Application and Network Orchestration Using HEAT & TOSCA

Nati Shalom

@natishalom

Samuel Bercovici

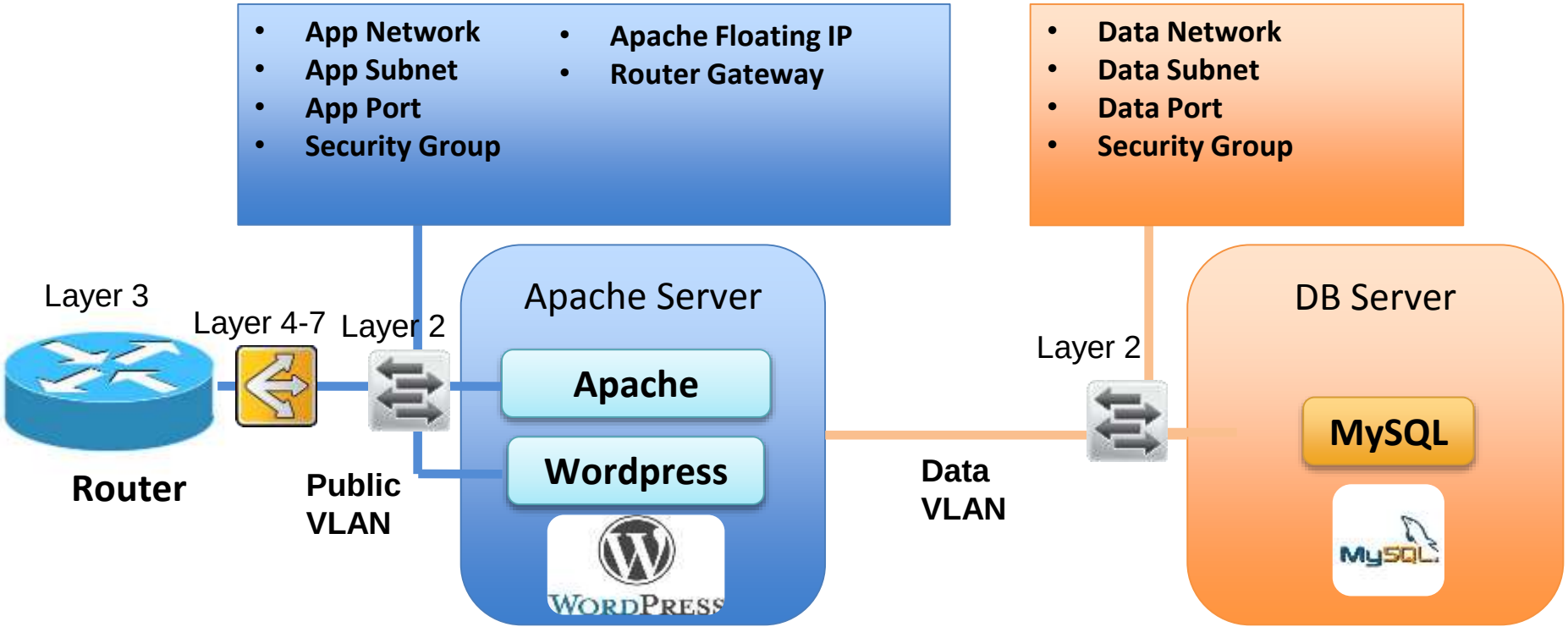
@samuelbercovici

Agenda

- Introduction
  - Introduction to Networking & Neutron
  - Introduction to Orchestration with HEAT & TOSCA
- Orchestrating Network & Applications by Example
  - Heat + Ceilometer Auto-Scaling Demo
  - Portable Version using TOSCA
- Real World (NFV) Example –
  - Creating “Your own Skype” On-demand.
- Summary

Let’s Start With a Quick Overview...

Networking & Application



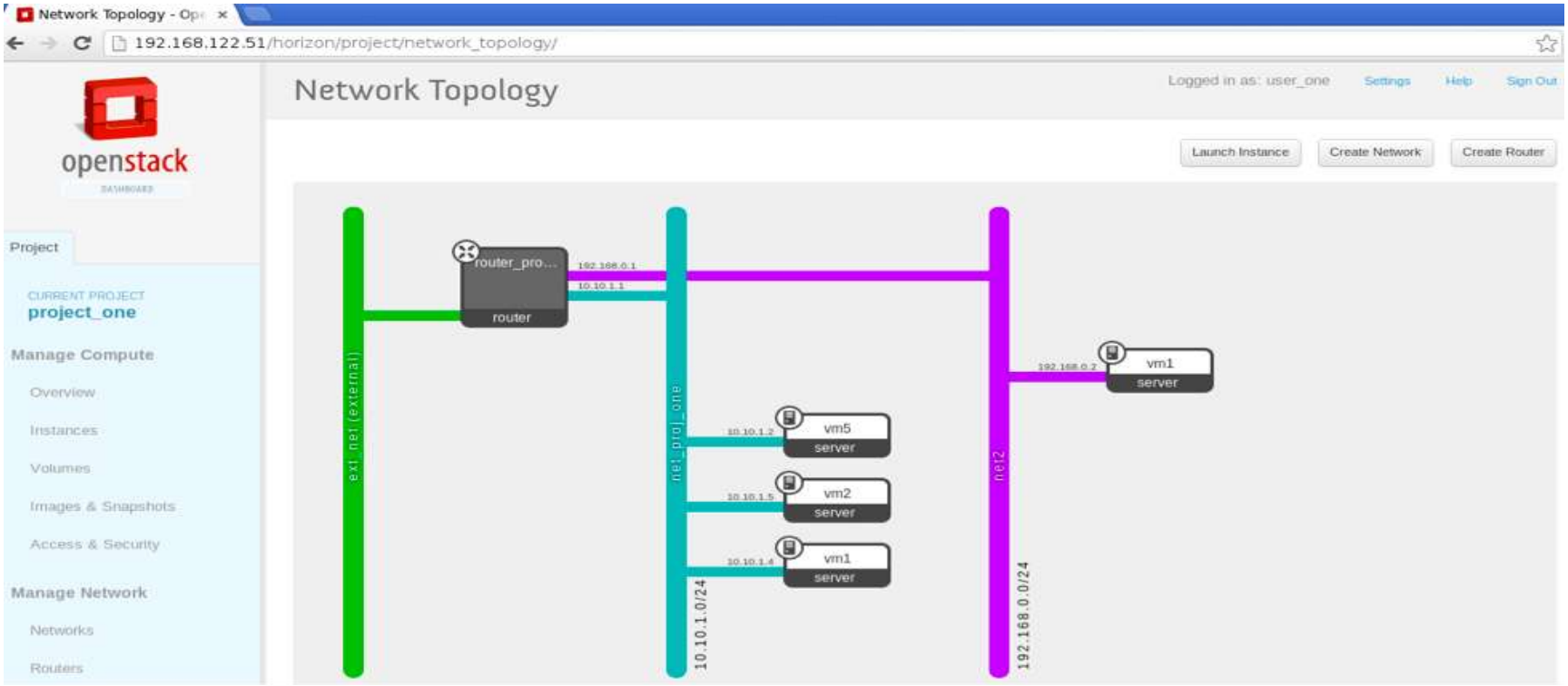
What is Neutron?

