# OpenStack Workload Reference Architecture: eCommerce

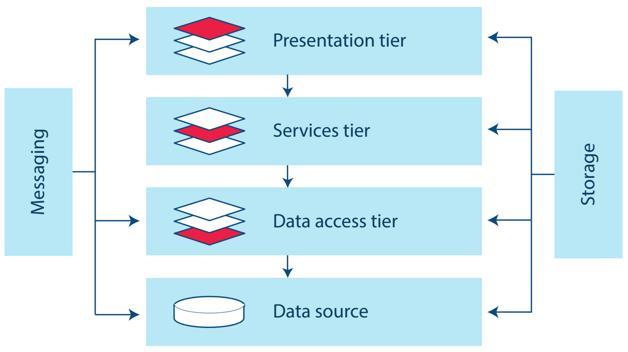
## Introduction

A cloud environment naturally lends itself as the foundation for an eCommerce solution. eCommerce is a transaction of buying or selling online, that draws on technologies such as mobile, electronic funds transfer, Internet marketing, online transaction processing, inventory management systems, and more. Best Buy, Walmart, overstock.com, eBay, Staples, Nike, Snapdeal and PayPal are just a few examples of businesses relying on OpenStack to drive their eCommerce sites.

OpenStack's popularity for the eCommerce workload is attributed to massive scalability, flexibility to architect high availability and global presence, fast-paced innovation, community support, and low cost. Digging deeper, the elasticity of eCommerce consumer demand requires an automated resource consumption model. In an OpenStack cloud environment, IT resources (CPU utilization, memory, etc.) can be automatically provisioned and configured to scale dynamically, up and down, based on load. This document provides a reference architecture for a basic eCommerce application running on an OpenStack cloud. It identifies and recommends the required OpenStack services and key options used to the implement the workload.

This workload reference architecture is intended for enterprise architects who are looking to deploy an eCommerce application on an OpenStack cloud. It is accompanied by two OpenStack Orchestration (Heat) templates that will install a sample eCommerce workload with the tiers and messaging shown below, on an existing OpenStack cloud. One template offers a manual scaling environment, the other implements auto-scaling.

Figure 1: eCommerce high-level architecture



This reference architecture is for a general workload developed by members of the Enterprise Working Group and is not representative of the contributors' companies.

**CONTRIBUTORS:**
**Sameer Adhikari,** *Principal Engineer,* Walmart
**Megan Rossetti,** *Cloud Operations,* Walmart
**Craig Sterrett,** *Software Architect,* Intel Corporation
**Yih Leong Sun, PhD,** *Senior Software Cloud Architect,* Intel Corporation

## OpenStack for eCommerce

For this reference architecture, the eCommerce application primarily consists of a web presentation tier, a services tier, and a persistent database tier. Messaging, load balancing, auto-scaling, and analytics support the interaction among the services running in the primary tiers.

- **Web presentation tier** – cluster of web servers that will be used to render either static or dynamically generated content for the user.
- **Services tier** – clusters of application servers that will be used to process the functional services of an eCommerce environment; some visible to the end user, some not.
- **Database tier** – cluster of database servers that stores data persistently.

