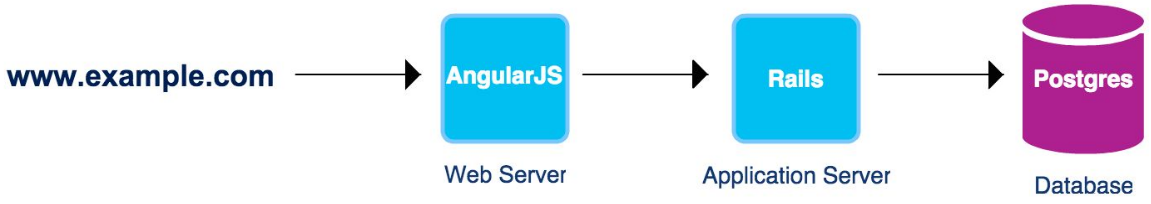
TITLE:
OpenStack with Docker

Team ID:
17

Team Members:
Ruchir Sharma 201405558
Priyanka Bajaj 201405540
Shikher Somal 201201091
Jaspreet Singh 201202078

Problem Description:

Objective: Configure OpenStack with docker and deploy a three-tier application. We want to create a three tier web architecture composed of a web server, an application server and a database server. Here is the architectural diagram for an example application:



Creating the containers: *A Dockerfile is a script used by Docker to build a container, step-by-step, layer-by-layer, automatically from a source base image.*

There will be three docker containers, one for each tier. These will be build using already available source based images.

Database: This is our persistence layer that will be populated with some demo data representing customers. The database will be a MYSQL database.
API Server: This tier represents Restful API services. It exposes the database.
WebClient: This is a really simple tier that demonstrates a web application. We are going for Python web server that just serves up a static HTML page along with corresponding Javascript.

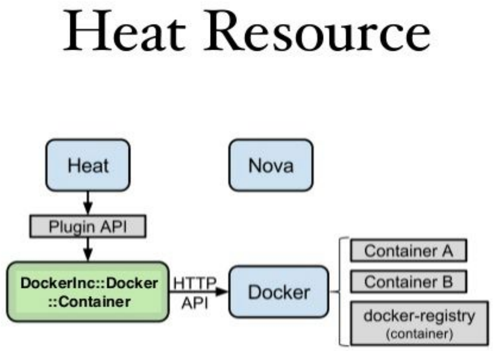
Background

N-tier denotes a software engineering concept used for the design and implementation of software systems using client/server architecture divided into multiple tiers. This decouples design and implementation complexity, thus allowing for the scalability of the deployed system.

In a three-tier application, the user interaction is managed by the presentation tier, which provides an easy-to-operate front end. The business rules are managed by the business tier, which controls and operates the entire application framework. The underlying data is stored and served by the data storage tier, also known as data persistence.

Since, the three tiers are loosely coupled to each other, with predetermined and stable interfaces. This decoupling allows for significant changes to occur within the design, implementation and scale of each tier, without impacting the other tiers.

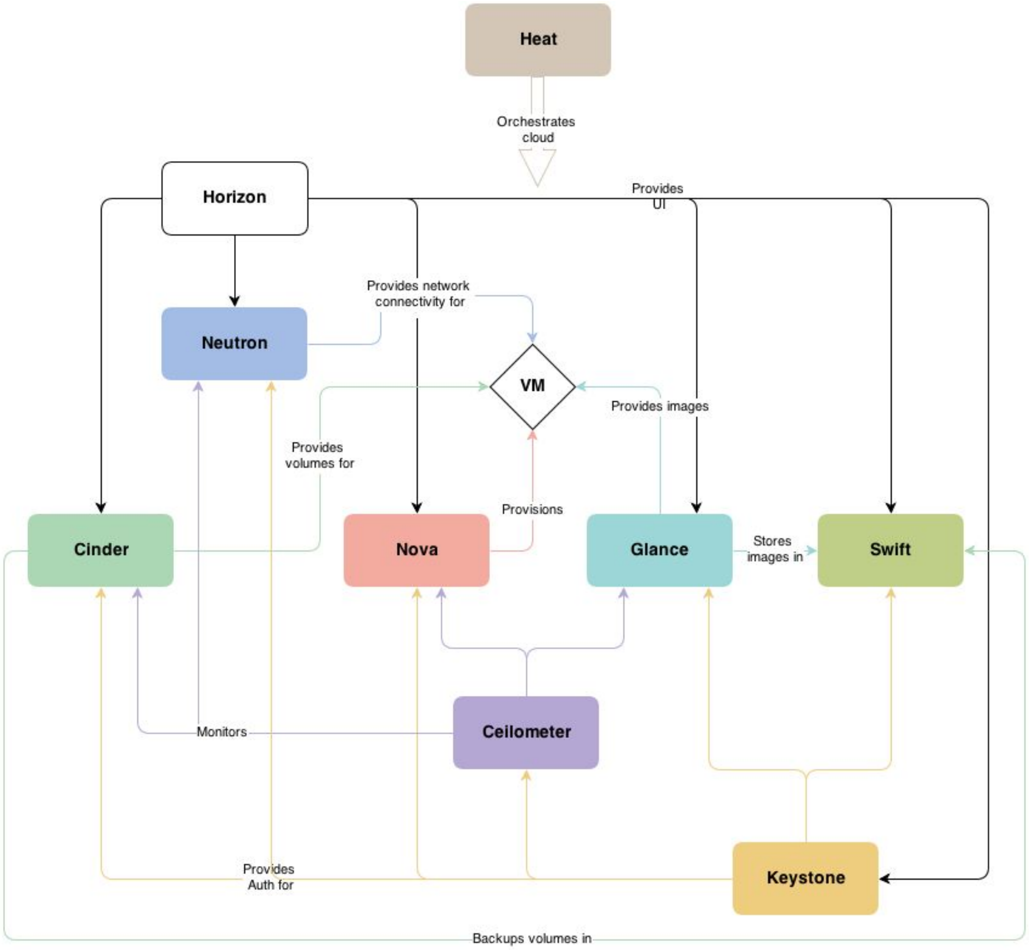
Docker suits this functionality really well, as it provides the containers to remain in isolation from one another, while exposing interfaces so containers can talk to each other. When the containers are created, the -P flag is used to automatically map any network port inside it to a random high port within an *ephemeral port range* on the Docker host. In case of three-tier applications, the web tier can expose itself by allowing clients to send requests to this port. While, the app tier would be responsible for handling all the logic aspect for this, and thus allow the web tier to talk to the app tier, and the app tier can then further talk to the database tier.



