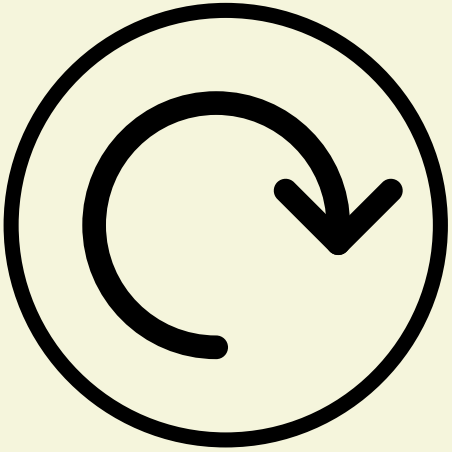
OpenStack Deployment in a Fog Computing Context: A Survey

Hélène Coullon, Dimitri Pertin
Inria, IMT Atlantique, Nantes, France
helene.coullon@inria.fr, dimitri.pertin@inria.fr

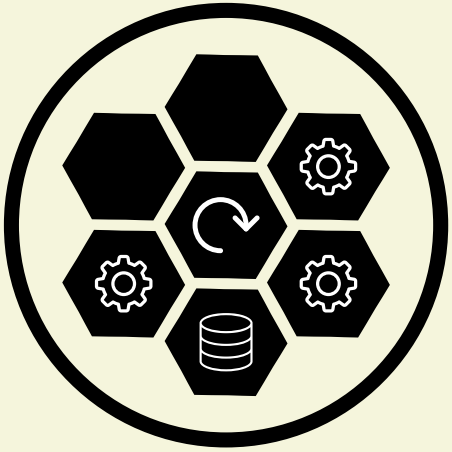
Christian Pérez
Inria, ENS, Lyon, France
christian.perez@inria.fr

Software architecture



Monolithic

- single entity
- multi-tasks



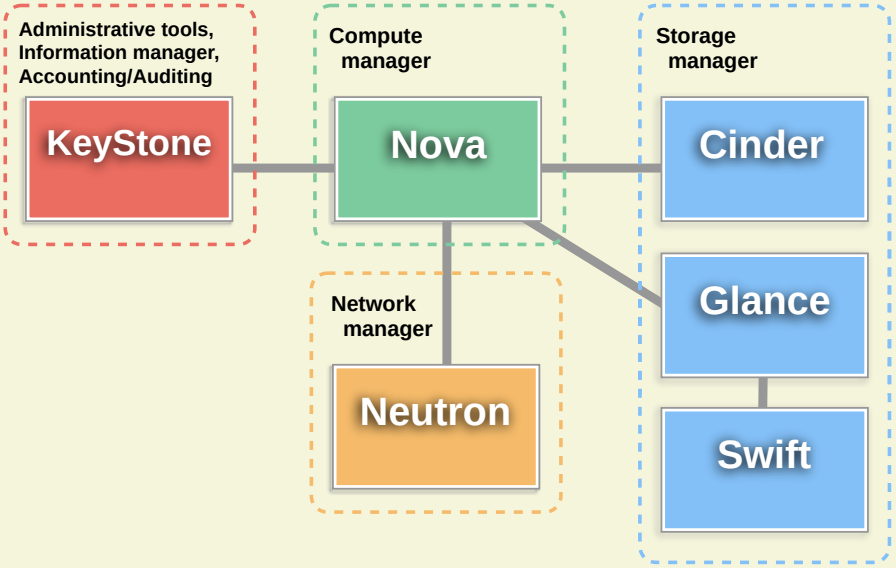
Modular

- separation of concerns
- multiple services
- interconnected through interfaces

Distributed software

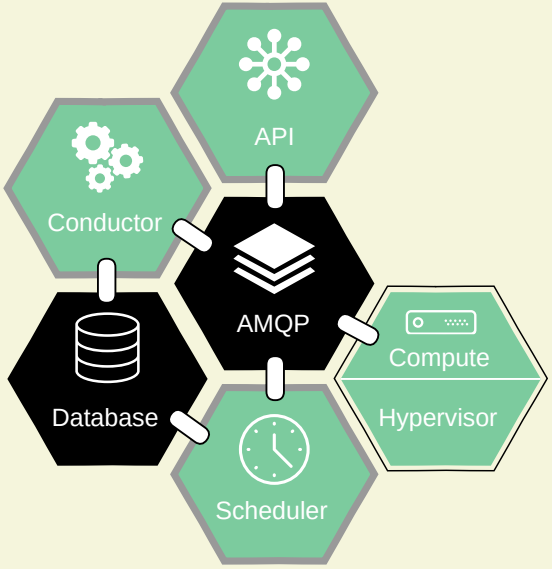
- multiple services
- distributed over multiple servers
- interconnected by a network

OpenStack



- IaaS software: Compute, Storage, Network
- Modular architecture
- Communication by REST APIs, Message Queues (AMQP) and Databases

