CumuLogic

CumuLogic User Guide

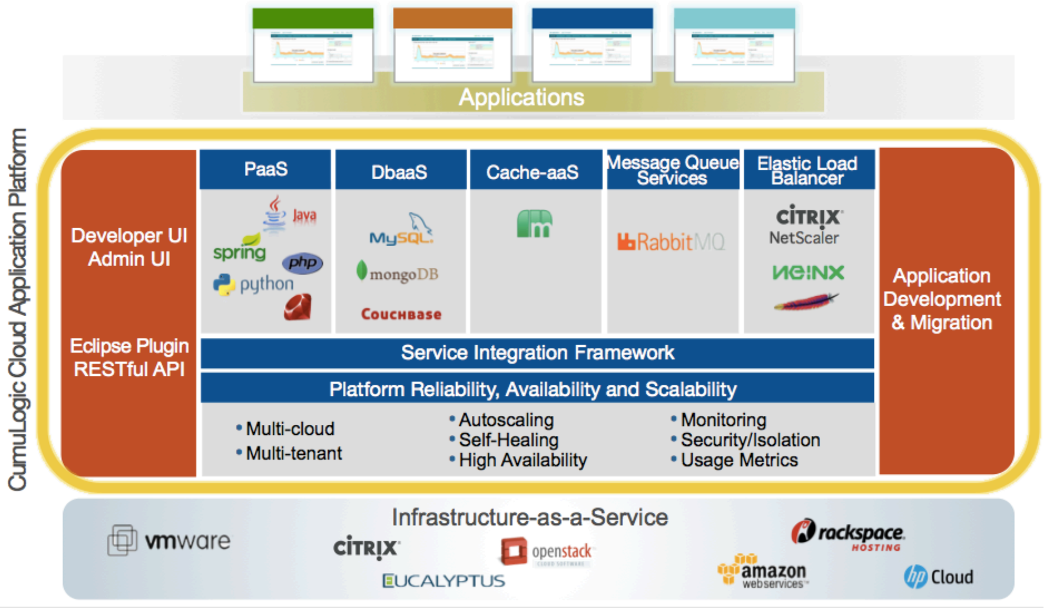
February 2013

Table of Contents

CumuLogic Cloud Application Platform

Welcome to the world's most comprehensive Cloud Application Platform!

CumuLogic is the only company that provides an easy-to-use Platform-as-a-Service (PaaS) with integrated cloud services, including Database-as-a-Service, Distributed Cache-as-a-Service, Elastic Load Balancer and more. CumuLogic PaaS with Cloud Services enable users to design flexible architectures for deploying and managing mission critical cloud applications.



This guides provides you with step-by-step instructions to get you started with each cloud service. Please refer to the latest **Release Notes** on our website for a complete list of services supported in the platform. CumuLogic releases additional cloud services, which may be available as updates and downloaded at cumulogic.com.

Terminology

- **Target Cloud** – This refers to one or more Infrastructure-as-a-Service clouds such as Amazon EC2, Rackspace Cloud Servers, HP Cloud or any other OpenStack- or CloudStack-powered clouds.
- **Services** – Application infrastructure services such as databases, load balancers, cache servers, etc. Each tier of infrastructure is delivered as a "service".
- **Application Platform** – Platform-as-a-Service, which is the core of CumuLogic solution. CumuLogic's platform is the combination of PaaS and cloud services.
- **Parameter Groups** – A set of parameters, which can be tweaked by users for a given service, instance to optimize that service instance for a specific type of application workload.
- **Access Groups** – Security or firewall configurations to control access to the service instance.
- **Automated Backups** – Backups of database instantiated by the platform, usually once in 24 hours for each running database instance.
- **Minor Updates** – Software patches that may pose no security exposure but enhances the performance or fixes some known issues in the service. These exclude any OS patches.
- **Autoscaling** – Functionality of the platform to horizontally scale application instances and load balancers in order to handle peak loads of the application.
- **Failover** – functionality of the platform to recover cloud service instances from any failures. This usually includes restarting a failed service or re-provisioning failed instances.

Infrastructure-as-a-Service (IaaS) Clouds Supported

CumuLogic's platform is integrated with the following IaaS clouds. Please refer to the latest release notes on cumulogic.com for the updated list.

Private Clouds

1. Apache CloudStack
2. Citrix CloudPlatform
3. Eucalyptus
4. Nebula
5. OpenStack
6. VMware vCloud Director
7. VMware vSphere virtualized environments

Public Clouds

Please refer to cumulogic.com for a list of public clouds supported.

Use CumuLogic Console to

1. Deploy Java and PHP applications on any cloud
2. Create fully managed MySQL database in the cloud
3. Create fully managed MongoDB database in the cloud
4. Launch Memcached cache cluster in the cloud

What You Need to Know to Deploy Your First Application

CumuLogic's platform makes it easy to quickly deploy applications. All you need to do is push application to the platform and the platform will provision the required services and resources to keep your application running. To deploy an application, you'll need the following information:

1. **Target IaaS cloud credentials**
   If you are using public IaaS clouds such as Amazon, Rackspace, HP Cloud, VMware vCloud or any other cloud based on Apache CloudStack, OpenStack or Citrix CloudPlatform, you'll need to register with one of those cloud providers and add your credentials to CumuLogic's platform under "My Account" on the top right hand corner of the **Dashboard**. For private clouds or VMware vSphere, you may already have this information in your "My Account." Contact your platform administrator if you don't have credentials.

2. **Application**
   You need your application artifacts packaged in war format for Java and Spring applications, and tar.gz format for PHP apps.

3. **CumuLogic deployment descriptor file** – You need to create a **cumulogic-app.xml** file or download a sample descriptor file from CumuLogic Documentation site. This is the manifest file for describing the application tiers of and for requesting containers, frameworks and services other than the default ones provisioned by the platform. See cumulogic-app.xml section for details.

To deploy an application, go to the **CumuLogic Application Dashboard**, select "Deploy App" and upload the artifacts and cumulogic-app.xml files, or you can use the API or command line tools.

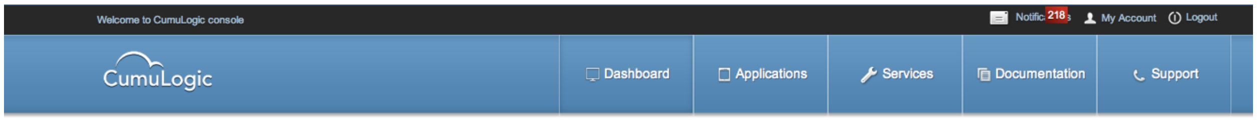


Figure 1 Dashboard

Below are the detailed steps to deploy your first application.

Register and Log into CumuLogic Console

Whether you are using a hosted or private version of CumuLogic's platform, you need to register and activate your account. You will receive a link to confirm your email and activate your account.

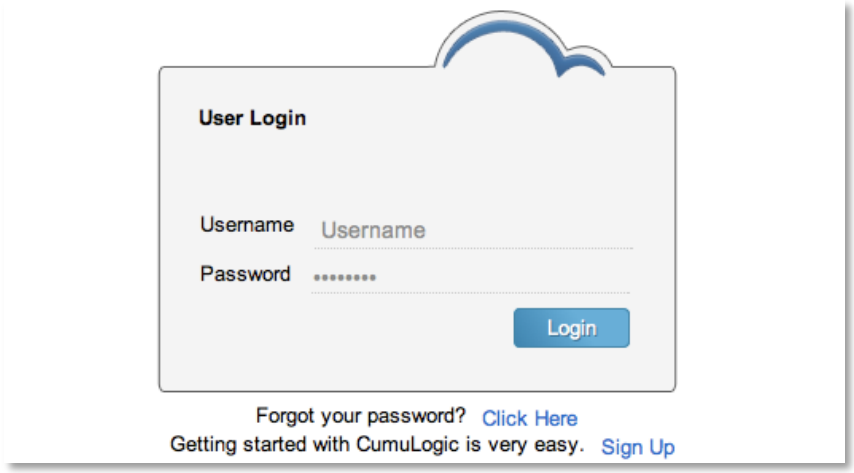
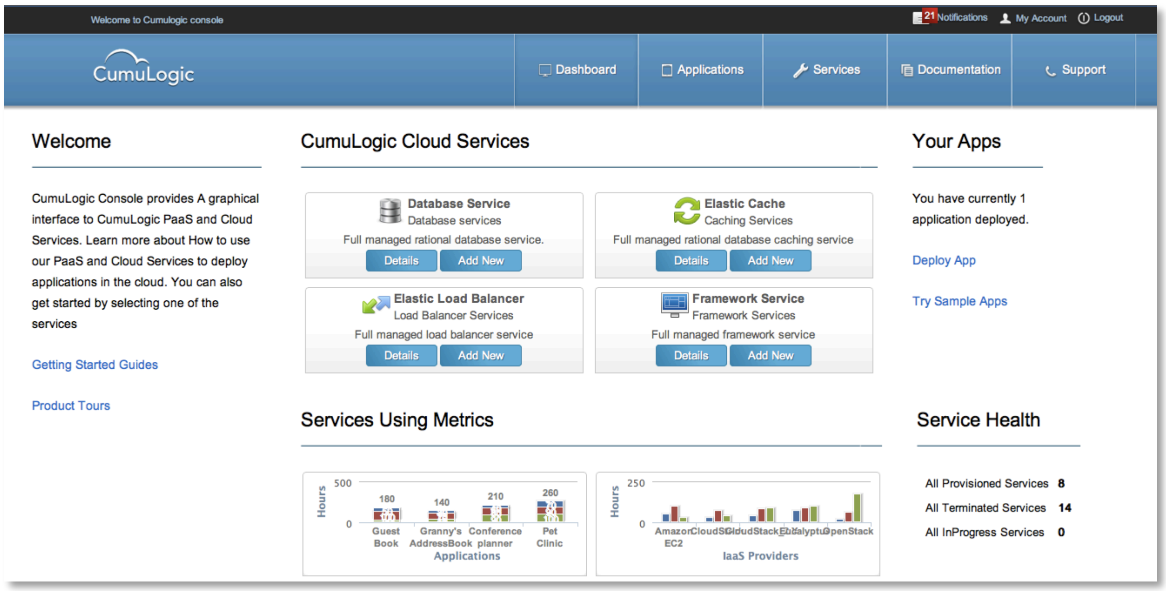

User Login
Username
Username
Password
••••••••
Login
Forgot your password? Click Here
Getting started with CumuLogic is very easy. Sign Up

Figure 2: CumuLogic Console Login Window

Dashboard

Your landing page is the CumuLogic Dashboard. The dashboard provides information about your applications, services that are provisioned, service health and visibility into the usage of your cloud resources. The dashboard also provides shortcuts to quickly deploy applications, try some sample applications or launch services directly from the dashboard. Please note that, the Dashboard on your platform may have been customized for your organization and may have different shortcuts available.



Welcome

CumuLogic Console provides A graphical interface to CumuLogic PaaS and Cloud Services. Learn more about How to use our PaaS and Cloud Services to deploy applications in the cloud. You can also get started by selecting one of the services

CumuLogic Cloud Services

Database Service
Database services
Full managed rational database service.

Elastic Cache
Caching Services
Full managed rational database caching service

Elastic Load Balancer
Load Balancer Services
Full managed load balancer service

Framework Service
Framework Services
Full managed framework service

Your Apps

You have currently 1 application deployed.

Services Using Metrics

Service Health

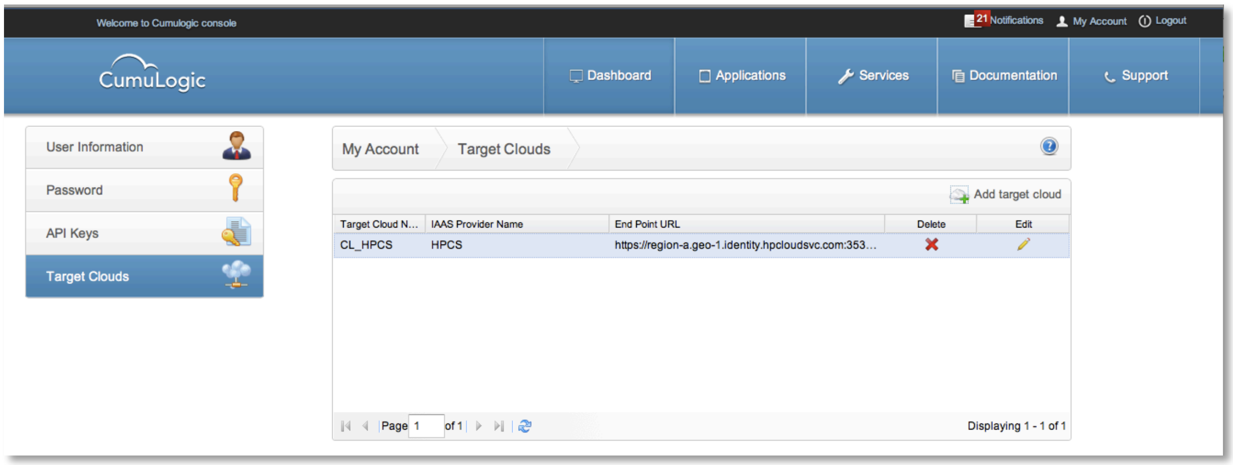
All Provisioned Services	8
All Terminated Services	14
All InProgress Services	0

Figure 3: CumuLogic Dashboard

Target IaaS Clouds

CumuLogic's platform allows you to deploy applications on multiple target Infrastructure-as-a-Service (IaaS) clouds to provision the applications. Target clouds can be one of the supported public clouds, including HP Cloud, Rackspace Cloud, Amazon EC2, Datapipe or any other clouds based on Apache CloudStack, Citrix CloudPlatform and OpenStack clouds. CumuLogic's platform also supports VMware vCloud and vSphere environments.

Based on the target cloud, you need to enter your credentials and access keys provided by your platform administrator or the Cloud Provider.



Target Cloud N...	IAAS Provider Name	End Point URL	Delete	Edit
CL_HPCS	HPCS	https://region-a.geo-1.identity.hpcloudsvc.com:353...	✗	✎

Figure 4: Adding target cloud credentials in the My Account screen

NOTE: For private clouds, the administrator may provide the target cloud credentials in your account, so you won't need to choose the target cloud(s).

