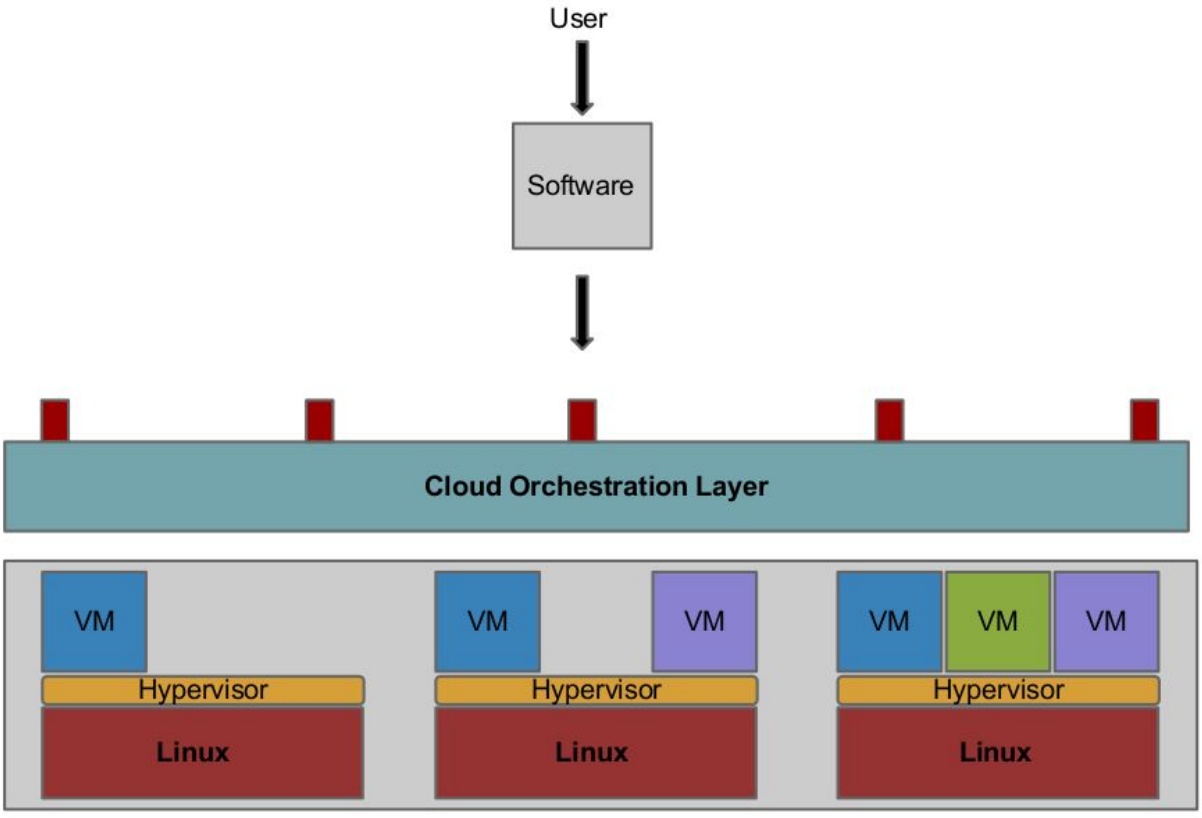
Orchestration Details

Consider the example of a data center consisting of a huge infrastructure - hundreds of servers, huge data store. A cloud orchestration suite acts a transparency layer. It combines all the resources and shows them as one(a cloud) to the users. Users can demand variable resources from the orchestration layer and pay as per the usage. This is a well established cloud model and as you witnessed in the demo during the tutorial, cloud orchestration layer plays a big role here.

An orchestration suite provides a number of services and allows the users access to these in the form of service endpoints. The small red boxes on top of the orch. layer in the figure below depicts these endpoints.