Tenant Facing API to manage L2-L7 network logical elements

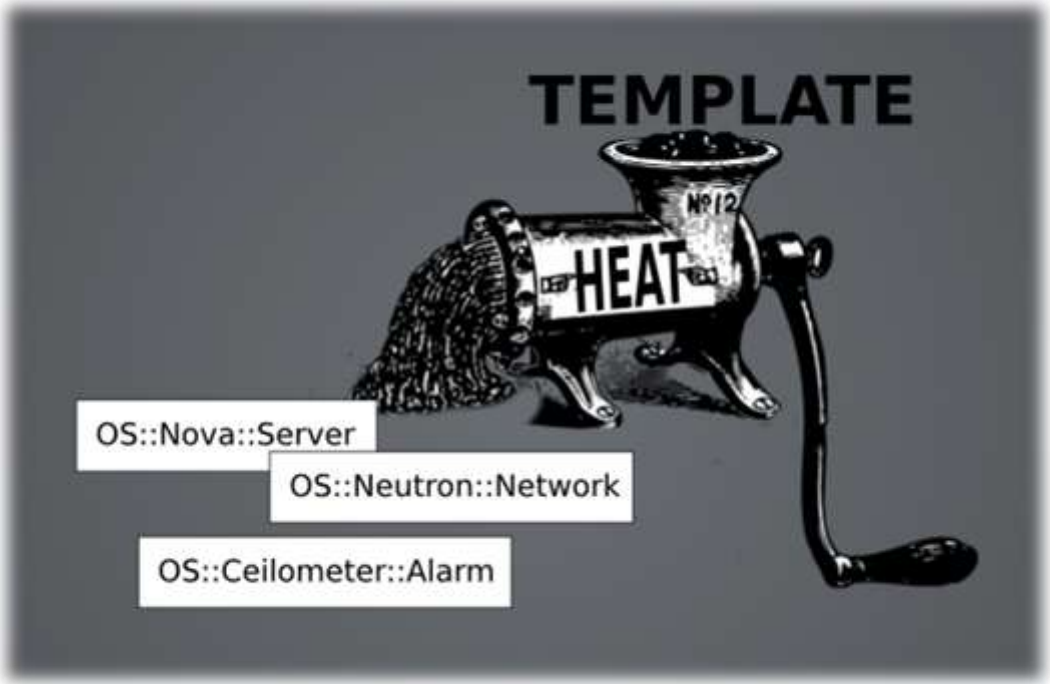
- Layer 2 networks
- Layer 3 subnets
  - IP address management – DHCP based
  - Router / gateway / NAT
  - Port (Security groups, Floating IP)
- Layer 4-7 Services - Load balancing, VPN, Firewall




Networking in OpenStack Horizon

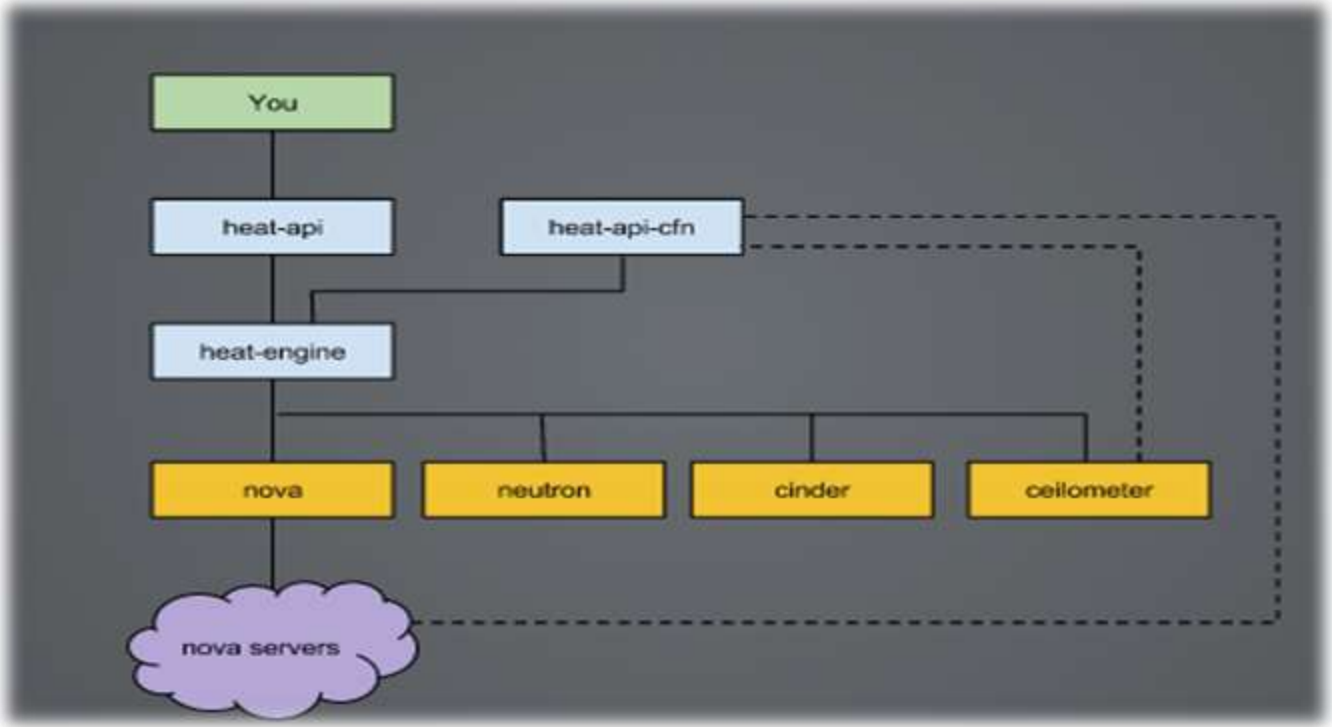


What is Heat?

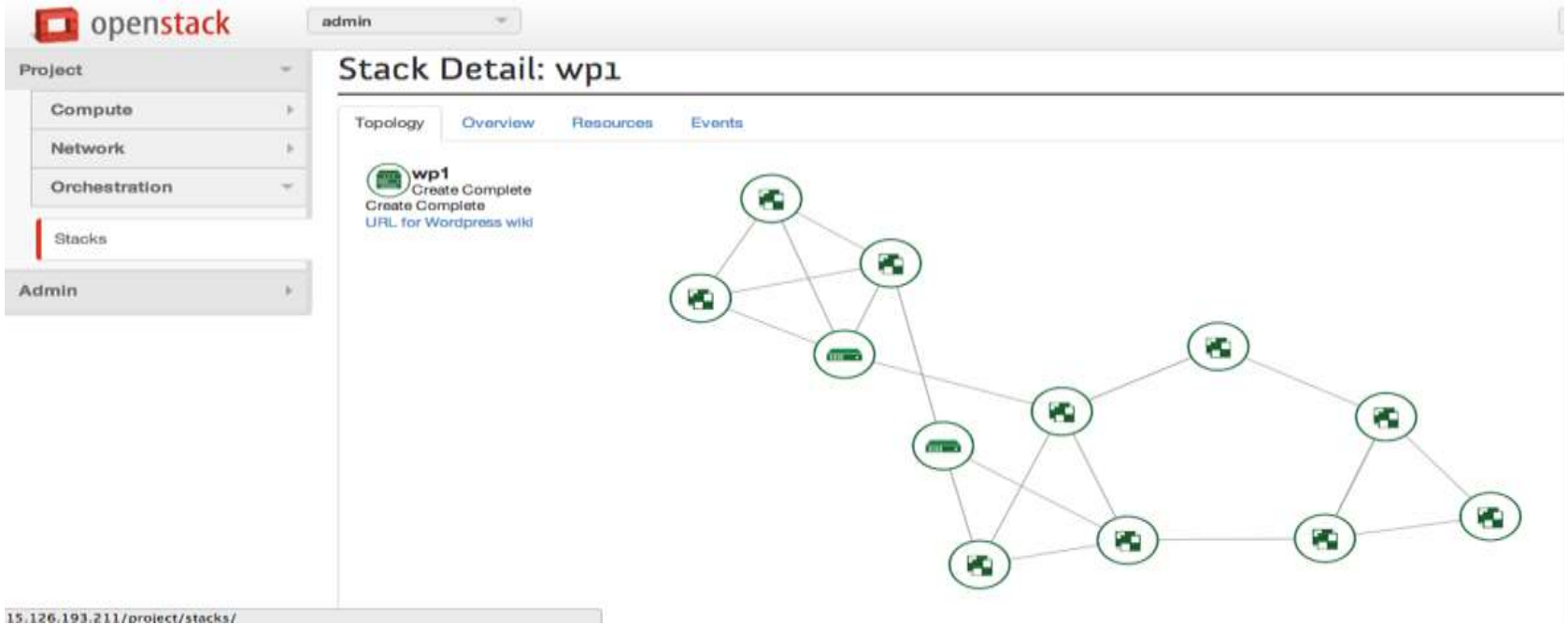


Heat provides a mechanism for orchestrating OpenStack resources through the use of modular templates.

