setup. The configuration of the SSH certificates was part of configuring our first grid. The installation for CA AppLogic installs the BMC automatically.

To gather the information in Figure 5, we used the following command:

```
ssh principledtech@grm.ca.net "3tbms meter dump
principledtech:ptmatrix-S20110125021432328856000070609 --detail
```

In this example, `principledtech` is the account name, and `ptmatrix-S20110125021432328856000070609` is the grid name.

APPENDIX A – SERVER CONFIGURATION INFORMATION

Figure 9 provides detailed configuration information about the test servers.

System	Dell PowerEdge 2950 server 1	Dell PowerEdge 2950 server 2	Dell PowerEdge 2950 server 3
Power supplies			
Total number	2	2	1
Vendor and model number	Dell 7001072-Y000	Dell N750PS0	Dell 7001072-Y000
Wattage of each (W)	750	750	750
Cooling fans			
Total number	4	4	4
Vendor and model number	NMB-MAT2415KL-04W-B96	PCF0612DE	NMB-MAT2415KL-04W-B96
Dimensions (h x w) of each	2” x 2”	2” x 2”	2” x 2”
Volts (V)	12	12	12
Amps (A)	2.10	1.68	2.10
General			
Number of processor packages	2	2	2
Number of cores per processor	4	4	4
Number of hardware threads per core	1	1	1
CPU			
Vendor	Intel®	Intel	Intel
Name	Xeon®	Xeon	Xeon
Model number	5440	5440	E5405
Socket type	LGA771	LGA771	LGA771
Core frequency (GHz)	2.83	2.83	2.00
Bus frequency (MHz)	1,333	1,333	1,333
L1 cache (KB)	128	128	128
L2 cache (MB)	12	12	12
Platform			
Vendor and model number	Dell PowerEdge 2950	Dell PowerEdge 2950	Dell PowerEdge 2950
Motherboard model number	0H603H	0CX396	0M332H
Motherboard chipset	Intel 5000X	Intel 5000X	Intel 5000X
BIOS name and version	Dell 2.6.1	Dell 2.6.1	Dell 2.6.1
BIOS settings	Virtualization enabled	Virtualization enabled	Virtualization enabled

System	Dell PowerEdge 2950 server 1	Dell PowerEdge 2950 server 2	Dell PowerEdge 2950 server 3
Memory module(s)			
Total RAM in system (GB)	16	32	16
Vendor and model number	Samsung M395T5750EZ4-CE66	Samsung M395T5750EZ4-CE66	Samsung M395T5750EZ4-CE66
Type	PC2-5300F	PC2-5300F	PC2-5300F
Speed (MHz)	667	667	667
Speed running in the system (MHz)	667	667	667
Timing/latency (tCL-tRCD-tRP-tRASmin)	5-5-5-15	5-5-5-15	5-5-5-15
Size (GB)	2	4	2
Number of RAM module(s)	8	8	8
Chip organization	Double-sided	Double-sided	Double-sided
Rank	Dual	Dual	Dual
Hard disk			
Hard disk 1			
Vendor and model number	Seagate ST973452SS	Western Digital WD 1600AAJS	Seagate ST3300656SS
Number of disks in system	2	1	2
Size (GB)	73	160	300
Buffer size (MB)	16	8	16
RPM	15,000	7,200	15,000
Type	SAS	SATA	SAS
Disk controller	Dell SAS 6/IR	Dell PERC 6/i	Dell SAS 6/IR
Hard disk 2			
Vendor and model number	N/A	Fujitsu MBA 3147RC	N/A
Number of disks in system	N/A	1	N/A
Size (GB)	N/A	146	N/A
Buffer size (MB)	N/A	16	N/A
RPM	N/A	15,000	N/A
Type	N/A	SAS	N/A
Disk controller	N/A	Dell PERC 6/i	N/A
Operating system			
Name	CentOS Release 5.3	CentOS Release 5.3	CentOS Release 5.3
Kernel release	2.6.18.8-xen0	2.6.18.8-xen0	2.6.18.8-xen0

System	**Dell PowerEdge 2950 server 1**	**Dell PowerEdge 2950 server 2**	**Dell PowerEdge 2950 server 3**
Kernel version	#1 Thu May 13 11:28:43	#1 Thu May 13 11:28:43	#1 Thu May 13 11:28:43
File system	ext 3	ext 3	ext3
Language	English	English	English
Ethernet			
NIC Type 1			
Vendor and model number	Broadcom® 5708 NetXtreme® II 5708 Ethernet Adapter	Broadcom 5708 NetXtreme II 5708 Ethernet Adapter	Broadcom 5708 NetXtreme II 5708 Ethernet Adapter
Type	Onboard	Onboard	Onboard
NIC Type 2			
Vendor and model number	N/A	N/A	Intel 82571EB Quad Port Low Profile Adapter
Type	N/A	N/A	PCI-E
Optical drive(s)			
Vendor and model number	TEAC CD-ROM CD-224E-N	TEAC CD-ROM CD-224E-N	TEAC CD-ROM CD-224E-N
USB ports			
Number	4	4	4
Type	2.0	2.0	2.0

Figure 9: Configuration details for the test servers.

ABOUT PRINCIPLED TECHNOLOGIES

Principled Technologies, Inc.
1007 Slater Road, Suite 300
Durham, NC, 27703
www.principledtechnologies.com

We provide industry-leading technology assessment and fact-based marketing services. We bring to every assignment extensive experience with and expertise in all aspects of technology testing and analysis, from researching new technologies, to developing new methodologies, to testing with existing and new tools.

When the assessment is complete, we know how to present the results to a broad range of target audiences. We provide our clients with the materials they need, from market-focused data to use in their own collateral to custom sales aids, such as test reports, performance assessments, and white papers. Every document reflects the results of our trusted independent analysis.

We provide customized services that focus on our clients' individual requirements. Whether the technology involves hardware, software, Web sites, or services, we offer the experience, expertise, and tools to help our clients assess how it will fare against its competition, its performance, its market readiness, and its quality and reliability.

Our founders, Mark L. Van Name and Bill Catchings, have worked together in technology assessment for over 20 years. As journalists, they published over a thousand articles on a wide array of technology subjects. They created and led the Ziff-Davis Benchmark Operation, which developed such industry-standard benchmarks as Ziff Davis Media's Winstone and WebBench. They founded and led eTesting Labs, and after the acquisition of that company by Lionbridge Technologies were the head and CTO of VeriTest.