openstack
CLOUD SOFTWARE

OpenStack Workload Reference Architecture: Web Applications

Web applications are the most prevalent applications in business today. They are driven by user interaction over the Internet using a web browser front-end. Common web applications include webmail, online retail sales, online auctions, online banking, instant messaging services, and more.

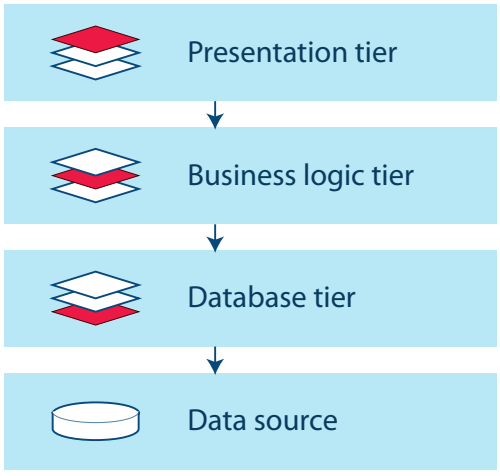
Web applications are typically characterized by IT resource requirements that fluctuate with usage, predictably or unpredictably. Failure to respond to either can impact customer satisfaction and sales. An automatically scaling web application and underlying infrastructure can be essential. Unlike a traditional, static environment, cloud computing allows IT resources to scale dynamically, both up and down, based on the application-generated load (CPU utilization, memory, etc.).

The OpenStack cloud platform offers auto-scaling for web applications as well as a comprehensive platform for all IT applications, offering agility and cost-effectiveness. OpenStack is open source cloud software that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard or API. Thousands of enterprises use OpenStack to run their businesses every day.

Intended for enterprise IT architects, this reference architecture describes the architecture and services required by a simple three-tier web application, using popular LAMP software on an OpenStack cloud. LAMP consists of Linux, Apache, MySQL, and PHP/Python/Perl and is considered by many as the platform of choice for development and deployment of high performance web applications.

We identify and recommend the required and optional OpenStack services for both a static virtualized implementation and a fully dynamic auto-scaling implementation. Lastly, we will provide tested implementation files you can use to install and instantiate an OpenStack web application environment using Wordpress as the sample application. These files are Heat templates that will create the virtual servers for each tier, networking, load balancing, and optionally, auto-scaling.

Figure 1: Three-tier web application architecture overview



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:
Craig Sterrett, *Software Architect,* Intel Corporation
Yih Leong Sun, *PhD, Senior Software Cloud Architect,* Intel Corporation
Shamail Tahir, *Offering Manager,* IBM

OpenStack for Web Applications

A three-tier web application consists of the web presentation, the application, and persistent database tiers.

- Web presentation tier – cluster of web servers that will be used to render either static or dynamically generated content for the web browser.
- Application tier – cluster of application servers that will be used to process content and business logic.
- Database tier – cluster of database servers that store data persistently.