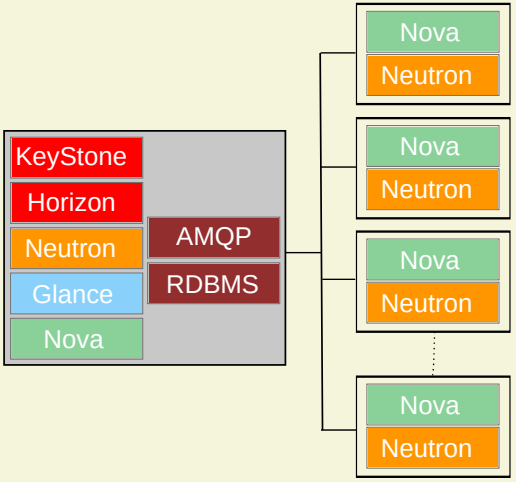
Example: Nova components



- 12 services in nova (only four represented here)
- **Control** (api, conductor, scheduler) vs **compute** services
- 164 services in OpenStack (based on Kolla container images)

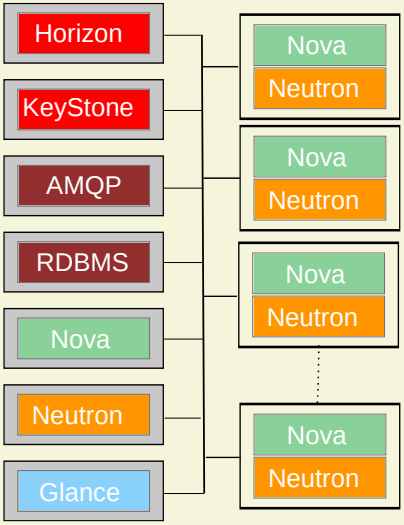
How are deployed such components?

Deployment scenarios (1/3)



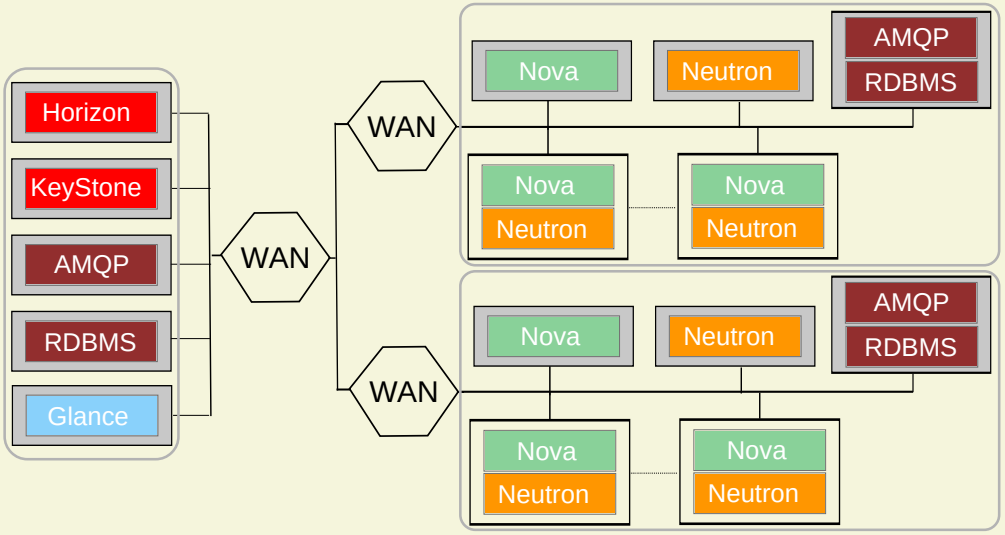
Single controller, multiple compute nodes

Deployment scenarios (2/3)



Multiple controllers, multiple compute nodes

Deployment scenarios (3/3)



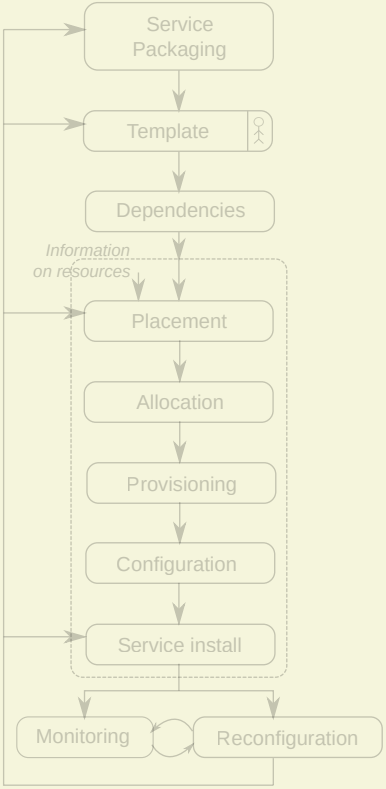
Multiple controllers, multiple compute nodes, mutiple regions

Agenda

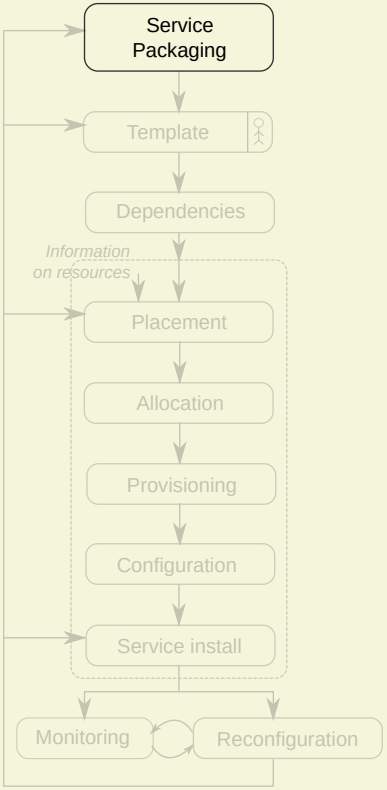
1. Application deployment model
2. OpenStack deployment tools
    - Kolla, Enos, Juju, Kubernetes, TripleO
3. OpenStack deployment challenges for Fog Computing
    - Fog Computing
    - Challenges
4. Future works
    - Description language to deploy OpenStack for Fog Computing