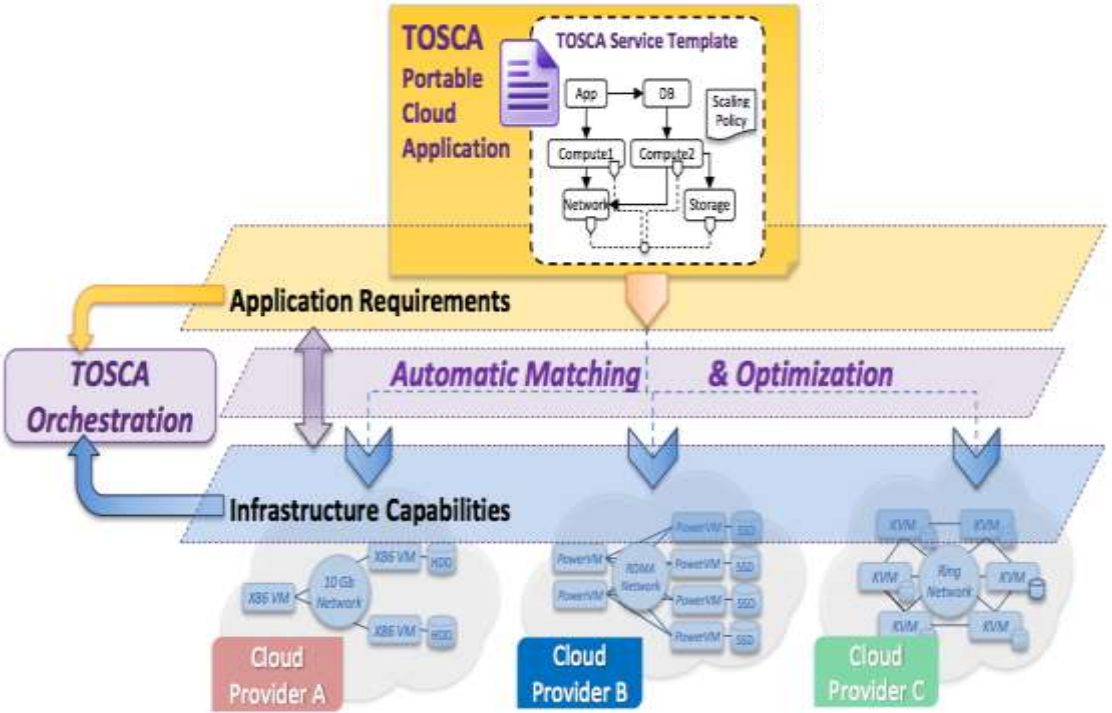
Heat Architecture



Heat Topology View

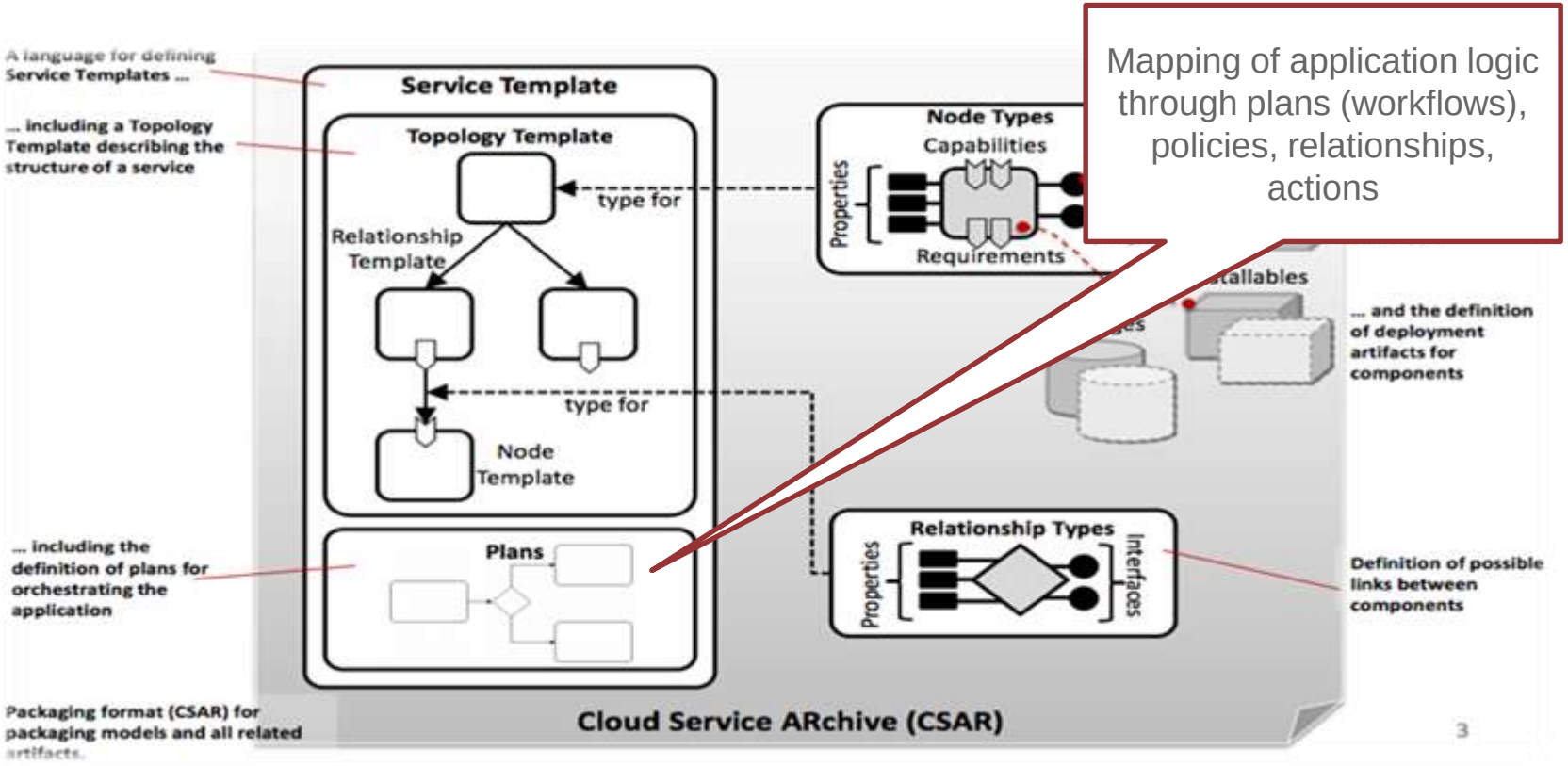


What is TOSCA?



TOSCA defines the interoperable description of applications; including their components, relationships, dependencies, requirements, and capabilities….

TOSCA in a Nutshell



TOSCA State of the Union

- Top four cloud open standard (Forrester)
- 5000+ participants
- 65+ countries
- TOSCA Parser Integrated into the Heat Project

