XLcloud
Energy efficiency for supporting HPC as a Service

François Rossigneux
francois.rossigneux@inria.fr

November 29, 2013 - Université de Lille

Context
XLcloud

XLcloud:

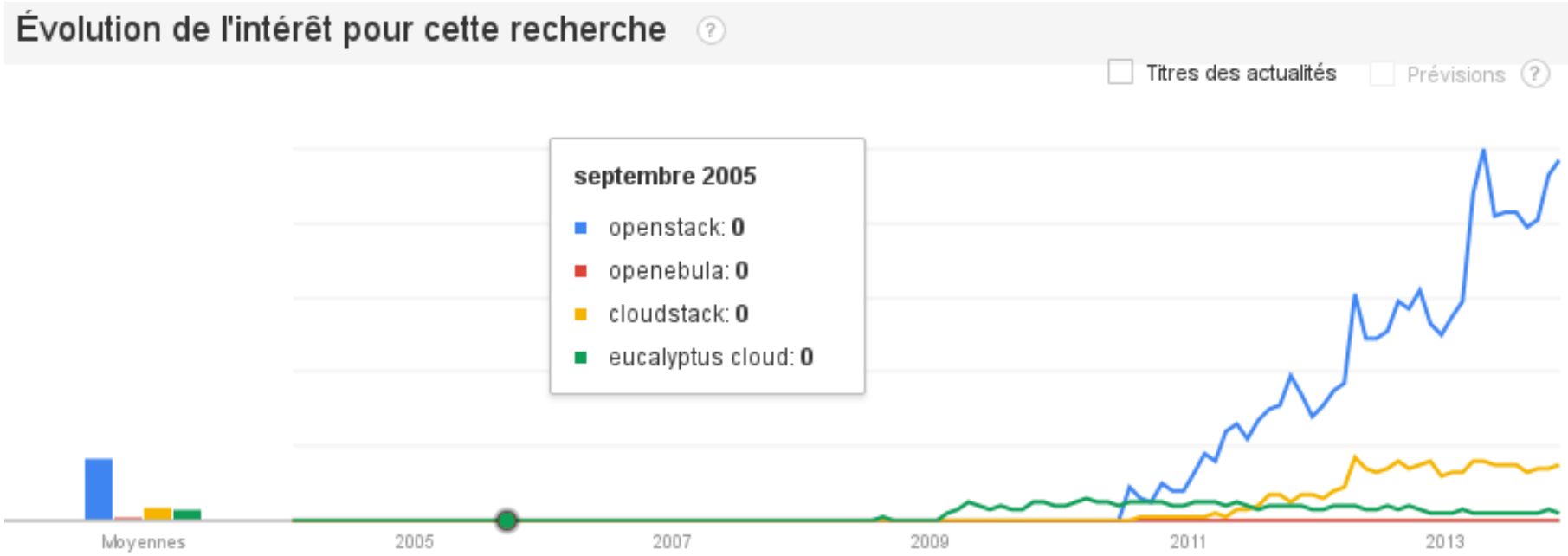
- HPC-as-a-Service (based on OpenStack)
- Funded by the "Fonds national pour la Société Numérique"
- Three-year long collaborative project
- Open source license

Some use cases:

- Compute plants (HPC clouds Research & Industry)
- Cloud gaming (industry of 3D video games)
- Interactive and collaborative virtual prototyping
- Human body visualization (research areas in online medical education)
- Interactive computation fluid dynamics for the cloud

Context
Why OpenStack?

OpenStack is becoming a major open source cloud computing solution!



Context
Consortium

Context
Topics

Our team is working on energy topics:

- Telemetry (taking measurements)
- Scheduling (placing virtual machines)
- Turning off unused machines (sleep modes)

Context

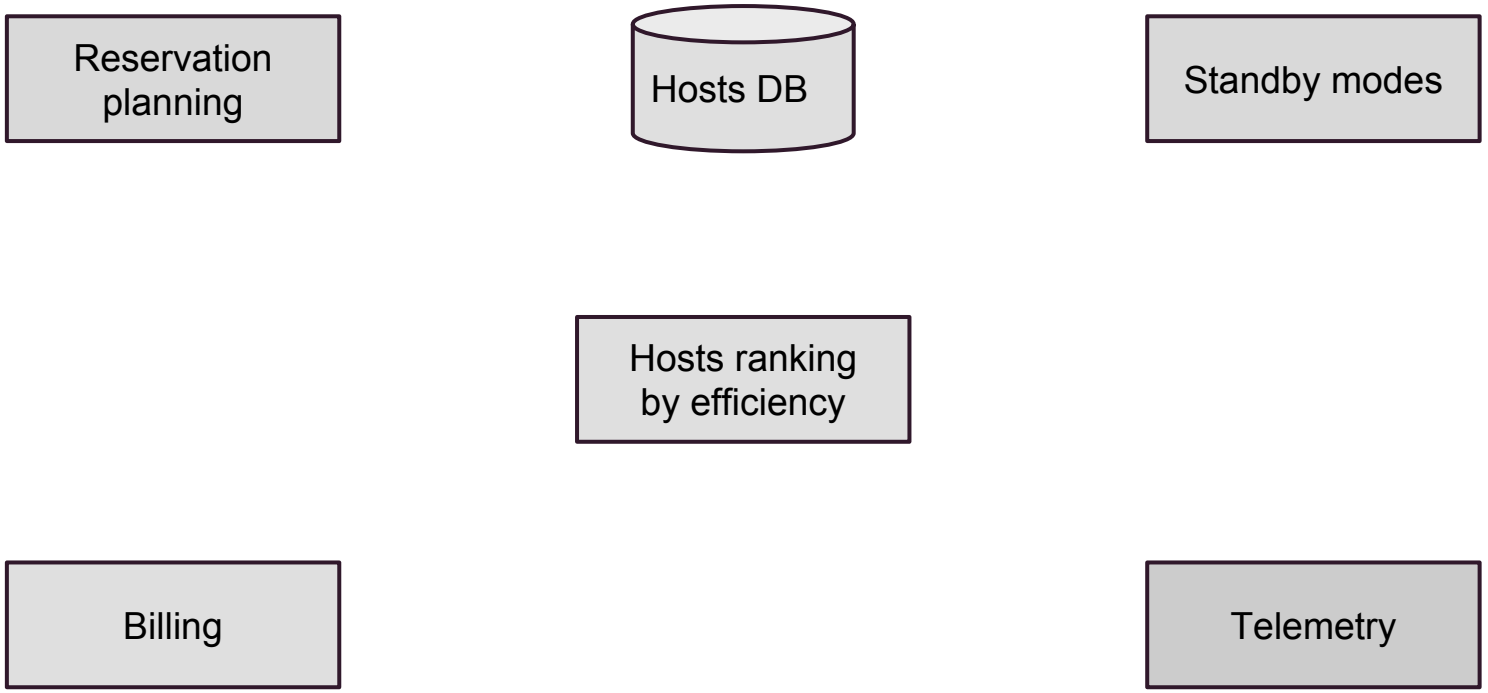
OpenStack main components:

- Compute (**Nova**)
- Object Storage (Swift)
- Block Storage (Cinder)
- Networking (Neutron)
- Identity (**Keystone**)
- Dashboard (Horizon)
- Metering / billing (**Ceilometer**)

Incubation:

- Reservation (**Climate**) interacting with Nova
- Energy (**Kwapi**) interacting with Ceilometer

Modules


Reservation planning
Hosts DB
Standby modes
Hosts ranking by efficiency
Billing
Telemetry

Modules
Current development status

Telemetry


Reservation planning
Standby modes
Hosts ranking by efficiency
Billing
Telemetry
Wattmeter

Telemetry
Main objectives

Create a framework fully integrated in OpenStack:

- Use the common code (provided by Oslo)
- Use the OpenStack conventions and repositories (StackForge)

Collect power measurements from various wattmeters:

- IPMI cards
- PDUs

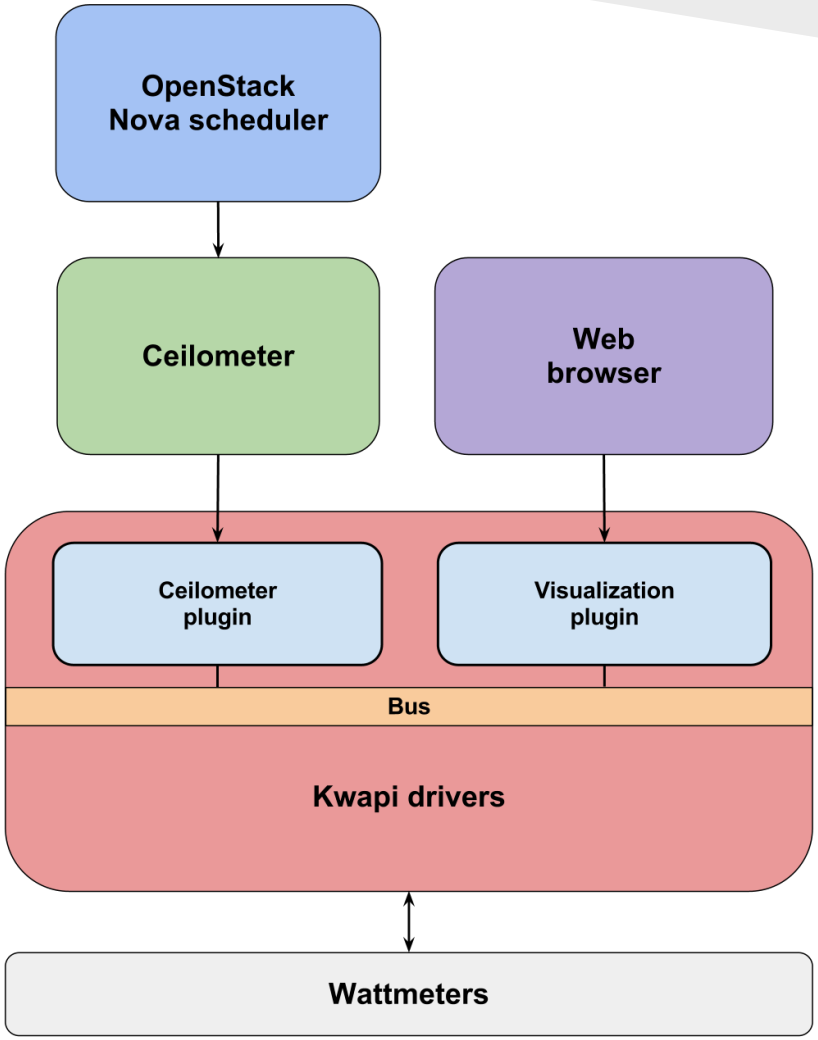
Be scalable (thousands of sensors, potentially in different locations):