1. Application deployment model

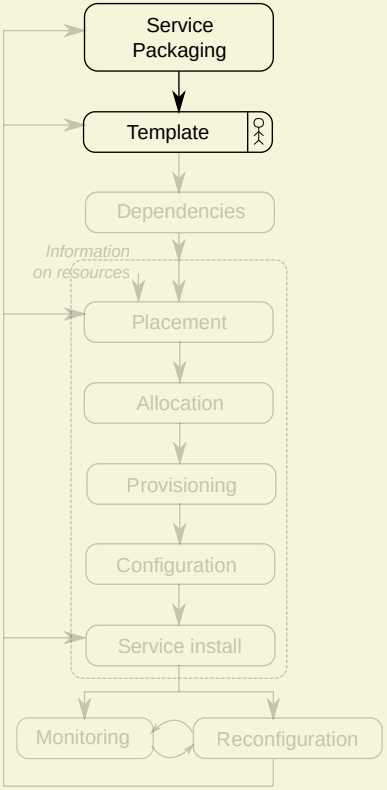
Deployment model



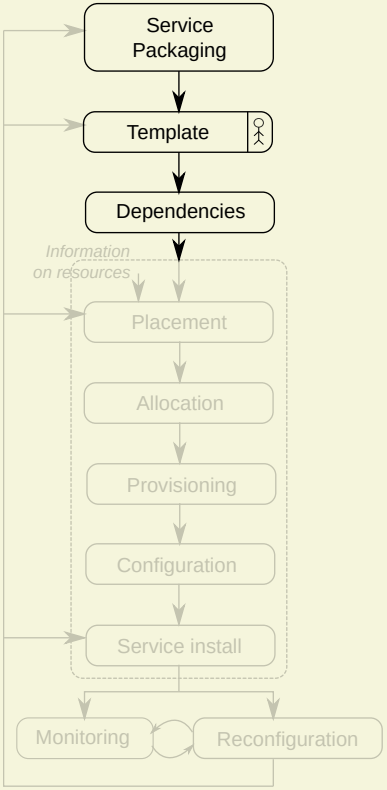
Deployment model



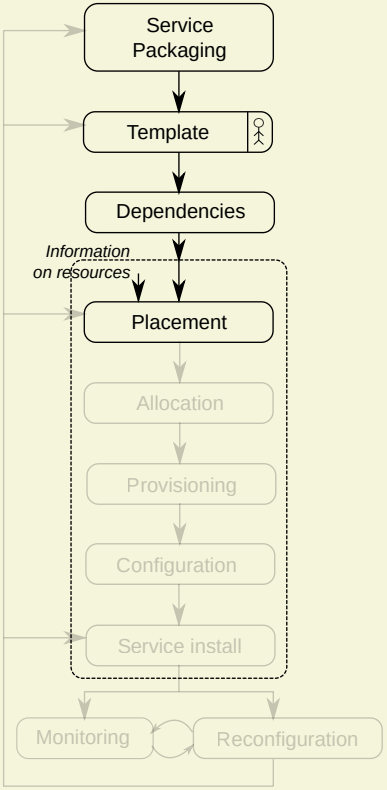
Deployment model



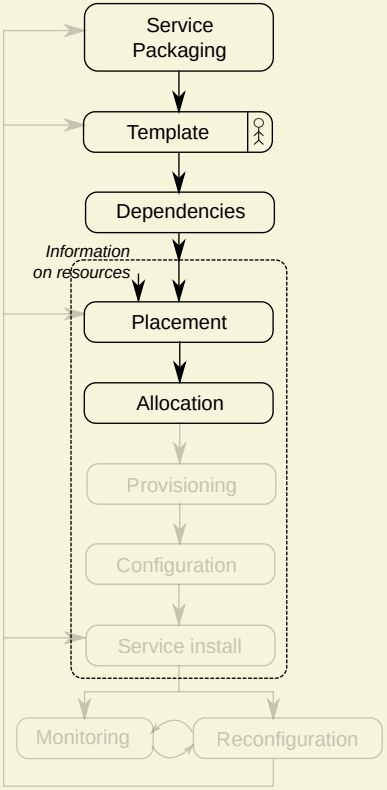
Deployment model



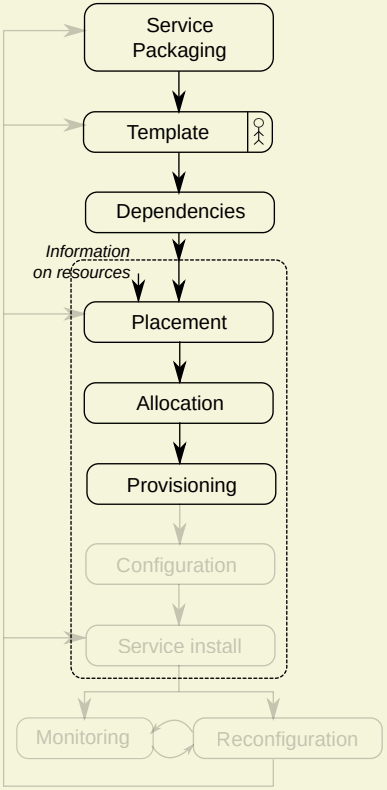
Deployment model



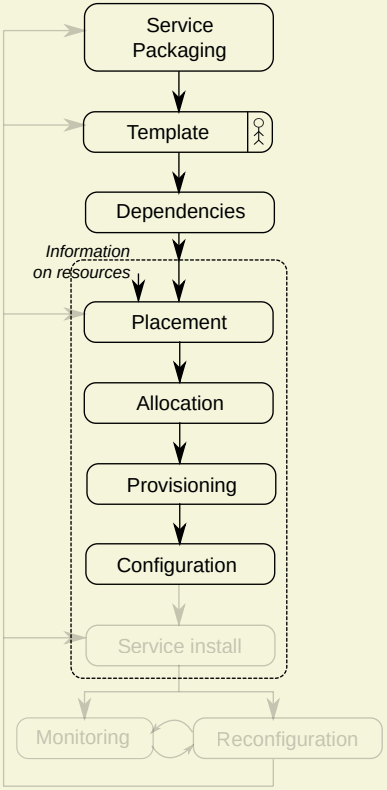
Deployment model



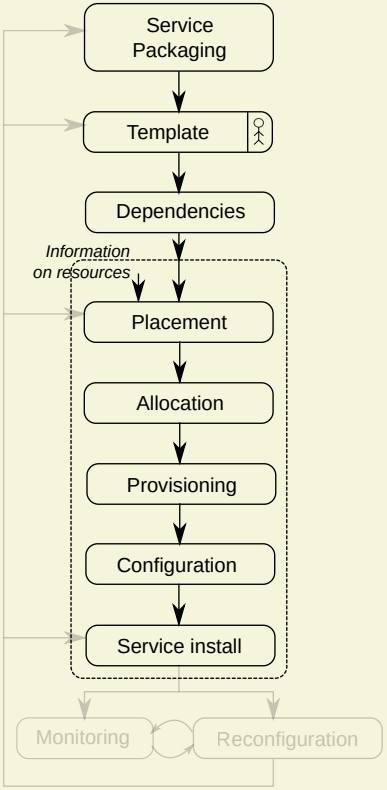
Deployment model



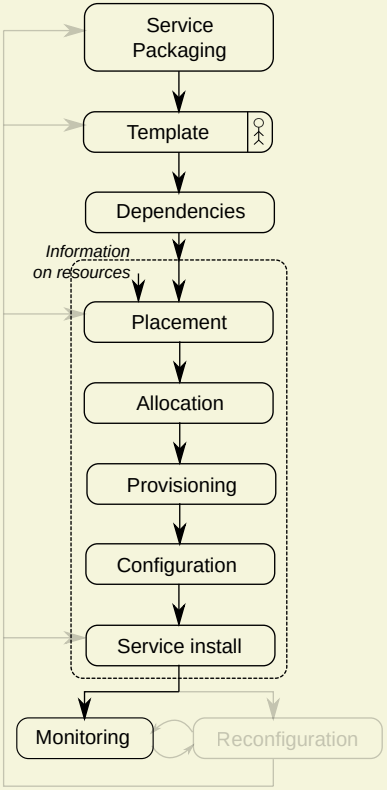
