Sublimated Configuration of Infrastructure as a Service Deployments

Ming: A Model- and View-Based Approach for Cloud Datacenters

6th International Conference on Cloud Computing and Services Science
Roma, Italy, 2016–04–23

Ta'id Holmes
Infrastructure Cloud, Deutsche Telekom Technik GmbH

Outlook
KEY LEARNINGS & MESSAGES

- overview of different systems and projects for
  - bare metal installation
  - OpenStack deployment
- the need for a standard for describing DC deployments
- MING, a model-based approach, permitting the tool agnostic, declarative configuration of IaaS deployments in DCs

Software Installation — Phase 1
BARE METAL INSTALLATION

- base operating system (OS) installation on each node
- automated, over the network (IPMI, PXE, DHCP, TFTP)
- MAC addresses of NICs

projects: Cobbler, FAI, Ironic, MAAS

Software Installation — Phase 2

IAAS SERVICE DEPLOYMENT

Node Type	IaaS Service
▪ storage	▪ Ceph [Weil et al., 2006]
▪ compute	▪ Nova
▪ network	▪ Neutron
▪ management	▪ Keystone, Nova cloud controller, Heat, Horizon, ...

projects: Compass, Crowbar, Fuel, JuJu, Packstack, RDO

Configuring the Deployment of Datacenters

configuration information

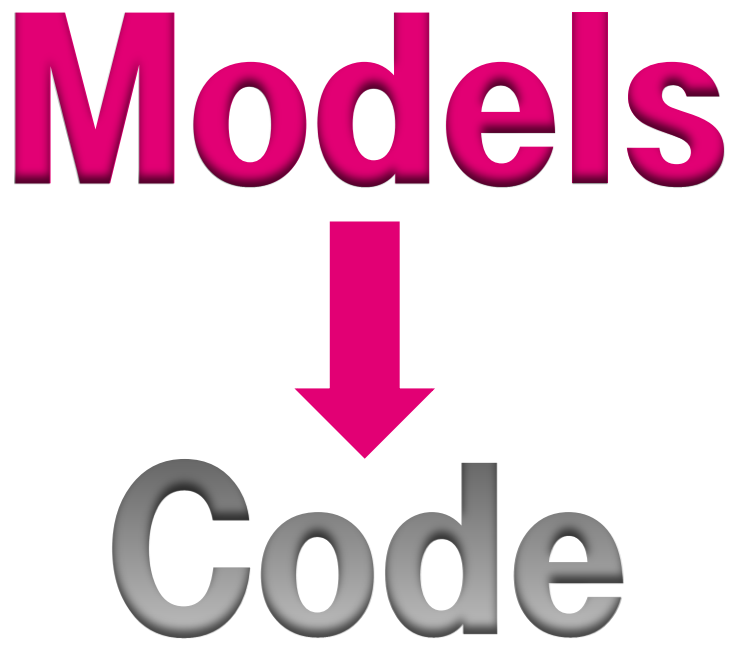
- **aggregated** in files
- by **experts** familiar with deployment technologies
- **scattered** and **tangled**
  - relating to different aspects
  - needs to be kept **consistent**

Need for A Common Datamodel

- Configuration cannot be reused in other projects, currently.
- How to evaluate different projects?
- How to avoiding tool dependencies?

Motivating a Model-Based Approach
FOR IAAS DEPLOYMENTS

- improve
  - understandability
  - maintainability
- avoid redundancy
- gain independence from technologies
- automate as much as possible