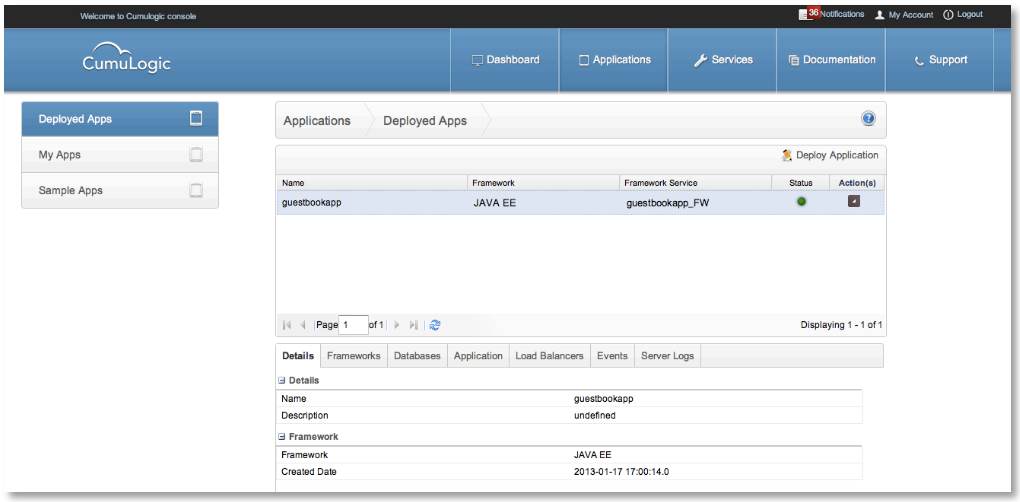
Deploy Applications

Once you have added credentials for at least one target cloud, you can deploy your applications following these steps:

I. Select the **Deploy Apps** or **Deploy Sample Apps** shortcut from the Dashboard or
II. Select the **Applications Menu** from the top bar

Under the **Applications Tab**, there are three views available:

a. **Deployed Apps** – This is the list of all the applications currently deployed in your account. The list will be empty if you haven't deployed any apps yet.



Name	Framework	Framework Service	Status	Action(s)
guestbookapp	JAVA EE	guestbookapp_FW	●	

Details

Name	guestbookapp
Description	undefined

Framework

Framework	JAVA EE
Created Date	2013-01-17 17:00:14.0

Figure 5: Deployed Apps

In figure 4, the **Details** tabs provide details about the application and the services such as databases, load balancers, etc. The information is provided as read-only information.

The status field shows the current status of the application. The status is "green" if the application is deployed and running and "red" when the application has been un-deployed or has encountered errors.

Refer to section "How to Deploy Apps" below for information on deploying new applications on the platform.

b. **My Apps** – List all your applications deployed or uploaded to the platform.

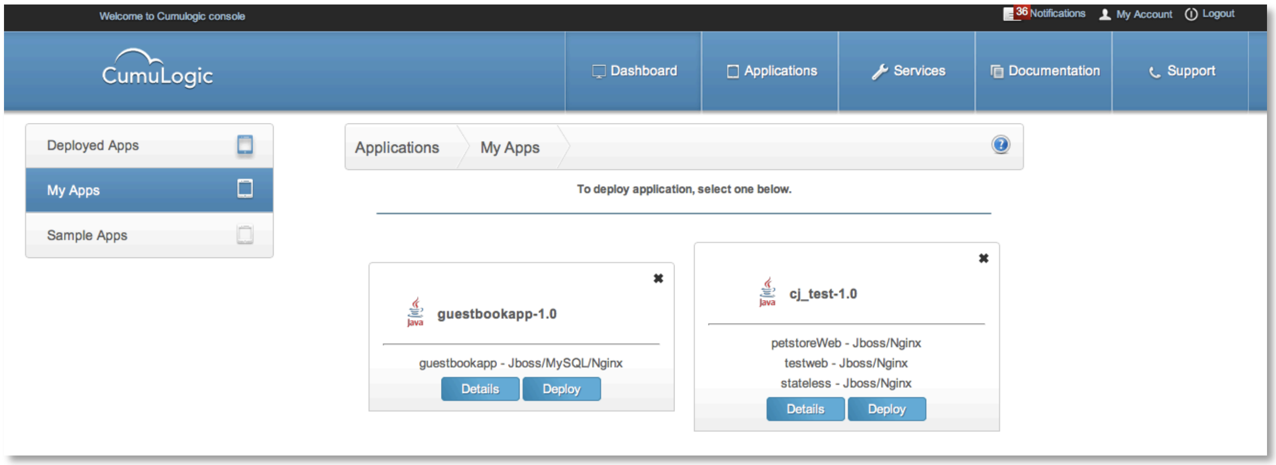


Figure 6: My Apps

My Apps is similar to having your personal **App Store**. All the applications you deploy to the platform are uploaded to the App Store with the application artifacts; the application services manifest file (cumulogic-app.xml), the database schema and any bootstrap data.

My Apps gives allows you to deploy applications on any target cloud with a single click.

c. **Sample Apps** – Some sample applications provided in CumuLogic platform. Use these sample apps to get a feel of the platform and familiarize yourself with the features of the platform.

You can download the source code of sample applications to understand the platform features and how applications can source information, such as datasource information, IP addresses of cache nodes or database instances using environment variables.

Refer to the section on "How to Deploy Apps" below for information on deploying new applications to the platform.

Deploy Your First Application

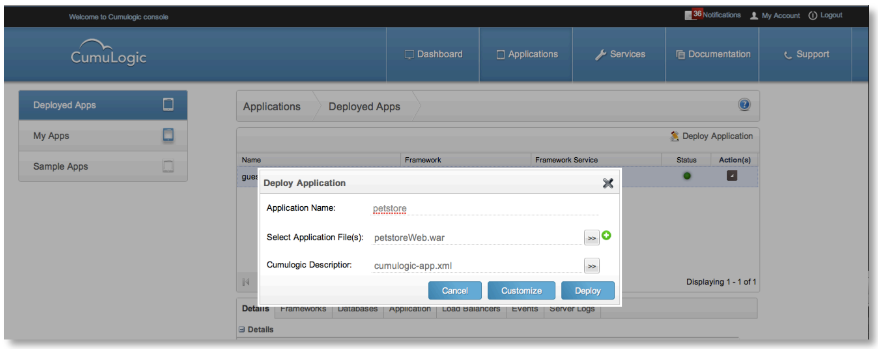
To deploy an application, you need the application artifacts (package them as war files for Java Web apps, Jar files for JVM apps and tar.gz format for PHP applications). Refer to the "How to Package Your Cloud Apps" section for details.

You can optionally create a *CumuLogic deployment descriptor file,* cumulogic-app.xml and define the application datasource information and infrastructure services that your application requires. For example, if you wish to deploy your application using the JBoss container instead of the default Tomcat container, or if you wish to front-end your application with Apache web server or load balancer, you can define these requirements in the descriptor file. Refer to the "CumuLogic Deployment Descriptor File Schema" for details on all supported configuration elements.

Please note that in the absence of the **cumulogic-app.xml** file, all default services are provisioned for your application. In a private cloud scenario, the platform administrator specifies the default services. Please check with your administrator for details.

Petstore Application Example

Let's look at how to deploy a sample application. On the **Deploy Apps** menu, select "Deploy Application" on the top right hand corner of the table.



Deploy Application

Application Name: petstore

Select Application File(s): petstoreWeb.war

Cumulogic Descriptior: cumulogic-app.xml

Figure 7

Provide the following information:

I. Application name – name of the application.
II. Application files – you can upload one or more war files (if the application has different modules, you can deploy them as individual war files on different application contexts. Each war file must have its own web.xml file).
III. CumuLogic Descriptor – cumulogic-app.xml file.

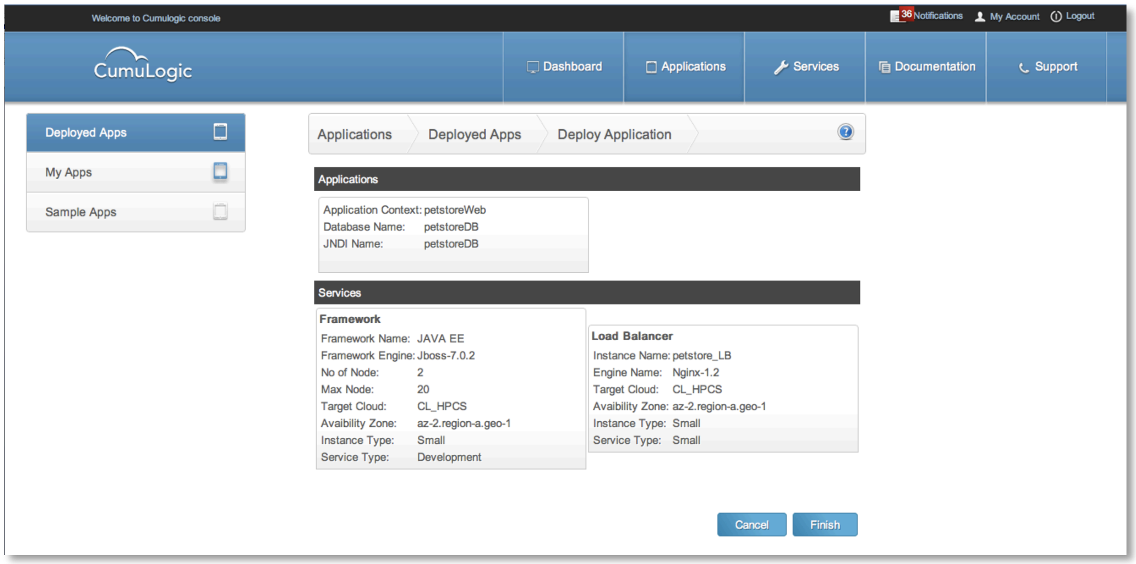
To deploy an application using the default stack, select **Deploy**. Or if you prefer to customize the application datasource information, database type app or context names, you can select the "**Customize"** option on this screen.

Deploying applications using the default stack:

In order to identify the datasource and application context information,

CumuLogic's platform will introspect the web.xml, app-context.xml or jboss.xml file if available in the war files.

Just select **Deploy** and the following screen with details about the infrastructure services will be displayed. These are the values detected from web.xml or other deployment descriptor files.



Applications

Application Context: petstoreWeb
Database Name: petstoreDB
JNDI Name: petstoreDB

Services

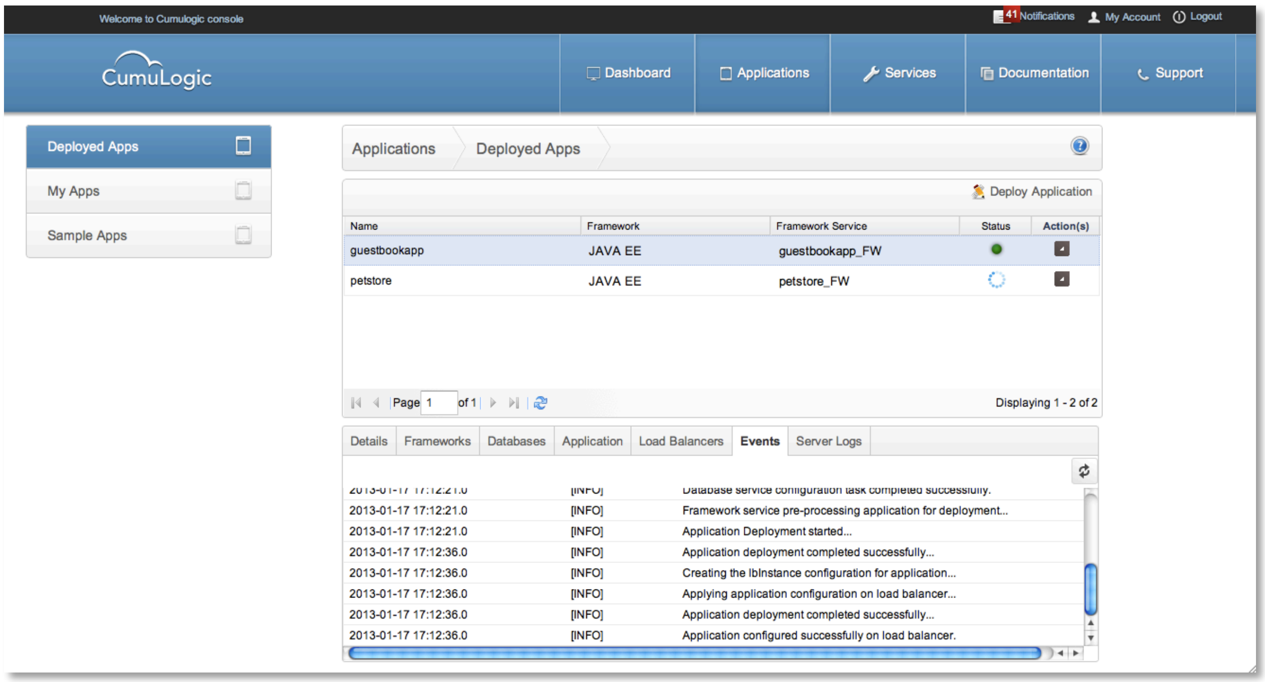
Framework
Framework Name: JAVA EE
Framework Engine: Jboss-7.0.2
No of Node: 2
Max Node: 20
Target Cloud: CL_HPCS
Avaibility Zone: az-2.region-a.geo-1
Instance Type: Small
Service Type: Development

Load Balancer
Instance Name: petstore_LB
Engine Name: Nginx-1.2
Target Cloud: CL_HPCS
Avaibility Zone: az-2.region-a.geo-1
Instance Type: Small
Service Type: Small

Figure 8: Deploy Application

Select **Deploy** to confirm and deploy the application. The platform will provision the required services, configure the stack and deploy the application. You can check the **Details** tabs for information on all the resources allocated to your application.

To track the progress of the application deployment, please refer to the **Events** tab as illustrated below:



Name	Framework	Framework Service	Status	Action(s)
guestbookapp	JAVA EE	guestbookapp_FW	●	
petstore	JAVA EE	petstore_FW		

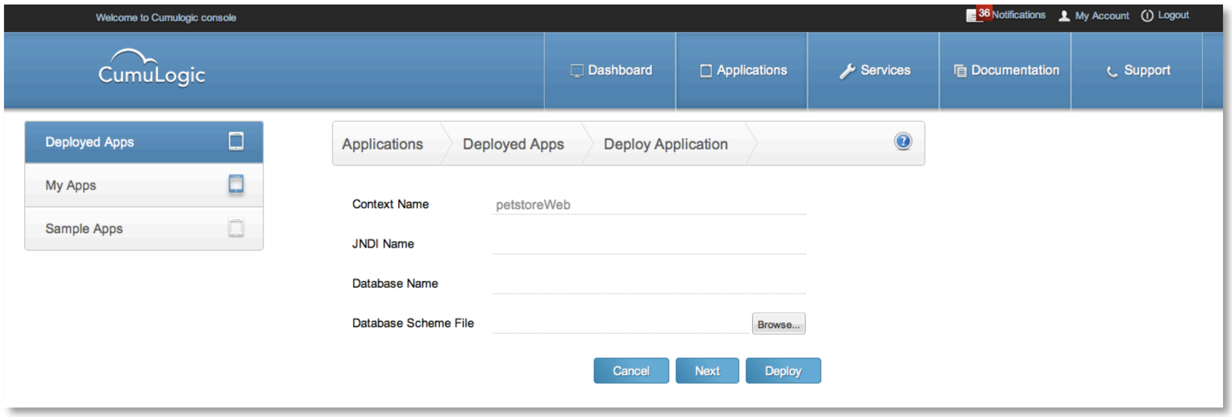
```
2013-01-17 17:12:21.0    [INFO]    Database service configuration task completed successfully.
2013-01-17 17:12:21.0    [INFO]    Framework service pre-processing application for deployment...
2013-01-17 17:12:21.0    [INFO]    Application Deployment started...
2013-01-17 17:12:36.0    [INFO]    Application deployment completed successfully...
2013-01-17 17:12:36.0    [INFO]    Creating the lbInstance configuration for application...
2013-01-17 17:12:36.0    [INFO]    Applying application configuration on load balancer...
2013-01-17 17:12:36.0    [INFO]    Application deployment completed successfully...
2013-01-17 17:12:36.0    [INFO]    Application configured successfully on load balancer.
```

Figure 9: Application Deployment Events

TIP: Click on the **Applications** tab to find the application URL.