Deployment model



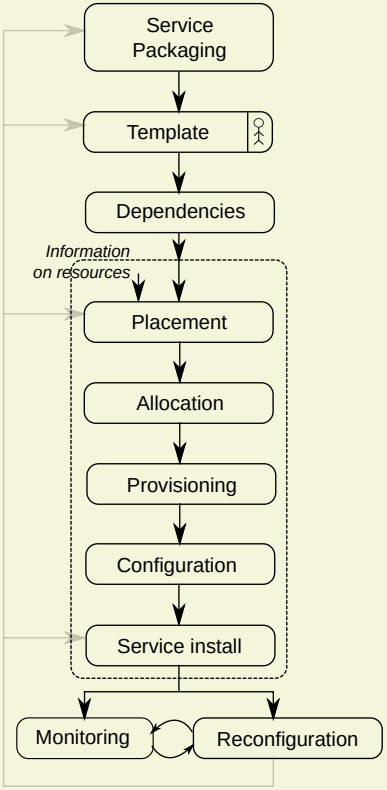
Deployment model



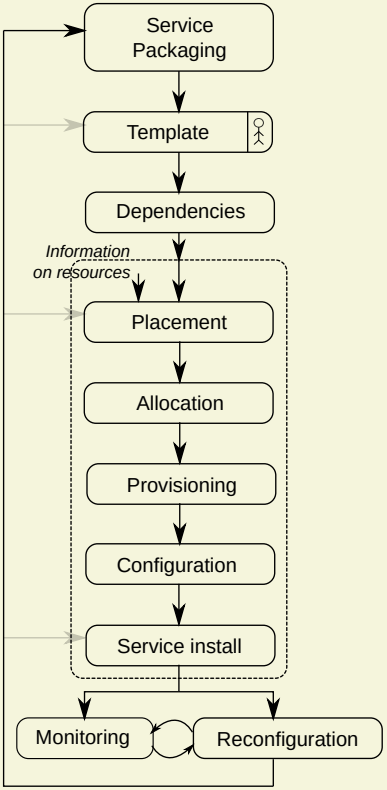
Deployment model



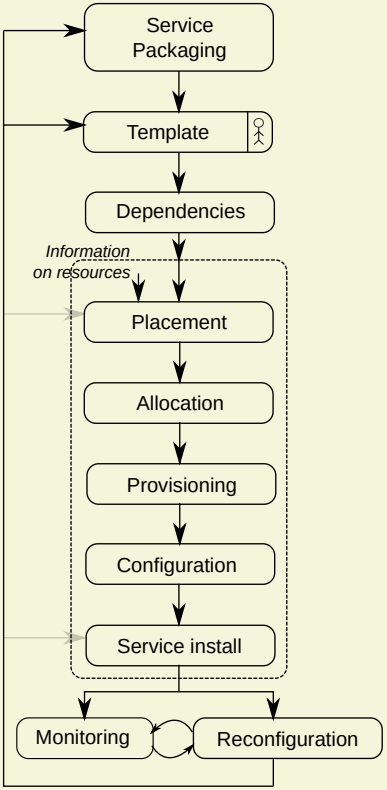
Deployment model



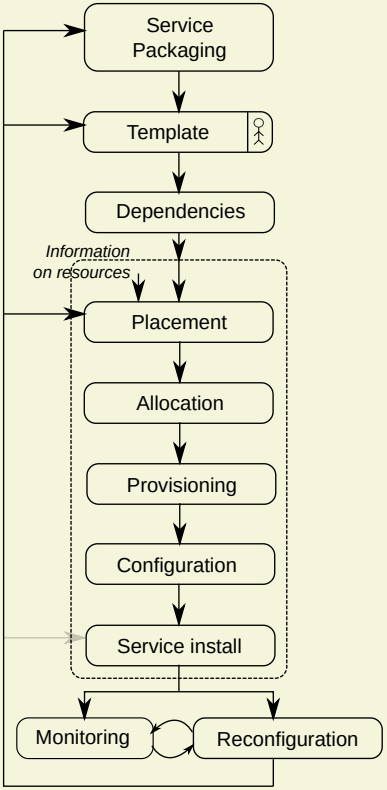
Deployment model



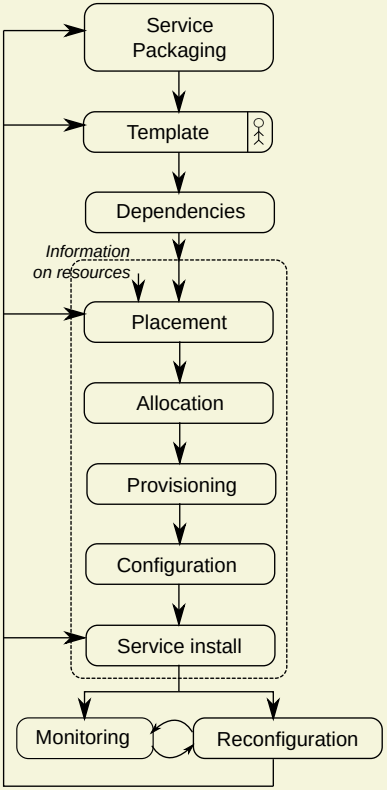
Deployment model



Deployment model

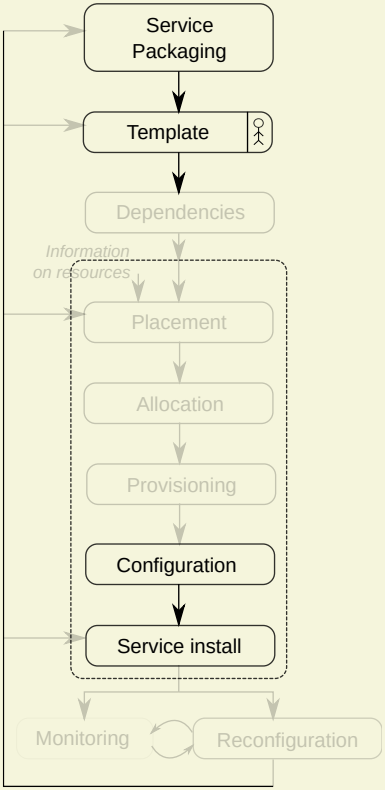


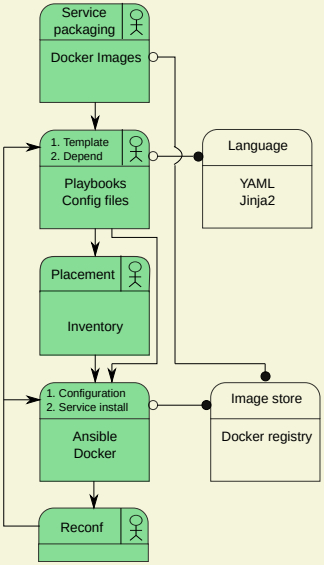
Deployment model



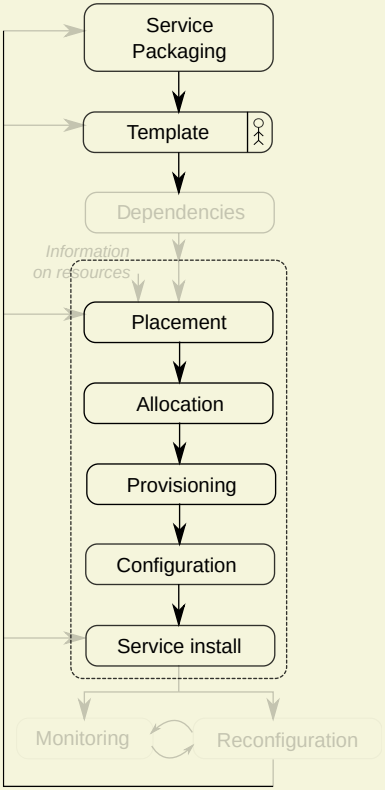
2. OpenStack deployment tools