Figure 2: Logical representation of OpenStack components

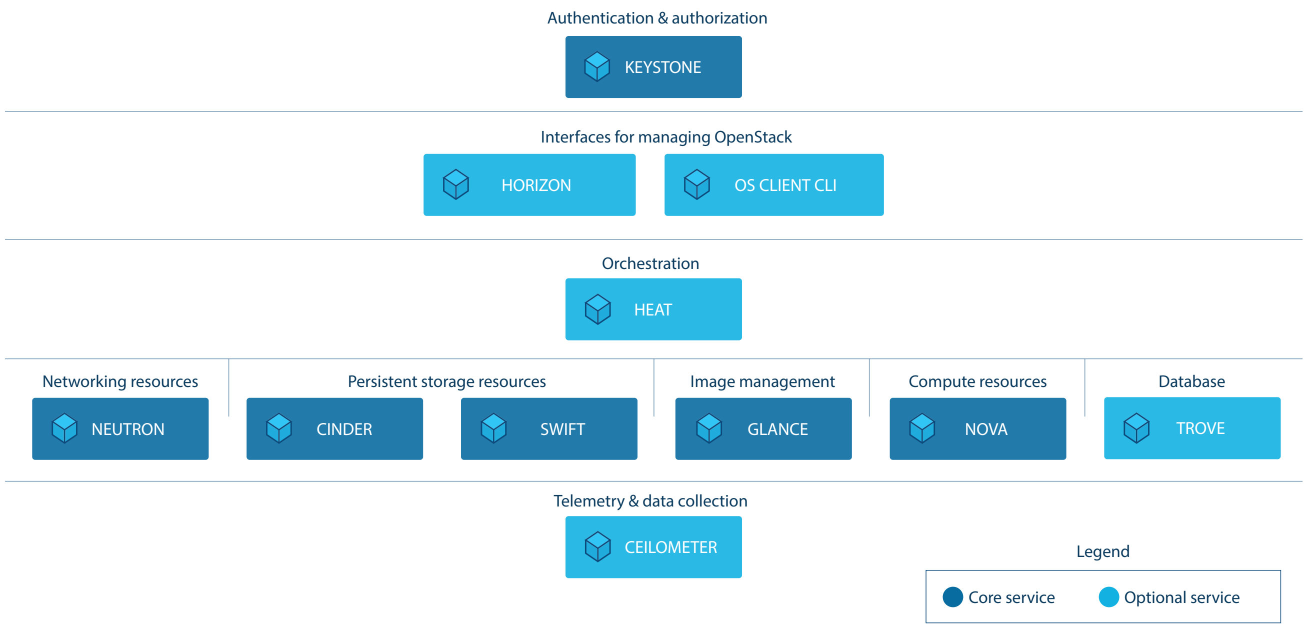


Figure 2 shows the required and optional services in relation to one another, and the services to confirm are available in your OpenStack cloud.

Brief descriptions of the services used for a simple eCommerce application follow. The OpenStack Project Navigator provides additional information.

| | |
|---|---|
| **COMPUTE (NOVA)** | Manages the life cycle of compute instances, including spawning, scheduling, and decommissioning virtual machines (VMs) on demand. |
| **IMAGE SERVICE (GLANCE)** | Stores and retrieves VM disk images. Used by OpenStack Compute during instance provisioning. |
| **BLOCK STORAGE (CINDER)** | Virtualizes the management of block storage devices and provides a self-service API to request and use those resources regardless of the physical storage location or device type. Supports popular storage devices. |
| **NETWORKING (NEUTRON)** | Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API to define networks and their attachments. Supports popular networking vendors and technologies. Also provides Load Balancing-as-a-Service (LBaaS) and Firewall-as-a-Service (FWaaS). |
| **IDENTITY SERVICE (KEYSTONE)** | Provides authentication and authorization for the other OpenStack services. |
| **OBJECT STORAGE (SWIFT)** | Stores and retrieves arbitrary unstructured data objects via a RESTful HTTP-based API. Highly fault-tolerant with data replication and scale-out architecture. |

## OPTIONAL SERVICES

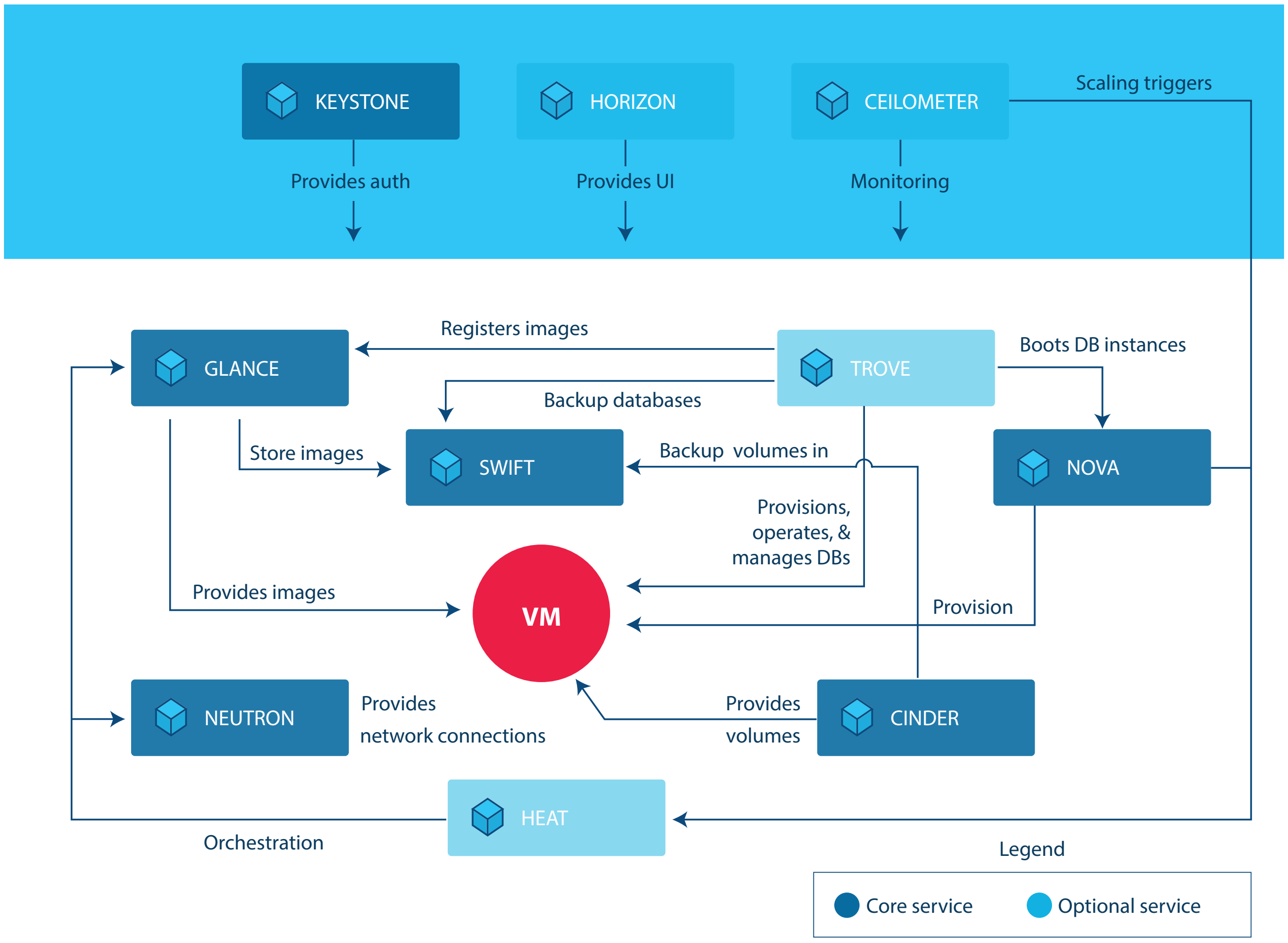
Optional services can be added for more functionality. These are the most popular options, although many more can be used to build out a more specific eCommerce infrastructure—using container services, for example.