Deploying applications using custom stack settings:

To customize the database type, datasource name, database schema and any services, such as frameworks, select the **Customize** option and provide information on successive screens.



Context Name: petstoreWeb

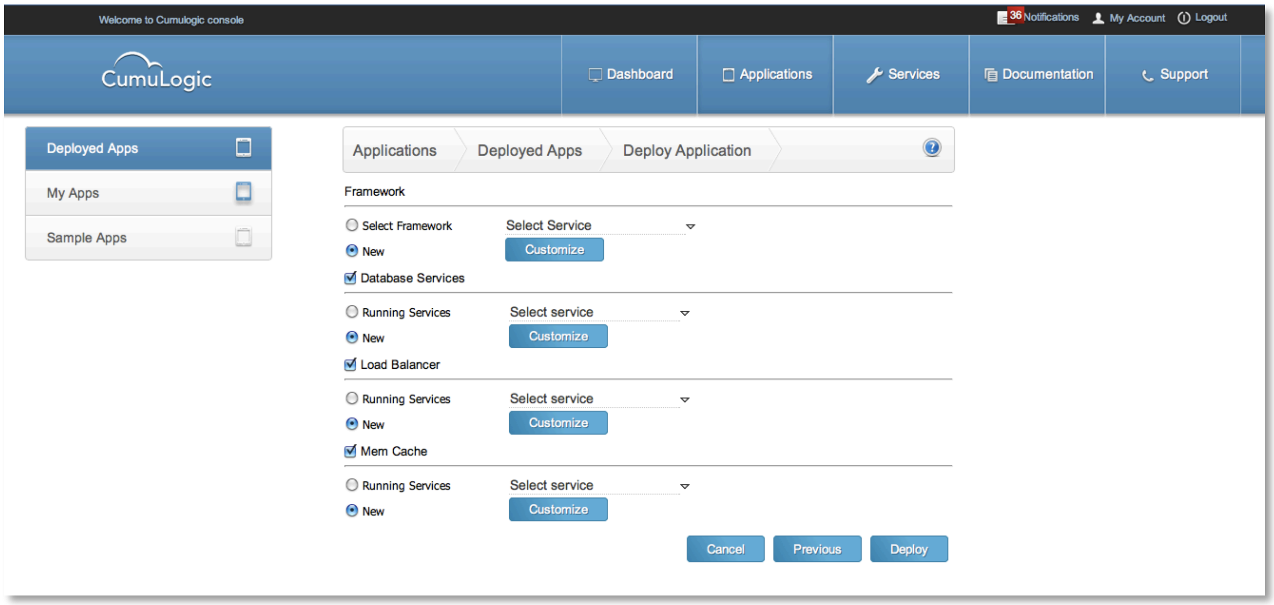
JNDI Name:

Database Name:

Database Scheme File:

Figure 10: Customize Application Settings

The next screen allows you to customize containers, frameworks and other services.



Framework

( ) Select Framework — Select Service
(●) New — Customize

☑ Database Services

( ) Running Services — Select service
(●) New — Customize

☑ Load Balancer

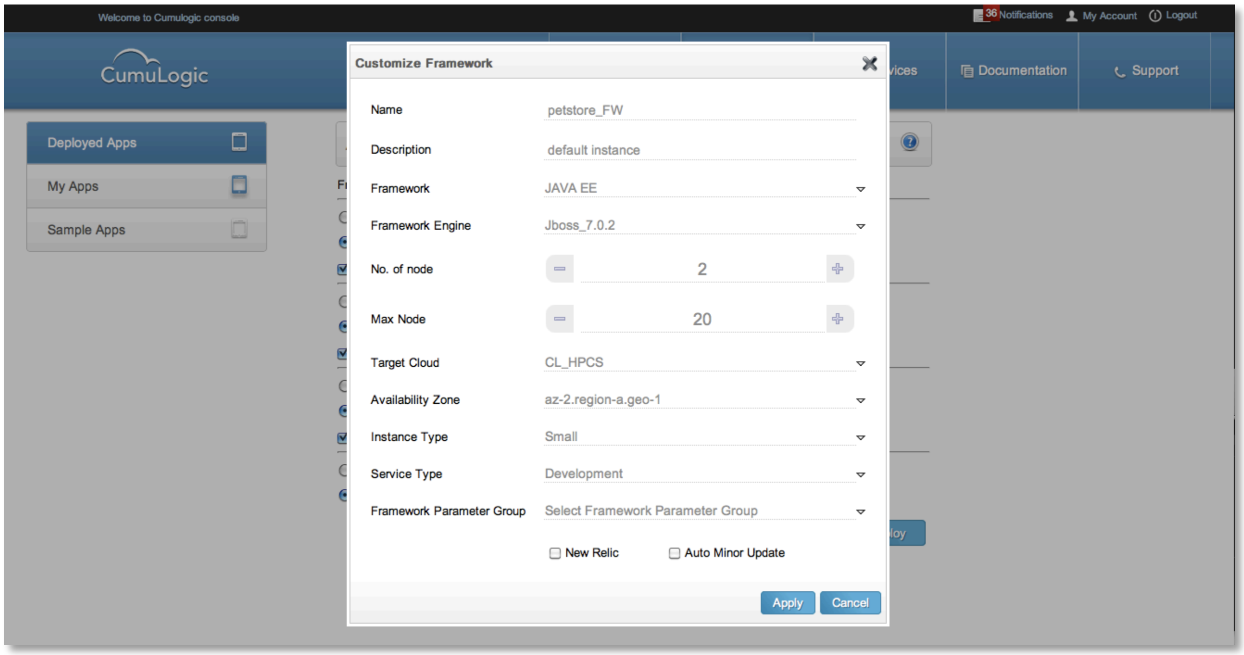
( ) Running Services — Select service
(●) New — Customize

☑ Mem Cache

( ) Running Services — Select service
(●) New — Customize

Figure 11: Customize Services

Select **Next** to choose a different framework option.



Customize Framework

Name	petstore_FW
Description	default instance
Framework	JAVA EE
Framework Engine	Jboss_7.0.2
No. of node	2
Max Node	20
Target Cloud	CL_HPCS
Availability Zone	az-2.region-a.geo-1
Instance Type	Small
Service Type	Development
Framework Parameter Group	Select Framework Parameter Group

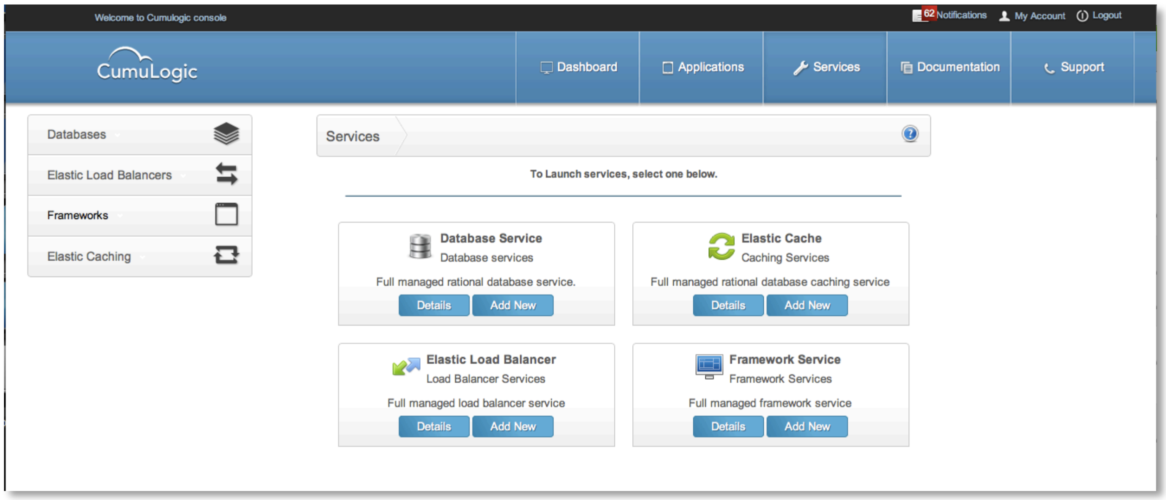
☐ New Relic ☐ Auto Minor Update

Figure 12: Customize Frameworks

Select the **Deploy** option to deploy applications using the customized stack and settings.

CumuLogic Services Dashboard

The services dashboard shows a listing of all application infrastructure services available on the platform.



To Launch services, select one below.

Full managed rational database service.

Full managed rational database caching service

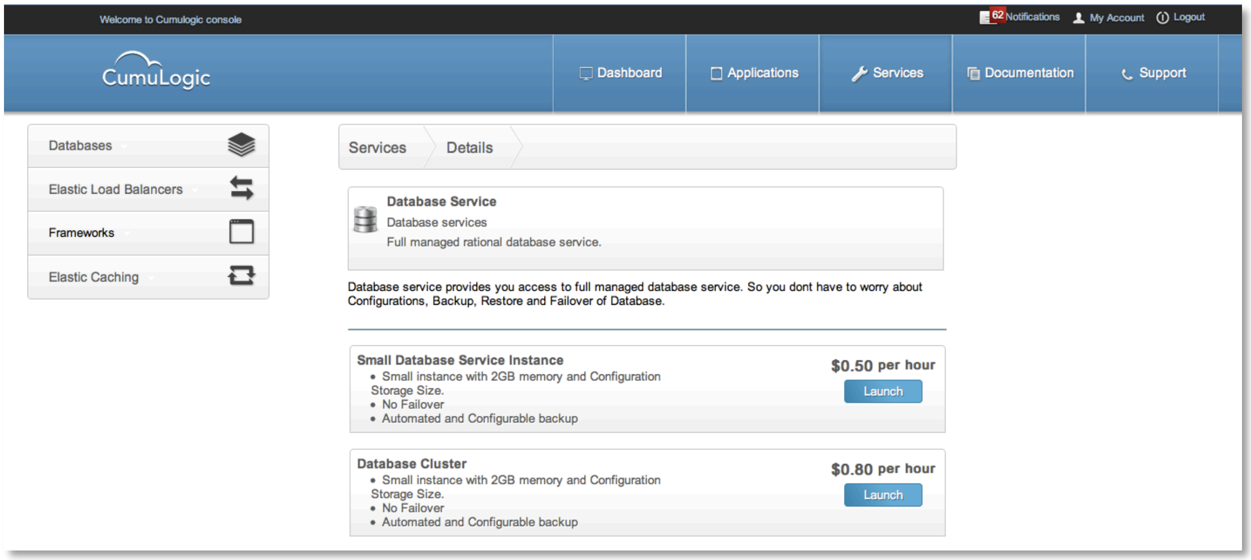
Full managed load balancer service

Full managed framework service

Figure 13: Cloud Services

Details of each service are available along with service description, size of the service instance and in some cases, the cost of the service runtime.

For example, below is a screenshot of a database service, which provides the price of various service options and description of what is included in the service.



Database Service
Database services
Full managed rational database service.

Database service provides you access to full managed database service. So you dont have to worry about Configurations, Backup, Restore and Failover of Database.

Small Database Service Instance — **$0.50 per hour**
- Small instance with 2GB memory and Configuration Storage Size.
- No Failover
- Automated and Configurable backup

Database Cluster — **$0.80 per hour**
- Small instance with 2GB memory and Configuration Storage Size.
- No Failover
- Automated and Configurable backup

Figure 14: Database Cloud Services details

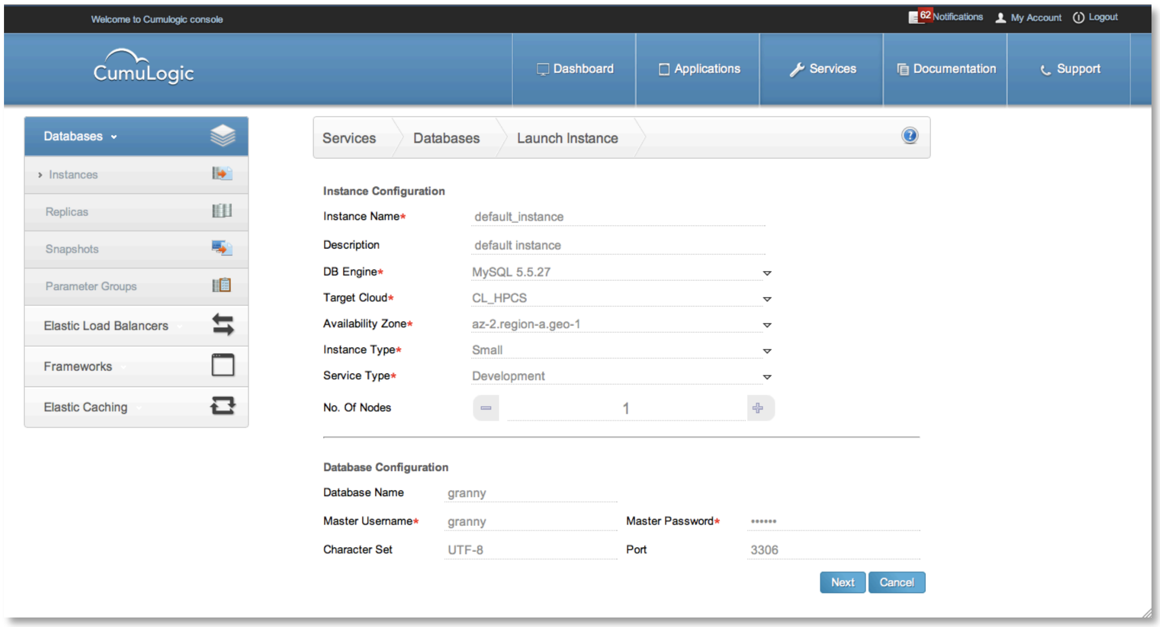
Please note that your platform may have different set of services, service offerings and pricing.

CumuLogic Database Service

Use the Database Service to launch the fully managed MySQL database, MySQL Cluster or MongoDB database in any target cloud.

Launching a MySQL Database Instance

You can launch a MySQL database instance or MySQL Cluster from the **Services Dashboard** by choosing a standard configuration as illustrated below or you can select "**Launch Instance**" and customize the parameters, including target cloud, database or storage size, and other backup or maintenance options.