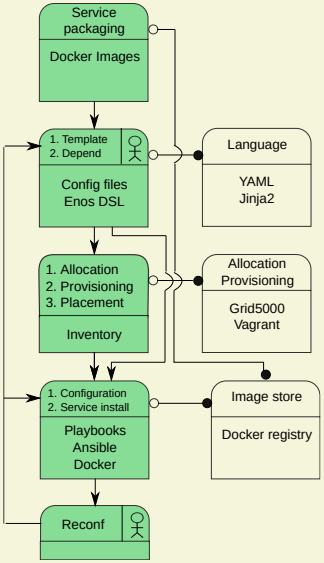
1. Kolla


Service Packaging
Template
Dependencies
Information on resources
Placement
Allocation
Provisioning
Configuration
Service install
Monitoring
Reconfiguration

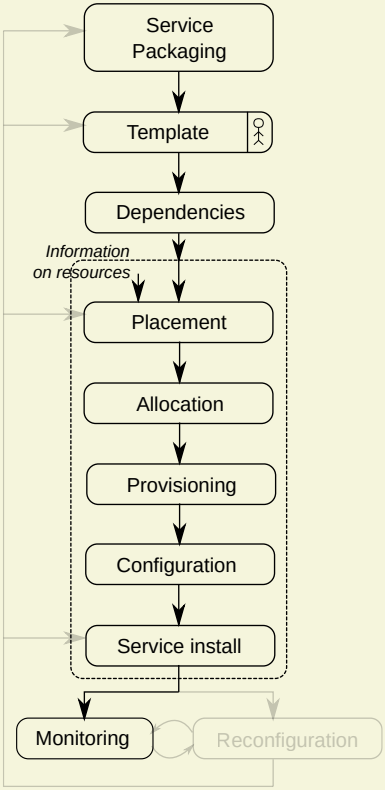

Service packaging
Docker Images
1. Template
2. Depend
Playbooks
Config files
Language
YAML
Jinja2
Placement
Inventory
1. Configuration
2. Service install
Ansible
Docker
Image store
Docker registry
Reconf

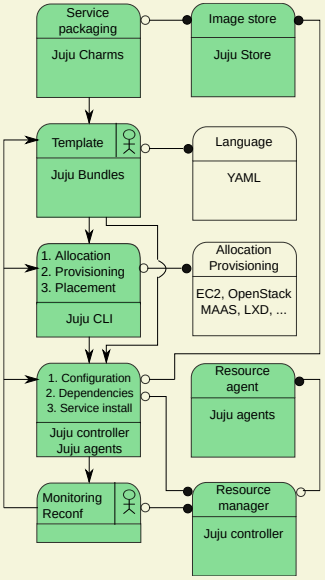