| | |
|---|---|
| DASHBOARD (HORIZON) | A graphical user interface that provides an extensible web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses, or configuring access controls. |
| ORCHESTRATION (HEAT) | Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API. Heat allows automating workload deployment and is also used to deploy the sample application provided with this reference architecture. |
| TELEMETRY (CEILOMETER) | Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistics. |
| DATABASE (TROVE) | Database-as-a-service that automates and provisions the user's choice of relational and non-relational database engines. Trove is an option for eCommerce databases on OpenStack but is not used in this reference architecture. |

Together, Orchestration (Heat) and Telemetry (Ceilometer) enable **dynamic scaling** as load increases and decreases. Manual scaling—through operator intervention—does not require these services. Manual scaling can be used for piloting or stable-load eCommerce sites; however, dynamic scaling is most often desired.

Figure 3 illustrates the basic functional interaction between these services. For further details: OpenStack Conceptual Architecture Diagram.

Figure 3: Functional interaction between OpenStack components



## Structuring an eCommerce Application

The requirements for an eCommerce architecture are derived from the desired customer experience. A user starts with a home page that is delivered based on their information (e.g. geographic location, time of day). When each user creates an account and signs in, related information from previous orders can be used to generate information on the page. While a user is viewing the details of an item, related content like description, image, price, other sellers, and related recommendations is also provided. When the user finds products they like, they place them in a cart. To complete the order, the user follows a checkout process. Finally, the user tracks or receives notifications about shipping and delivery.

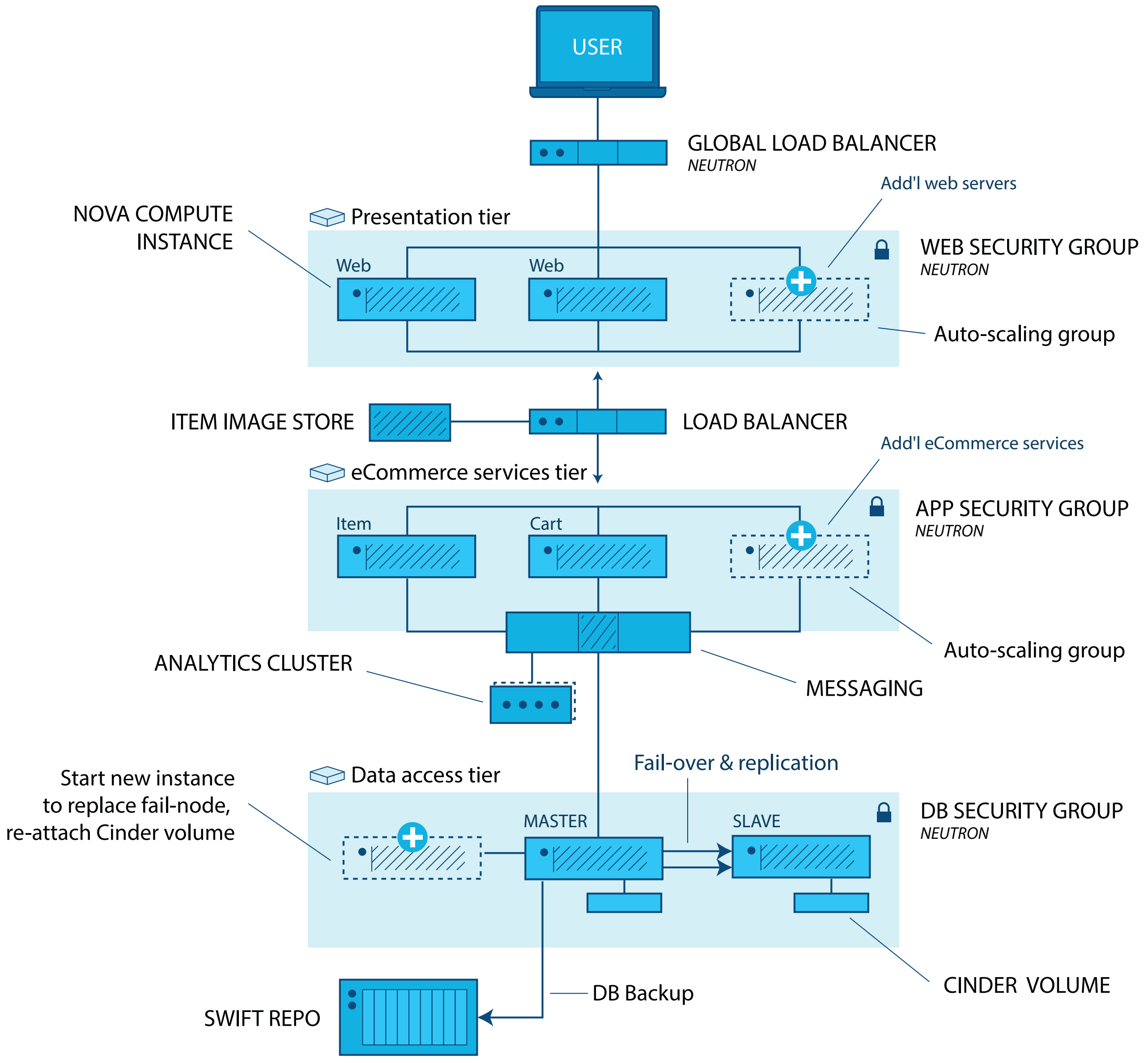
The point of this model is the customer experience drives the need for dynamic content generation at scale. This leads to a service-oriented architecture (SOA) for the applications. Each customer view is a result of collaboration between many services. To illustrate, when a customer is using the search box, the application is talking to one service, the images on the page are coming from another service, the related product recommendations are coming from yet another service, the item details from another one, and so on.