- Use a high-performances bus
- Aggregate the collected data

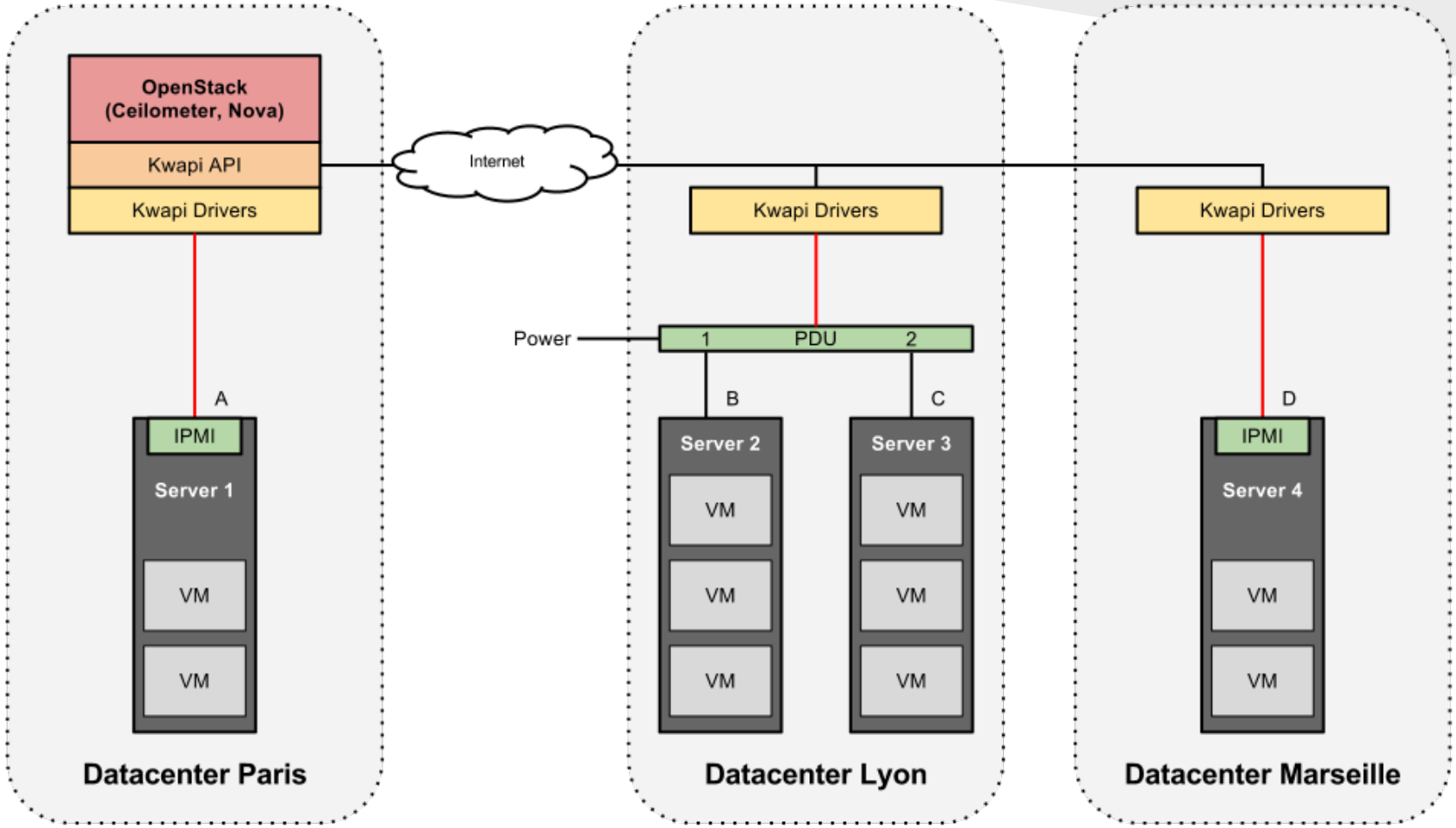
Be extensible:

- Easily write drivers to support new wattmeters
- Easily write plugins to bring new features (graphs, log…)

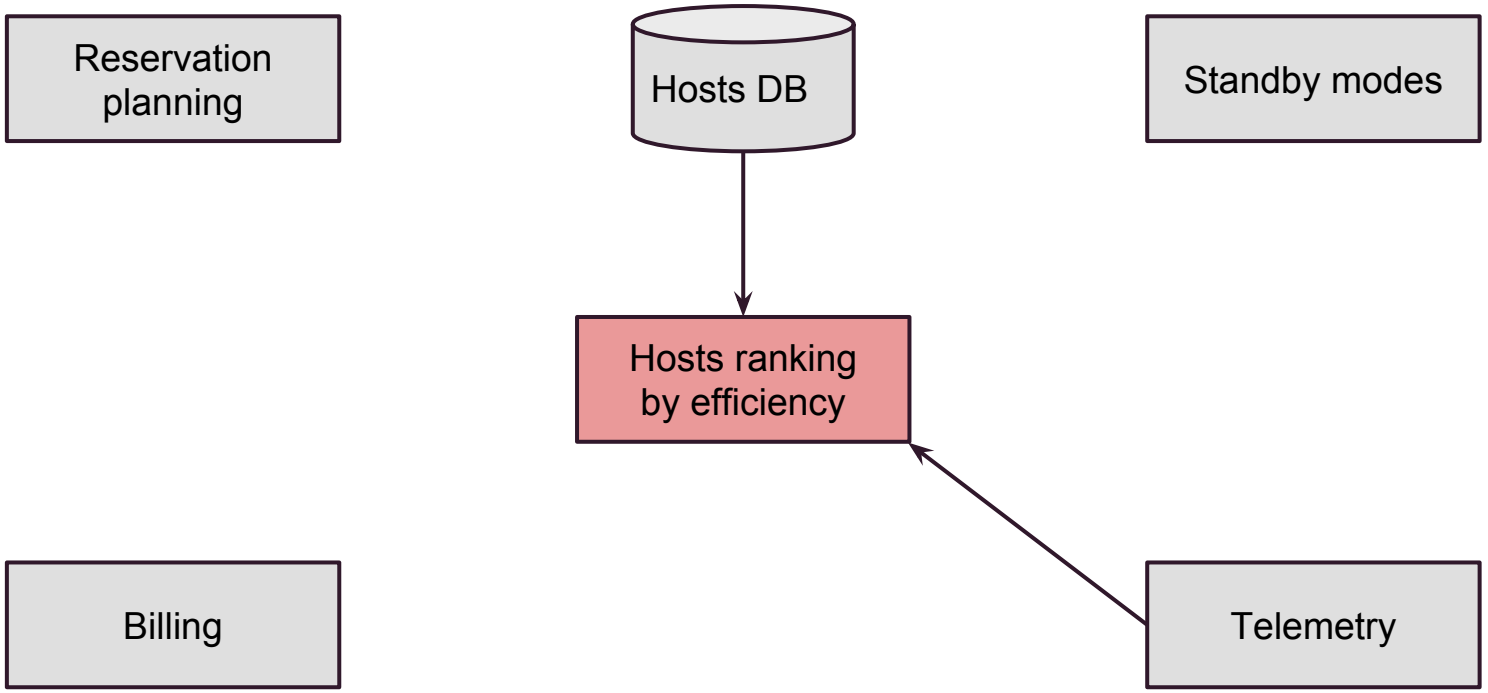
Telemetry architecture
Software layers



Telemetry architecture
Datacenter overview



Ranking


Reservation planning
Hosts DB
Standby modes
Hosts ranking by efficiency
Billing
Telemetry

Ranking

Static approach:

- Benchmark (run a job, and look how much energy is needed to complete it)

Dynamic approach:

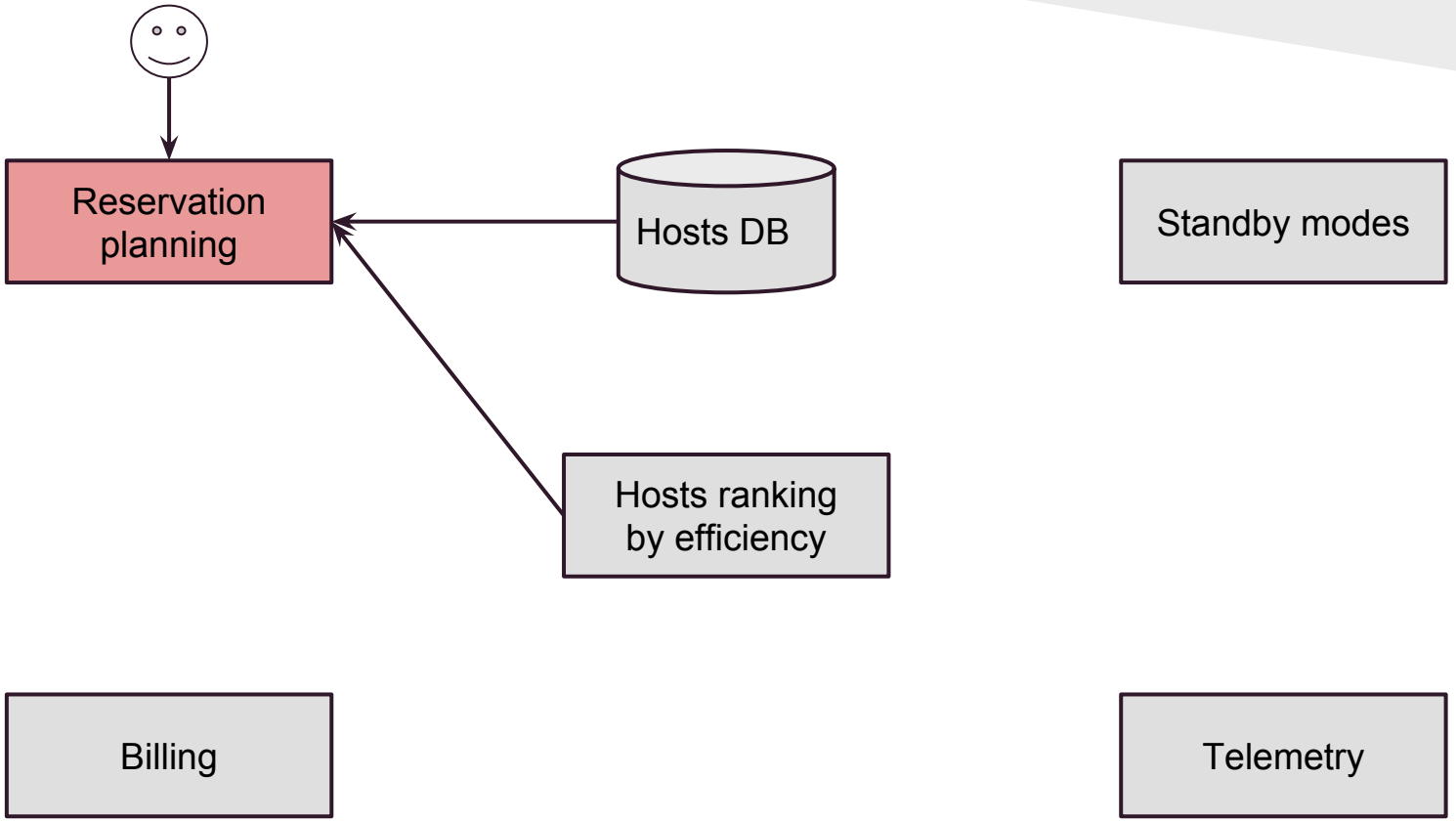
- 2 CPUs with same model have homogeneous performances

  We can build a DB to associate each CPU model with its performance metric

- 2 CPUs with same model have heterogeneous energy consumption (up to 20%)

  We need to compute the efficiency performance for each machine in the datacenter

Reservation planning



Reservation planning
Why?

Primary objective:

Provide a mono-tenancy environment with stable and good performances

=> A reservation service could fill this need, and also bring useful features.