2. Enos


Service Packaging
Template
Dependencies
Information on resources
Placement
Allocation
Provisioning
Configuration
Service install
Monitoring
Reconfiguration

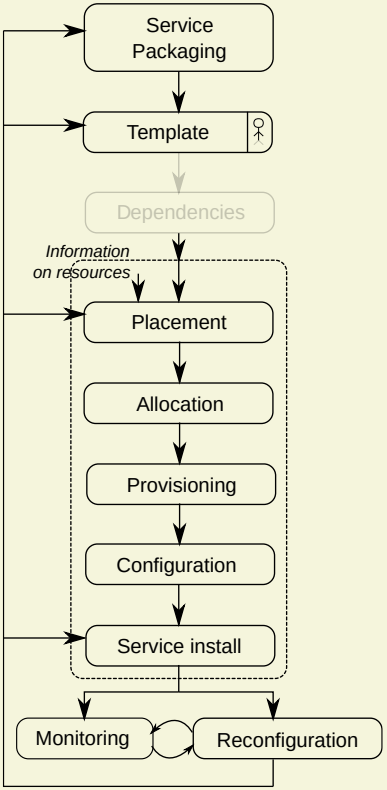

Service packaging
Docker Images
1. Template
2. Depend
Config files
Enos DSL
Language
YAML
Jinja2
1. Allocation
2. Provisioning
3. Placement
Inventory
Allocation Provisioning
Grid5000
Vagrant
1. Configuration
2. Service install
Playbooks
Ansible
Docker
Image store
Docker registry
Reconf