The application architecture can be split into the following components:

| ARCHITECTURAL COMPONENT | DESCRIPTION |
|---|---|
| **Web presentation tier** | This tier serves the interface for the user mobile or desktop devices. It dynamically assembles the content for the view by interacting with the services tier. |
| **Services tier** | This tier consists of the services that implement the business logic functionality; some visible to the end user and some not. Each service focuses on its core competency, like browse, search, cart, checkout, ship, image, recommendation, item, etc. The services interact with each other using the published web service API. |
| **Database tier** | The processing done in an application depends on data such as catalog, inventory, user, price, transaction history, etc. The database layer is responsible for providing the storage and retrieval mechanisms for data. |
| **Messaging** | The API for each service provides a synchronous communication mechanism. There is also a need for an asynchronous communication mechanism provided by messaging. |
| **Storage** | Persistent storage is required for all databases. |
| **Analytics** | An eCommerce workload meets the criteria for big data: volume, variety, and velocity. The analytics layer is responsible for ensuring the transformation of the raw logs, metrics, and meters into information and insights, which are then stored in databases, and consumed by services online (customer interaction) or offline (business decisions). |

OpenStack enables the cloud manifestation of this SOA framework in a natural and comprehensive manner, as shown in Figure 4.

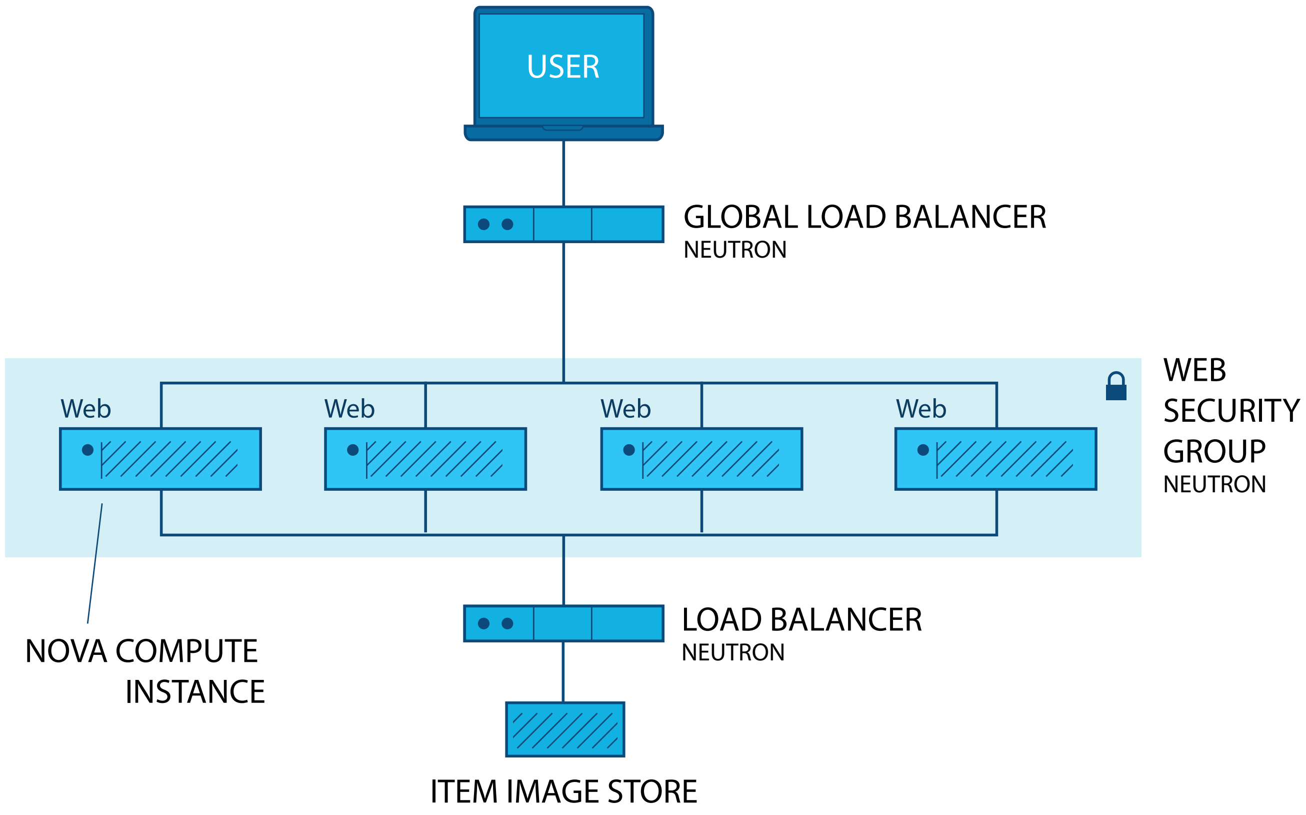
Figure 4: Overall OpenStack eCommerce architecture



## WEB PRESENTATION TIER

A load balancer directs customer requests to the web tier, that then generates dynamic web pages incorporating responsive web design. The load balancer is deployed as a scalable service on OpenStack. OpenStack Nova provides the VMs, which are instantiated with images stored in Glance. Heat can be used to orchestrate the applications in this tier, and to auto-scale the instance count based on the customer traffic pattern to the site. The VMs in this tier are usually smaller flavors. Each of them does a small amount of work, but many are needed to support the various types of work at the presentation tier.

Figure 5: eCommerce presentation tier architecture

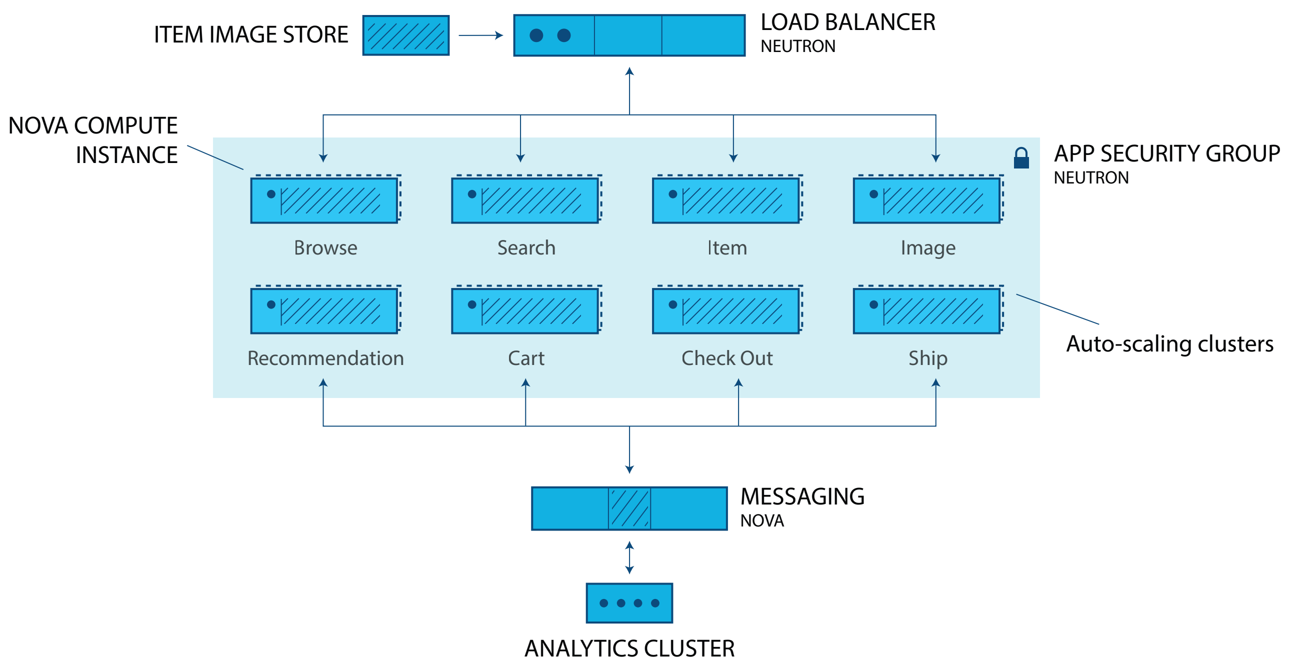


## SERVICES TIER

The services tier of scalable application servers is the heart of the eCommerce application. The different components of an eCommerce shopping experience are implemented in this tier. Similar to the presentation tier, Nova is used to provide the VMs for running these components, with a load balancer to spread the load across various services. Glance stores and provides the images for each. Heat orchestration can be used to configure the required software for each service. The roles of the different services are as follows:

| ECOMMERCE SERVICE | DESCRIPTION |
|---|---|
| **Browse** | Allows the user to browse for products by category. |
| **Search** | Allows the user to search for products with keywords. |
| **Item** | Provides the details of an item such as description, features, price. |
| **Image** | Provide image files for all item pictures shown to a user. |
| **Recommendation** | Provides products related to the one the user is viewing. |
| **Cart** | Allows the user to gather items they want to buy. |
| **Checkout** | Allows the user to pay for the items they want to buy. |
| **Ship** | Provides delivery status and details after purchase. |

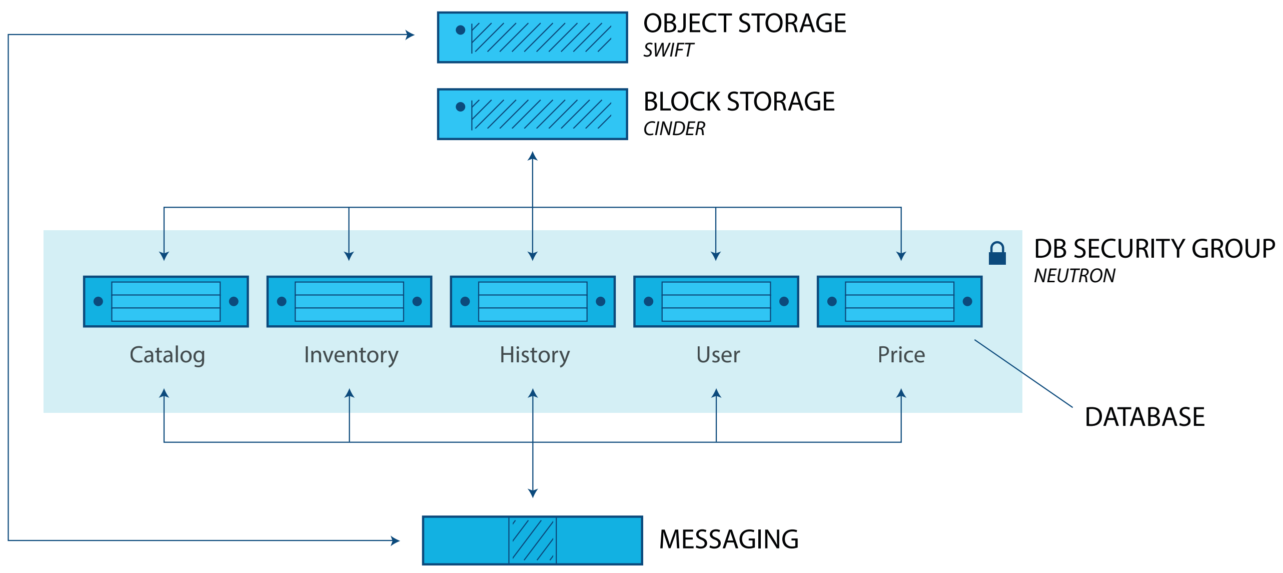
Figure 6: eCommerce services tier architecture



### DATABASE TIER

The underlying support of the eCommerce application is the database tier. This set of virtual servers organizes, stores, and manages large collections of product information and images. Databases accept customer requests and instruct the operating system to transfer the appropriate customer data while maintaining security protocols and data integrity. The databases store different types of information for the business and eCommerce operation. The performance of the database tier, including data access, data retrieval, and online completion time, can impact the customer experience both negatively and positively.

Figure 7: eCommerce database tier architecture



## Supporting OpenStack Services

OpenStack cloud provides various capabilities to help deploy the eCommerce web application architecture.

### COMPUTE INSTANCES

Depending on the scale and architecture of the e-commerce application, the components of different tiers can be deployed on various compute instances (VMs). The web presentation, services, and database tiers can be configured on separate clusters of servers with OpenStack Nova instances. For larger and more complex deployments, the various components used in the services tier can be split into multiple dedicated clusters to serve a particular function (for example, a dedicated cluster for Search and a separate cluster for Checkout). In a simple deployment, a single database instance can be provisioned with Nova. Dedicated database clusters can be provisioned as Nova compute instances for the various data stores (user, catalog, price, history, inventory) for a large-scale environment. In this example architecture, a single database instance is provisioned for all data stores. Database-as-a-Service (Trove) can be considered to simplify the provisioning and management of the database tier.