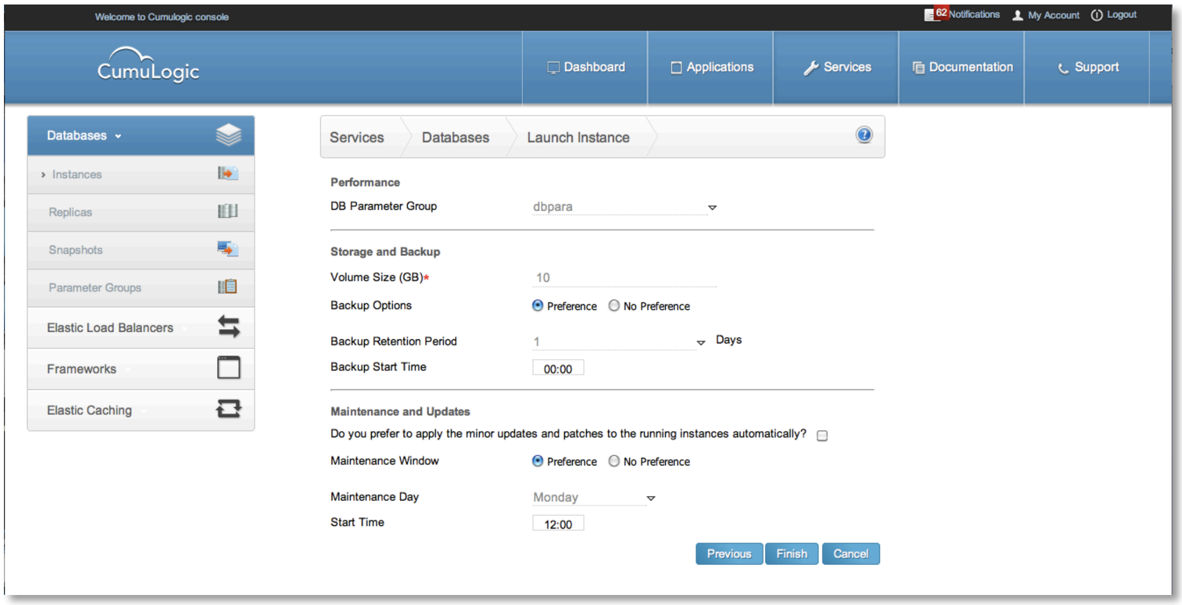
Instance Configuration

Instance Name*	default_instance
Description	default instance
DB Engine*	MySQL 5.5.27
Target Cloud*	CL_HPCS
Availability Zone*	az-2.region-a.geo-1
Instance Type*	Small
Service Type*	Development
No. Of Nodes	1

Database Configuration

Database Name	granny		
Master Username*	granny	Master Password*	••••••
Character Set	UTF-8	Port	3306

Figure 15: Launch DB Instance



Performance

DB Parameter Group: dbpara

Storage and Backup

Volume Size (GB)*: 10

Backup Options: [✓ Preference] No Preference

Backup Retention Period: 1 Days

Backup Start Time: 00:00

Maintenance and Updates

Do you prefer to apply the minor updates and patches to the running instances automatically? ☐

Maintenance Window: [✓ Preference] No Preference

Maintenance Day: Monday

Start Time: 12:00

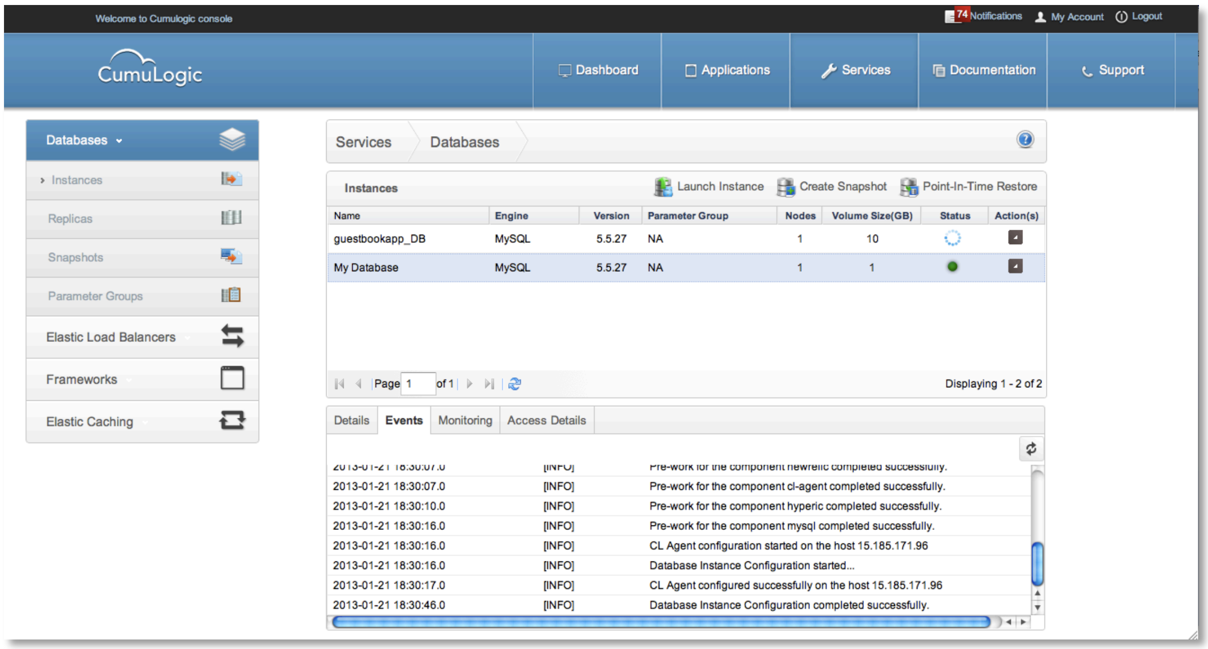
Figure 16: Launch DB Instance Options

Below is a description of each field and the default values:

Data Field	Description	Default
Instance Name	Name of your database instance	default_instance
Description	Describe your database instance	default_instance
DB Engine	Choose database engine from one of the supported types such as MySQL or MySQL Cluster	MySQL 5.x
Target Cloud	Target IaaS cloud to provision the database on	Default cloud from user's My Account Profile
Availability Zone	Availability zone of target IaaS cloud	Default Availability Zone from user's target cloud
Instance Type	Size of database instance	Small. Usually 2GB RAM
Service Type	Tag the lifecycle of database instance	Development
No of nodes	Number of nodes for database instance. MySQL cluster requires at least one master node and two data nodes	1 for MySQL 5.x and 3 for MySQL cluster
DB Parameter	Optimized Parameter Group for your	Default Parameter Group

Data Field	Description	Default
group	workload.	
Volume size	Storage size allocated to the database instance	None. Must be specified by the user
Backup Retention period	How long to retain automated backups	One day
Backup Start Time	Start time every day to initiate automated backups	Midnight in user's local time zone
Maintenance Day	Day of the week when automated updates and minor patches can be applied to running instance	Monday
Start Time	Start time of two-hour window when automated updates and minor patches can be applied	Midnight in user's local time zone

Once the database instance has been provisioned, you should see the listing and status as shown below.



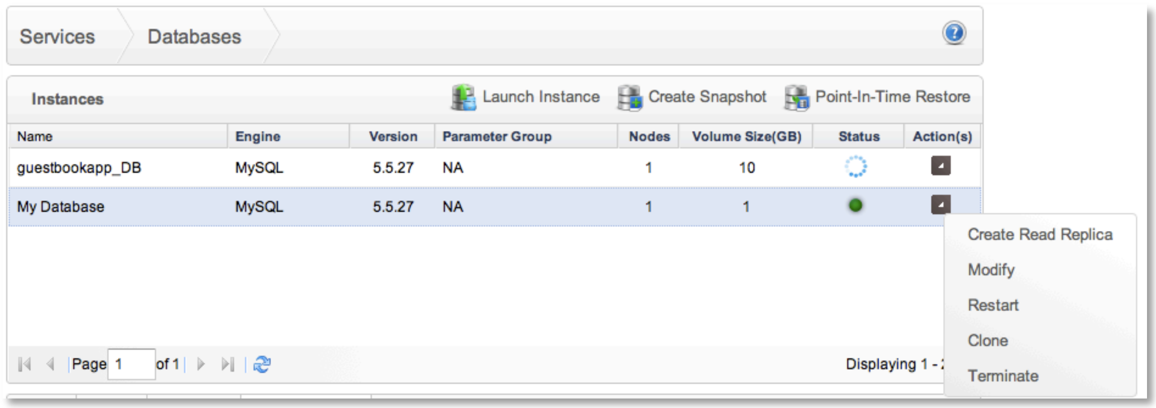
Name	Engine	Version	Parameter Group	Nodes	Volume Size(GB)	Status	Action(s)
guestbookapp_DB	MySQL	5.5.27	NA	1	10		
My Database	MySQL	5.5.27	NA	1	1	●	

```
2013-01-21 18:30:07.0    [INFO]    Pre-work for the component newrelic completed successfully.
2013-01-21 18:30:07.0    [INFO]    Pre-work for the component cl-agent completed successfully.
2013-01-21 18:30:10.0    [INFO]    Pre-work for the component hyperic completed successfully.
2013-01-21 18:30:16.0    [INFO]    Pre-work for the component mysql completed successfully.
2013-01-21 18:30:16.0    [INFO]    CL Agent configuration started on the host 15.185.171.96
2013-01-21 18:30:16.0    [INFO]    Database Instance Configuration started...
2013-01-21 18:30:17.0    [INFO]    CL Agent configured successfully on the host 15.185.171.96
2013-01-21 18:30:46.0    [INFO]    Database Instance Configuration completed successfully.
```

Figure 17: MySQL DB Instance ready for use

Click on the **Access Details** tab to see the access URL and the credentials for connecting to the database using your preferred client, such as Workbench or MyPHPadmin.

To create a **Read-Only replica** of an existing database instance or modify any configuration parameters, select **Actions** on the database as shown below.



Name	Engine	Version	Parameter Group	Nodes	Volume Size(GB)	Status	Action(s)
guestbookapp_DB	MySQL	5.5.27	NA	1	10		
My Database	MySQL	5.5.27	NA	1	1	●	

Figure 18: MySQL DB Instance actions

Creating a Read-Only Replica

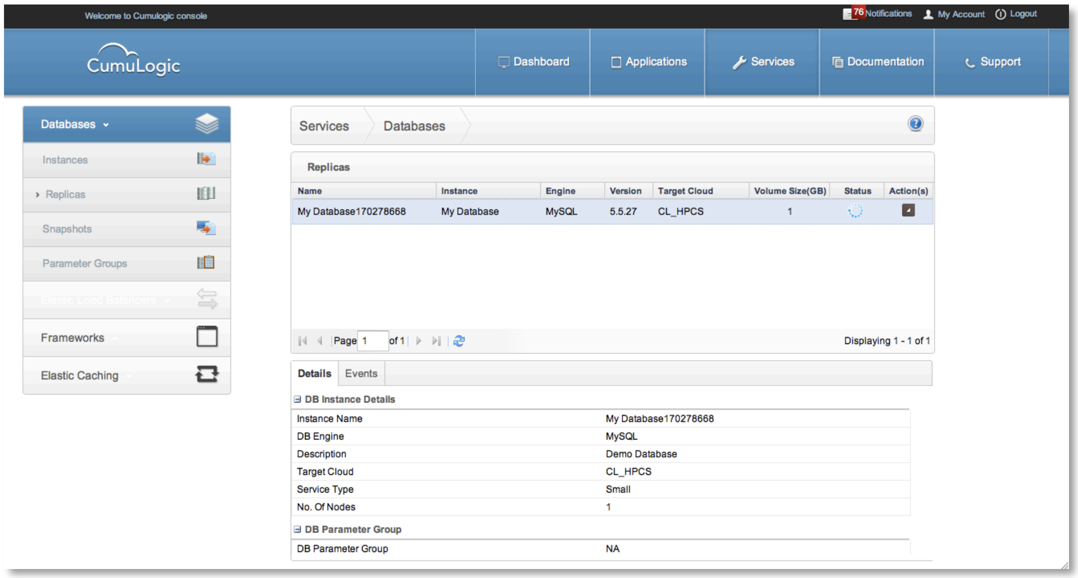
To scale your application for handling transient peak loads, you can launch one or more read-only replicas of an existing running database instance.

You can configure your application to perform read operations from one or more replicas and use a master database instance for write operations. Please note that replication happens one way only – from master to the read-only replicas, so avoid any write operations to the replicas, which may not be synced to the master database.

Read-only replicas can also be useful for building redundancy and fault-tolerance in your architecture. In case of failures on master database instance, you can promote a replica as master database instance.

To create a Read-only Replica, select the "**Create Replica**" action as shown below. This action will lock the master database instance for any writes, take a snapshot of existing database and transaction logs and setup the synchronization between the master and slave replica. Please note that the master database instance will be unavailable for writes until the replication is completed. The amount of time it takes to create a replica depends on the size of the database.

To view the list of available replicas, select **Replicas** from the left menu bar.



Replicas

Name	Instance	Engine	Version	Target Cloud	Volume Size(GB)	Status	Action(s)
My Database170278668	My Database	MySQL	5.5.27	CL_HPCS	1		

DB Instance Details

Instance Name	My Database170278668
DB Engine	MySQL
Description	Demo Database
Target Cloud	CL_HPCS
Service Type	Small
No. Of Nodes	1

DB Parameter Group

DB Parameter Group	NA

Figure 19: Creating Read-only replicas

Select the "Access Details" tab for information on access credentials to configure your applications to use read-only replicas.

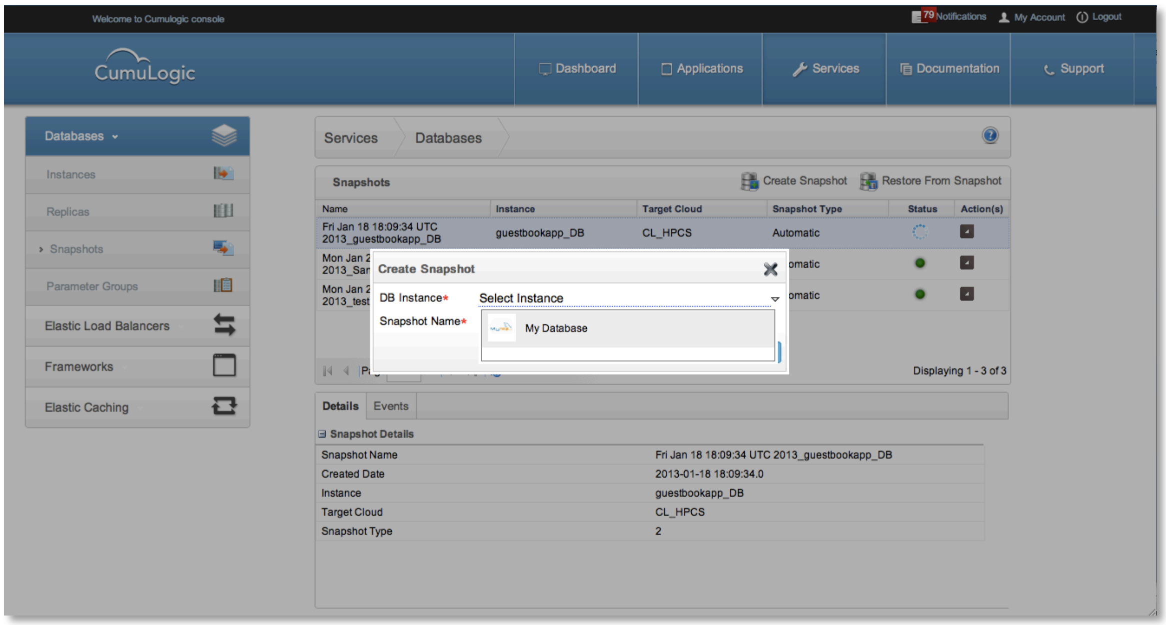
Taking a Snapshot of a Running Database Instance

If you are planning to make any major schema or data changes to your running database instance, you may consider storing the latest snapshot.