The focus of this session

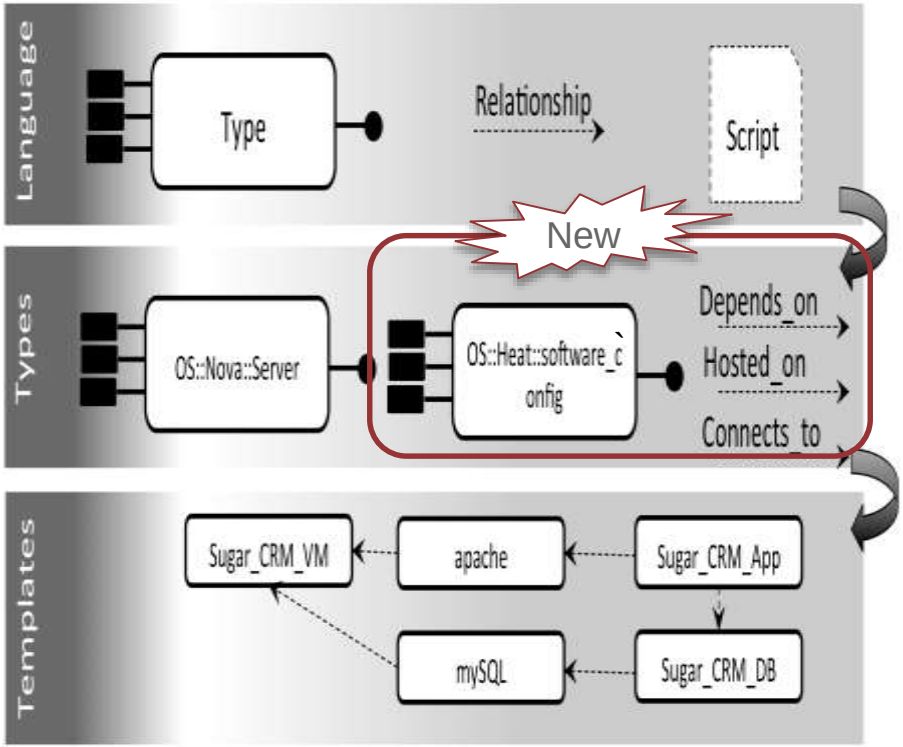
Comparing TOSCA & Heat

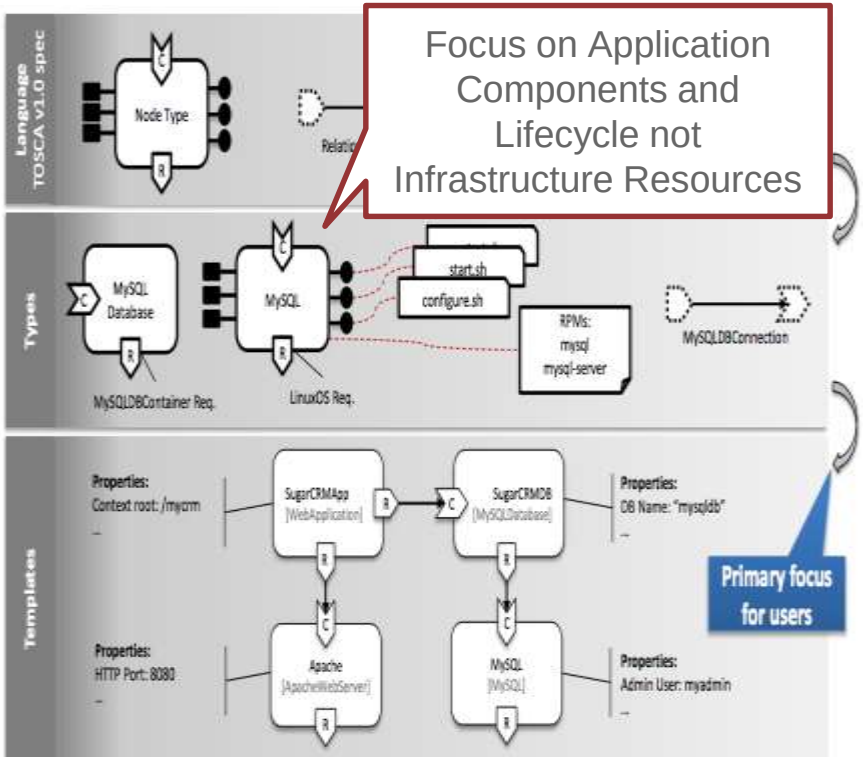
- **Heat** – Automate the configuration and setup of **OpenStack resources**
- Specific to OpenStack

- **TOSCA** – Automation of the **application** deployment and management lifecycle
- Portable

Merging Concepts

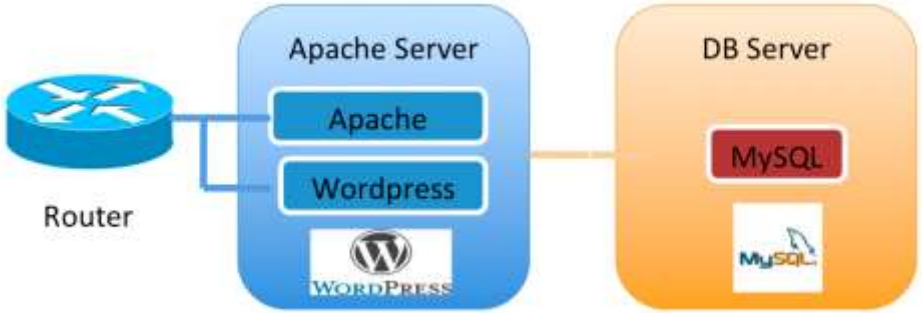
Heat/HOT vs TOSCA Syntax





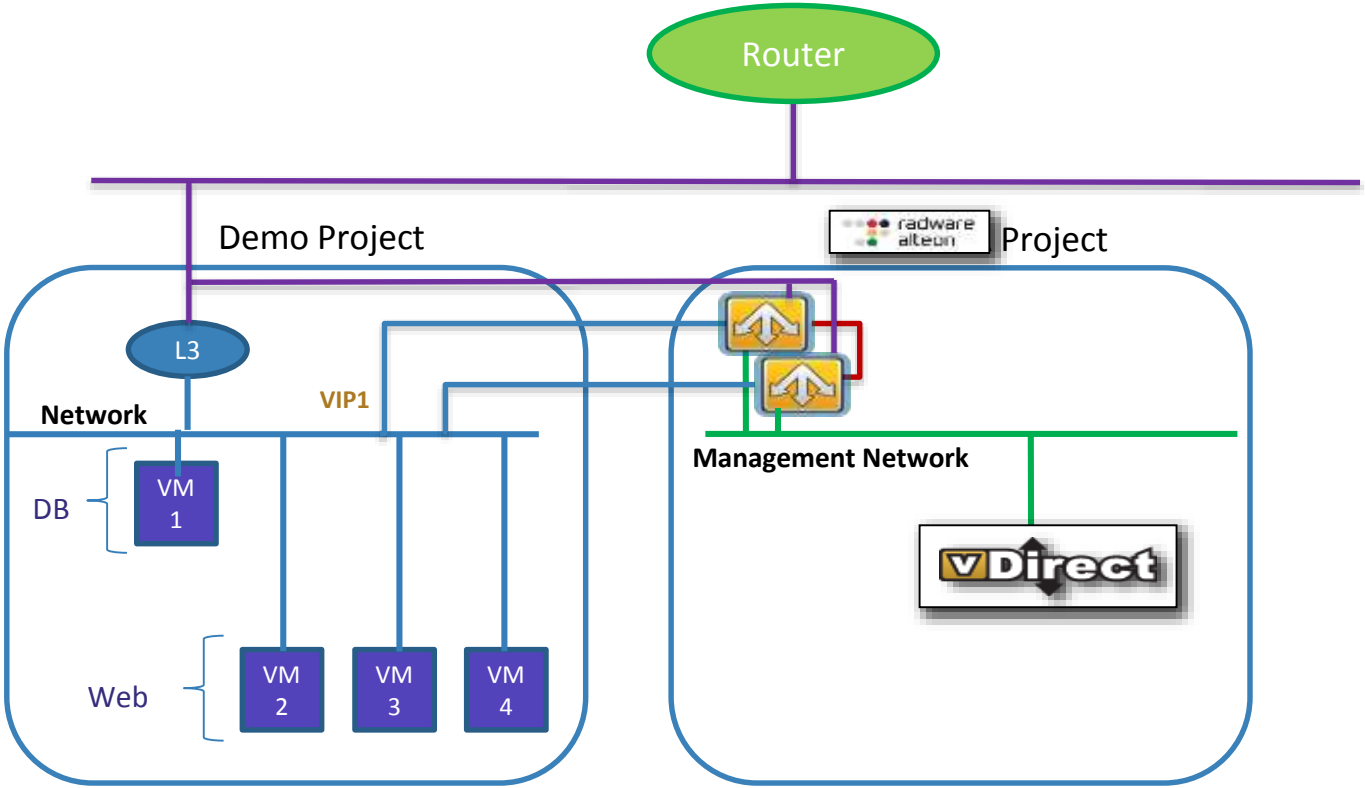
Orchestrating Networks & Applications by Example...