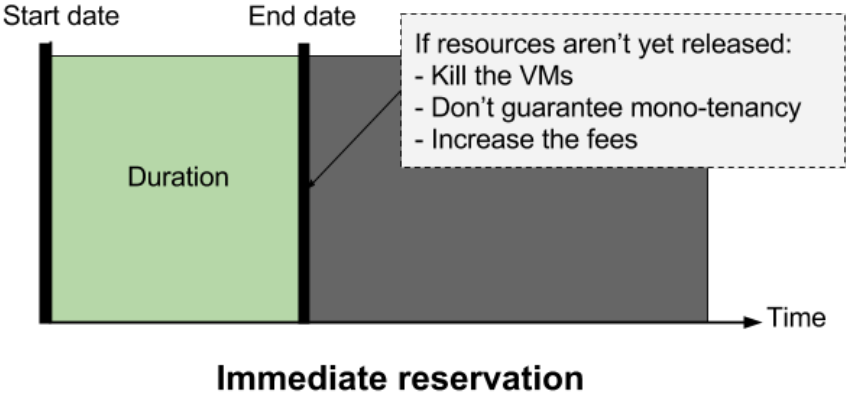
User benefits:

- Compatibility with HPC applications requirements (mono-tenancy environment)
- Lower hourly rates during off-peak hours (allow the user to delay its jobs)
- Guaranteed availability of the reserved resources (required by critical applications)

Infrastructure manager benefits:

- Load forecasts and dynamic hourly rates
- Better control over the datacenter usage
- Power capping
- Shut down the unused hosts

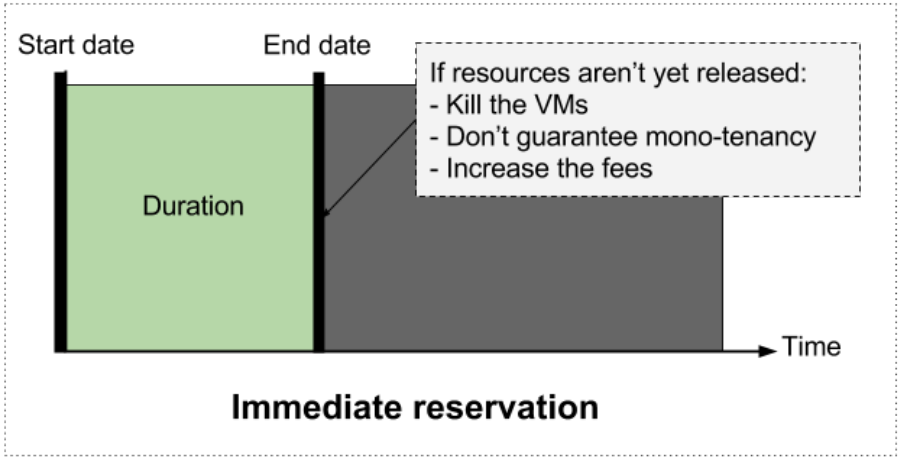
Reservation planning
Immediate reservation

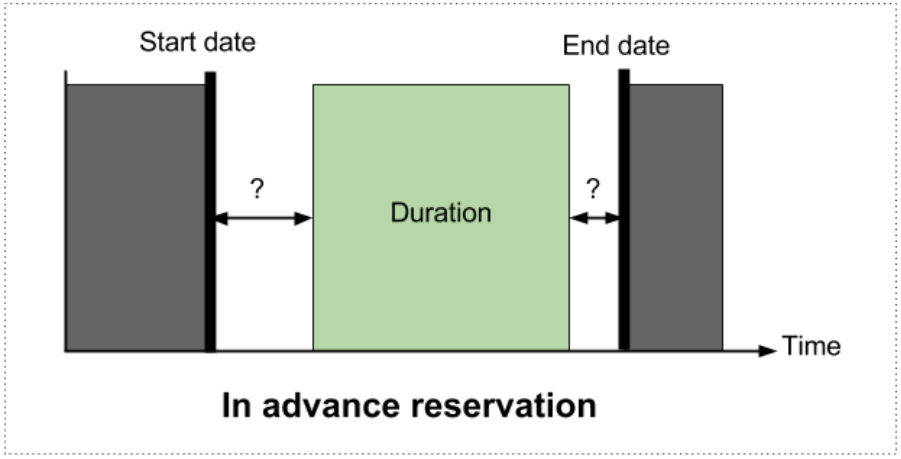


Immediate reservation

Supported by:
- Amazon
- Grid'5000
- OpenStack
- OpenNebula

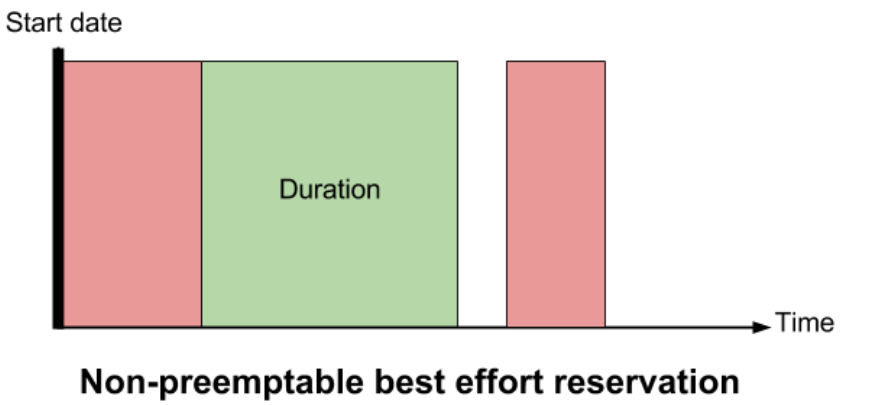
Reservation planning
In advance reservation



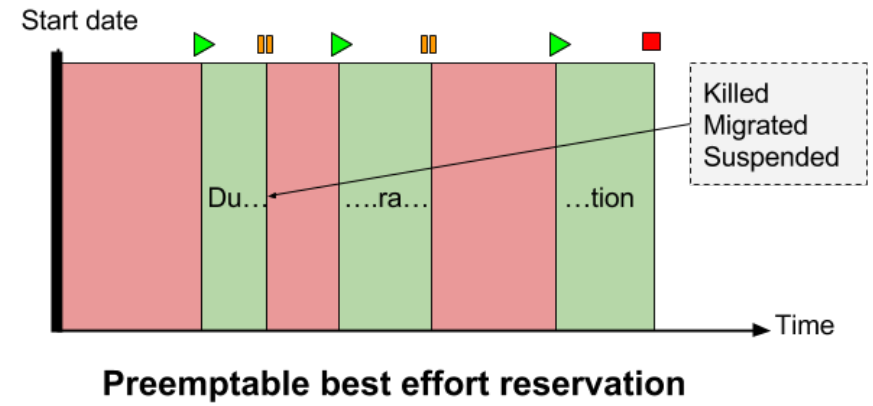


Supported by:
- Grid'5000
- OpenNebula (with Haizea)

Reservation planning
Best-effort reservation



Non-preemptable best effort reservation



Preemptable best effort reservation

Supported by:
- Amazon (spot instances)
- Grid'5000 (best effort)
- OpenNebula (with Haizea)

Reservation planning
Mono-tenancy

Amazon:

- And option “Dedicated instances” (mono-tenancy / network isolation)

Grid’5000:

- Supported

OpenStack:

- Not natively supported:

OpenNebula:

- Not natively supported

Reservation planning
Main features

Reservation of physical and virtual resources:

- Physical hosts
- Instances
- Network
- Volumes

Several reservation types covering the main use cases (Haizea-like):

- Immediate
- In advance
- Best-effort

Climate framework:

- Included in OpenStack
- Developed by Bull, Inria, and Mirantis

Reservation planning
Reservation request

Lease

start = 1385721907
end = 1385770000
duration = 7200

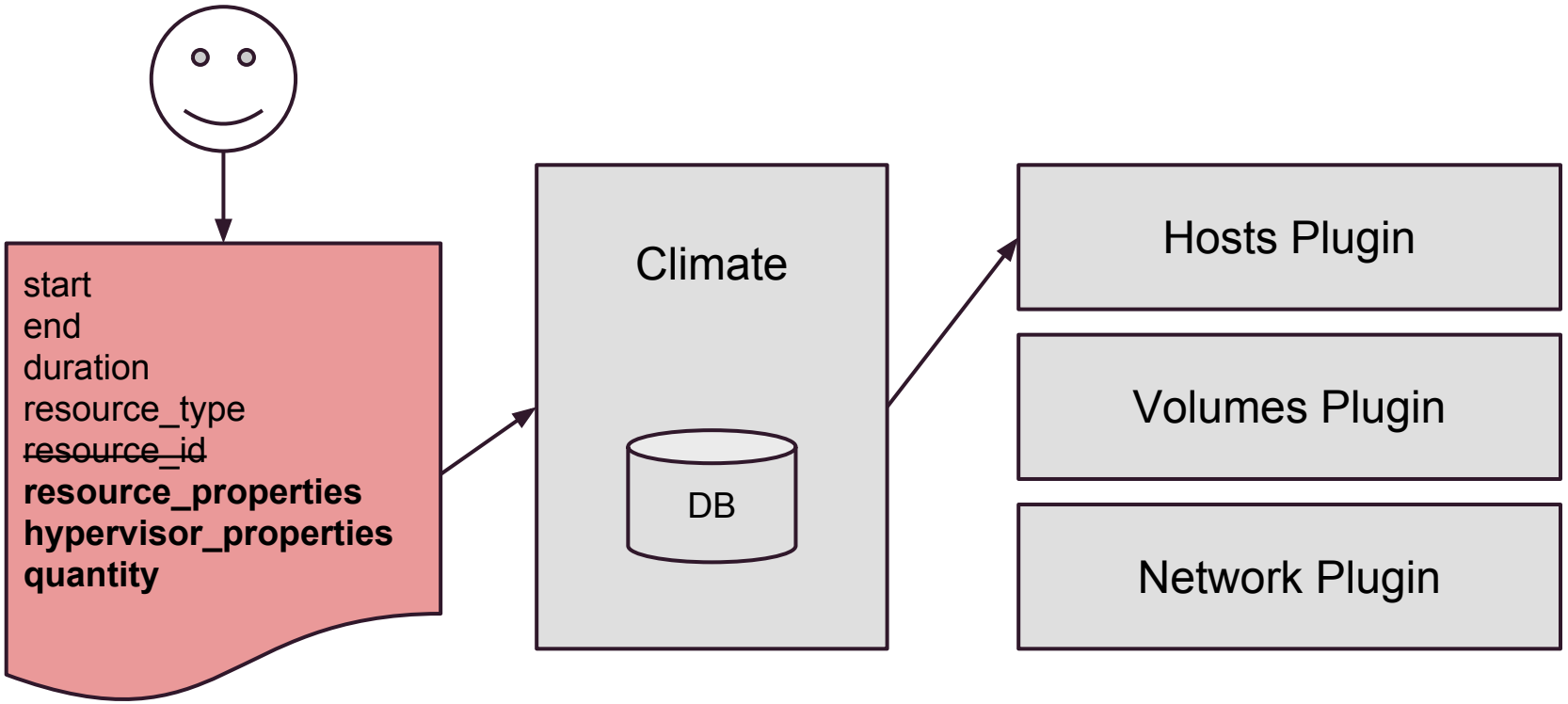
Reservation

resource_type = physical:host
resource_properties = RAM>4GB
hypervisor_properties = name=XEN
quantity = 5