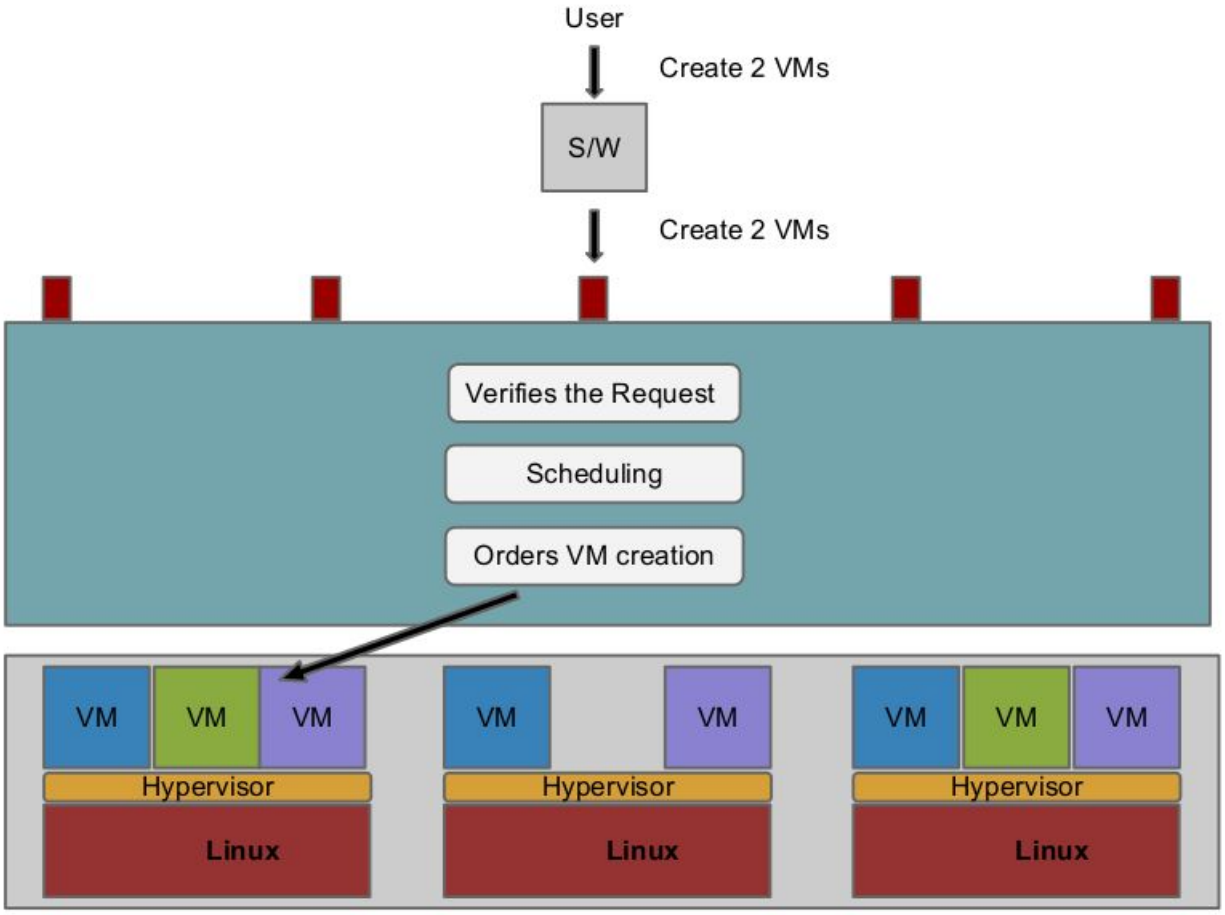
An endpoint is a contact point where an application or a user can contact the respective service of the system through the API. In our project, the endpoint is web based(a URL) and the API are based on HTTP calls. You can read more about them in the project description document.