- Define Networks as Parameters
- Create a New WordPress Stack
- WordPress Scale-out
- WordPress Scale-in



Reference Hot Templates: autoscaling.yaml and lb_server.yaml

Demo Scenario



openstack™
CLOUD SOFTWARE

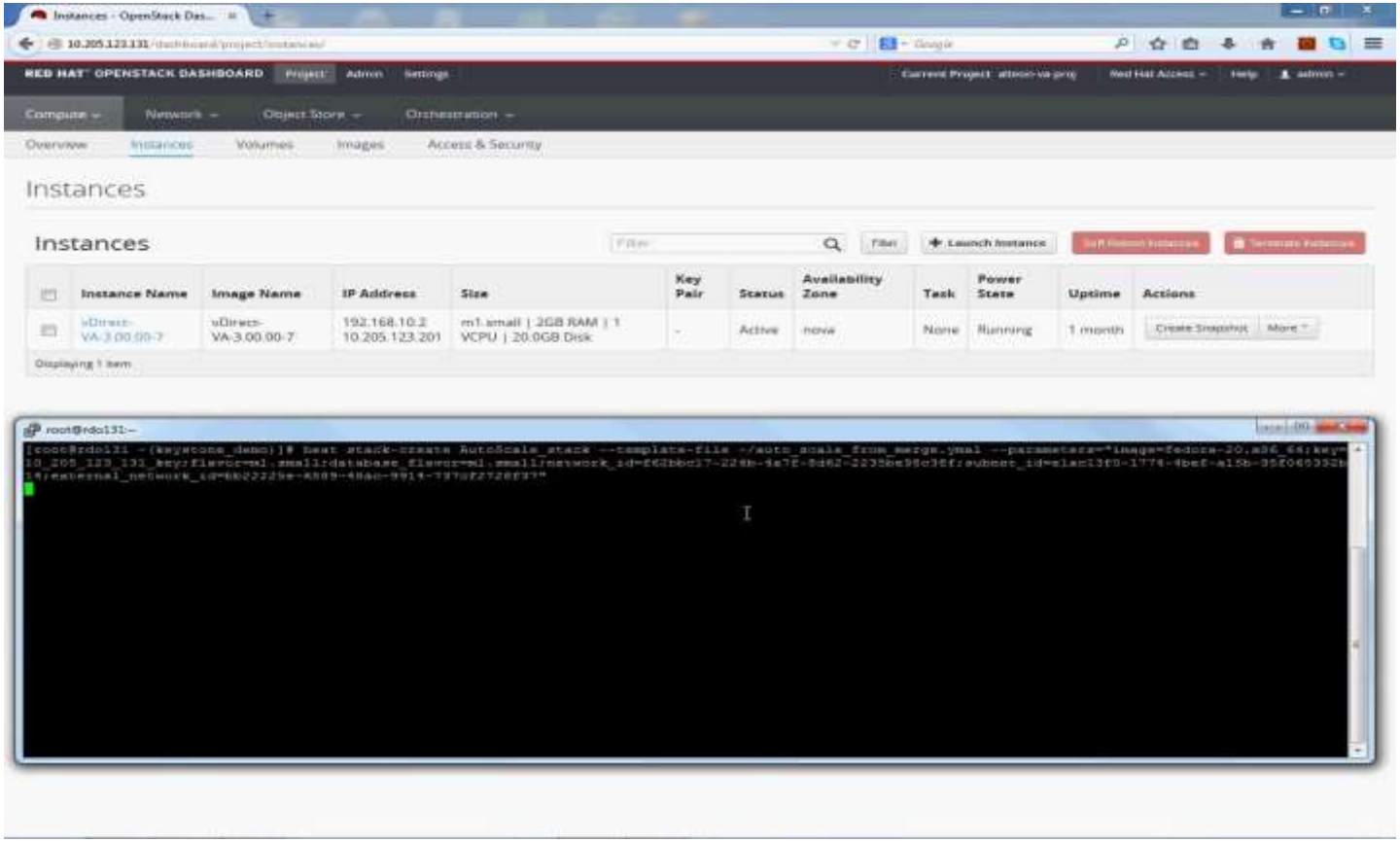
Auto Scaling - Key Elements

```
web_server_group:
  type: OS::Heat::AutoScalingGroup
  properties:
    min_size: 1
    max_size: 3
    resource:
      type: lb_server.yaml
      properties:
web_server_scaleup_policy:
  type: OS::Heat::ScalingPolicy
  properties:
web_server_scaledown_policy:
  type: OS::Heat::ScalingPolicy
  properties:
cpu_alarm_high:
  type: OS::Ceilometer::Alarm
  properties:
cpu_alarm_low:
  type: OS::Ceilometer::Alarm
  properties:
monitor:
  type: OS::Neutron::HealthMonitor
  properties:
pool:
  type: OS::Neutron::Pool
  properties:
lb:
  type: OS::Neutron::LoadBalancer
  properties:

# assign a floating ip address to the load balancer
# pool.
lb_floating:
  type: "OS::Neutron::FloatingIP"
  properties:
```

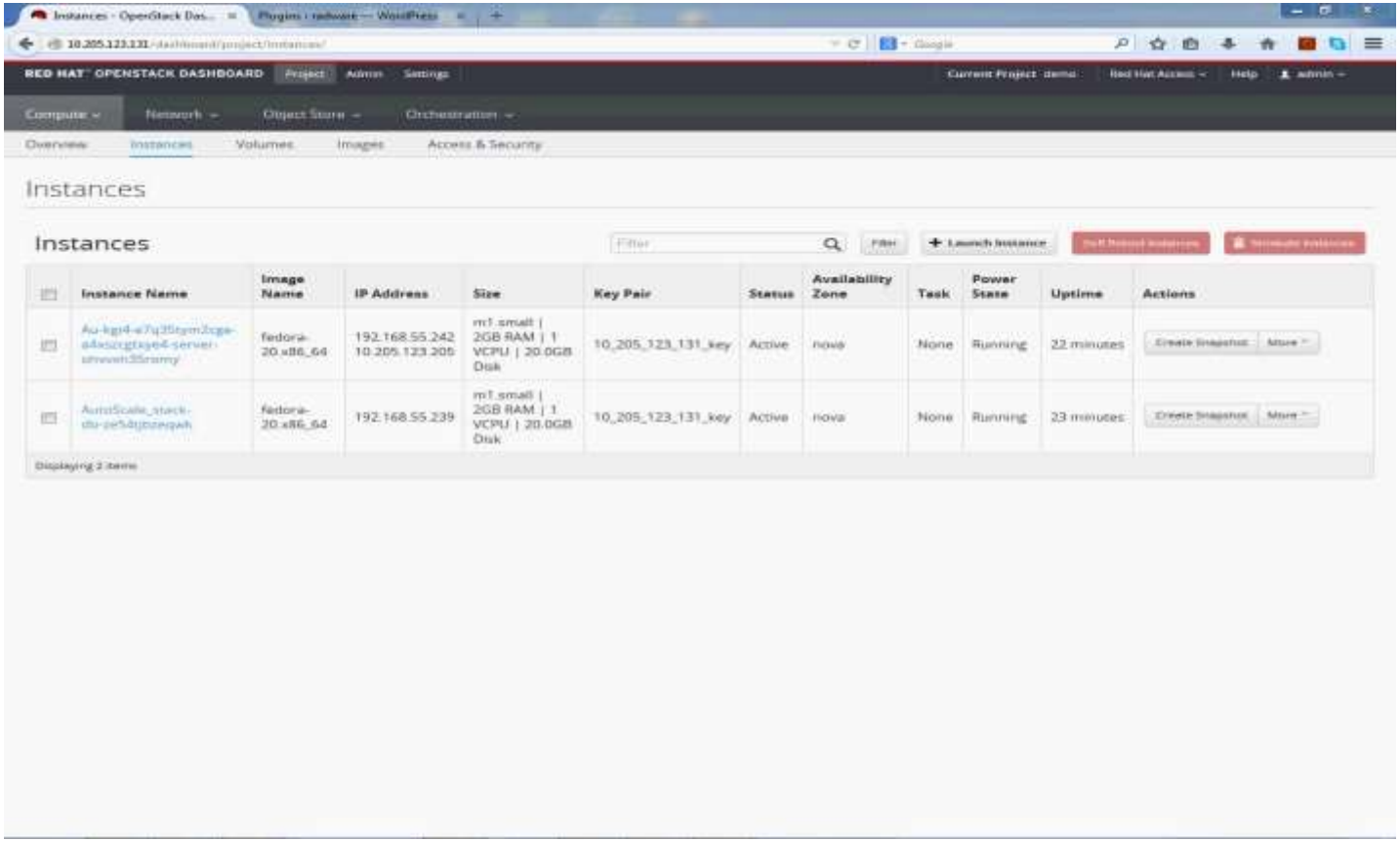
Create New Stack

```
#heat stack-create AutoScale_stack
  --template-file ~/autoscale.yaml
  --parameters="image=fedora_image_for_web_and_db;
                        key=some_key;
                        flavor=m1.small;database_flavor=m1.small;
                        network_id=servers_network_id;
                        subnet_id=servers_subnet_id;
                        external_network=floating_ip_network
```