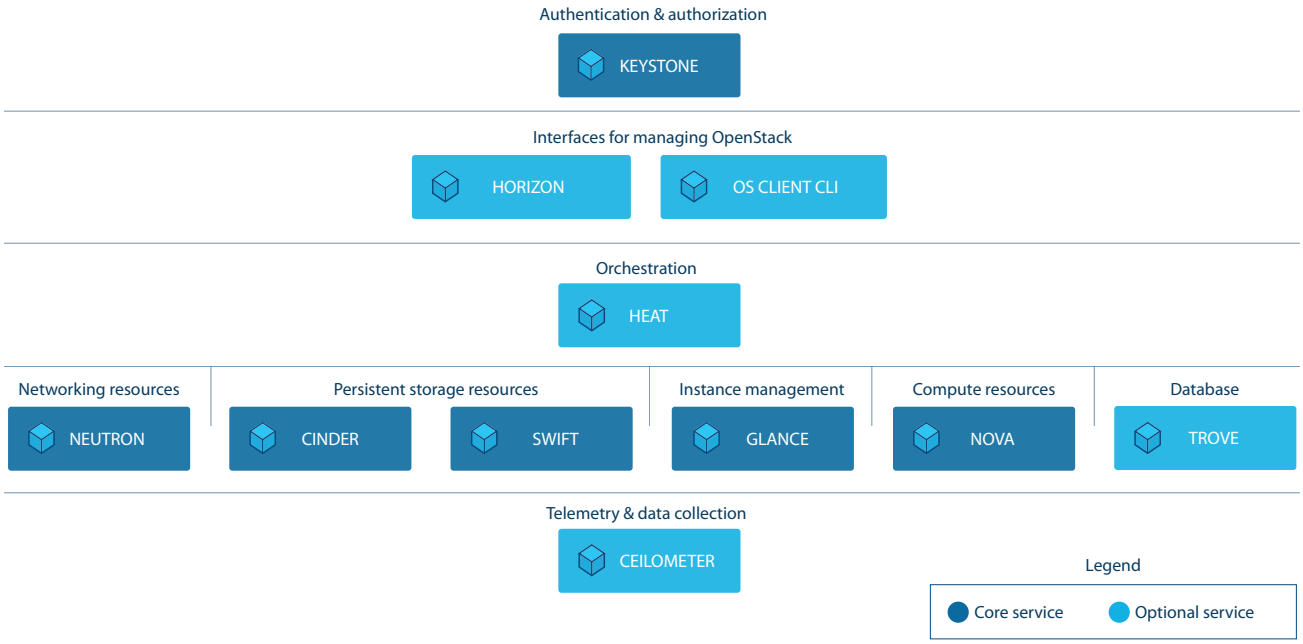
An OpenStack cloud is powered by many different services (also known as projects). Utilizing only the **core services,** a three-tier web services application can be deployed in a virtualized environment that can be **manually** scaled up and down as required with minimal effort.

Optional services can be added for more functionality:

- OpenStack Orchestration service (Heat project) allows automating workload deployment.
- Together, Orchestration and Telemetry (Ceilometer) enable **dynamic scaling** as load increases and decreases.
- OpenStack Database service (Trove) provides Database-as-a-Service (DBaaS) to automate database provisioning and administration. Trove is an option for web applications on OpenStack but is not used in this basic reference architecture.

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

Figure 2. Logical representation of OpenStack services for web applications



Brief descriptions of the core and optional services used for simple three-tier web applications follow. The OpenStack Project Navigator provides additional information.