These endpoints can be of the following nature:

- **Cloud computing fabric controller.** It controls the lifecycle of Virtual Machines(VMs) - spawn, pause, terminate etc.
- **Storage Endpoint.** It controls and serves storage requirements - providing a raw virtual hard disk of any desired size.
- **Identity Endpoint**. It controls user management - authentication and authorization.
- **Networking Endpoint**. It allows the creation of different virtual networks with different routing rules applying to different VMs.
- **Object Store Endpoint.** It provides a highly scalable and durable storage infrastructure. Its similar to a distributed file system of huge capacity and high availability.

For this project, you are only required to implement the first one.

Let us consider a small example where a user requests for 2 VMs. Here is a rough sketch of the steps involved.



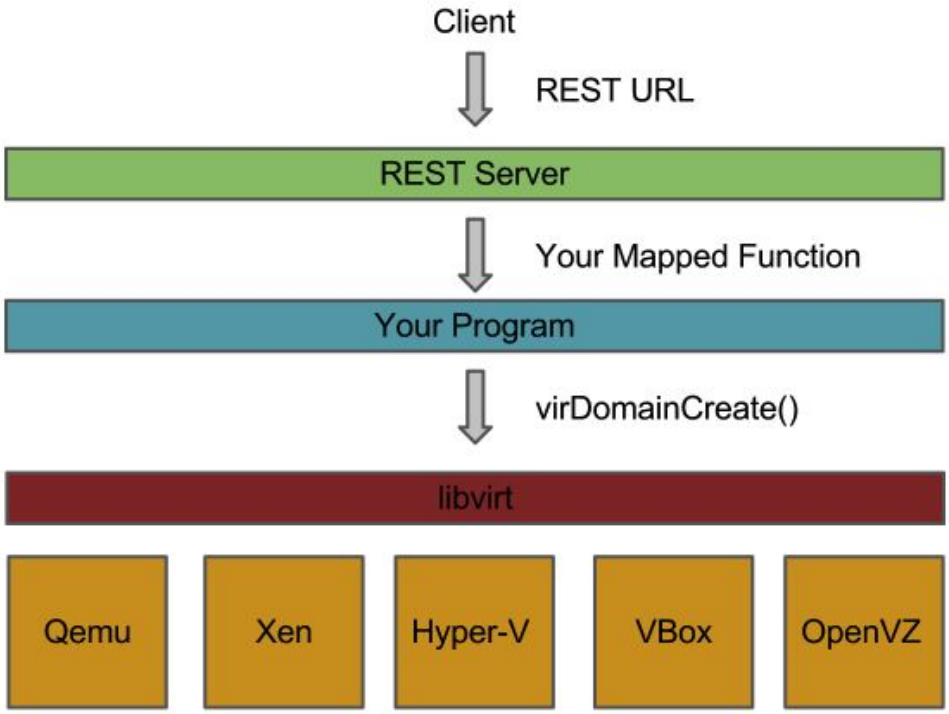
Lets take a closer look at each step taken in the orchestration layer.

- Request Verification - A web server(apache or your own code) will be listening on port 80 for incoming requests. Once the request is received, based on the URL, the required action will be decided. Refer to the API guide in the project description document for more details. In this case, the action is to spawn two instances.
- Scheduler - At this step, the task is clear - spawn 2 Virtual machines. The scheduler's job is to decide which hosts are suitable for spawning these VMs. One can use a variety of algorithms for the scheduler. A good design would be to modularize scheduler implementation. This will allow easy plugin and testing of various algorithms.
- Order VM Creation - Once the hosts are decided, libvirt is used to contact the hypervisors. The benefit of using libvirt is the ease of controlling various types of hypervisors using the same code.
- Status Reporting to User - The user must be notified about the status (success/failure) of the request.

FAQ:

1. Why libvirt ?
Ans: LibVirt provides a common API for virtualization. This allows one to use the same code for managing and handling a multi-hypervisor environment.

2. What I need to do ? or How do I start ?
Ans: If it is not clear from the above description following picture should make things


Client
REST URL
REST Server
Your Mapped Function
Your Program
virDomainCreate()
libvirt
Qemu
Xen
Hyper-V
VBox
OpenVZ