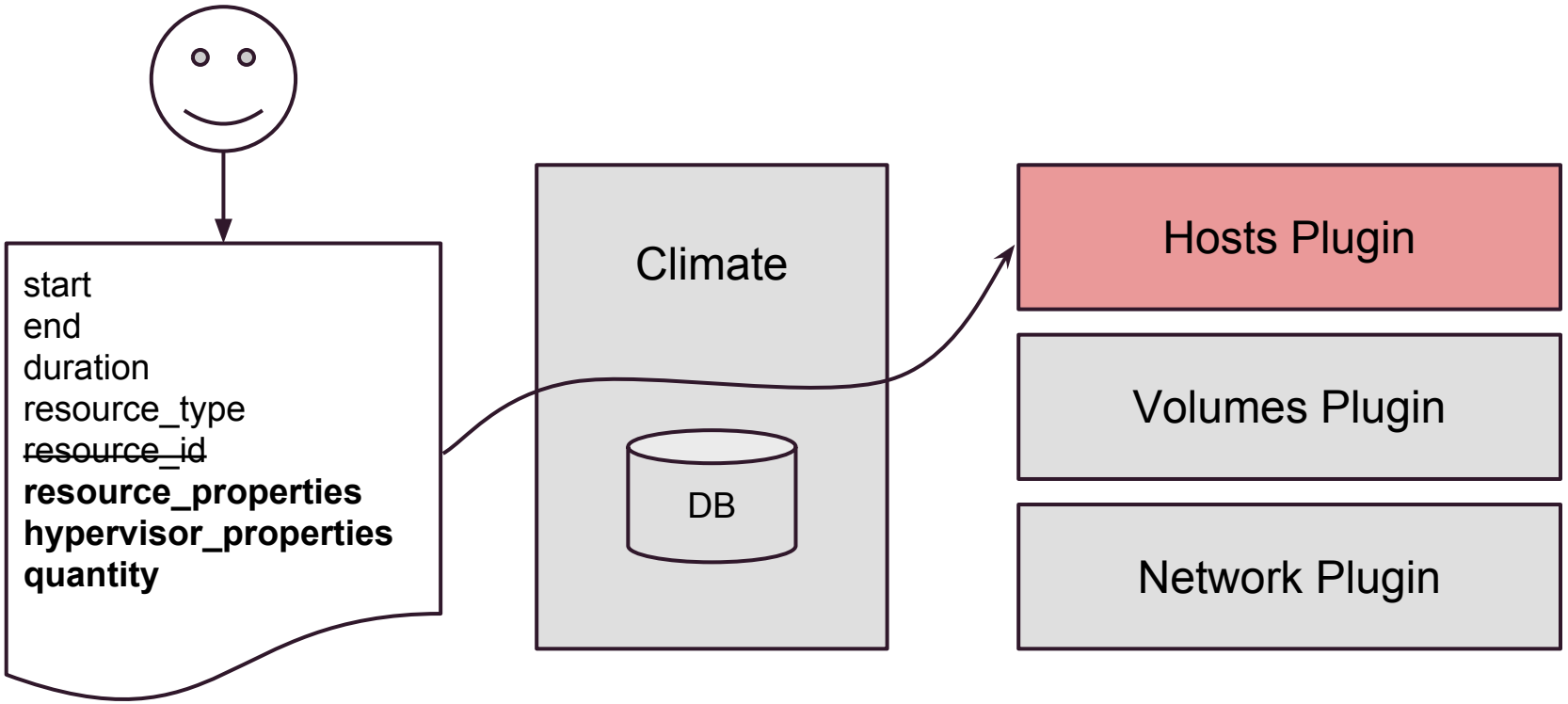
Reservation

resource_type = volume
resource_id = 8594-4940-3380-1048

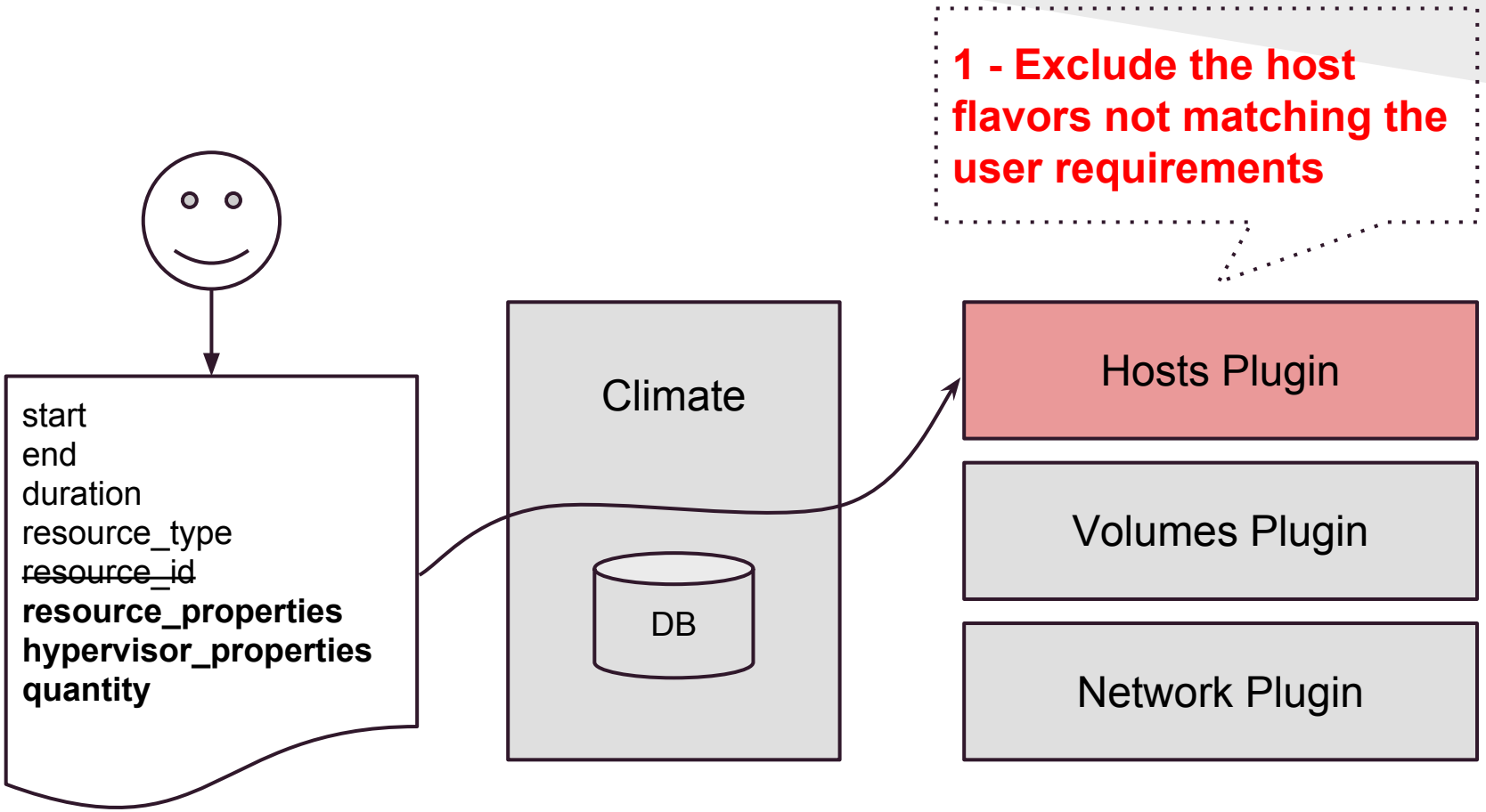
Reservation planning
Workflow



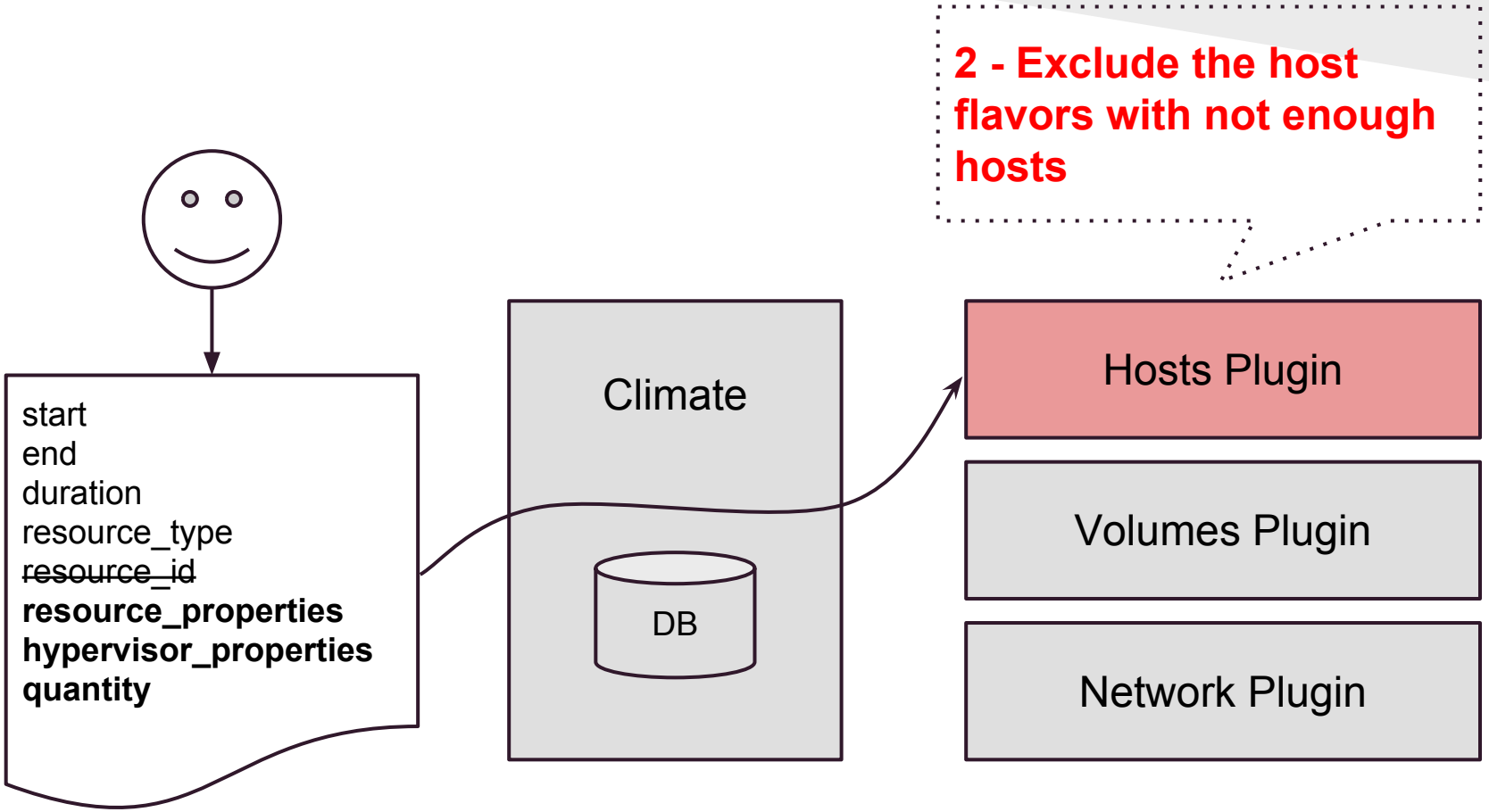
Reservation planning
Workflow



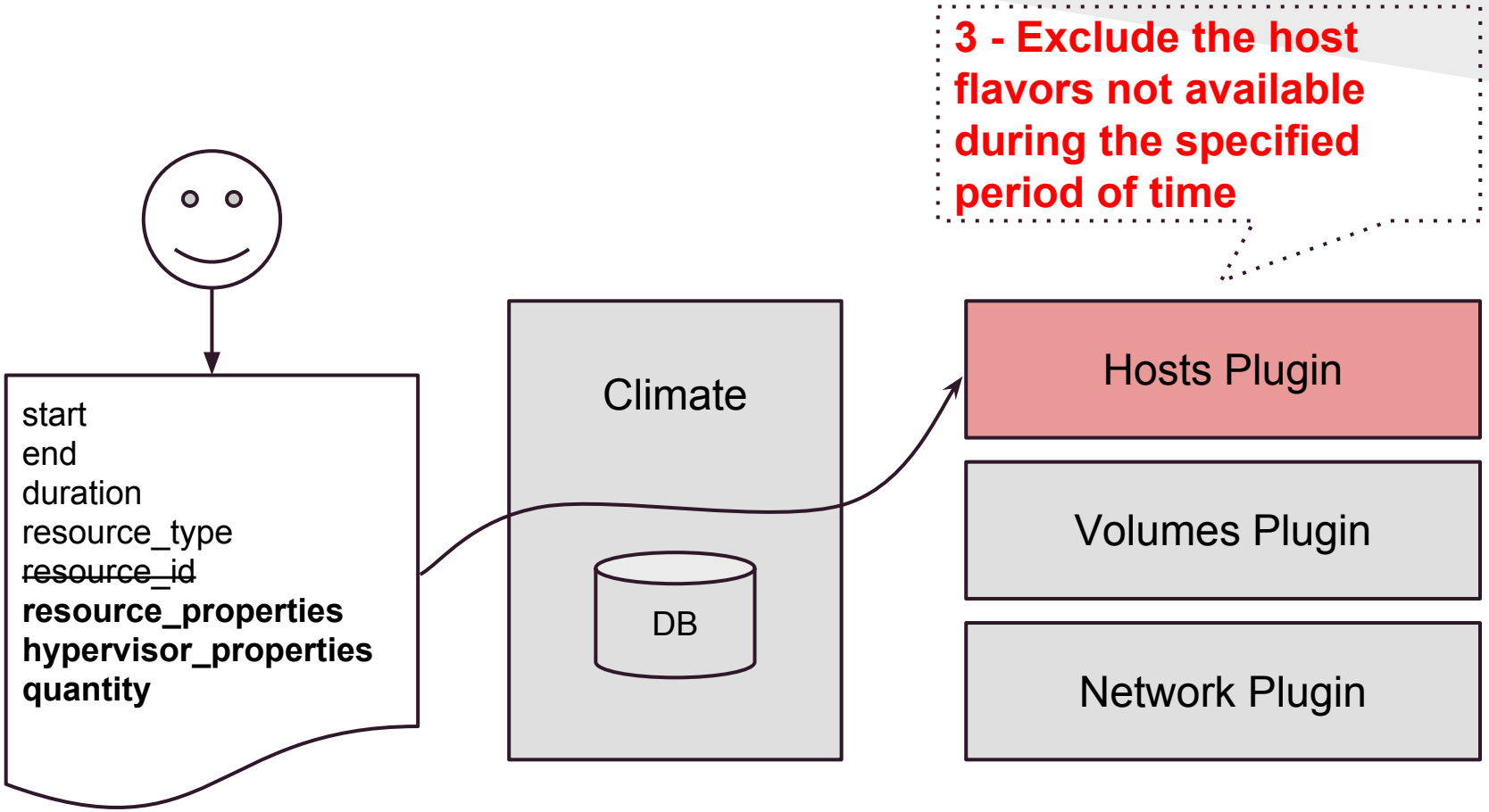
Reservation planning
Workflow



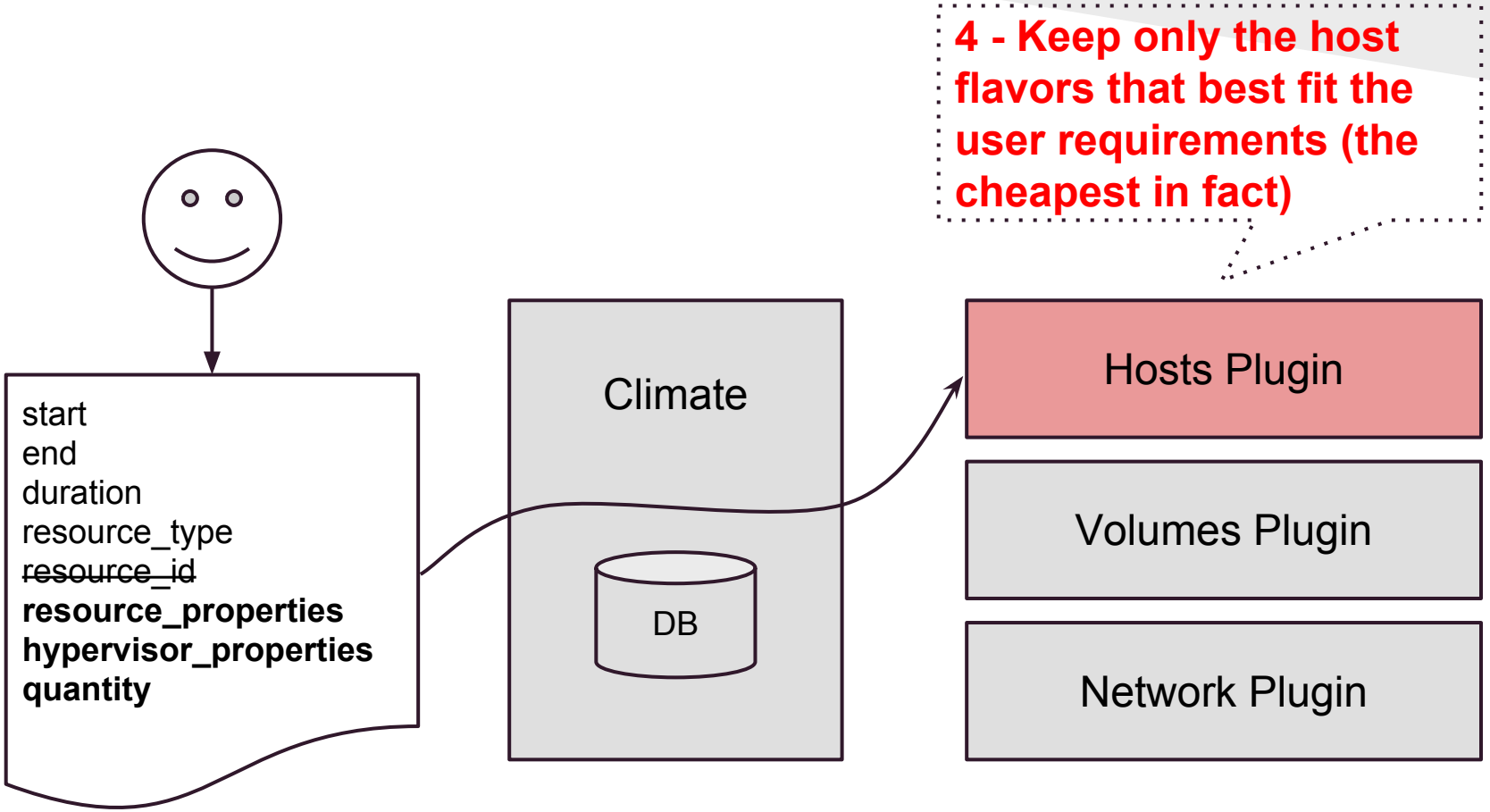
Reservation planning
Workflow



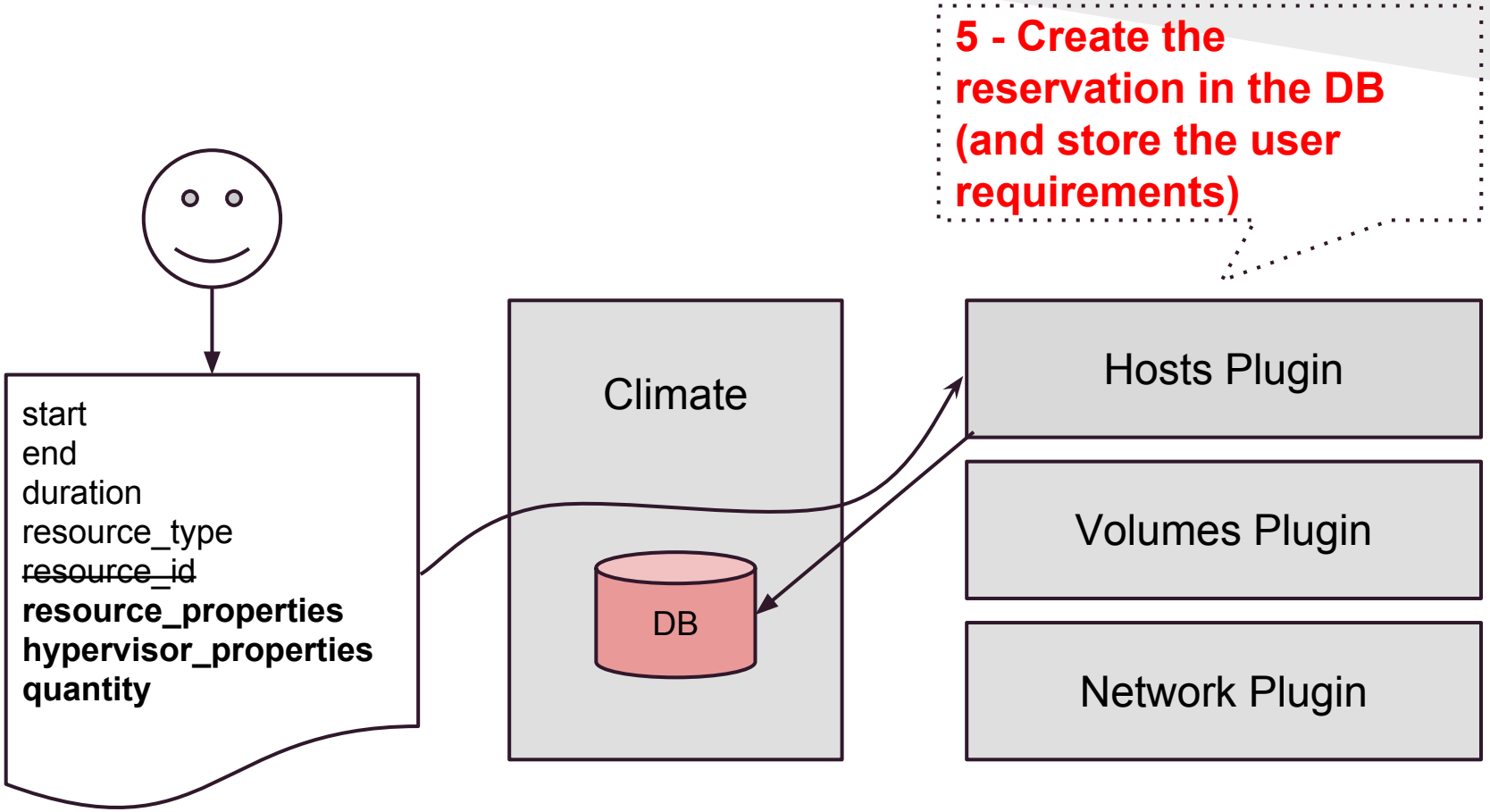
Reservation planning