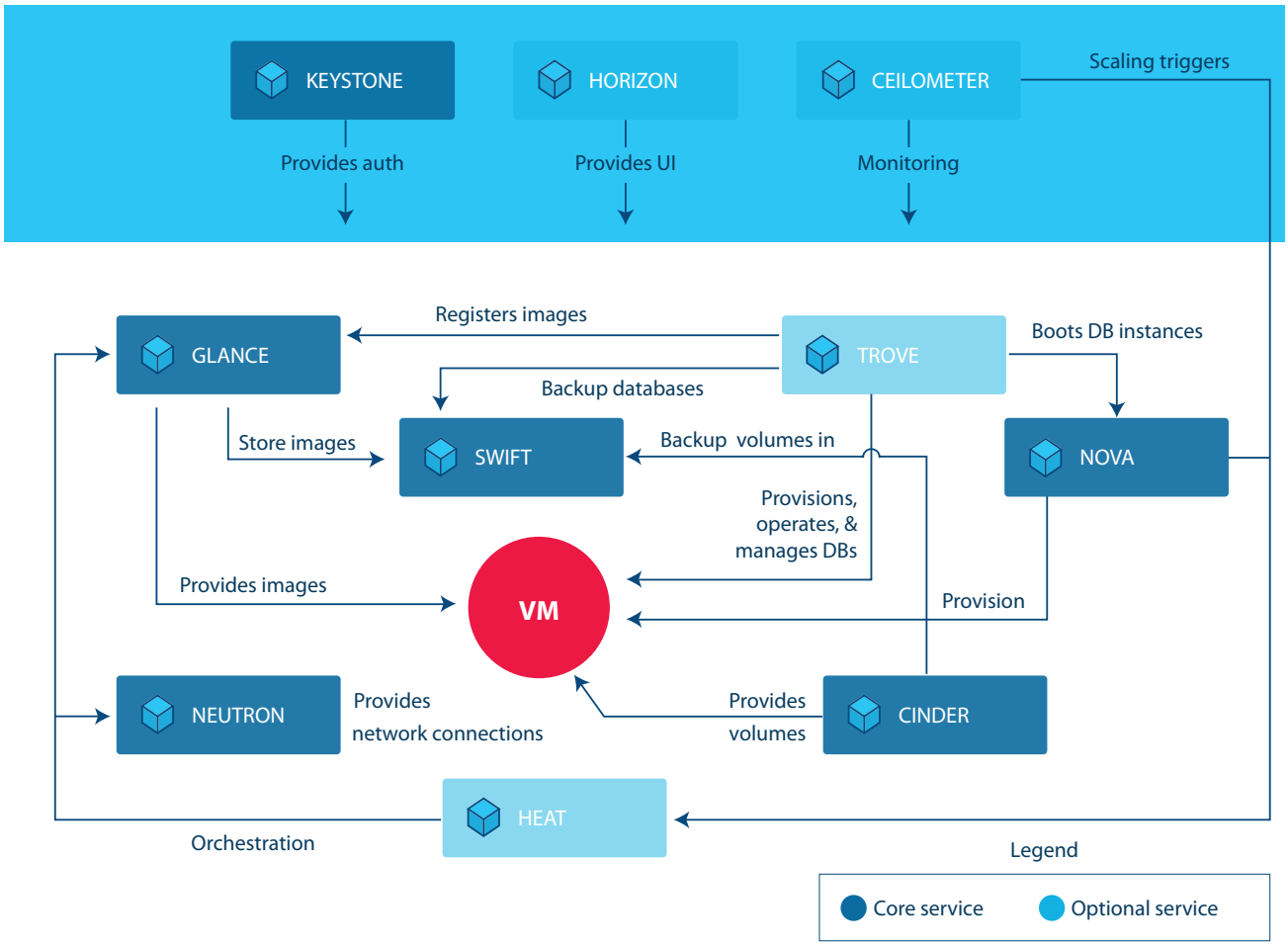
COMPUTE (NOVA)	Manages the life cycle of compute instances, including spawning, scheduling, and decommissioning of virtual machines (VMs) on demand.
IMAGE SERVICE (GLANCE)	Stores and retrieves VM disk images. Used by OpenStack Compute during instance provisioning.
BLOCK STORAGE (CINDER)	Virtualizes the 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 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.
DASHBOARD (HORIZON)	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.

Optional services

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.
TELEMETRY (CEILOMETER)	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
DATABASE (TROVE)	A database-as-a-service that provisions relational and non-relational database engines.