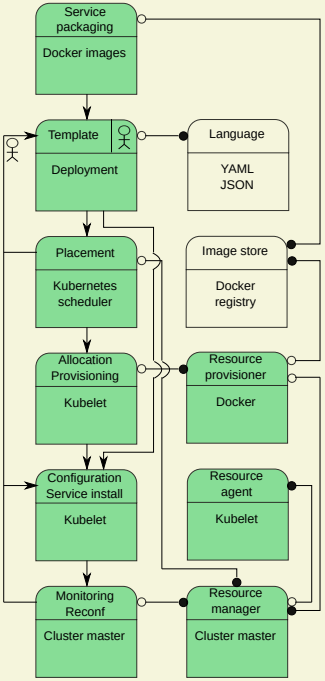
3. Juju


Service Packaging
Template
Dependencies
Information on resources
Placement
Allocation
Provisioning
Configuration
Service install
Monitoring
Reconfiguration

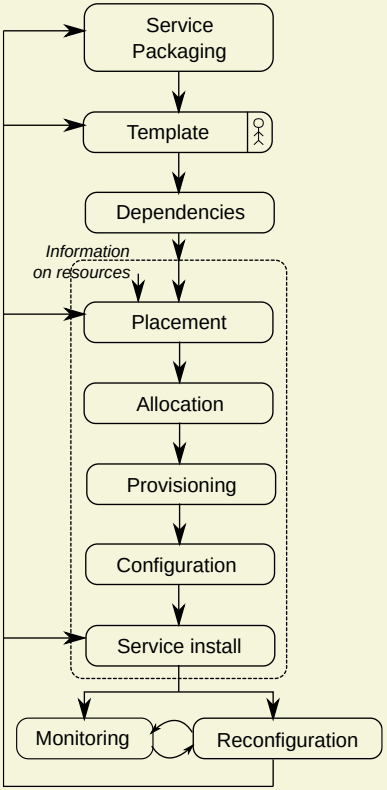

Service packaging
Juju Charms
Image store
Juju Store
Template
Juju Bundles
Language
YAML
1. Allocation
2. Provisioning
3. Placement
Juju CLI
Allocation Provisioning
EC2, OpenStack MAAS, LXD, ...
1. Configuration
2. Dependencies
3. Service install
Juju controller
Juju agents
Resource agent
Juju agents
Monitoring
Reconf
Resource manager
Juju controller

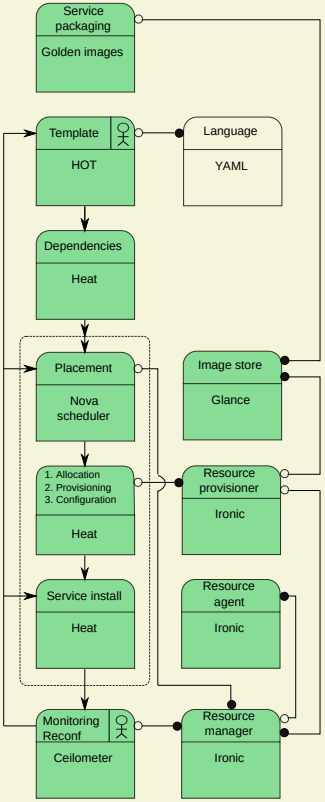
4. Kubernetes





5. TripleO


Service Packaging
Template
Dependencies
Information on resources
Placement
Allocation
Provisioning
Configuration
Service install
Monitoring
Reconfiguration


Service packaging
Golden images
Template
HOT
Language
YAML
Dependencies
Heat
Placement
Nova scheduler
Image store
Glance
1. Allocation
2. Provisioning
3. Configuration
Heat
Resource provisioner
Ironic
Service install
Heat
Resource agent
Ironic
Monitoring Reconf
Ceilometer
Resource manager
Ironic

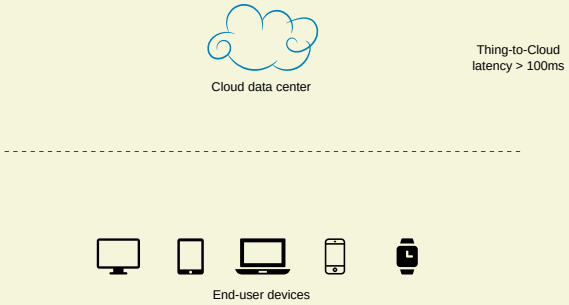
Deployment tools

	Kolla	**Enos**	**Juju**	**Kubernetes**	**TripleO**
Packaging	containers	containers	scripts	containers	disk images
Template	-	-	~	~	+
Dependencies	-	-	~	-	+
Placement	-	~	~	+	+
Allocation	-	~	~	+	+
Provisioning	-	~	~	+	+
Mon/Reconf	-	-	~	++	+

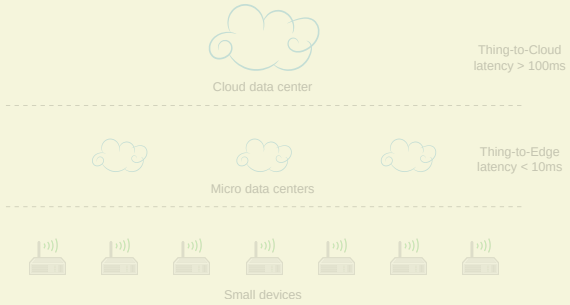
Kolla **Enos** **Juju** **Kubernetes** **TripleO**