HEAT

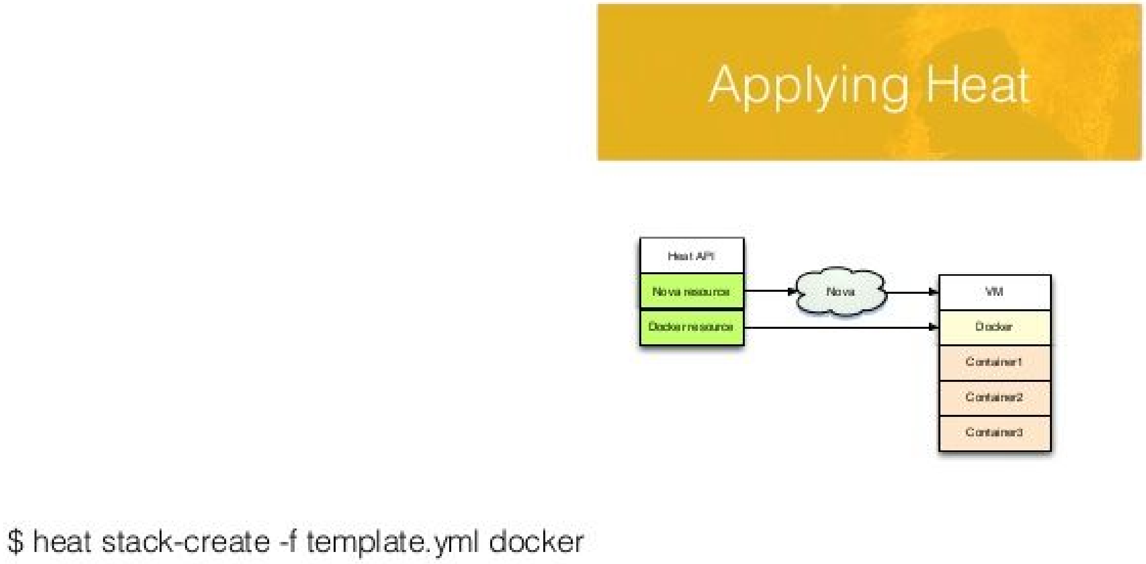
Heat orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

The following diagram shows how heat works as an orchestration API to manage resources.



This functionality suits the fundamental design of Docker containers as each container can be mapped to a certain tier. Instead of manually deploying these containers using Docker client API, we will be using the OpenStack Heat plugin to deploy these containers as micro services on a VM which was spawned by Openstack.

High Level Design: OpenStack with Docker



Applying Heat

```
$ heat stack-create -f template.yml docker
```

Nova typically manages VMs. In this approach, Nova driver is extended to spawn Docker Containers. The Linux Containers deployed with Docker have multiple advantages over the “normal” virtual machines usually deployed by Nova. Those advantages are speed, efficiency, and portability.

Scope:

Creating a Heat Orchestration Template file

i. Heat template specifies resources
ii. Each resource is part of container
iii. Specify the image of the container
iv. Specify the environment variables that it may need e.g. three-tier-app.yaml in the repository.

Running HEAT to deploy the application using the template.

Once the stack has been created, presentation layer is hosted and this is been connected to the Business/App tier, which can further be connected to a database layer in another container.

Technologies used:
OpenStack is a free and open-source cloud-computing software platform. Users primarily deploy it as an infrastructure-as-a-service (IaaS). The technology consists of a group of interrelated projects that control pools of processing, storage, and networking resources throughout a data center—which users manage through a web-based dashboard, through command-line tools, or through a RESTful API.

Docker is an open-source engine which automates the deployment of applications as highly portable, self-sufficient containers which are independent of hardware, language, framework, packaging system and hosting provider.