Select **Snapshots** from the left menu bar to take a full database snapshot of an existing database instance. The system will suspend operations on the database instance for the duration of time it takes to take a complete snapshot.

The Snapshot listing also displays all the available database backup snapshots, which are initiated automatically by the database service.

You can launch a new database instance using any of the existing snapshots.



Snapshots

Name	Instance	Target Cloud	Snapshot Type	Status	Action(s)
Fri Jan 18 18:09:34 UTC 2013_guestbookapp_DB	guestbookapp_DB	CL_HPCS	Automatic		
Mon Jan 2 2013_San			omatic	●	
Mon Jan 2 2013_test			omatic	●	

Displaying 1 - 3 of 3

Create Snapshot

DB Instance* Select Instance

Snapshot Name* My Database

Snapshot Details

Snapshot Name	Fri Jan 18 18:09:34 UTC 2013_guestbookapp_DB
Created Date	2013-01-18 18:09:34.0
Instance	guestbookapp_DB
Target Cloud	CL_HPCS
Snapshot Type	2

Figure 20: Taking a database snapshot

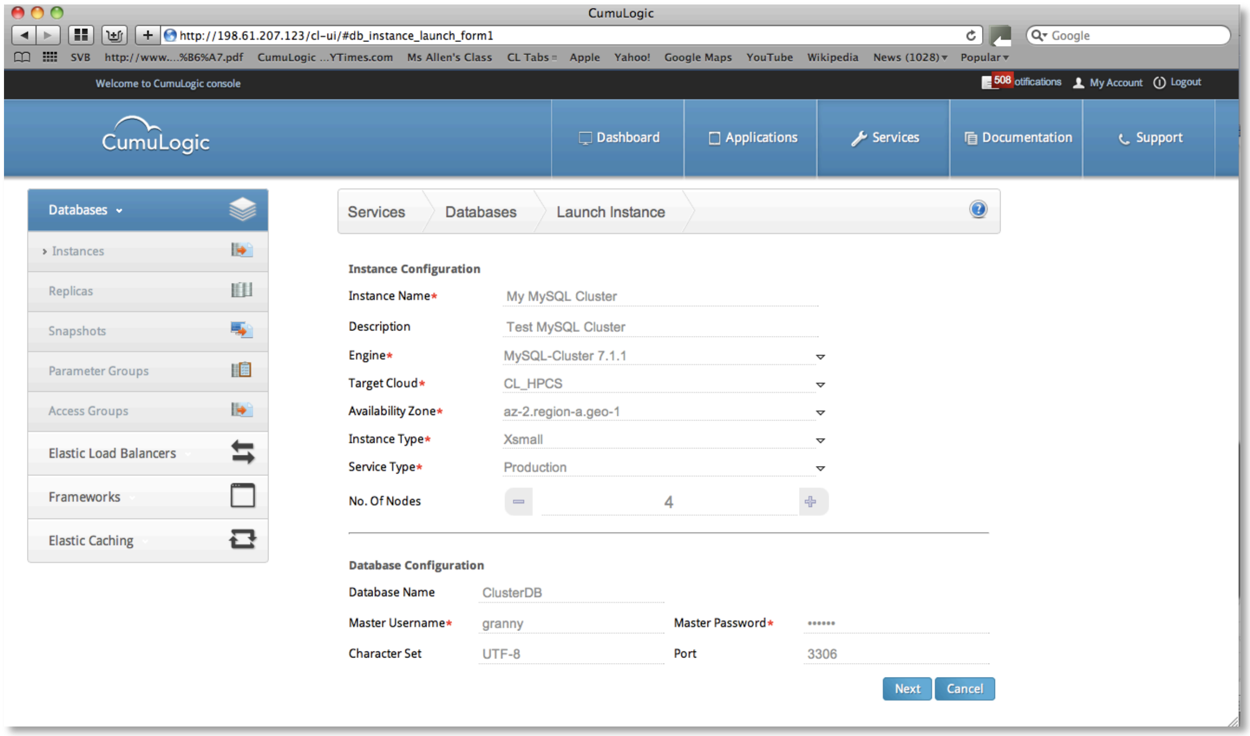
Cloning a Database Instance

In situations where you want to re-create an application environment or promote a development environment to production environment, you can clone an existing database instance by using the **Clone** option from the actions menu on the database instance screen.

Cloning a database instance creates a new database instance with a snapshot of the current database. Please note that any reads and writes on the master database are suspended until a clone is fully created. The difference between a clone and a read-only replica is that a clone is a copy of the existing running database instance and is available for read and write operations, whereas, read-only a replica doesn't have write operations enabled.

Launching a MySQL Cluster

To launch a redundant database service, you may want to use MySQL Cluster. MySQL cluster requires at least two nodes, one for the master database and others for data nodes.



Instance Configuration

Instance Name*	My MySQL Cluster
Description	Test MySQL Cluster
Engine*	MySQL-Cluster 7.1.1
Target Cloud*	CL_HPCS
Availability Zone*	az-2.region-a.geo-1
Instance Type*	Xsmall
Service Type*	Production
No. Of Nodes	4

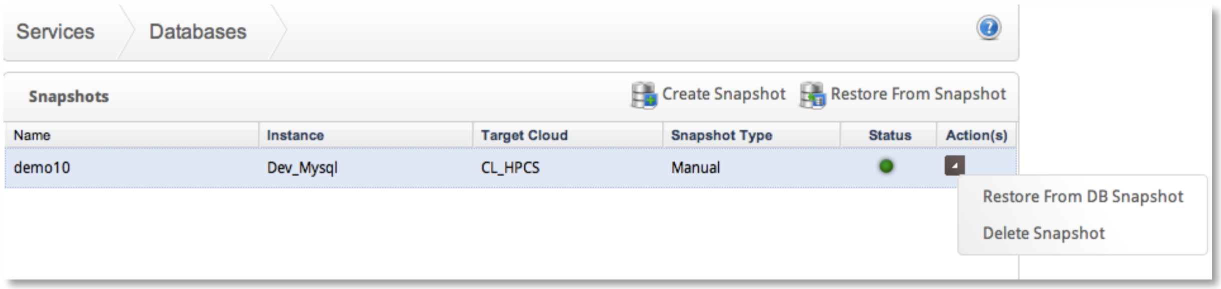
Database Configuration

Database Name	ClusterDB		
Master Username*	granny	Master Password*	••••••
Character Set	UTF-8	Port	3306

Figure 21: Launching MySQL Cluster

Restoring From Database Snapshots

To restore a database snapshot, go to the **Restore From Snapshot** option on **Snapshots** and provide information, such as database instance, size, etc.



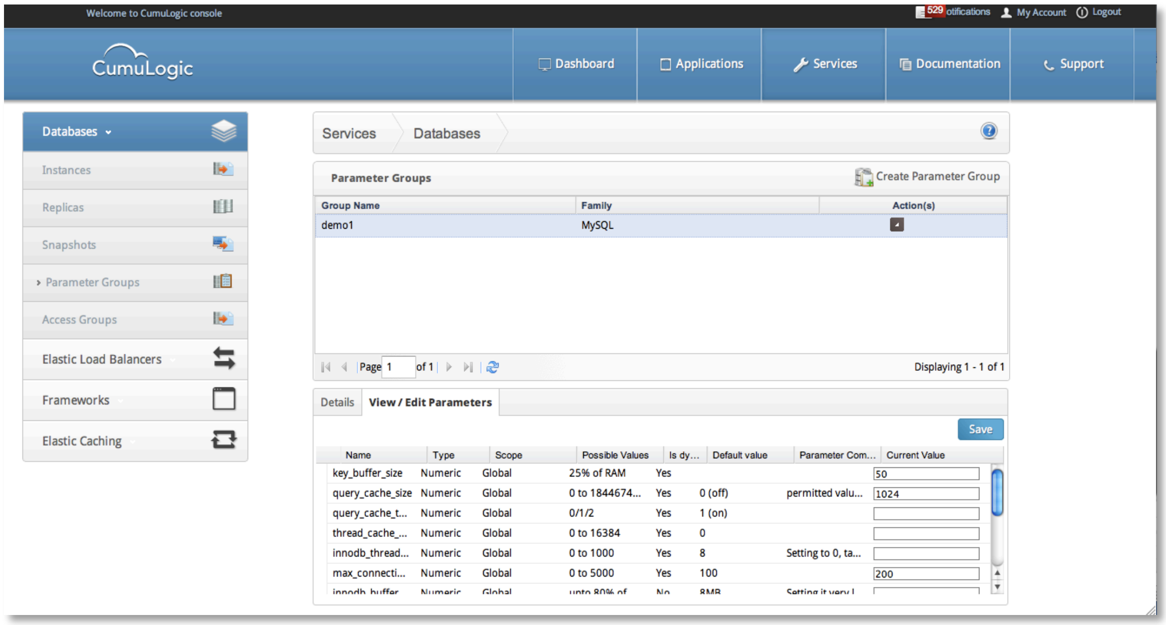
Name	Instance	Target Cloud	Snapshot Type	Status	Action(s)
demo10	Dev_Mysql	CL_HPCS	Manual	●	

Figure 22: Launching MySQL Cluster

Restore from Snapshot allows users to launch a new database instance of a different size than the source database snapshot instance. This process also allows users to scale their database instance to handle additional workloads.

Creating Parameter Groups

Parameter groups are a set of configuration and optimization parameters that CumuLogic database service exposes for users to modify to meet specific workload requirements. For instance, you may want to optimize MySQL parameters to tune your database for read-heavy workloads or increase the query cache size. If you are familiar with optimization techniques for databases, you can create your own Parameter Groups and change the values for specific parameters and apply them to any database instance.



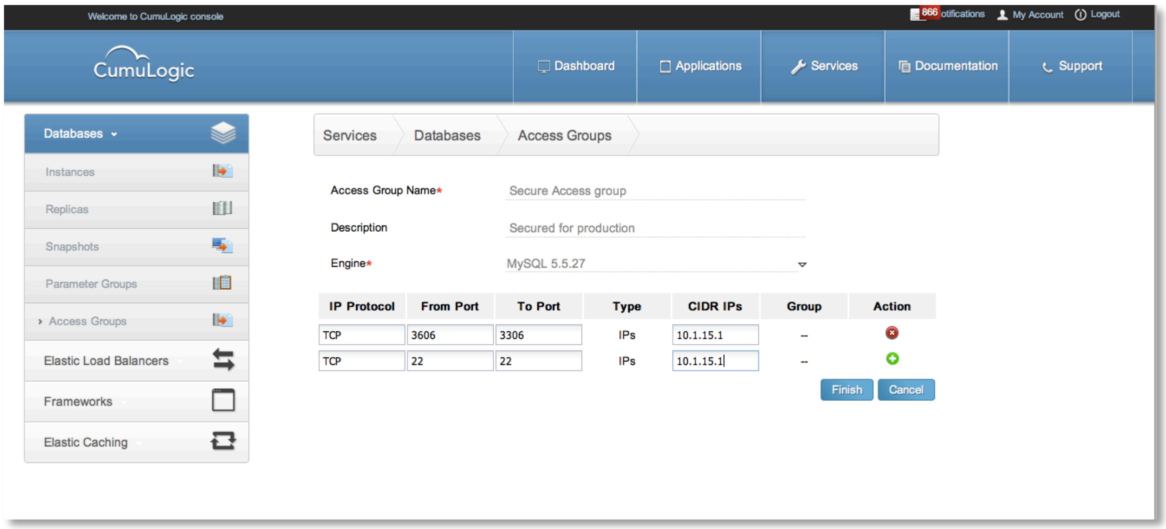
Group Name	Family	Action(s)
demo1	MySQL	

Name	Type	Scope	Possible Values	Is dy…	Default value	Parameter Com…	Current Value
key_buffer_size	Numeric	Global	25% of RAM	Yes			50
query_cache_size	Numeric	Global	0 to 1844674…	Yes	0 (off)	permitted valu…	1024
query_cache_t…	Numeric	Global	0/1/2	Yes	1 (on)		
thread_cache_…	Numeric	Global	0 to 16384	Yes	0		
innodb_thread…	Numeric	Global	0 to 1000	Yes	8	Setting to 0, ta…	
max_connecti…	Numeric	Global	0 to 5000	Yes	100		200
innodb_buffer	Numeric	Global	upto 80% of	No	8MB	Setting it very l	

Figure 23: Creating Parameter Groups

Access Groups

Access groups such as Security Groups provide firewall-like functionality to secure database instances. By default, access to database instances via any protocol is blocked. In order to connect to the database instances, users can create and configure Access Groups to allow access from specific IP addresses or servers. For example, to allow applications running on HOST A to be able to connect to a database instance, you can create an Access Group to open access from Host A as shown below.



Access Group Name* Secure Access group

Description Secured for production

Engine* MySQL 5.5.27

IP Protocol	From Port	To Port	Type	CIDR IPs	Group	Action
TCP	3606	3306	IPs	10.1.15.1	--	✖
TCP	22	22	IPs	10.1.15.1	--	✚

Figure 24: Create and Configure Access groups

In the example above, the Access Group called "Secure Access group" is used for production systems and opens access to the MySQL database instance only from 10.1.15.1 through port 3306 and port 22 for SSH.

Access Groups can be edited and applied to running database instances in real time without impacting database operations.

Load Balancer

CumuLogic Load Balancer is based on the open source Ngnix load balancer and supports HTTP and HTTPs protocols. It can be used to load balance applications deployed on CumuLogic PaaS and by default, a load balancer instance is configured to load balance applications deployed on CumuLogic PaaS.

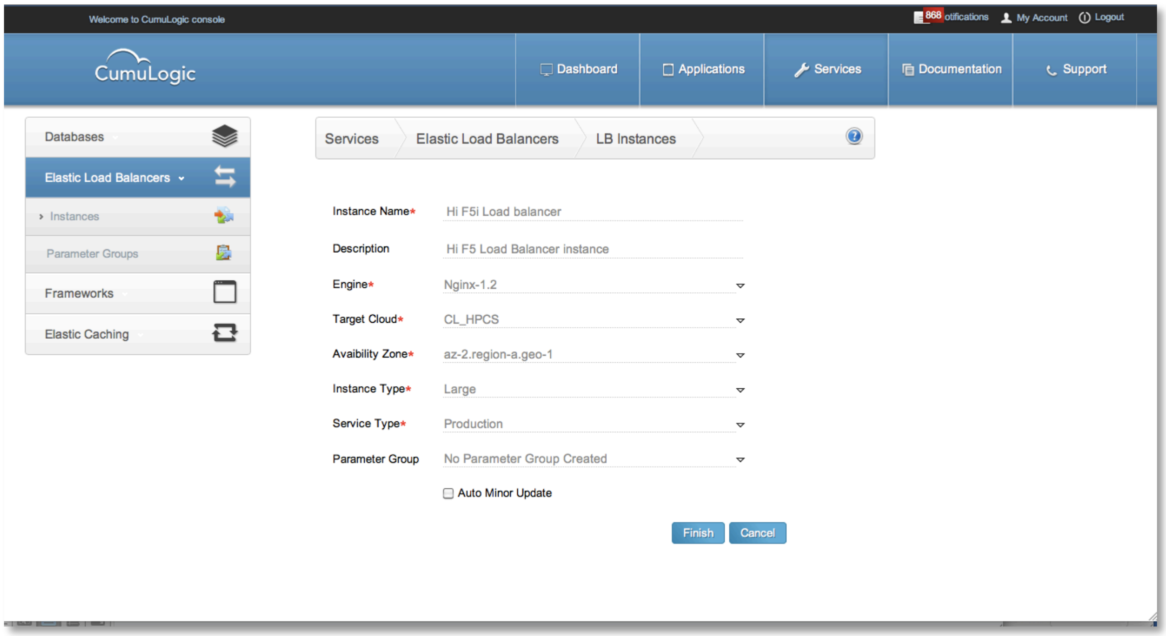
You can also use a load balancer instance to configure and load balance an application instance deployed on any cloud. CumuLogic Load Balancer is fully managed and is application node-aware, automatically detecting node failures and reconfiguring routing to avoid routing traffic to failed nodes.