Workflow



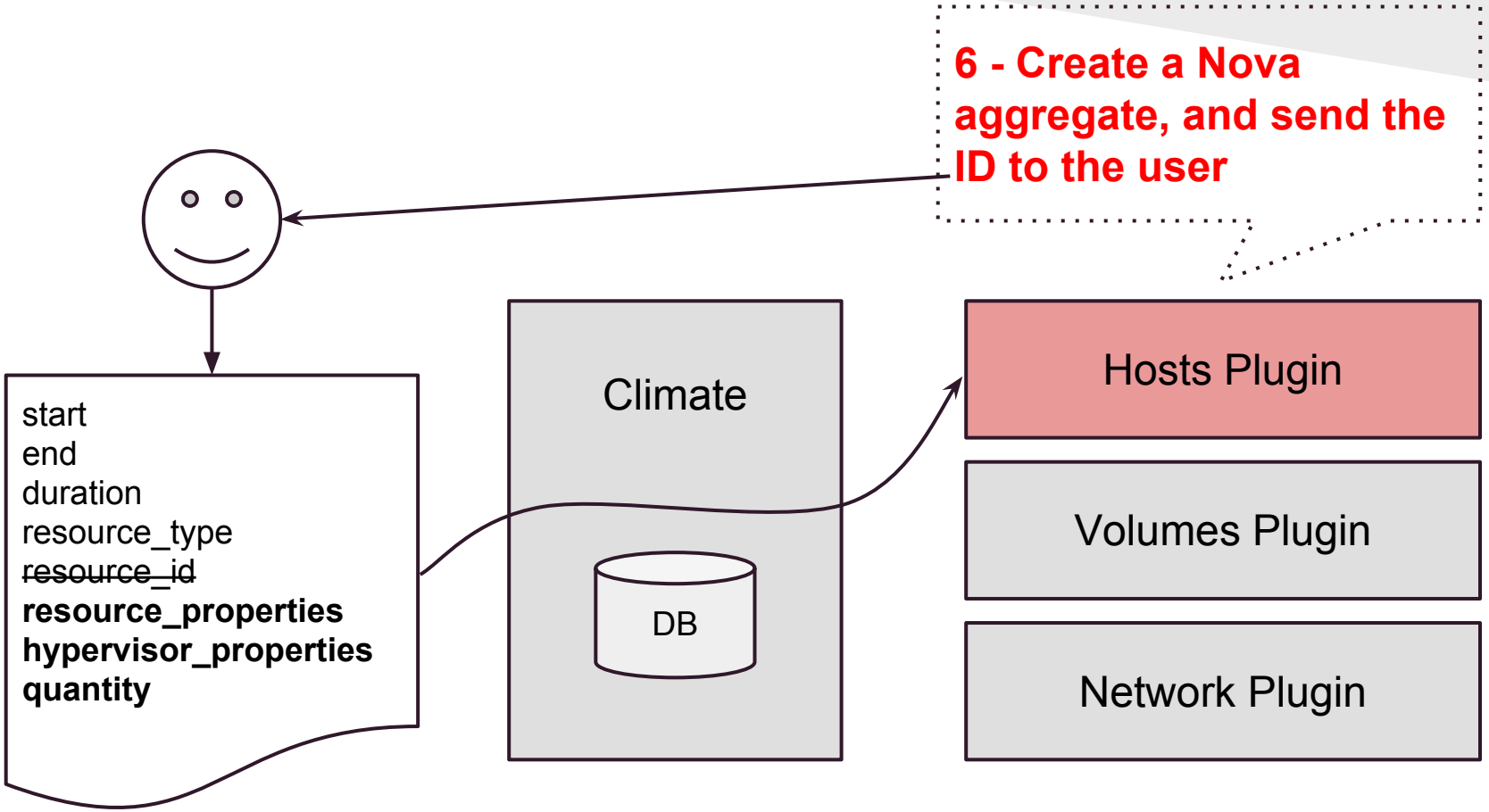
Reservation planning
Workflow



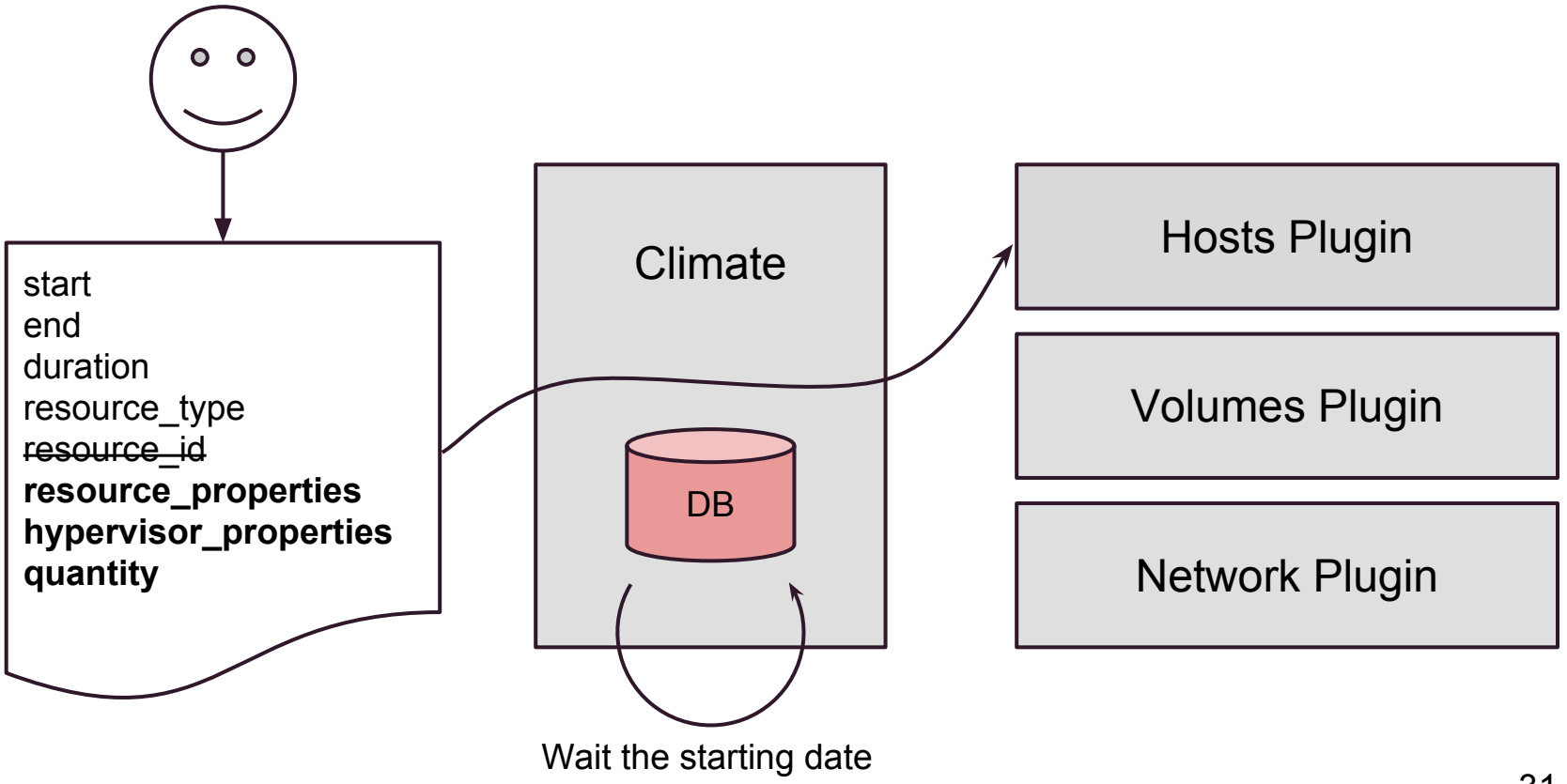
Reservation planning
Workflow



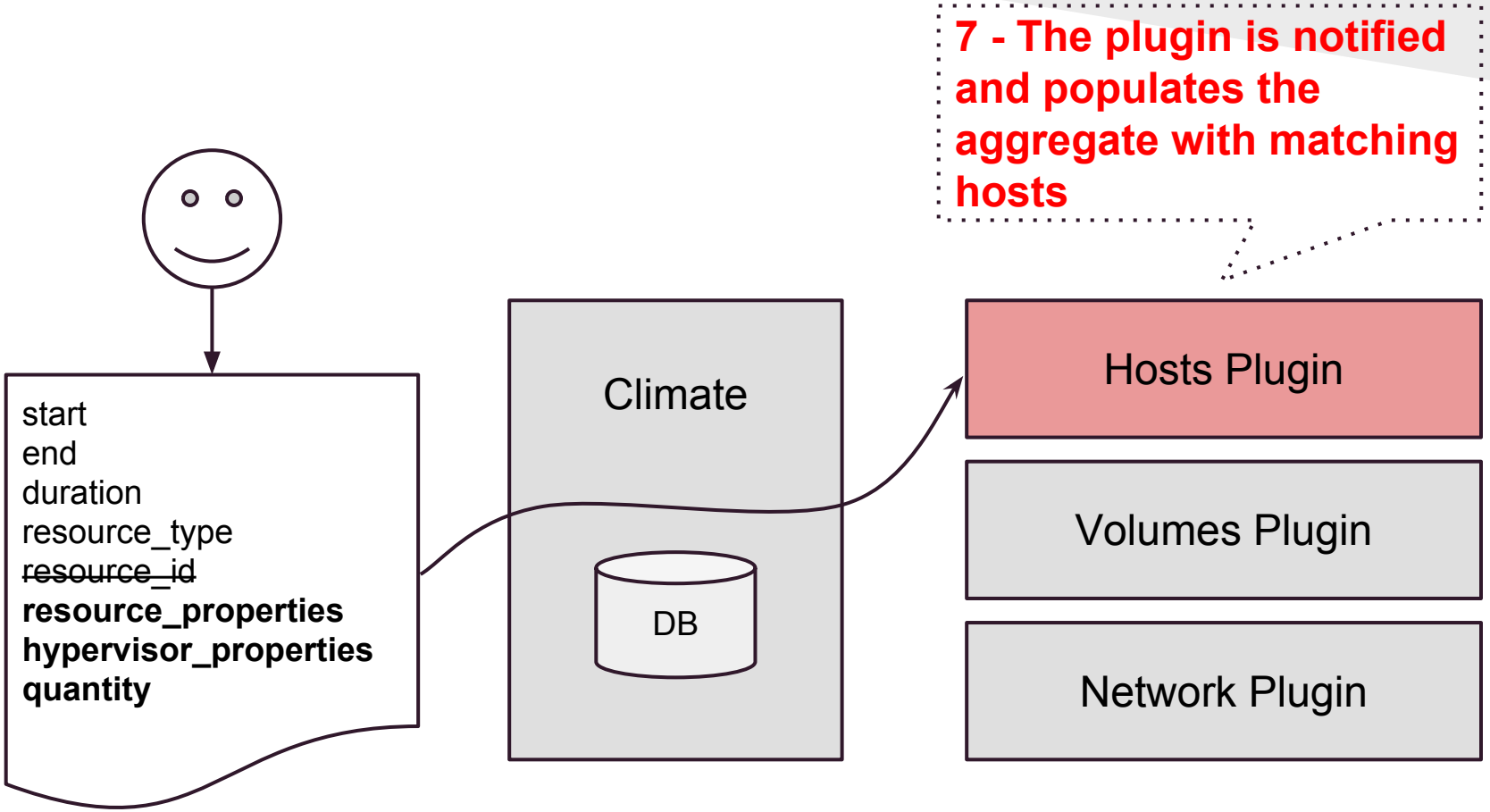
Reservation planning
Workflow



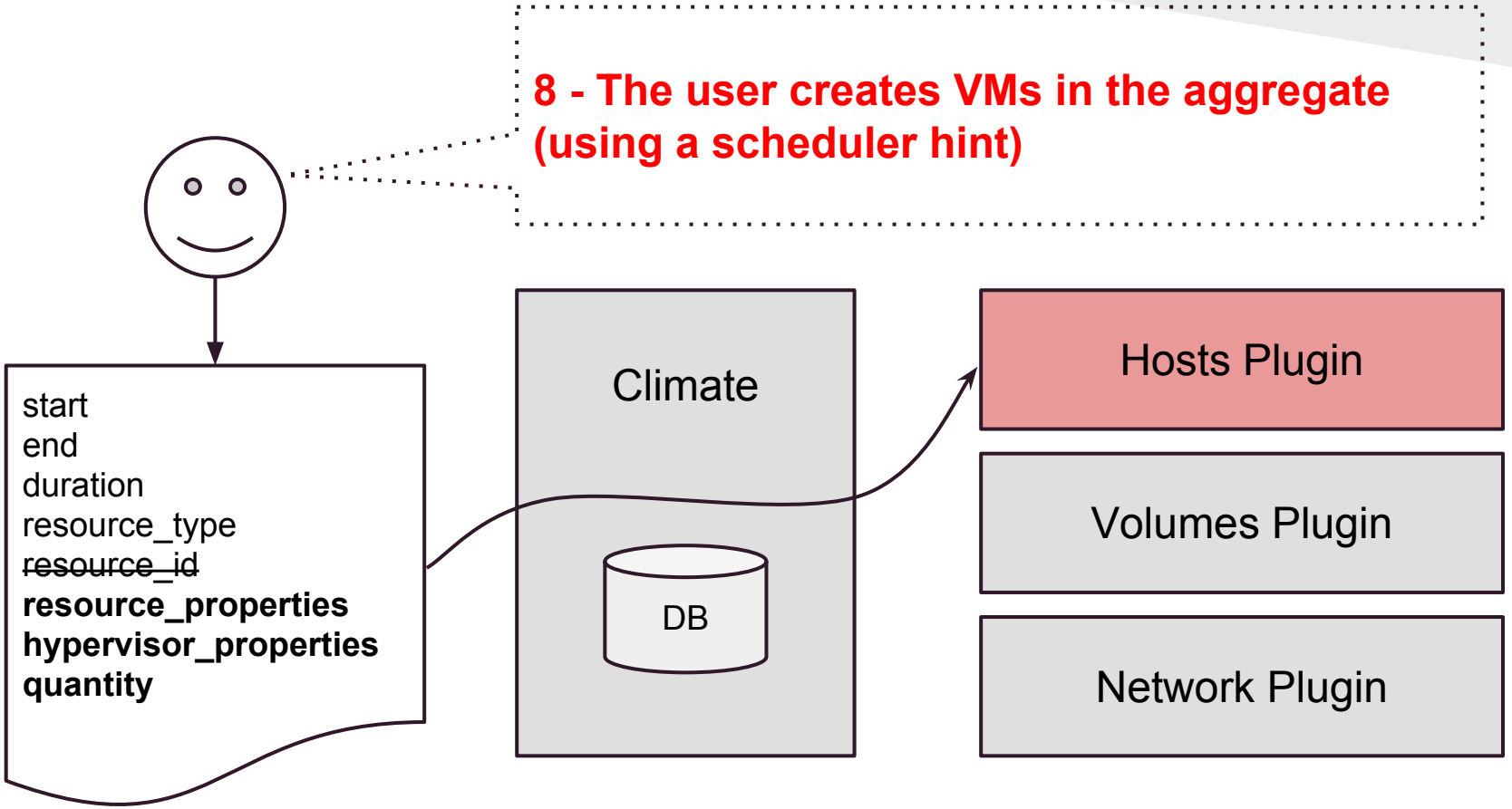
Reservation planning
Workflow



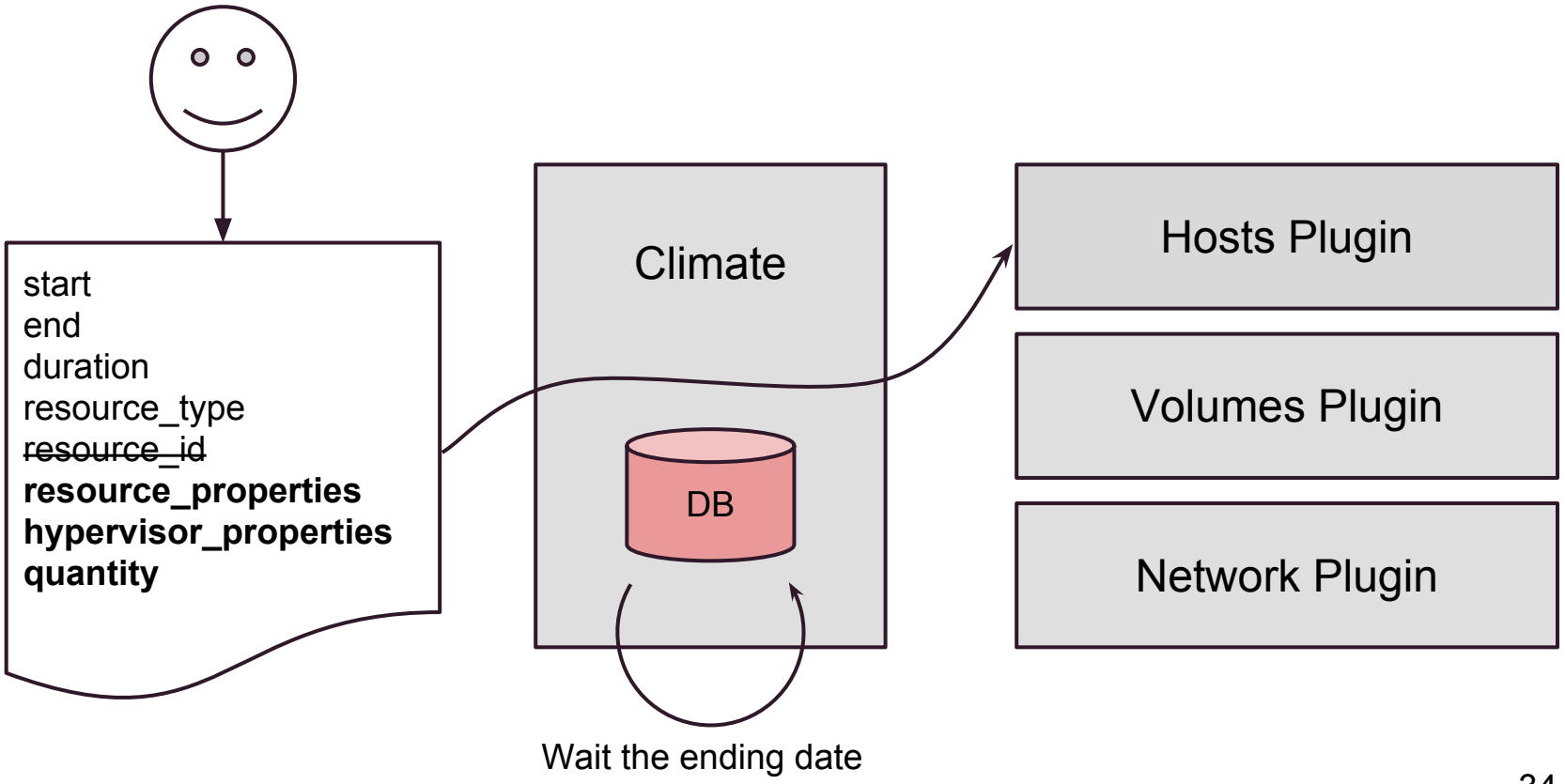
Reservation planning
Workflow



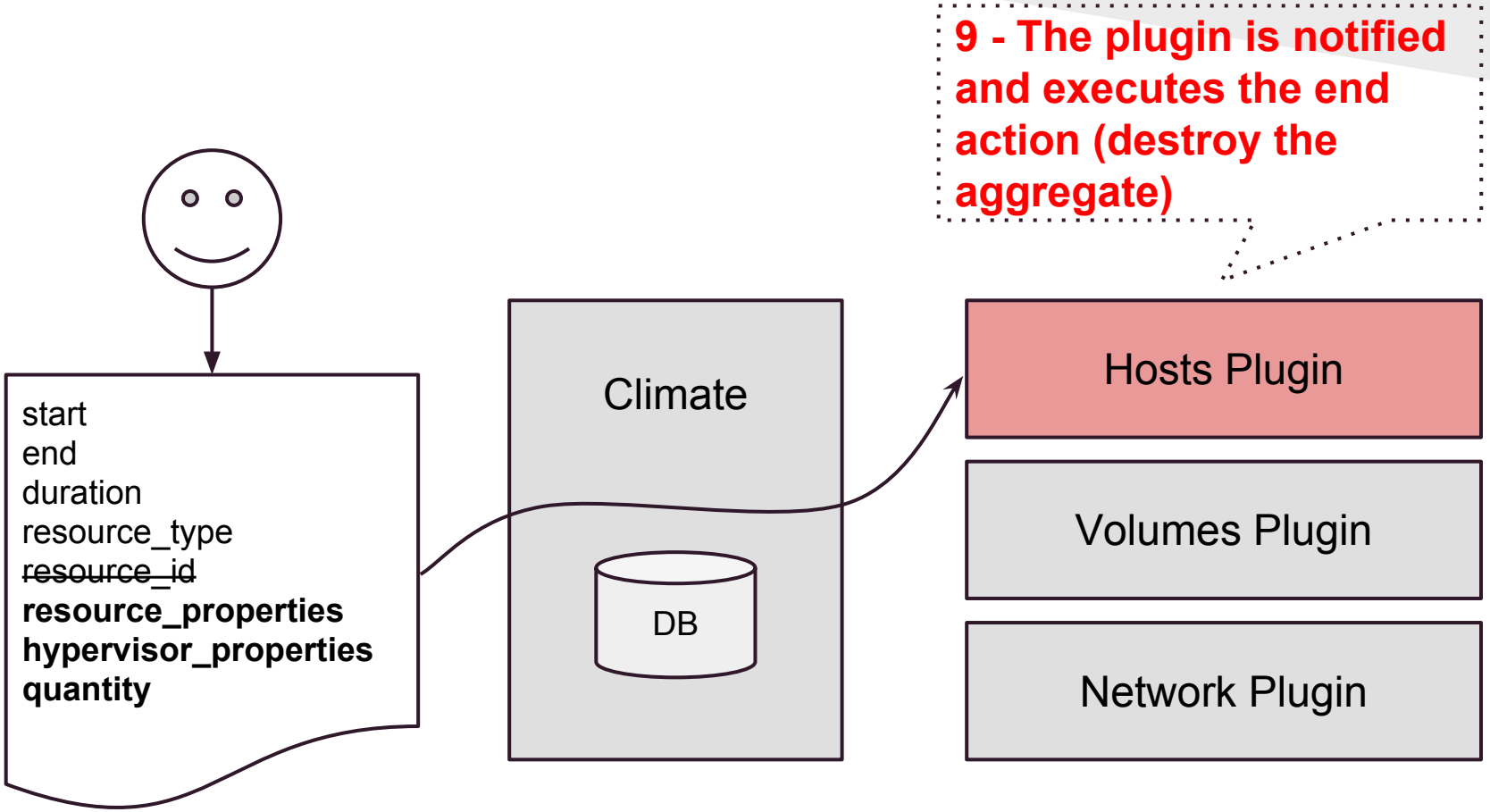
Reservation planning
Workflow



Reservation planning