## LOAD BALANCING

Load balancing can be based on round robin, least connections, or random. If the application is not cloud-native and needs to maintain session state, LBaaS can be configured to always direct the request to the same VMs. Neutron supports its own proprietary and open-source LBaaS technologies to drive load balancing of requests, allowing the OpenStack operator to choose. Neutron LBaaS V2.0 is used for this reference architecture. V2.0 became available with the OpenStack Liberty release and supports Octavia as well as HAProxy backends. An alternative to Neutron LBaaS is to setup a software load balancer by launching instances with HAProxy.

## IMAGE MANAGEMENT

The OpenStack Image Service (Glance) allows us to store and retrieve VM images. There are multiple options and tools to provide configuration of servers when spawning instances of the web, services, and database VMs. On-the-fly configuration allows greater flexibility but can increase spawning time. The images can also be pre-configured to contain all of the files, packages, and patches required to boot a fully operational instance. Pre-configuration can reduce instance build time, but includes its own set of problems, such as patching and keeping licensing up-to-date. For this example, the orchestration features built into Heat are used to spawn and configure the three tiers of servers on-the-fly.

## PERSISTENT STORAGE

Similar to an external hard drive, Cinder volumes are persistent block-storage devices that may be mounted and dismounted from the VM by the operating system. Cinder volumes can be attached to only one instance at a time. This reference architecture creates and attaches Cinder volumes to the database VM to meet the performance requirements for the database tier. In the case of a database VM failure, a new VM can be created and the Cinder volume can be re-attached to the new VM.

Swift provides highly available, distributed, eventually consistent object/BLOB storage. Unlike a physical device, Swift storage is never mounted to the instance. Objects and metadata are created, modified, and obtained using the Object Storage API, which is implemented as a set of REpresentational State Transfer (REST) web services. If the web application requires hosting of static content (e.g. image, video), use Swift to store it, and configure Swift to serve the content over HTTP. In this reference architecture, Swift is also used for storing and archiving the database backup files.

## NETWORK SUBNETS

In addition to using the Neutron LBaaS feature for this workload, Neutron is used to create multiple subnets, one for each tier: a web subnet, an application subnet, and a data subnet. Neutron routers are created to route traffic between the subnets.

## NETWORK SECURITY

Filtering of inbound traffic is done through the use of security groups. Different security groups can be created and applied to the instances in each tier to filter unnecessary network traffic. OpenStack security groups allow specification of multiple rules to allow/deny traffic from certain protocols, ports, or IP addresses or ranges. One or more security groups can be applied to each instance. All OpenStack projects have a "default" security group, which is applied to instances that have no other security group defined. Unless changed, the default security group denies all incoming traffic.

### ORCHESTRATION

OpenStack Orchestration (Heat) uses template files to automate the deployment of complex cloud applications and environments. Orchestration is more than just standing up virtual servers. It can also be used to install software, apply patches, configure networking and security, and more. The Heat templates provided with this reference architecture allow the user to quickly and automatically set up and configure a sample eCommerce application on three-tier web application environment: OpenCart on LAMP.

### AUTO-SCALING

The ability to scale horizontally is one of the greatest advantages of cloud computing. Using a combination of Heat orchestration and Ceilometer, an OpenStack cloud can be configured to automatically launch additional VMs for the web and application tiers when demand exceeds preset thresholds. Ceilometer performs the system resource monitoring and can be configured to alarm when thresholds are exceeded. Heat then responds to the alarm according to the configured scale-up policy. Scaling can also be done in the opposite direction, reducing resources when the demand is low, saving money.

## Demonstration and Sample Code

This section describes the Heat templates provided as resources for this workload. They have been created for reference and training and are not intended to be used unmodified in a production environment.

The Heat templates demonstrate how to configure and deploy OpenCart, a popular open source eCommerce shopping cart application running on top of a three-tier LAMP architecture and OpenStack (reference: OpenStack Web Applications reference architecture). There are two versions of the primary template: one creates a static environment (manual scaling), and the other integrates with Ceilometer to provide auto-scaling of the web and services tiers based on CPU load.

The Heat templates can be downloaded from the OpenStack Community Application Catalog here. Each includes these products:

| TIER | FUNCTION | DETAILS |
|---|---|---|
| **Web** | Reverse Proxy Server | Apache + mod_proxy |
| **Services** | Application Server | OpenCart |
| **Database** | Database Server | MySQL |

## Heat file details

The Heat template uses a nested structure, with two different primary yaml files, both of which use the same four nested files. The files contain inline comments identifying possible issues and pitfalls when setting up the environment. The templates were tested using the Mitaka release of OpenStack, Ubuntu server 14.04, and Centos 7.

***eCommerceStatic.yaml:*** Run this yaml file for a static environment. It creates two load-balanced web servers, two load-balanced application (services tier) servers, and a single database server using Cinder block storage for the database. This yaml file uses Heat resource groups to call launch_services_layer.yaml and launch_web_layer.yaml, launching multiple copies of the web and application servers.