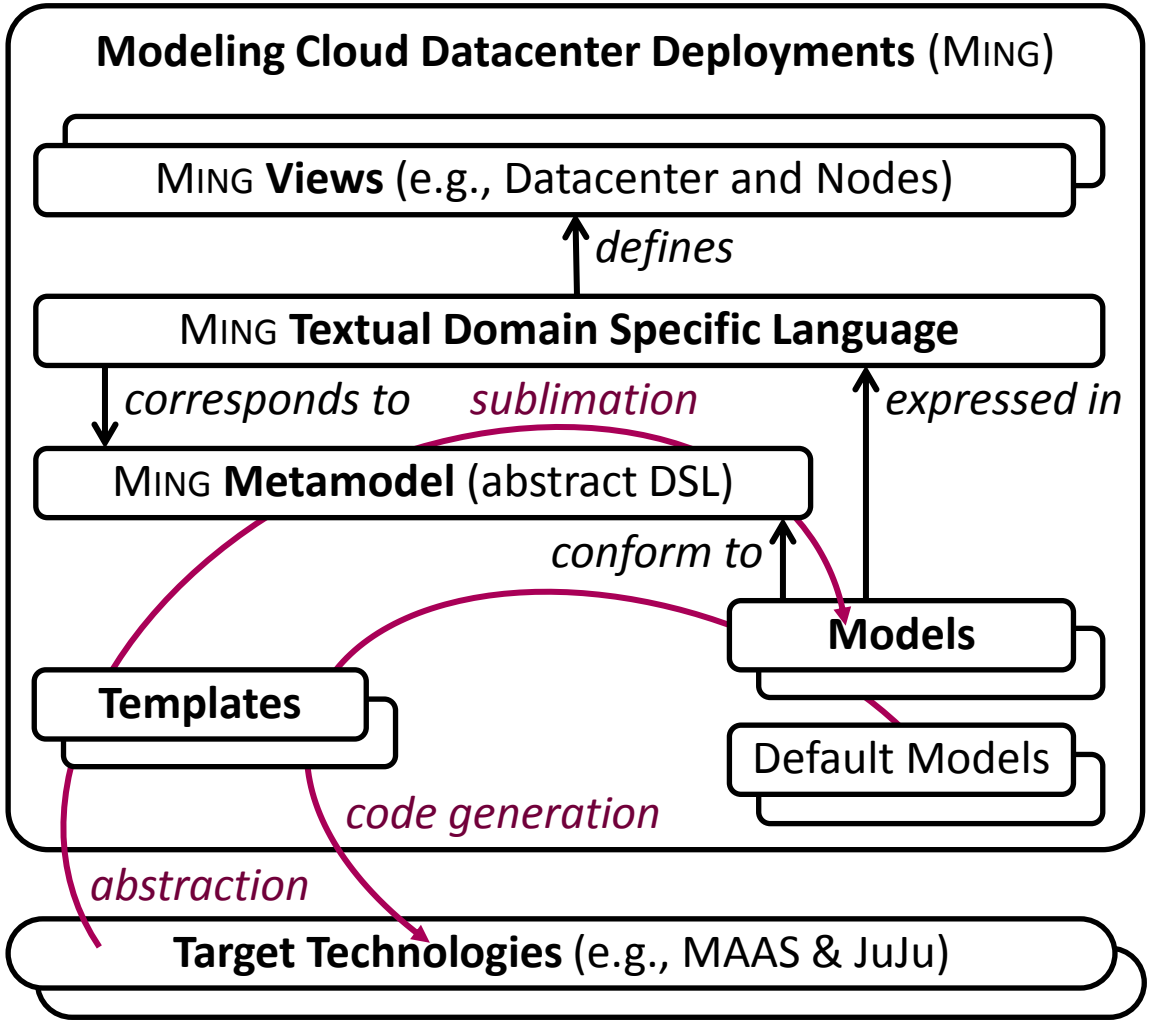
ModelING Cloud Datacenter Deployments (MING 明)

Aims at a

- **tool agnostic,**
- **declarative specification** of configuration

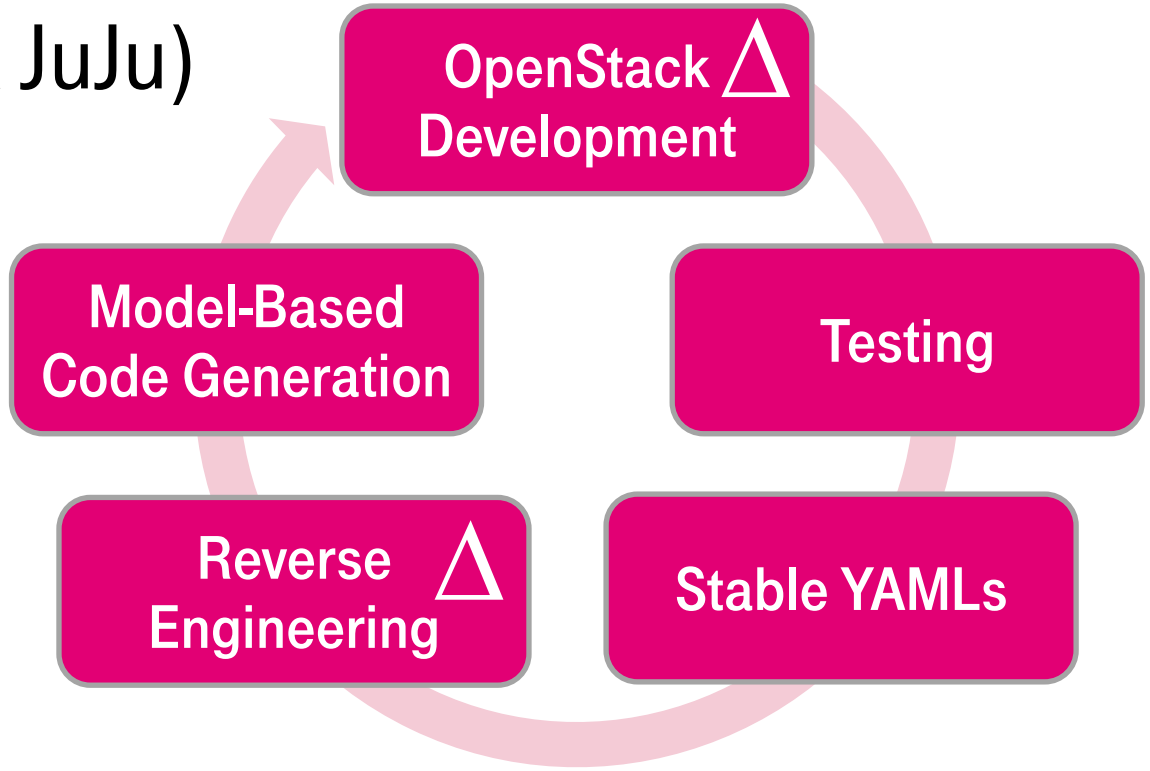
for realizing IaaS deployments in DCs from bare machines.