Workflow



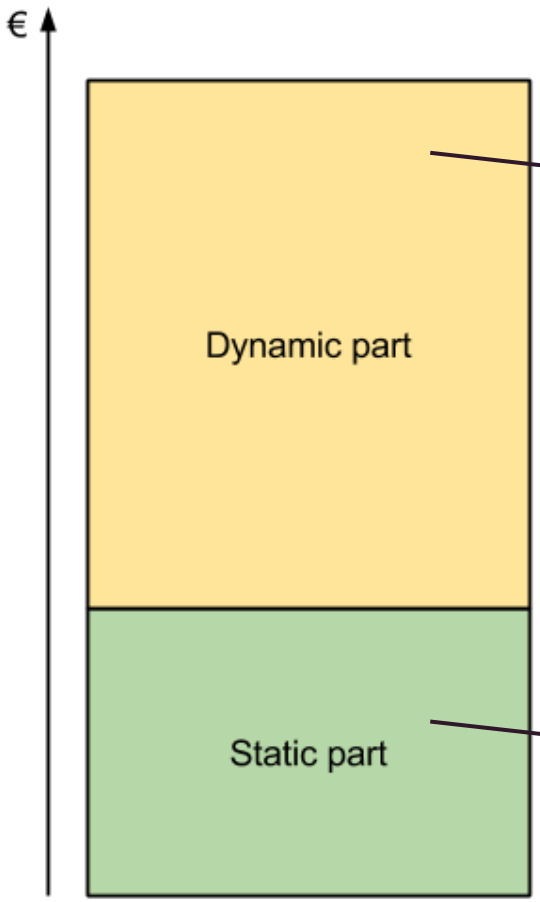
Reservation planning
Workflow



Billing


Reservation planning
Standby modes
Hosts ranking by efficiency
Billing
Telemetry

Billing



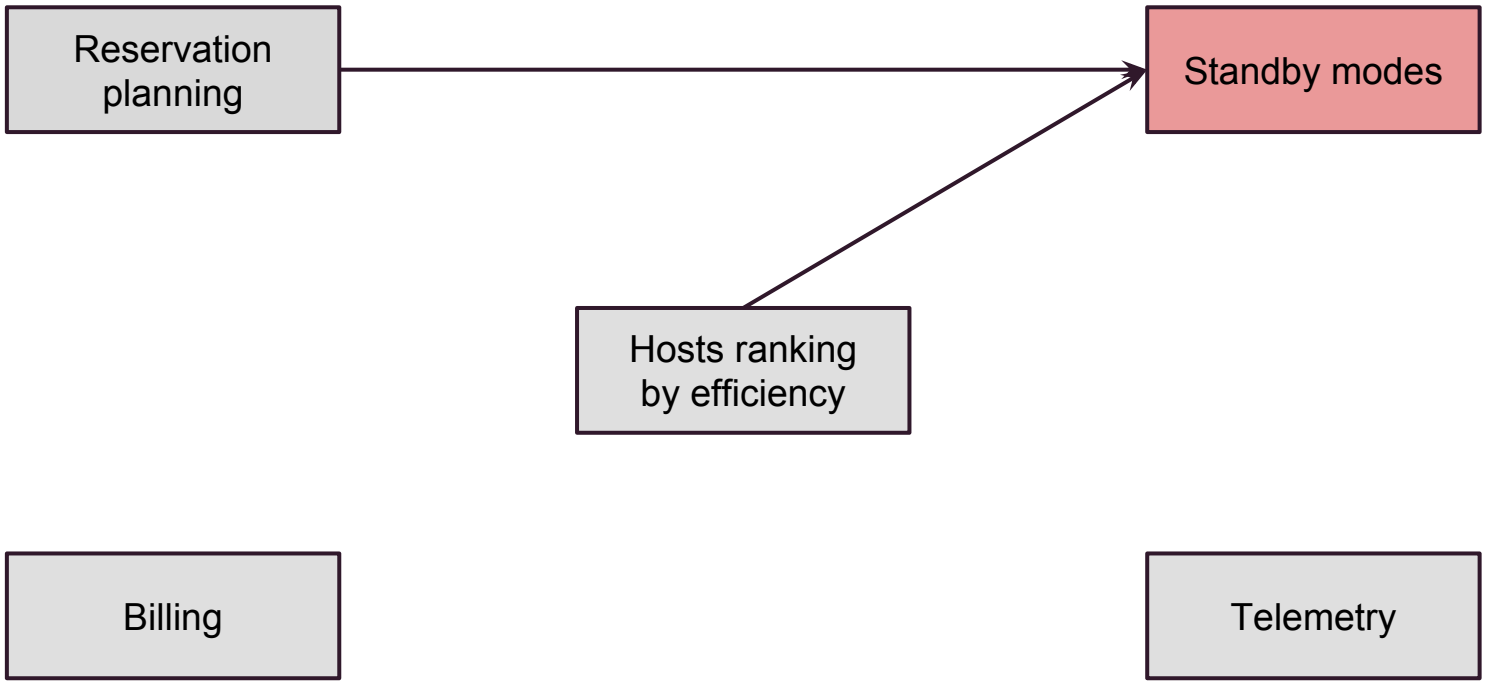
Dynamic (including energy)

Power metrics are tied to the customers:
- Implies that a physical machine is not shared among multiple customers
- Only possible while a reservation is active

Static

- Based on the reservation properties
