Why TOSCA?

• First step was to evaluate available orchestration options
  • HEAT, CloudFormation
  • Both tied to specific implementations
• TOSCA appeared as a viable common denominator
  • Topology definition
  • End user applications
• Existing codebase
  • TOSCA parser, HEAT translator
  • Reusable in other contexts
• Growing support in different communities
Complex example: Batch Cluster

node_templates:

elastic_cluster_front_end:
  type: tosca.nodes.indigo.ElasticCluster
  properties:
    deployment_id: orchestrator_deployment_id
  requirements:
    - lrms: torque_front_end
    - wn: wn_node

torque_front_end:
  type: tosca.nodes.indigo.LRMS.FrontEnd.Torque
  requirements:
    - host: torque_server

torque_server:
  type: tosca.nodes.indigo.Compute
  properties:
    public_ip: yes
  capabilities:
    host:
      properties:
        num_CPUs: 1
        mem_size: 1 GB

wn_node:
  type: tosca.nodes.indigo.LRMS.WorkerNode.Torque
  capabilities:
    wn:
      properties:
        max_instances: 5
        min_instances: 0
  requirements:
    - host: torque_wn

torque_wn:
  type: tosca.nodes.indigo.Compute
  properties:
    public_ip: no
  capabilities:
    scalable:
      properties:
        count: 0
    host:
      properties:
        num_CPUs: 1
        mem_size: 1 GB
  os:
    properties:
      type: linux
      distribution: ubuntu
      version: 14.04
Conclusions

• Many use cases already covered / implemented in TOSCA
  • Galaxy portal / SLURM
  • Mesos
  • DisVis
• Significant contributions upstream
  • Miguel core dev of the OpenStack TOSCA parser (and 2nd top contributor)
  • Mathieu core dev of the OpenStack Heat Translator (and 2nd top contributor)
• Available in the Indigo DataCloud Release 1
• Integration with the common openstack client