Architectural Overview



Development Process
ENGINEERING AND MODEL-BASED REVERSE ENGINEERING

- starting from code (MAAS & JuJu)
- establishing models through abstraction
- implementing model transformations
- generating original target code from models



Model Transformation

XTEND TEMPLATE FOR GENERATING MAAS YAML

```
nodes:
  «FOR az : dc.azs»
  «FOR rack : az.racks»
  «FOR node : rack.nodes»
  - name: «model.deployment.name»-«node.name»
    «IF node.nodeType == NodeTypeName.CN»
    tags: «getComputeAggregate(zones, node, az)»
    «ELSEIF node.nodeType == NodeTypeName.SN»
    tags: storage-«getCephPool(zones, node).name»
    «ELSEIF node.nodeType == NodeTypeName.MN»
    tags: api
    «ELSEIF node.nodeType == NodeTypeName.NN»
    tags: gateway-«getGatewayZone(zones, node).name»
    «ELSE»
    tags: standby
    «ENDIF»
    architecture: «node.arch»/generic
    mac_addresses:
    «FOR nic : getNICs(node)»
    - «nic.mac»
    «ENDFOR»
    power:
      type: ipmi
      address: «getIPMI(node)»
      user: «model.credentialsIPMI.username»
      pass: «model.credentialsIPMI.password»
      driver: LAN_2_0
    «enrichWithIPs(node)»
    «FOR nic : getNICs(node).filter[it.ip4 != null]»
    sticky_ip_address:
      mac_address: «nic.mac»
      requested_address: «nic.ip4»
    «ENDFOR»
  «ENDFOR»
  «ENDFOR»
  «ENDFOR»
```

OpenStack Configuration Options
VIEW IN DSL EDITOR

```
IaaS Deployment SongThrush @ MingDC8
OpenStack Liberty on Trusty

Compute
        liveMigration
        mtu 9000

Ceph
        osdFormat btrfs
        osdReformat

Percona
        maxConnections 12345

Neutron
        externalBridge "br-ext"
        overlayNetType "gre vxlan"
        l3_ha
        l3_agents 2–4
        mtu 9000

Heat
        workers 8
        hiddenTags "generated"
```

Separation of Models in Views
BENEFITS

- discovery of nodes and components
  automatically yields a certain view
- credentials stored in another view
  can be generated initially
- parts of the overall configuration are provided
  DSL user can focus on the other aspects of the deployment

Model-Based Approach
LIMITATIONS

- MING metamodel does not reflect all fine-grained configuration options
  - if desired metamodel can be extended
- not all technologies support the same features
  - early feedback in DSL editor
  - fallback during code generation

Conclusion

- MING, model-based approach
  - declarative textual DSL
  - tool agnostic
  - view-based
    - realizing separation of concerns (SoC)
    - supporting different stakeholders

Acknowledgements

LIFE IS FOR SHARING.

THANK YOU!

QUESTIONS?

Dr.techn. Ta'id HOLMES, DEA
Expert Key Projects Technology
Infrastructure Cloud, Deutsche Telekom Technik GmbH
T: +49 6151 680-5763 | M: +49 151 46.75.40.18 | E: t.holmes@telekom.de | W: http://t.holmes.info

LIFE IS FOR SHARING.