- Examples: performances, electricity rates

Standby Modes



Standby Modes

Managing each group of identical machines:

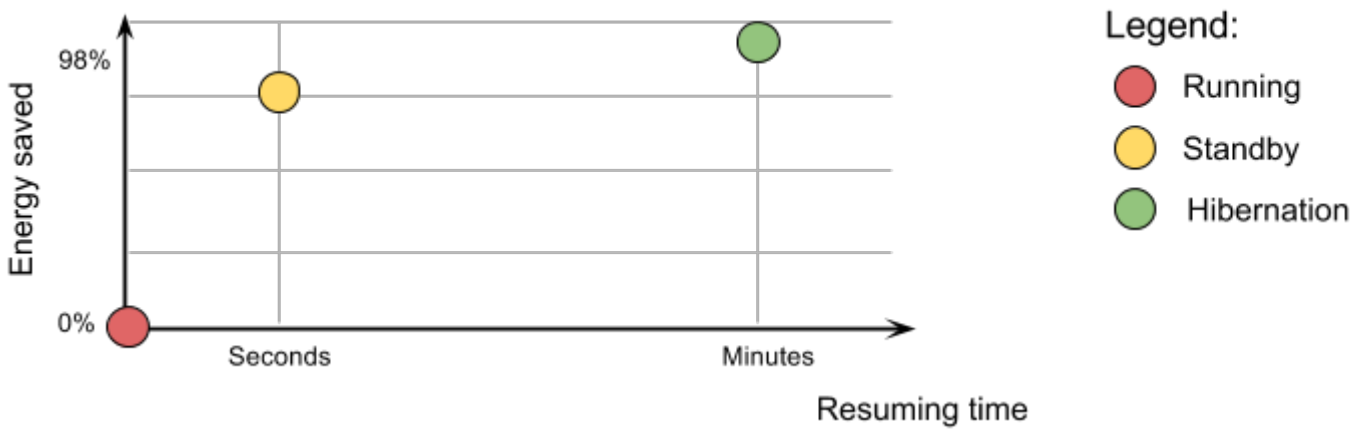
Defining dynamically the proportion of machines in each mode

Anticipating demand peaks:

- By consulting the reservation planning
- By looking at the history

Standby Modes

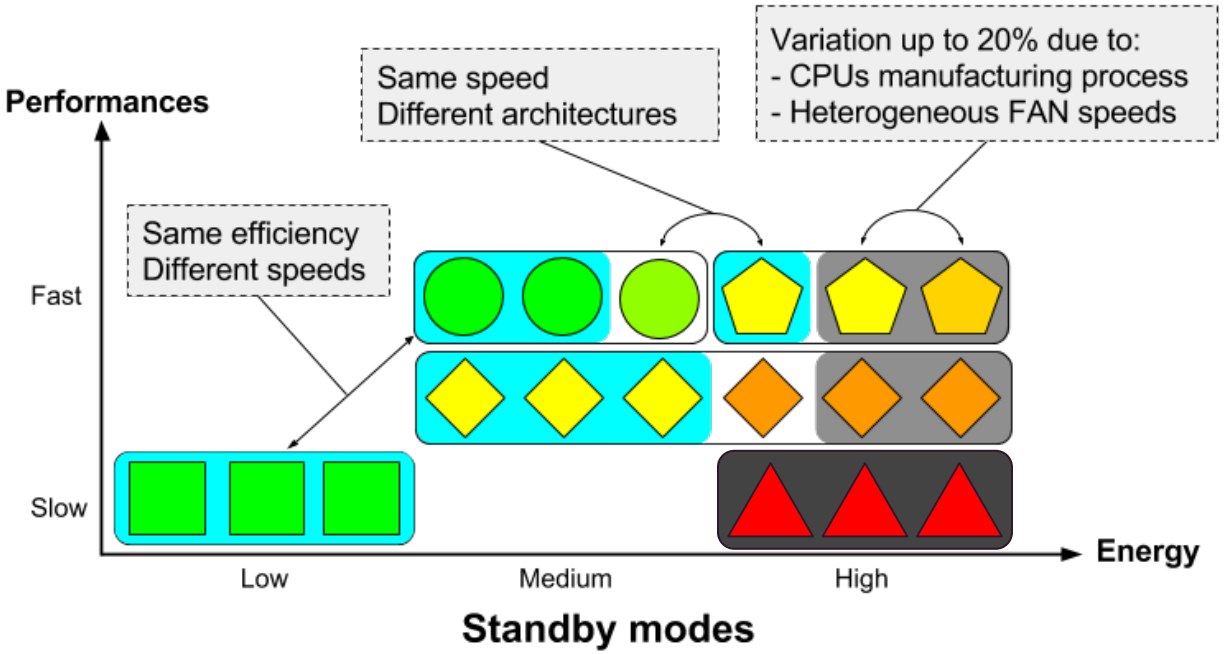
Energy saved / resuming time:

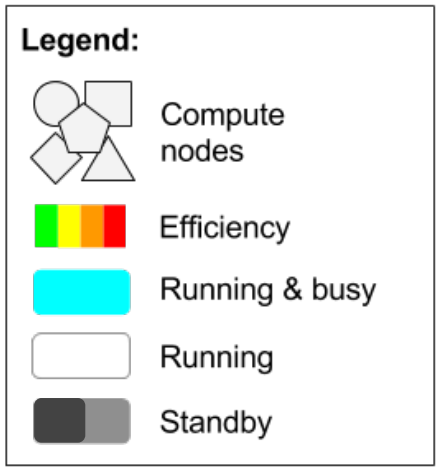


Over-consuming during resuming could ruin our efforts

Avoiding too frequent shutdown / start-up cycles

Standby Modes





Conclusion & Ongoing works

Conclusion:

The reservation service is the cornerstone of a lot of useful features, for the user and the provider. It is currently actively developed, and the other services which will be developed later will plug on it.

Ongoing works:

- Climate: very active development (Mirantis joined us recently)
- Standby modes: todo

Thank you
for your attention