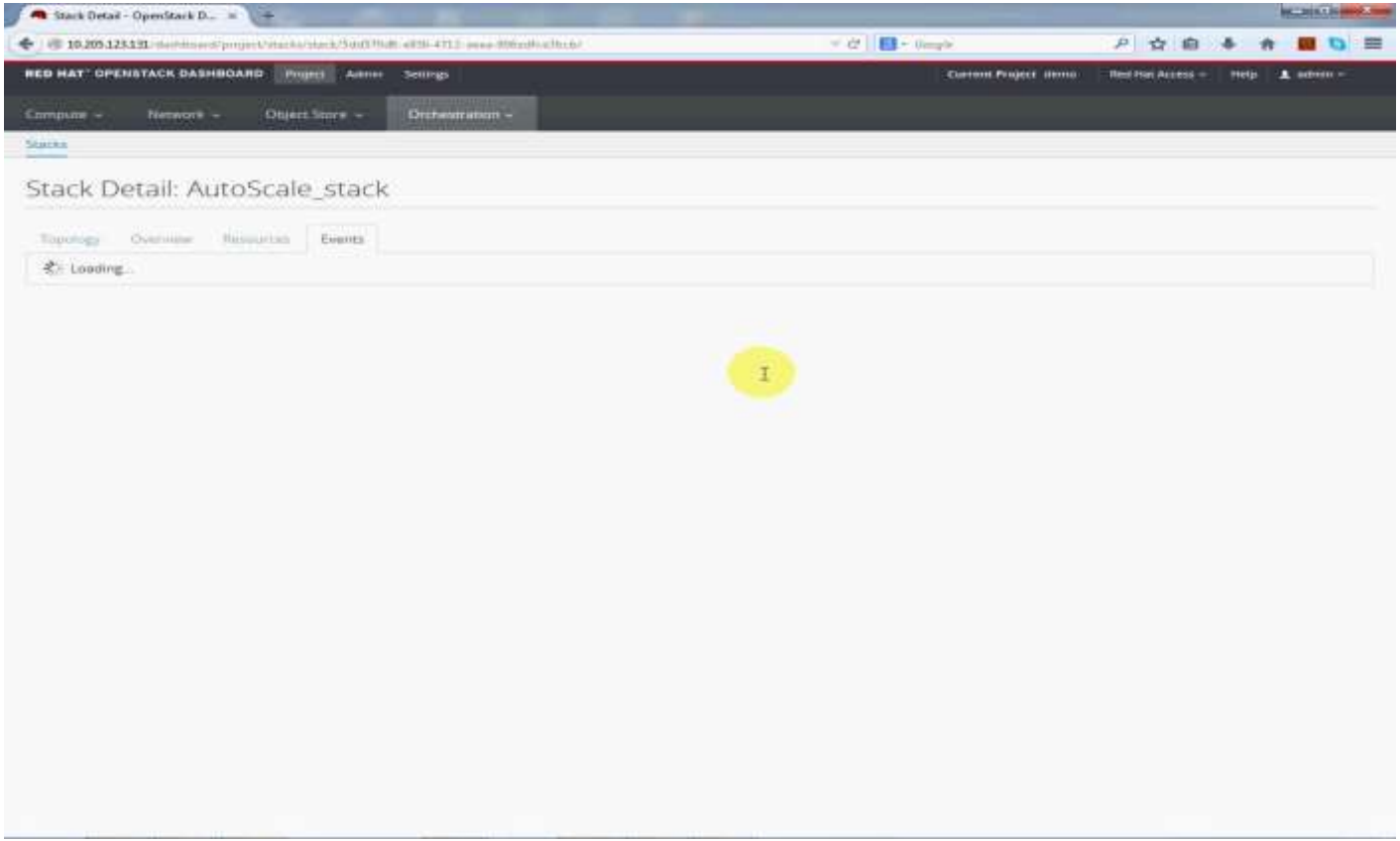
Create New Stack



WordPress Scale-out



WordPress Scale-In



Portable TOSCA Alternative

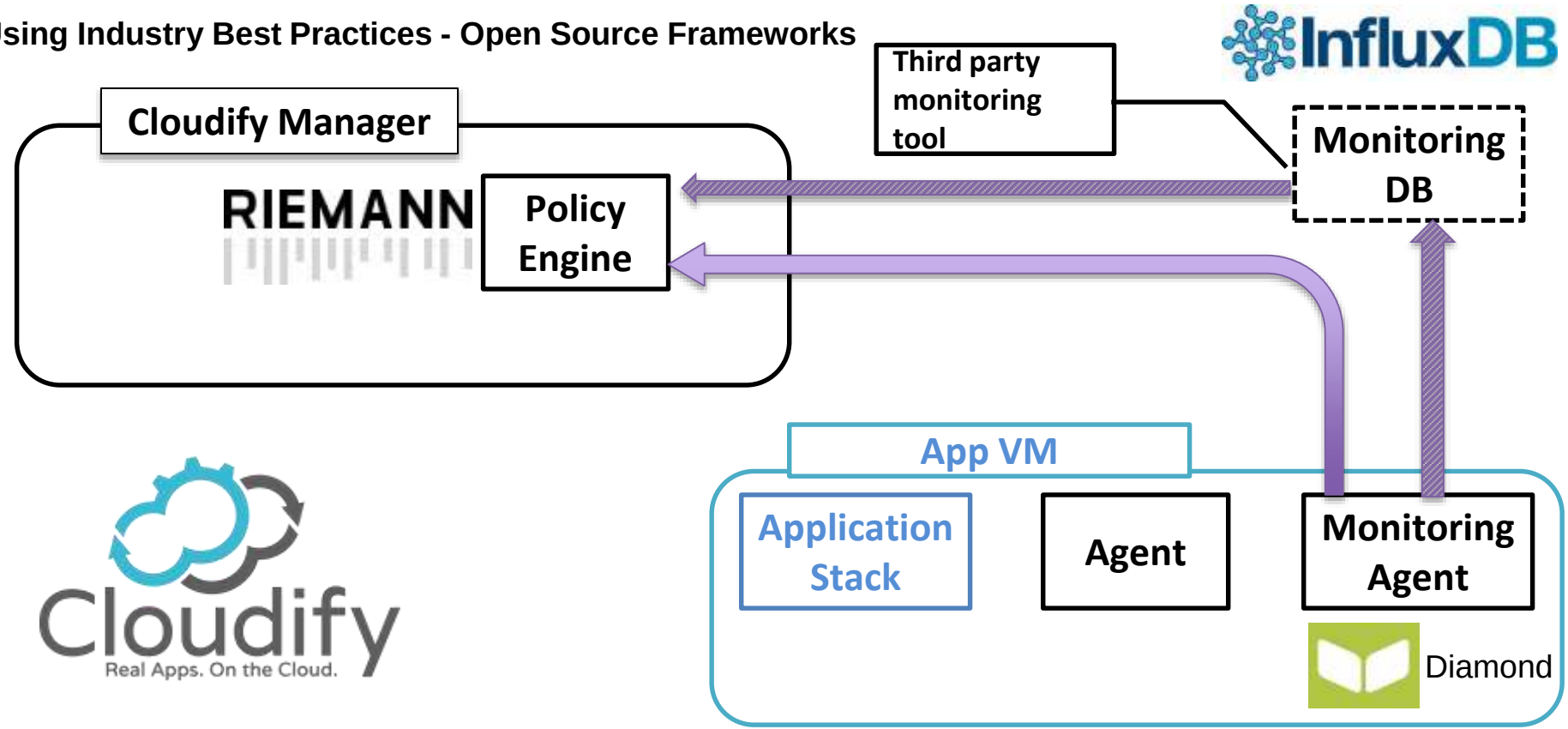
```
node_templates:
  wordpress:
    type: tosca.nodes.WebApplication.WordPress
    properties:
      admin_user: { get_input: wp_admin_username }
      admin_password: { get_input: wp_admin_password }
      db_host: { get_property: [ db_server, ip_address ] }
    requirements:
      - host: wordpress_webserver
      - database_endpoint: wordpress_db

  wordpress_webserver:
    type: tosca.nodes.WebServer.Apache
    properties:
        http_port: 8080
        https_port: 8443
    requirements:
      - host: web_server
    interaces:
      Lifecycle:
        inputs:
          db_host: { get_property: [ db_server, ip_address ] }
          db_port: { get_property: [ wordpress_db, db_port ] }
          db_name: { get_property: [ wordpress_db, db_name ] }
          db_user: { get_property: [ wordpress_db, db_user ] }
          db_password: { get_property: [ wordpress_db, db_password ] }
        configure: scripts/installandconfigurewp.sh

  web_server:
    type: tosca.nodes.Compute
    properties:
      # Compute properties
      num_cpus: 2
      mem_size: 4
      disk_size: 10
      # host image properties
      os_arch: x86_64
      os_type: linux
      os_distribution: ubuntu
      os_version: 12.04
```