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

Figure 3. Functional interaction between OpenStack components



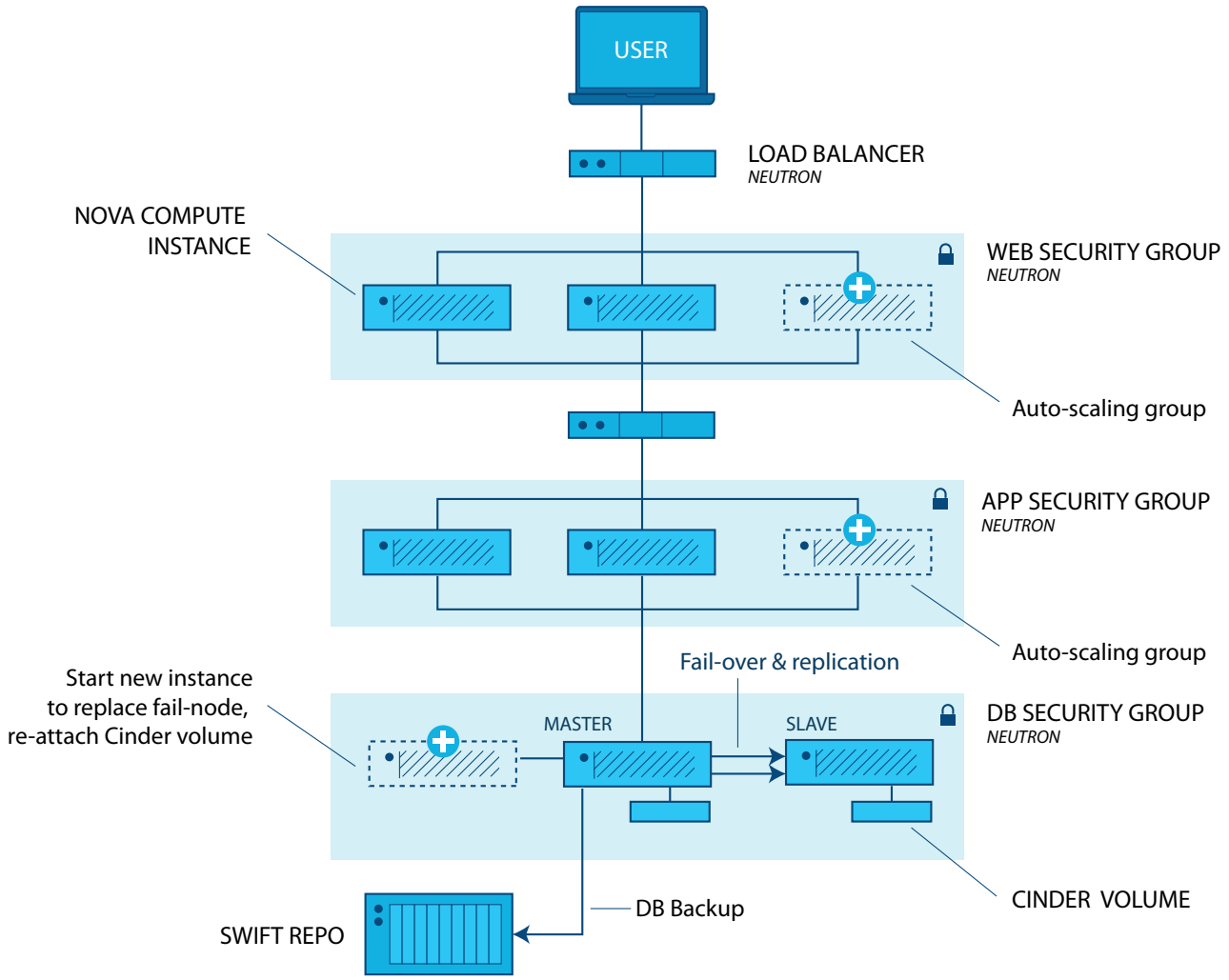
Structuring an OpenStack Web Application

Generally a three-tier web application consists of a web presentation tier, application tier, and persistent database tier. This chapter discusses these and additional architectural components and considerations for an OpenStack-based web application.

ARCHITECTURAL COMPONENTS	DESCRIPTION
Web presentation tier	A cluster of web server used to render static or dynamically generated content for the web browser.
Application tier	A cluster of application servers used to process content and business logic.
Database tier	A cluster of database servers used to store data persistently.