3. OpenStack deployment for Fog Computing

Fog Computing



Cloud computing

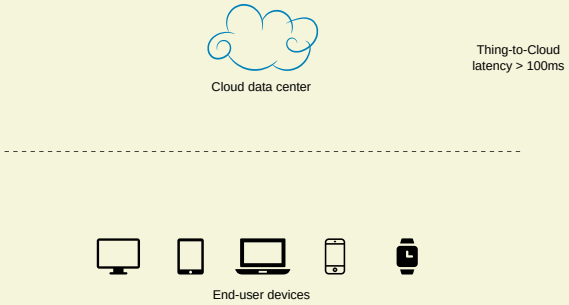


Fog computing

Bottleneck and high-latency for:

- Internet of Object
- Real-time application
- Smart-cities
- ...

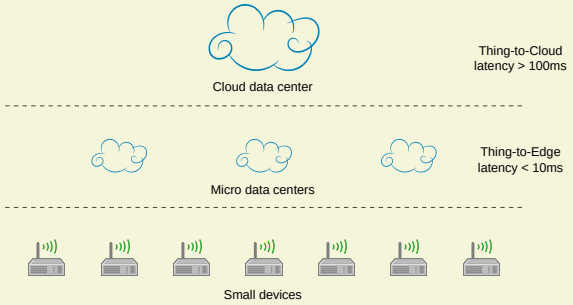
Fog Computing



Cloud computing

Bottleneck and high-latency for:

- Internet of Object
- Real-time application
- Smart-cities
- ...



Fog computing

Intermediate layer at edge/core:

- Heterogeneous resources
- Large-scale
- ...

Challenges (1/2)

Resource heterogeneity

- provisioning (manage multiple resources: bare-metal, VM, container)
- packaging (multiple images)
- resource manager (keep track of resources)

Large-scale

- decentralized services (not the case currently for OpenStack)
- dependency graph (performance)
- placement (performance, NP-hard)

Challenges (2/2)

Abstraction language

- complex infrastructure
  - declare service dependencies
  - scaling rules
- new constraints (hardware, locality/latency, energy, ...)

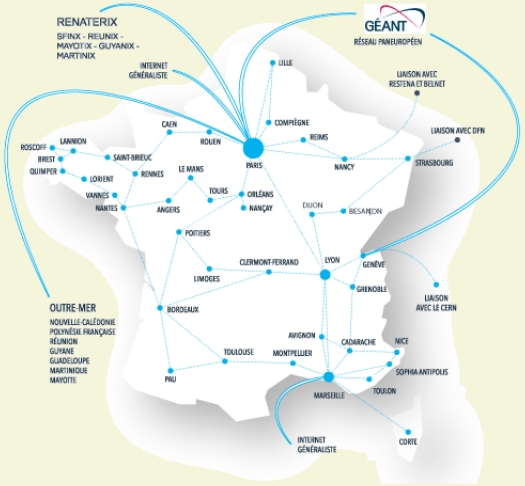
Dynamic reconfiguration

- recovering from failures
- update
- auto-scaling
- topology modification (moving mobile edge)

4. Future works

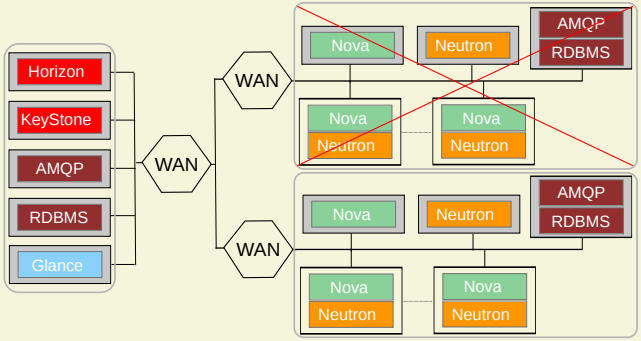
High abstraction language (1/2)

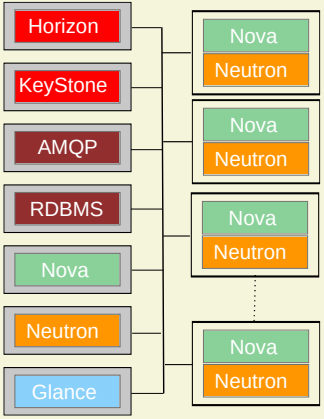
- First step: Propose a DSL to automatically deploy applications
  - e.g. automatically deploy OpenStack with *n* compute nodes
  - managing automatically the number of regions
  - managing automatically the component distribution
  - based on given criteria



High abstraction language (2/2)

- Second step: Add reconfiguration capabilities
  - redeployment
  - fault-tolerance
  - auto-scaling





Conclusion

1. We gave a **model** of the application deployment process...
2. as a base to **compare** the state-of-the-art tools to deploy OpenStack
    - Kubernetes and TripleO showed great features
3. We provided related Fog computing **challenges**
4. We plan to focus on the research problem of designing a high **abstraction language** to:
    - automatically deploy application
    - manage reconfiguration
    - not limited to OpenStack