Software components are defined just like any other resource

Infrastructure resource matched by their properties

Portable Auto-Scaling

Using Industry Best Practices - Open Source Frameworks



Portable TOSCA (Like) Metrics

```
wordpress_vm
    type: cloudify.openstack.server
    properties:
       server: { get_input: server }
       image: { get_input: image }
       flavor_name: { get_input: flavor_name }
       security_groups: { get_input: security_groups }
    relationships:
        -   type: cloudify.relationships.depends_on
            target: Everything_Allowed
    interfaces:
        cloudify.interfaces.monitor_lifecycle:
            - start:
                mapping: diamond.diamond_agent.tasks.install
                properties:
                    diamond_config:
                    interval: 5
                    collectors:
                        CPUCollector: {}
            - stop: diamond.diamond_agent.tasks.uninstall
```

Set Diamond Collector

Cloudify
Real Apps. On the Cloud.

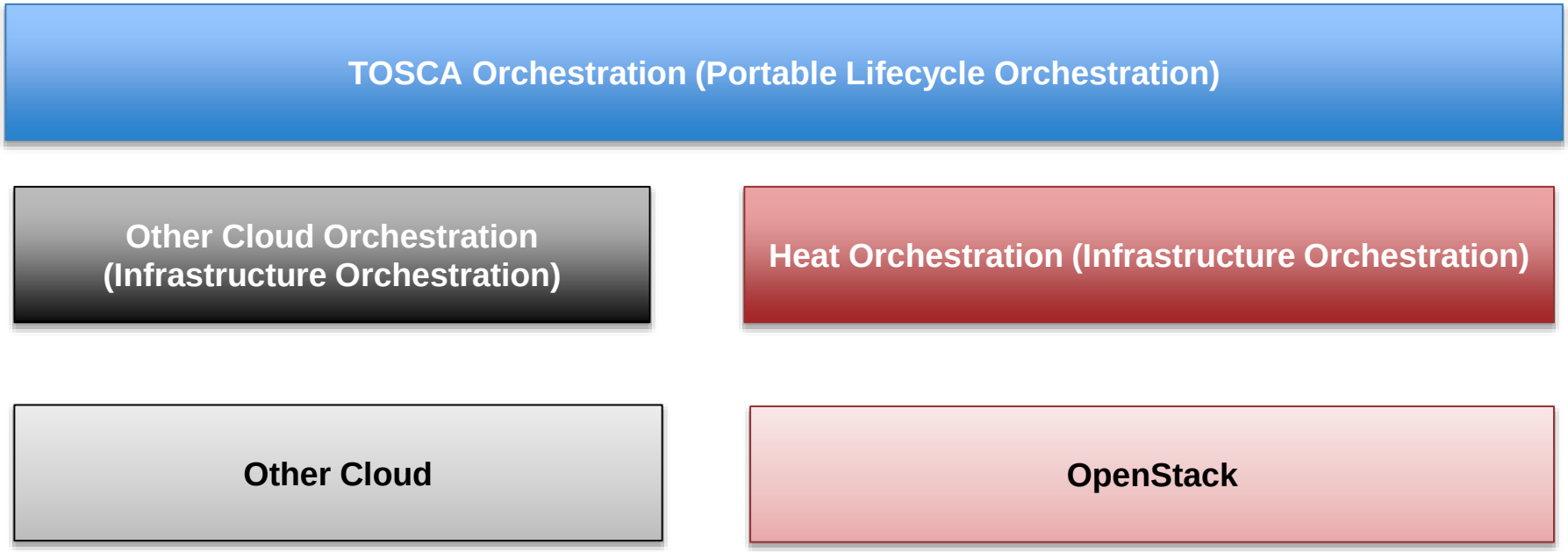
Portable TOSCA (Like) Auto-Scaling

```
groups:
    wordpress_group:
        members: [wordpress_app]
        policies:
            monitoring_policy:
                type: threshold_policy
                properties:
                    metric: CPU_PERCENTAGE
                    upper_bound: 50
                    lower_bound: 15
                triggers:
                    scaleup_trigger:
                        type: cloudify.policies.triggers.execute_workflow
                        parameters:
                            workflow: scaleup
                            workflow_parameters:
                                node_id: { get_properly: [SELF, node_id ] }
                                scale_by: 1
                    scaledown_trigger:
                        type: cloudify.policies.triggers.execute_workflow
                        parameters:
                            workflow: scaledown
                            workflow_parameters:
                                node_id: { get_properly: [SELF, node_id ] }
                                scale_by: 1
```

Set Auto-Scaling Policy Based on the App Metric

Cloudify
Real Apps. On the Cloud.