CumuLogic Load Balancer can also be configured to route traffic to additional new nodes added to the application. When an application deployed on CumuLogic PaaS auto-scales the application server or framework nodes, it automatically reconfigures the load balancer to route traffic to new nodes as well.

CumuLogic Load Balancer is also monitored and has self-healing capabilities to recover failures.

Creating a Load Balancer Instance

To create a new instance of the load balancer, select **Load Balancer** from menu bar on the left and choose the size of the load balancer instance as shown below.

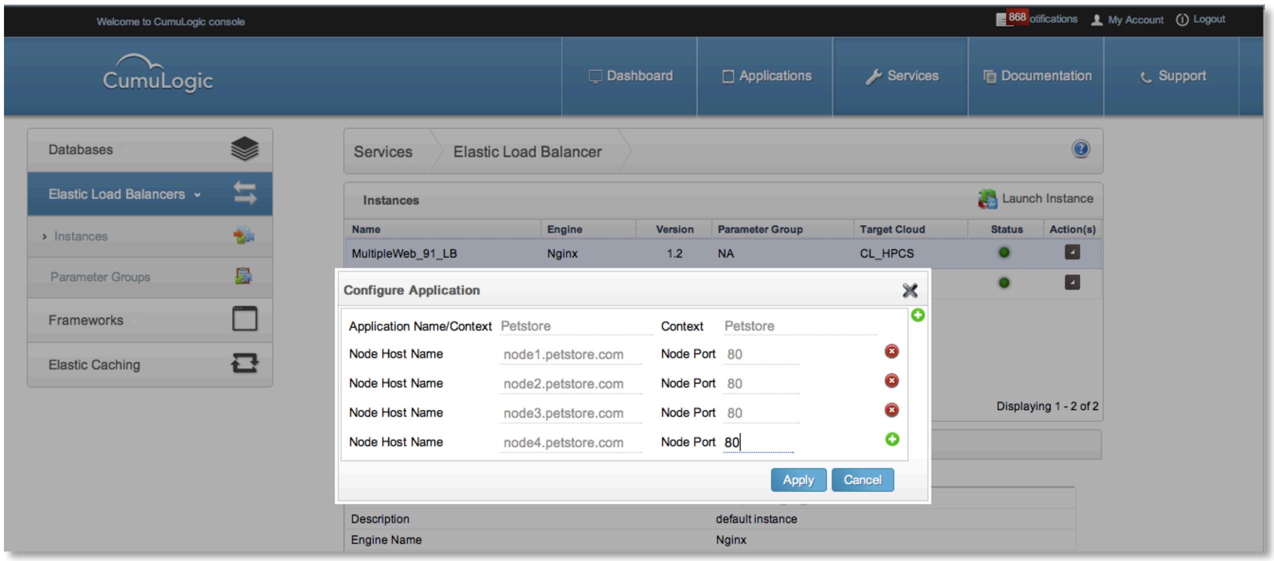


Instance Name* Hi F5i Load balancer
Description Hi F5 Load Balancer instance
Engine* Nginx-1.2
Target Cloud* CL_HPCS
Avaibility Zone* az-2.region-a.geo-1
Instance Type* Large
Service Type* Production
Parameter Group No Parameter Group Created
Auto Minor Update

Figure 25: Creating Load Balancer Instance

Configuring the Load Balancer for Applications

To configure the load balancer instance to load balance between multiple nodes of an application, select the **Add Application** option from the action menu.



Instances

Name	Engine	Version	Parameter Group	Target Cloud	Status	Action(s)
MultipleWeb_91_LB	Nginx	1.2	NA	CL_HPCS	●	
					●	

Displaying 1 - 2 of 2

Configure Application

Application Name/Context	Petstore	Context	Petstore
Node Host Name	node1.petstore.com	Node Port	80
Node Host Name	node2.petstore.com	Node Port	80
Node Host Name	node3.petstore.com	Node Port	80
Node Host Name	node4.petstore.com	Node Port	80

Description	default instance
Engine Name	Nginx

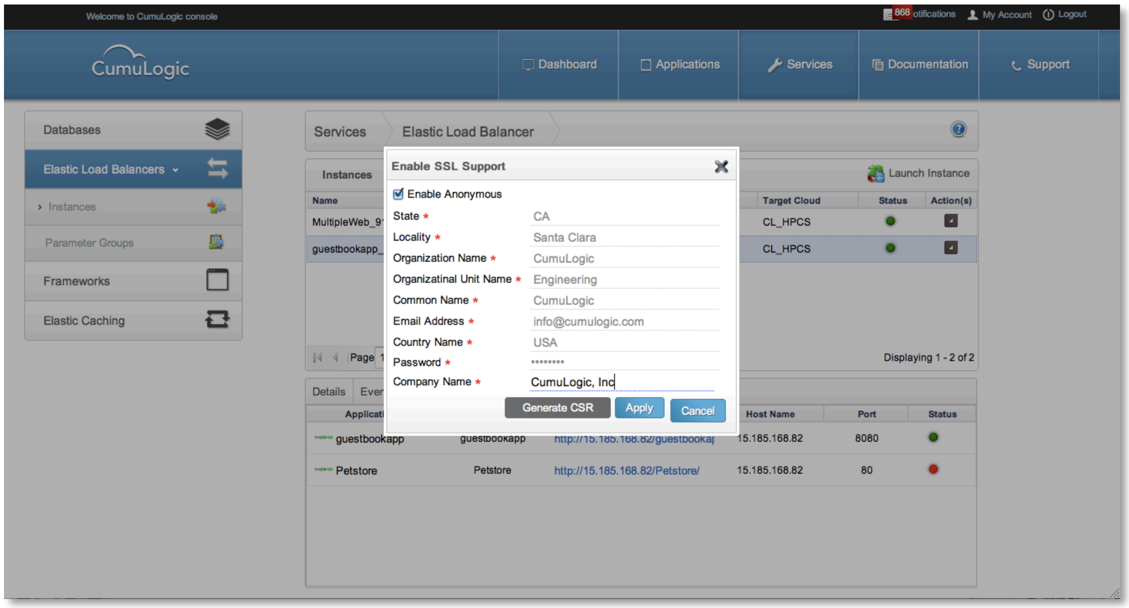
Figure 26: Configuring a load balancer instance

As shown in the screen above, you can add host names or IP addresses of nodes used for running the application. In this, the load balancer is configured to route traffic to the petstore application, which runs on four nodes.

Enabling SSL on the Load Balancer

You can enable SSL on the load balancer instance by selecting **Enable SSL** from the action menu on the instance.

It is also possible to enable SSL with self-signed certificates or CA certificates. If you want to use a self-signed certificate, select "Anonymous" on Enable SSL and fill in the required information.



Enable SSL Support

- [x] Enable Anonymous

State *	CA
Locality *	Santa Clara
Organization Name *	CumuLogic
Organizatinal Unit Name *	Engineering
Common Name *	CumuLogic
Email Address *	info@cumulogic.com
Country Name *	USA
Password *	••••••••
Company Name *	CumuLogic, Inc

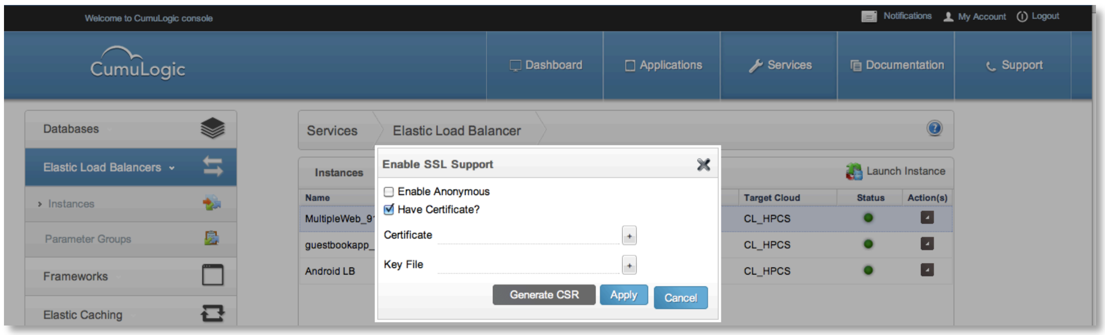
Generate CSR | Apply | Cancel

Name	Target Cloud	Status	Action(s)
MultipleWeb_9	CL_HPCS	●	
guestbookapp_	CL_HPCS	●	

Application			Host Name	Port	Status
guestbookapp	guestbookapp	http://15.185.168.82/guestbooka	15.185.168.82	8080	●
Petstore	Petstore	http://15.185.168.82/Petstore/	15.185.168.82	80	●

Figure 27: Enable SSL with self signed certificate

To enable SSL with a Root CA certificate, you can select "have CA Certificate" from the Enable SSL screen and upload the certificate files.



Enable SSL Support

- [ ] Enable Anonymous
- [x] Have Certificate?

Certificate	
Key File	

Generate CSR | Apply | Cancel

Name	Target Cloud	Status	Action(s)
MultipleWeb_9	CL_HPCS	●	
guestbookapp_	CL_HPCS	●	
Android LB	CL_HPCS	●	

Figure 28: Enable SSL with CA certificate

Cloning a Load Balancer Instance

To launch a new instance of the load balancer with the same configuration as an existing instance, you can use the **Clone** action. The Clone option launches a new instance with the same configuration and parameter group settings as the original load balancer instance. This is particularly useful if you want to create a new application environment for development, testing or debugging purposes.

Elastic Cache

CumuLogic Elastic Cache is a fully managed Memcached-compatible, distributed in-memory cache service. CumuLogic Elastic Cache can be used to improve the application scalability and performance for read-intensive workloads. To avoid expensive read operations on the database server for each read operation, it is useful to cache most frequently used data in the memory. Reading from the cache memory speeds up the access time for applications and scales applications to handle higher workloads.

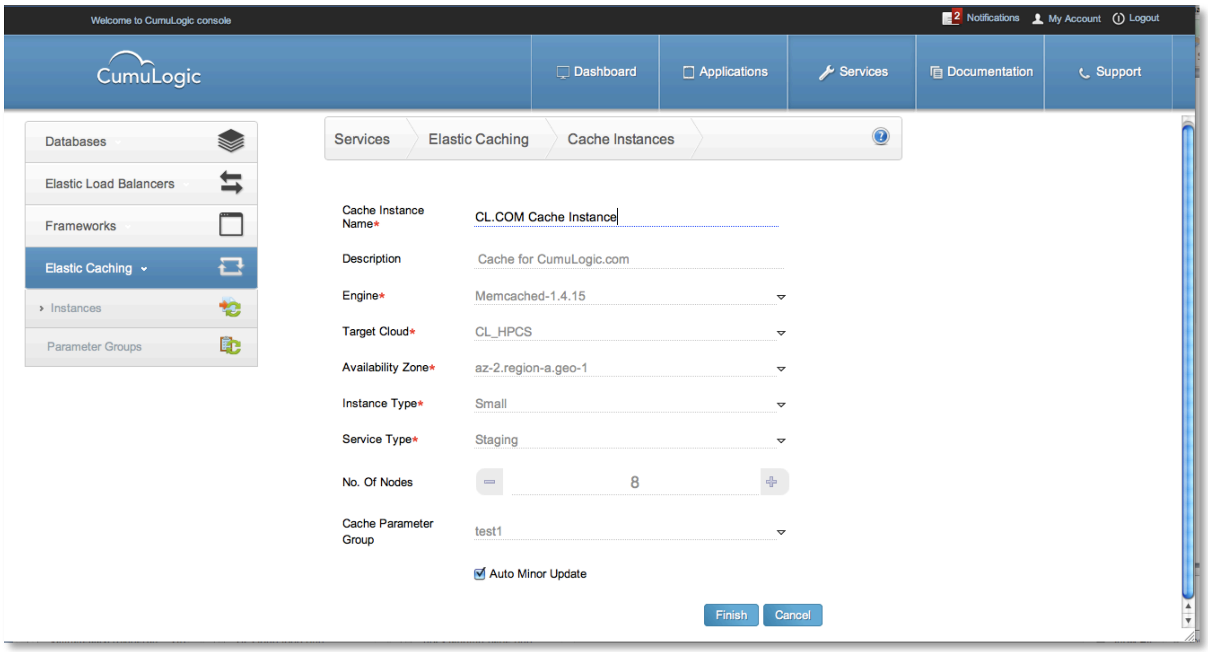
CumuLogic Elastic Cache service allows users to create an in-memory cache of by launching one ore more cache instances. CumuLogic Elastic Cache is distributed cache, so the total amount of memory available for the cache is equal to the amount of memory available in combined cache nodes.

It is preferable to use multiple smaller nodes of cache instead of using a large cache node for fault-tolerance purposes. For example, if you need a 4GB cache, it is advisable to use two 2GB cache nodes instead of one large 4GB cache node. In the event of a failure, there is a chance of at least one cache node being available. It is also possible to add a new cache node to replace a failed node in which case CumuLogic Elastic Cache will re-distribute the cached data between the nodes.

Creating a Cache Cluster

To create a cache cluster of one or more cache nodes, select **Elastic Cache** from the left side menu bar and select **Launch Instance.**

In the example below, we need a cache of 16GB, so we selected 8 nodes of small size where each node has 2GB memory. This 8-node cluster will approximately provide access to a 16GB cache memory.



Cache Instance Name*	CL.COM Cache Instance
Description	Cache for CumuLogic.com
Engine*	Memcached-1.4.15
Target Cloud*	CL_HPCS
Availability Zone*	az-2.region-a.geo-1
Instance Type*	Small
Service Type*	Staging
No. Of Nodes	8
Cache Parameter Group	test1
✓ Auto Minor Update	

Figure 29: Creating a cache cluster

Once the desired cache cluster is available for use, go to the **Hosts** tab to get the IP addresses of the cache nodes to use in your application.