ARCHITECTURAL COMPONENTS	DESCRIPTION
Load balancers	Two load balancers are required to equally distribute load. The first load balancer distributes the web traffic at the presentation tier. A separate load balancer is required to distribute the load among the application servers.
Relational Database Management System (RDBMS)	The database tier used in this example uses a master/slave RDBMS configuration. Data is kept in persistent block storage and backed-up periodically.
Firewalls	For security, a set of firewall rules must be enforced at each tier.
Network configuration	The network must be configured to filter unnecessary traffic at different tiers.
Auto-scaling	Auto-scaling is desirable to automatically respond to unexpected traffic spikes and resume to normal operation when the load decreases.

Figure 4: OpenStack web application architecture



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, Load-Balancing-as-a-Service (LBaaS) can be configured to always direct the request to the same VMs. Neutron allows for proprietary and open-source LBaaS technologies to drive load balancing of requests, allowing the OpenStack operator to choose. Neutron LBaaS V1.0, is used for this reference architecture. V2.0 is 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

There are multiple options and tools to provide configuration of servers when spawning instances of the web, application, 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 licenses 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 virtual 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 a Cinder volume to the database VM to meet the data persistency 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

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

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 setup and configure a LAMP-based web services environment.

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 WordPress, a popular web application, on a three-tier LAMP architecture. There are two versions of the primary template: one that creates a static environment (manual scaling) and one that integrates with Ceilometer to provide auto-scaling of the web and application tiers based on CPU load.

The Heat templates can be downloaded from http://www.openstack.org/software/sample-configs#web-applications

TIER	FUNCTION	DETAILS
Web	Reverse Proxy Server	Apache + mod_proxy
App	WordPress Server	Apache, PHP, MySQL Client, WordPress
Data	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 Mitaka release of OpenStack, and Ubuntu server 14.04 and Centos 7.

WebAppStatic.yaml: Run this yaml file for a static environment. It creates a static environment with two load-balanced web servers, two load-balanced application servers, and a single database server using Cinder block storage for the database. This yaml file utilizes Heat resource groups to call heat_app_tier.yaml and heat_web_tier.yaml, launching multiple copies of the web and application servers.

WebAppAutoScaling.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 application tiers. The high-CPU Ceilometer alarms are configured by default to add an instance when the average CPU utilization is greater than 50% over a five-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_net_sg.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 V1.0): one with a public IP that connects the web tier private network to the public network, and one with a private IP that connects the web tier network to the application tier network. The template also creates a router connecting the application network to the database network. In addition to the networks and routers, the template creates three security groups, one for each of the tiers.
- ***heat_web_tier.yaml:*** This template file launches the web tier nodes. In addition to launching instances, it installs and configures Apache and Apache modproxy, which is used to redirect traffic to the application nodes.
- ***heat_app_tier.yaml:*** This template file launches the application tier nodes. In addition to launching the instances, it installs Apache, PHP, MySQL client, and finally WordPress.
- ***heat_sql_tier.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 databases for the WordPress application.

Scope and Assumptions

The Heat templates provided and described above assume that the three-tier web application workload is deployed in a single-region, single-zone OpenStack environment. If the actual application requires higher SLA commitment, 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 OpenStack DBaaS that provisions relational and non-relational database engines. An update to this reference architecture to include Trove is under consideration.

Another OpenStack service that would be suitable for the three-tier architecture would be Neutron Firewall-as-a-Service (FWaaS). 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 is also under consideration for a future update.

Summary

There are many strategies for deploying a three-tier web application and there are choices for each OpenStack deployment. This reference architecture is meant to serve as a general guide to be used to deploy the LAMP stack on an OpenStack cloud using core and selected optional services. The Heat orchestration service is used; however, popular third-party deployment products such as Chef, Puppet, or Ansible can also be used. Other OpenStack services can be selected to enhance this basic architecture with additional capabilities.

This document shows how easily and quickly a three-tier LAMP and Wordpress environment can be implemented using just a few OpenStack services. We offer the Heat templates 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 LAMP/WordPress 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.