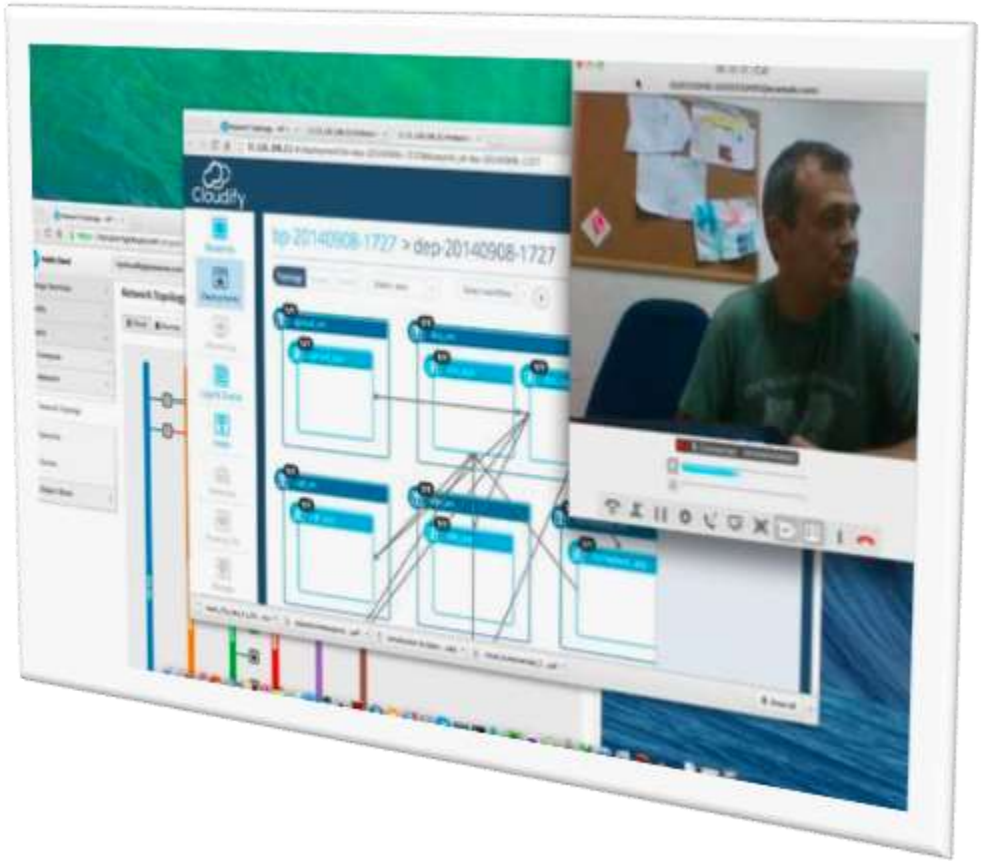
Mixing TOSCA & Heat

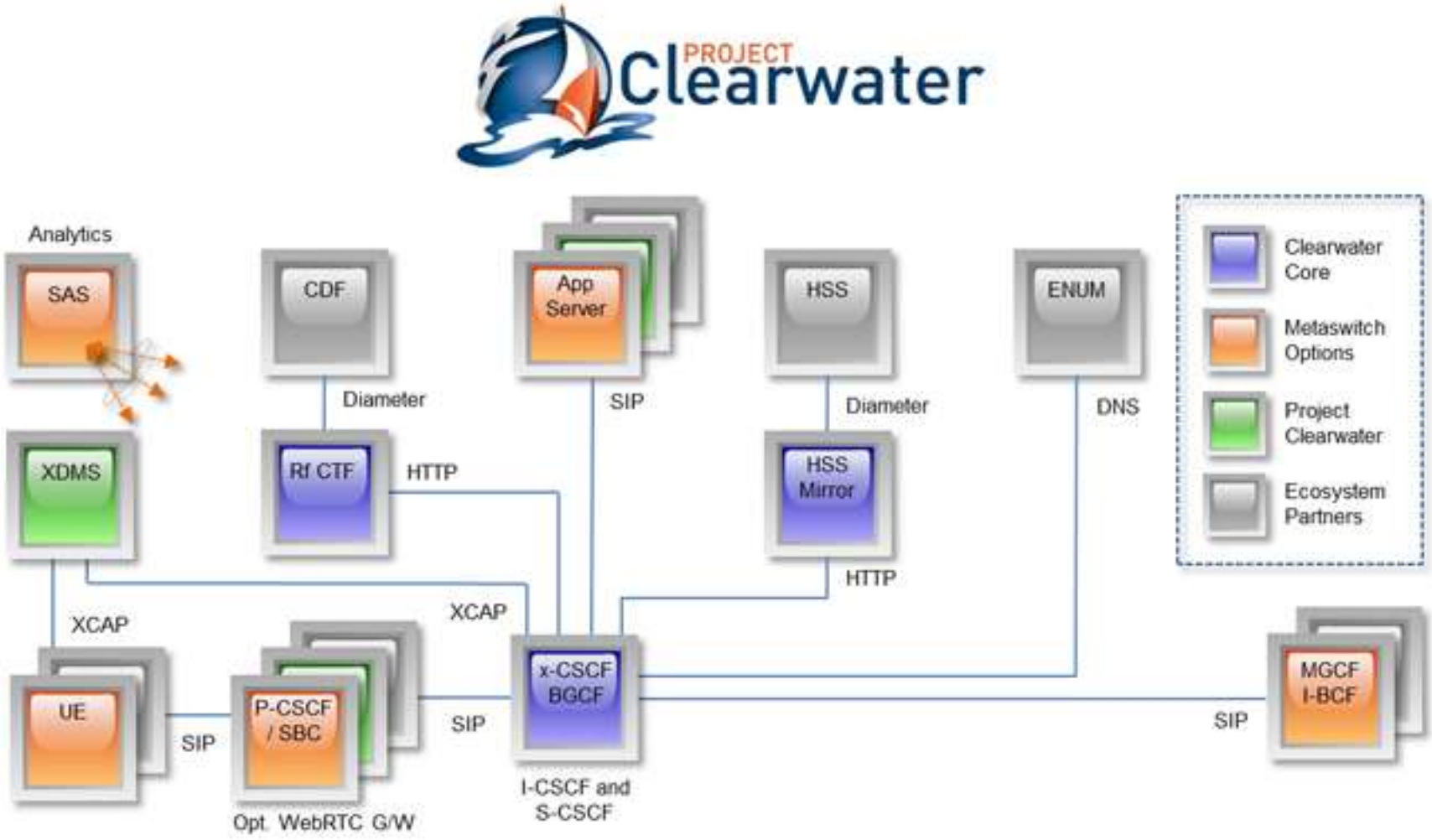


See Heat & TOSCA example here

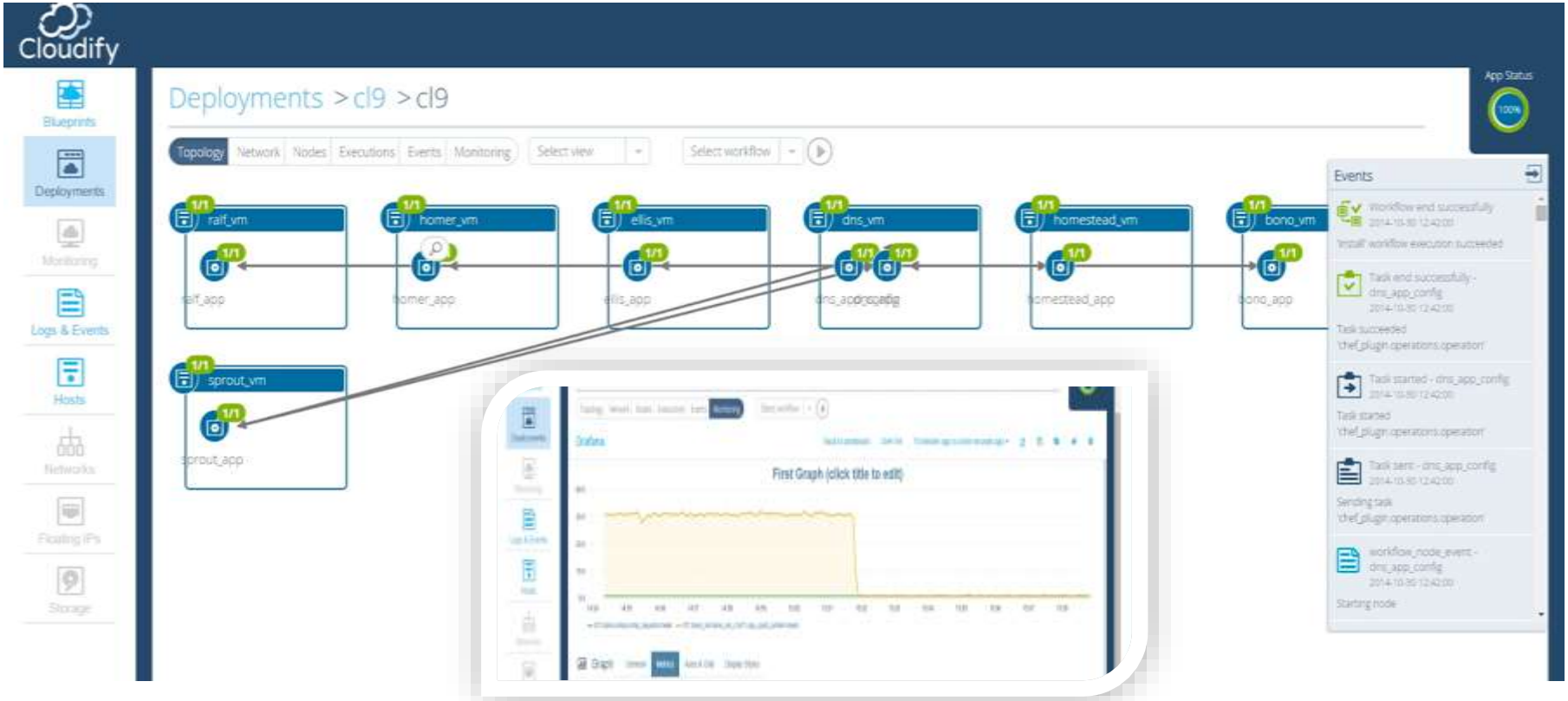
Real Life (NFV) Use Case

Create Your
Own “Skype”
On-demand ..




PROJECT
Clearwater
Analytics
SAS
CDF
App Server
HSS
ENUM
Clearwater Core
Metaswitch Options
Project Clearwater
Ecosystem Partners
Diameter
SIP
Diameter
DNS
XDMS
Rf CTF
HTTP
HSS Mirror
HTTP
XCAP
XCAP
UE
SIP
P-CSCF / SBC
SIP
x-CSCF BGCF
SIP
MGCF I-BCF
I-CSCF and S-CSCF
Opt. WebRTC G/W

Automating the Entire Stack



Summary

Putting networks & apps together enables:

- Simplify the deployment of complex apps
- Tighten the security
- Increase the agility

Available Today: Heat & Neutron, Ceilometer
TOSCA makes it portable

References

- Auto-Scaling WordPress Example (by Radware)
- ClearWater (NFV) Example (by Cloudify)
- Heat on DevStack
- Putting Heat & TOSCA Together
- Heat Technical Overview
- TOSCA Overview
- TOSCA Translator Project on StackForge