***eCommerceAutoScaling.yaml:*** For a dynamic auto-scaling environment, run this yaml file. It sets up Heat auto-scaling groups and Ceilometer alarms for both the web and services layers. The high-CPU Ceilometer alarms are configured by default to add an instance when the average CPU utilization is greater than 50% over a 10-minute period. The low CPU alarms are configured to remove an instance when the average CPU utilization drops below 20%. When configuring Ceilometer CPU alarms, it's important to keep in mind that the alarm by default looks at the average CPU utilization over all instances in the OpenStack project or tenant. Metadata can be used to create unique tags to identify groups of nodes, and then have the alarm trigger only when the average CPU utilization of the group exceeds the threshold. Ceilometer does not look at the CPU utilization on each of the instances; only the average utilization is reported. Another very important tip: ensure the selected "period" used to monitor the nodes is greater than the sampling rate configured in /etc/ceilometer/pipeline.config file. If the sampling rate is higher than the period, the alarm will never be activated.

The following yaml files are called by the primary files above:

- ***setup_network.yaml:*** This is the first file called by the main templates. This file creates three separate private networks, one for each tier. In addition, it creates two load balancers (using Neutron LBaaS V2.0): one with a public IP that connects the web presentation tier private network to the public network, and one with a private IP that connects the web presentation tier network to the services tier network. The template also creates a router connecting the services network to the database network. In addition to the networks and routers, the template creates three security groups, one for each of the tiers.
- ***launch_web_layer.yaml:*** This template file launches the web presentation tier nodes. In addition to launching instances, it installs and configures Apache and Apache modproxy, which is used to redirect traffic to the services tier nodes.
- ***launch_services_layer.yaml:*** This template file launches the services tier nodes. In addition to launching the instances, it installs Apache, PHP, MySQL client, and finally the OpenCart application.

- ***launch_sql_layer.yaml:*** This template file launches the database tier node. It also creates a Cinder block device to store the database files, and the required users and database for OpenCart. One database is deployed in this reference architecture, but multiples can be used.

## Scope and Assumptions

The Heat templates provided and described above assume that the eCommerce web application workload is deployed in a single-region, single-zone OpenStack environment. If the actual application requires higher SLA commitments, it is recommended to deploy OpenStack in a multi-zone, multi-region environment. This deployment is out of the scope of this reference architecture and will be described in a separate one.

As mentioned, Trove is not used in this implementation at this time. Trove is the OpenStack DBaaS that provisions relational and non-relational database engines. An update to this reference architecture to include Trove is under consideration. Containers are an emerging technology that can be applied to eCommerce, and will be discussed in a future article.

Another OpenStack service that would be suitable for the eCommerce architecture would be Neutron Firewall-as-a-Service (FWaaS) V2.0. FWaaS operates at the perimeter by filtering traffic at the Neutron router. This distinguishes it from security groups, which operate at the instance level. FWaaS details are also under consideration for a future update.

## Summary

There are many possible strategies and choices or strategies for an OpenStack eCommerce solution, and there are many possible variations in OpenStack deployment. This document is meant to serve as a generic reference architecture that can be used to deploy an eCommerce application on an OpenStack cloud. The basic architecture is intended to provide an example and show the reader how easily and quickly an eCommerce application can be deployed, using just a few OpenStack services. It does not include many of the additional OpenStack projects users may elect to implement, such as Trove, Sahara, Freezer, Monasca, etc.

In addition, the Heat orchestration service is used. Alternatively, popular third-party deployment options such as Chef, Puppet, or Ansible can be selected. The Heat templates are provided to help you get started and become familiar with OpenStack.

These additional resources are recommended to delve into more depth on overall OpenStack cloud architecture and the components and services covered in this reference architecture. The vibrant, global OpenStack community and ecosystem can be invaluable for their experience and advice. Visit openstack.org to get started or click on these resources to begin designing your OpenStack-based web applications.

| RESOURCE | OVERVIEW |
|---|---|
| OpenStack Marketplace | One-stop resource to the skilled global ecosystem for distributions, drivers, training, services and more. |
| OpenStack Architecture Design Guide | Guidelines for designing an OpenStack cloud architecture for common use cases. With examples. |
| OpenStack Networking Guide | How to deploy and manage OpenStack Networking (Neutron). |
| OpenStack Security Guide | Best practices and conceptual information about securing an OpenStack cloud. |
| OpenStack High Availability Guide | Installing and configuring OpenStack for high availability. |
| Complete OpenStack documentation | Index to all documentation, for every role and step in planning and operating an OpenStack cloud. |
| Community Application Catalog | Download this OpenCart-based eCommerce sample application and other free OpenStack applications here. |
| Welcome to the community! | Join mailing lists and IRC chat channels, find jobs and events, access the source code and more. |
| User groups | Find a user group near you, attend meetups and hackathons—or organize one! |
| OpenStack events | Global schedule of events including the popular OpenStack Summits and regional OpenStack Days. |