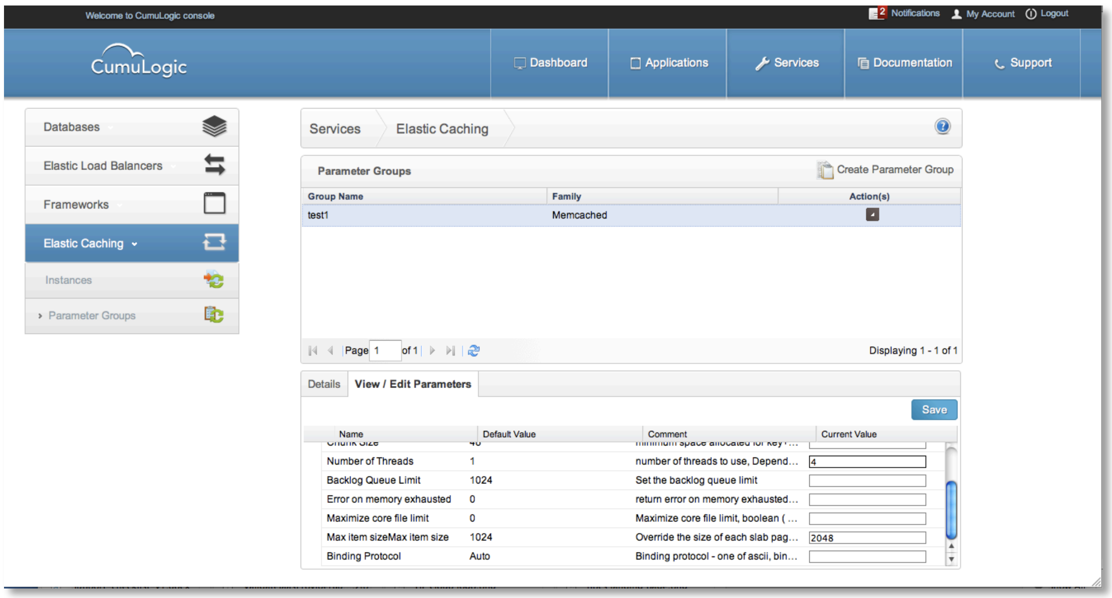
The IP addresses of the newly launched cluster nodes are also available in the environment variables CL_CACHE_SERVERS (IP address:port), which could be sourced from your application.

Cache nodes are monitored by the **Health Monitor** and will be recovered in case of most failures. When new nodes are provisioned, the cache service will trigger re-distribution of data across the nodes.

Cache Parameter Groups

Cache analytics are available for users to review the performance and utilization metrics. You can tweak the parameters to optimize the cache for your particular workload type. To make changes to the default configurations, select **Create Parameter Group** and go to the **Edit/Modify** tab to change the values of the parameters. The CumuLogic Controller makes real time modifications to the cache nodes without requiring a restart.

Please note that any changes to the parameters may affect your cache and application performance negatively if not appropriately used. Please read the documentation for details on each parameter and how it affects the cache service before making any changes to the parameter groups.



Parameter Groups

Group Name	Family	Action(s)
test1	Memcached	

Name	Default Value	Comment	Current Value
Number of Threads	1	number of threads to use, Depend...	4
Backlog Queue Limit	1024	Set the backlog queue limit	
Error on memory exhausted	0	return error on memory exhausted...	
Maximize core file limit	0	Maximize core file limit, boolean (...	
Max item sizeMax item size	1024	Override the size of each slab pag...	2048
Binding Protocol	Auto	Binding protocol - one of ascii, bin...	

Figure 30: Cache Parameter Groups

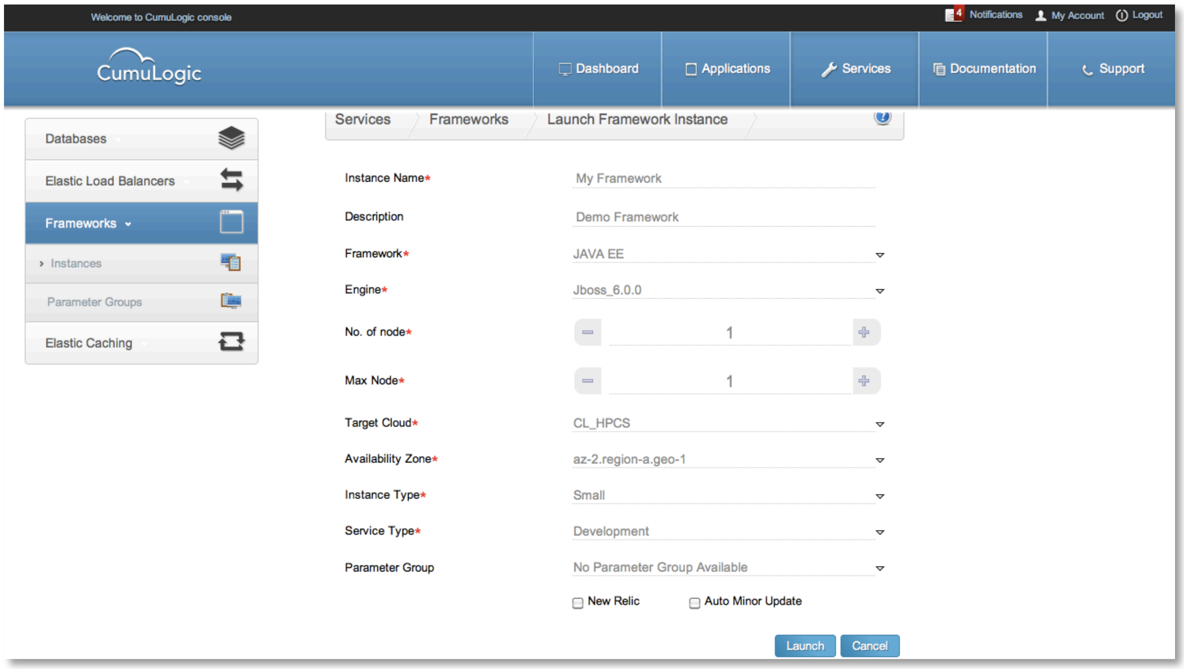
Frameworks

CumuLogic's platform supports Java, Spring and Apache PHP frameworks. *Please refer to the latest release notes on the Documentation website for details on current frameworks and languages supported.*

You should use **CumuLogic application deployment wizard** to deploy applications on the CumuLogic platform, which provisions and configures all the services required by the applications. All services are then monitored and managed by the **PaaS Controller**, so no manual application management tasks are required. However, in some cases, you may need to manually deploy applications directly on the frameworks and configure any services desired. An example is when you need to design an architecture requiring different modules, or modules that are not currently supported by the CumuLogic platform. In these scenarios, the CumuLogic platform allows users to create framework instances, customize them for desired workloads, configure the framework properties and containers, and have the flexibility to build composite architectures.

Creating Framework Instances

You can select **Frameworks** from the left side menu bar on the CumuLogic Console and select **Launch Instance** and select the Framework type and other information as illustrated in Figure 31 below.



Instance Name* My Framework
Description Demo Framework
Framework* JAVA EE
Engine* Jboss_6.0.0
No. of node* 1
Max Node* 1
Target Cloud* CL_HPCS
Availability Zone* az-2.region-a.geo-1
Instance Type* Small
Service Type* Development
Parameter Group No Parameter Group Available
New Relic
Auto Minor Update

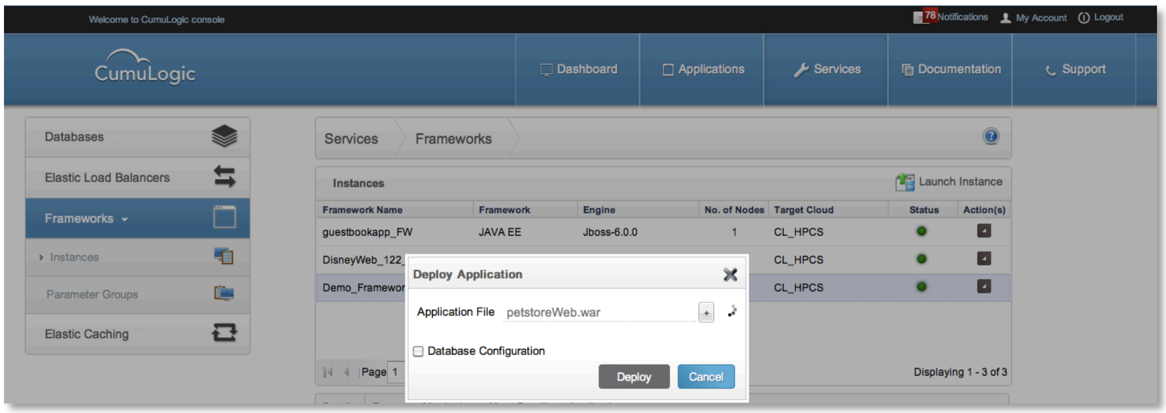
Figure 31: Launch Framework

Instance Name	**Name or identifier for your framework instance. Usually, the same name as Application name.**
Framework	Choose based on the application type. For example, Java EE or Spring for Java and Spring applications.
Engine	Framework or container type in case more than one are supported. For example, for Java EE, the engine could be JBoss or Tomcat based on your application needs.
No. of nodes	Number of instances of framework you need. All the instance nodes must be load balanced with CumuLogic Load Balancer.
Max no. of node	Upper limit to number of nodes you would like to scale when autoscaling provisions additional nodes. The load balancer will automatically limit traffic to this number of nodes.
Target Cloud	Target IaaS cloud where you prefer to launch this framework.
Availability Zone	Availability zone of the target cloud.
Instance Type	Size of the instance or flavor of the VM size on target IaaS cloud. Typically, small, medium, large and extra large.
Service Type	Tag as Development, Testing, Staging or Production.
Parameter group	Parameter group used to configure the framework properties.

Deploying Applications on Frameworks

To deploy an application on a framework, choose **Deploy Application** on the provisioned framework instance. Note that the preferred way to deploy applications is through the **Applications tab** with a defined **cumulogic-web.xml** manifest file so that the platform manages all the services allocated to the applications. In some cases, when architecture is not supported by the platform, it is recommended to manually launch the framework instances and deploy the applications or modules.

Applications must be packaged as war files or tar.gz, which include the application files and the **cumulogic-app.xml** deployment descriptor file. The applications type, framework type and datasource requirements are automatically detected from the application artifacts such as web.xml for Java apps.



Framework Name	Framework	Engine	No. of Nodes	Target Cloud	Status	Action(s)
guestbookapp_FW	JAVA EE	Jboss-6.0.0	1	CL_HPCS	●	
DisneyWeb_122				CL_HPCS	●	
Demo_Framewor				CL_HPCS	●	

Page 1 Displaying 1 - 3 of 3

Deploy Application

Application File petstoreWeb.war

☐ Database Configuration

Deploy Cancel

Figure 32: Deploying Application on Framework

In the **sample application** above, we uploaded the petstoreWeb.war file with a custom cumulogic-web.xml manifest or descriptor file. The <cumulogic-app> tag defines the application's datasource JNDI lookup name and database type (no datasource is used in this application as petstoreWeb sample app manages its own Derby database).

The <services> tab declares a single node framework (Java EE) and the load balancer to be provisioned for the application. Please refer to the latest release notes or documentation for framework types and services and their engines supported in the current build of CumuLogic PaaS Server.

If no services are declared in the manifest file, the platform will provision the default container (which is defined by the PaaS administrator in private PaaS deployments) and no other services such as databases or load balancers.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.cumulogic.com/schema/v1cumulogic-
app.xsd">

<cumulogic-app>
<application-context>petstoreWeb</application-context>
        <jndi-name></jndi-name>
        <db-name></db-name>
        <db-type></db-type>
<application-file-name>petstoreWeb.war</application-file-name>
</application>

 <services>
        <framework>
                <type>Java EE</type>
                <engine>Jboss-7.0.2</engine>
                <no-of-nodes>1</no-of-nodes>
                <max-nodes>2</max-nodes>
         </framework>
        <elastic-load-balancer>
                <engine>Nginx-1.2</engine>
        </elastic-load-balancer>
  </services>
</cumulogic-app>
```

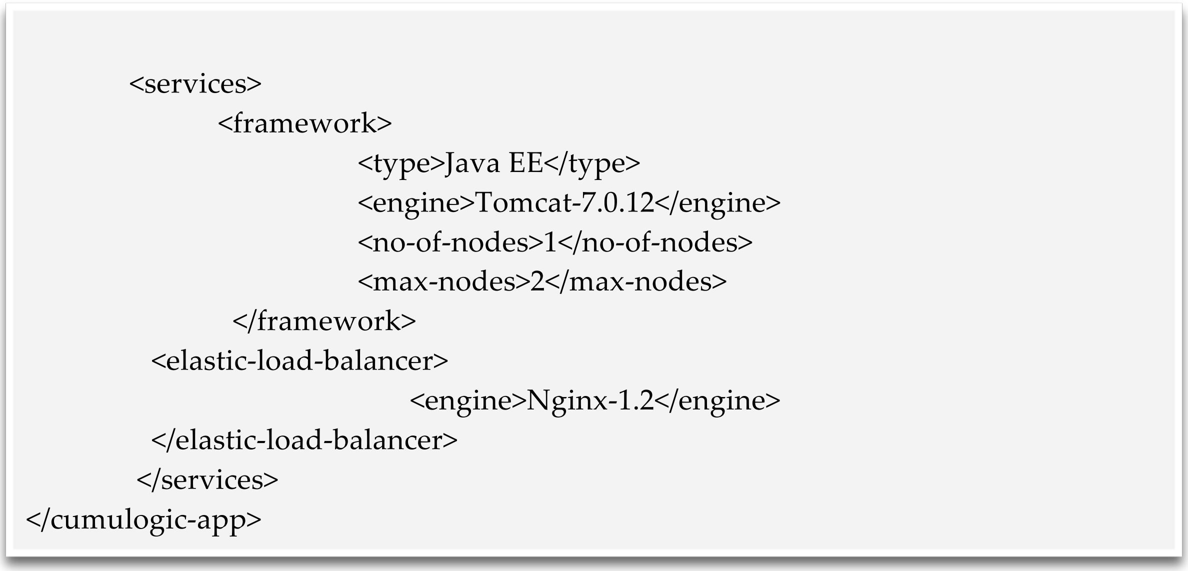
Figure 33: cumulogic-app.xml manifest file for sample app

Deploying Multiple Applications on Shared Frameworks

If you have a set of dependent application modules or applications, you can deploy them all with a single deployment manifest file.

It is possible to deploy multiple modules or multiple applications on shared containers, shared database instances and load balancers. Below is a sample cumulogic-app.xml file.

```
<cumulogic-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.cumulogic.com/schema/v1/cumulogic-
app.xsd">>
        <application>
                <application-context>testweb</application-context>
                 <jndi-name></jndi-name>
                 <db-name></db-name>
                 <db-type></db-type>
                <application-file-name>testweb.war</application-file-name>
         </application>

        <application>
                 <application-context>petstoreWeb</application-context>
                 <jndi-name></jndi-name>
                 <db-name></db-name>
                 <db-type></db-type>
                <application-file-name>petstoreWeb.war</application-file-name>
        </application>

        <application>
                 <application-context>stateless</application-context>
                 <jndi-name></jndi-name>
                 <db-name></db-name>
                 <db-type></db-type>
               <application-file-name>stateless.war</application-file-name>
        </application>
```



```
            <services>
                  <framework>
                              <type>Java EE</type>
                              <engine>Tomcat-7.0.12</engine>
                              <no-of-nodes>1</no-of-nodes>
                              <max-nodes>2</max-nodes>
                   </framework>
             <elastic-load-balancer>
                                  <engine>Nginx-1.2</engine>
             </elastic-load-balancer>
            </services>
</cumulogic-app>
```

Figure 34: cumulogic-app.xml manifest file for sample app

In this example, we deploy three sample applications, statesless.war, testweb.war and petstoreWeb.war on a Tomcat container (multiple instances, load balanced) and a single instance of load balancer for all three applications).

Deploying Multiple Applications in Heterogeneous Environments

In most large-scale deployments, the application environments are complex and require the deployment of heterogeneous applications that are dependent on each other. For example, a highly popular website may have several Java-based web applications, plain JVM-based applications or PHP modules, which could be deployed in hundreds of virtual machines. In order to scale these architectures, distributed applications are to be provisioned so each of them and their tiers can scale independently. CumuLogic platform allows the provisioning of complex environments with ease, eliminating the complexity of architecting, configuring and managing a complete application environment.

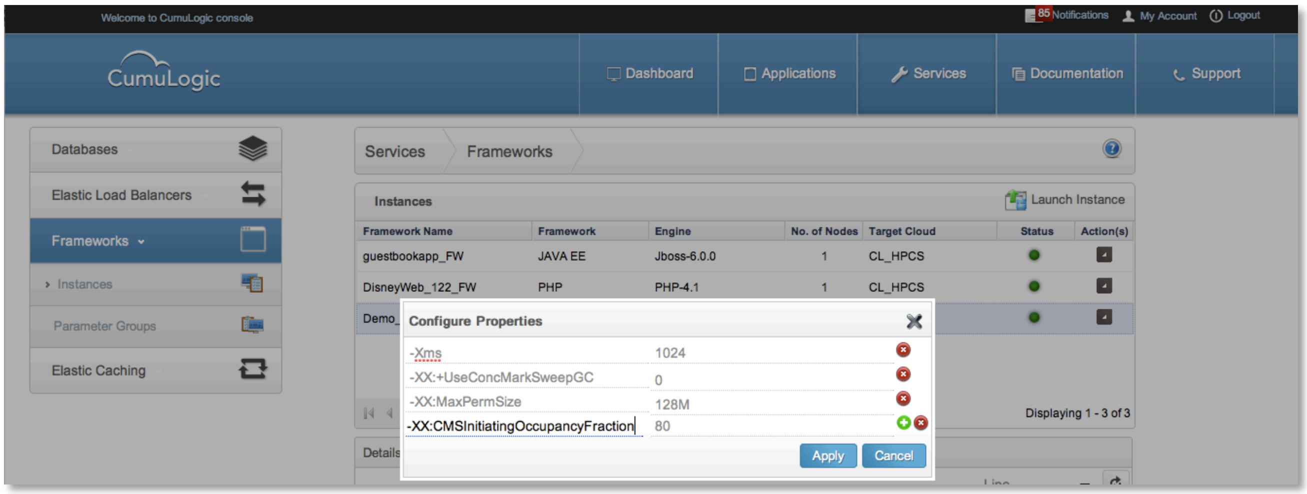
To deploy heterogeneous environments, refer to the deployment manifest below. In this example, we demonstrate the deployment of two applications (a Java app and a PHP app) where each application requires a different container and number of instances for scalability.

```
<cumulogic-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.cumulogic.com/schema/v1/cumulogic-
app.xsd">>

        <application>

                <application-context>testweb</application-context>
                 <jndi-name></jndi-name>
                 <db-name></db-name>
                 <db-type></db-type>
                 <application-file-name>testweb.war</application-file-name>
                <services>
                        <framework>
                                    <type>Java EE</type>
                                    <engine>Tomcat-7.0.12</engine>
                                    <no-of-nodes>10</no-of-nodes>
                                    <max-nodes>20</max-nodes>
                         </framework>
                  <elastic-load-balancer>
                                    <engine>Nginx-1.2</engine>
                  </elastic-load-balancer>
                  </services>
        </application>

        <application>
                 <application-context>guestbook-php</application-context>
                 <jndi-name>MySQLGBDS</jndi-name>
                 <db-name>guestbook</db-name>
                 <db-type>MySQL</db-type>
                <application-file-name>guestbook-php.tar.gz</application-file-name>
                <services>
                        <framework>
                                    <type>PHP</type>
                                    <engine>PHP-4.1</engine>
                                    <no-of-nodes>10</no-of-nodes>
                                    <max-nodes>50</max-nodes>
                                     <storage>100</storage>
                         </framework>
                  <elastic-load-balancer>
                                        <engine>Nginx-1.2</engine>
                  </elastic-load-balancer>
```

Figure 35: cumulogic-app.xml manifest for deploying heterogeneous environments

Custom Properties on Frameworks

To customize the system properties or modify the framework properties, you can choose **Configure Properties** on the framework and enter the desired properties and values. These properties will be applied to the JVM during startup and provisioning of applications.



Framework Name	Framework	Engine	No. of Nodes	Target Cloud	Status	Action(s)
guestbookapp_FW	JAVA EE	Jboss-6.0.0	1	CL_HPCS	●	
DisneyWeb_122_FW	PHP	PHP-4.1	1	CL_HPCS	●	
Demo_					●	

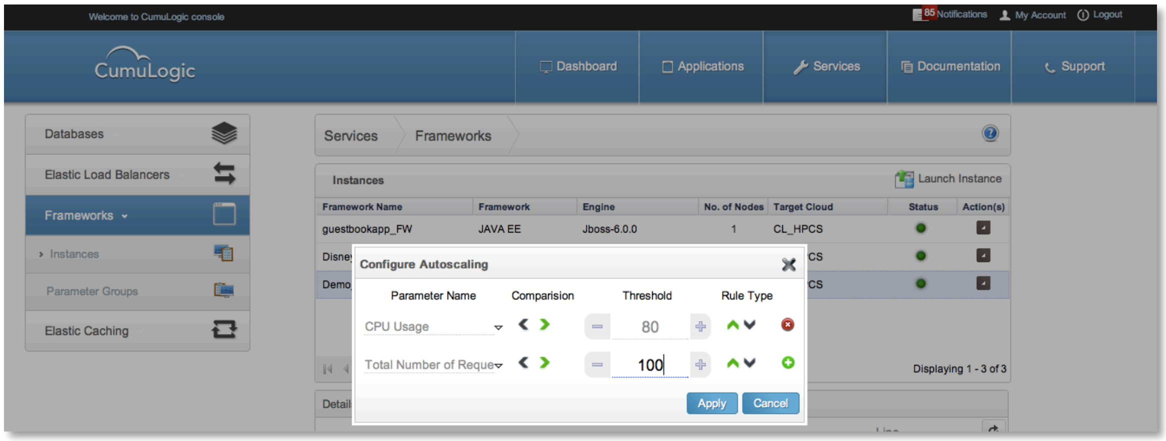
Configure Properties

-Xms	1024
-XX:+UseConcMarkSweepGC	0
-XX:MaxPermSize	128M
-XX:CMSInitiatingOccupancyFraction	80

Figure 36: Configure properties on Frameworks

Autoscaling Configuration

The CumuLogic platform automatically scales the resources required to maintain the health of running applications. CumuLogic's platform uses default thresholds for metrics such as CPU and memory usage to scale a particular tier (except data tier in case of MySQL or other SQL databases). You can configure custom threshold values for specific metrics and add new autoscaling rules.



Framework Name	Framework	Engine	No. of Nodes	Target Cloud	Status	Action(s)
guestbookapp_FW	JAVA EE	Jboss-6.0.0	1	CL_HPCS	●	
Disney				CS	●	
Demo				CS	●	

Parameter Name	Comparision	Threshold	Rule Type
CPU Usage		80	
Total Number of Reque		100	

Figure 37: Configuring autoscaling rules

In the example above, we've added two new rules for auto-scaling the application servers and load balancers when CPU thresholds and the total number of requests to the applications are reached. The platform will monitor the CPU usage and if it stays over the defined threshold value for 3 minutes, additional application servers will be provisioned. If the CPU usage goes below the lower threshold value for 5 minutes, the application server instances will be scaled down and the load balancer will be configured appropriately.

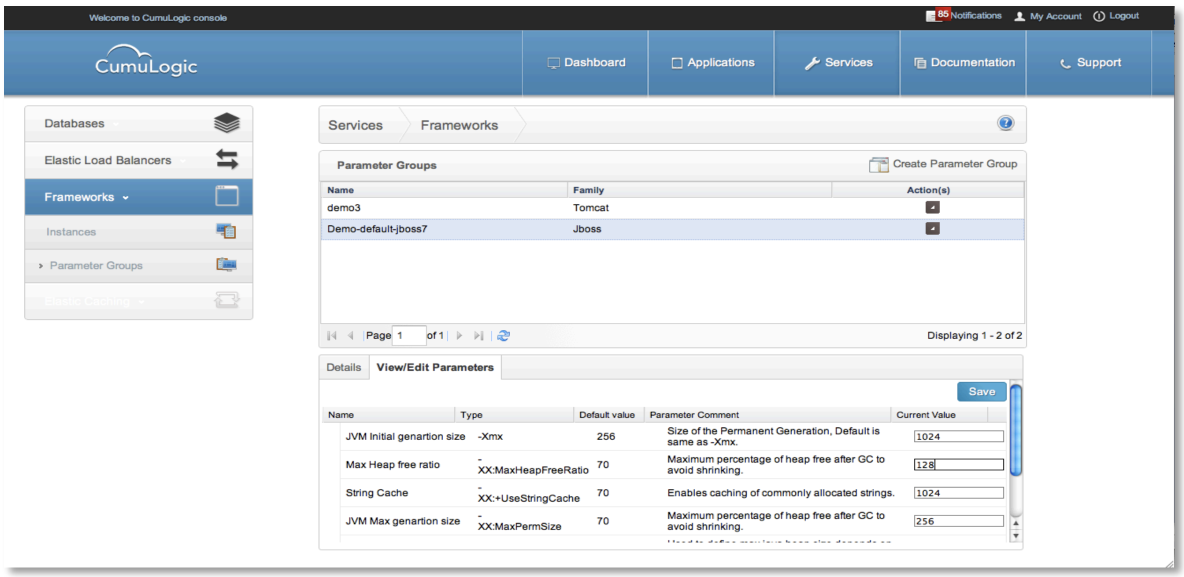
Cloning an Existing Framework

To clone a pre-configured Framework, choose **Clone**. This action makes a copy of the existing framework with all system properties and other configurations defined. This is an easy way to replicate an environment on a different cloud or infrastructure, or move workloads between different stages of applications.

Modifying Parameters Using Parameter Groups

The Framework Parameter Groups allow users to customize the performance parameters for frameworks. For instance, for an application that requires a large number of container threads or larger amount of Java heap, you can create a new Parameter Group and edit the parameters to suit your workloads.

As shown in the screenshot below, we created a demo3 parameter group and modified the values of heap sizes for the JVM. This parameter group can then be applied to an existing framework instance or to launch new instances. Please note that some parameters require frameworks to be restarted to take affect on running instances.



Name	Family	Action(s)
demo3	Tomcat	
Demo-default-jboss7	Jboss	

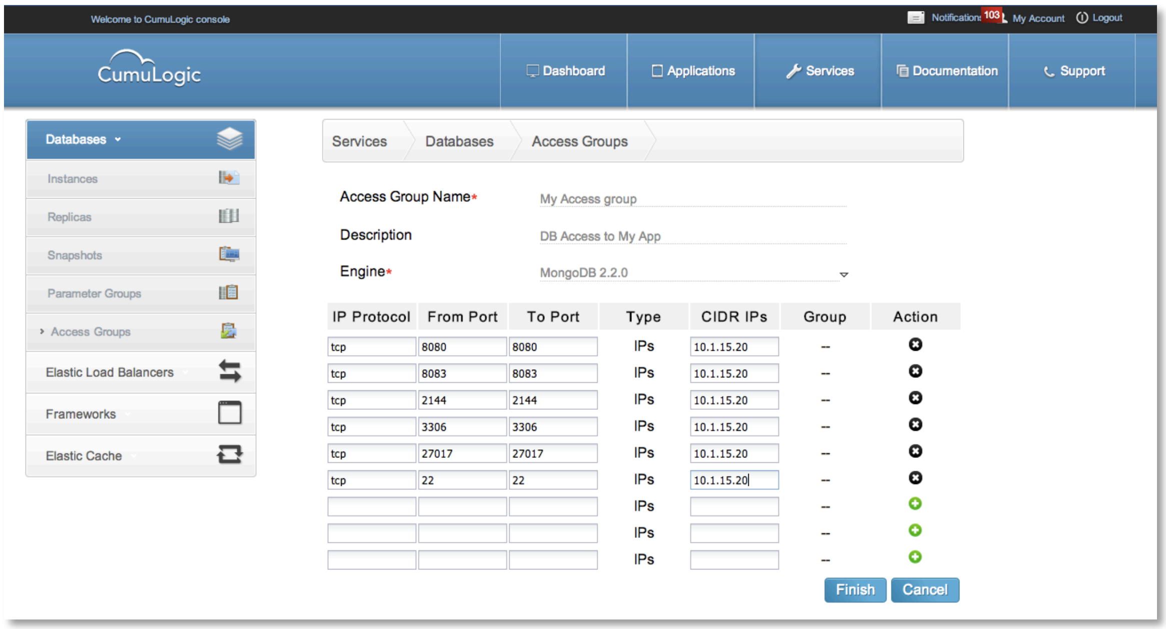
Name	Type	Default value	Parameter Comment	Current Value
JVM Initial genartion size	-Xmx	256	Size of the Permanent Generation, Default is same as -Xmx.	1024
Max Heap free ratio	-XX:MaxHeapFreeRatio	70	Maximum percentage of heap free after GC to avoid shrinking.	128
String Cache	-XX:+UseStringCache	70	Enables caching of commonly allocated strings.	1024
JVM Max genartion size	-XX:MaxPermSize	70	Maximum percentage of heap free after GC to avoid shrinking.	256

Figure 38: Parameter Groups

Security and Access Groups

An access group acts as a firewall that controls the traffic to any CumuLogic-powered cloud service in and out of the instance. If you wish to block access to a service or allow access from a particular IP address, you can create a custom access group and configure it just like you would configure firewall settings.

By default, network access to all CumuLogic services is blocked. To allow access to your services, you can create an access group and assign services as members of the access groups.



Access Group Name*	My Access group
Description	DB Access to My App
Engine*	MongoDB 2.2.0

IP Protocol	From Port	To Port	Type	CIDR IPs	Group	Action
tcp	8080	8080	IPs	10.1.15.20	--	✖
tcp	8083	8083	IPs	10.1.15.20	--	✖
tcp	2144	2144	IPs	10.1.15.20	--	✖
tcp	3306	3306	IPs	10.1.15.20	--	✖
tcp	27017	27017	IPs	10.1.15.20	--	✖
tcp	22	22	IPs	10.1.15.20	--	✖
			IPs		--	⊕
			IPs		--	⊕
			IPs		--	⊕

Figure 39: Access groups

As shown in the screenshot above, **My Access Group** allows access to a MongoDB instance only from one IP address which is the instance running the application. To revoke access to a service, remove the IP address from the authorization list and save the access groups.

The source can be a range of IP addresses specified in the form of xx.xx.xx.xx/end.

Any changes to the access groups are applied immediately to any service instance using that access group.