Orchestration with Openstack Heat

Why ‘Heat’? It makes the clouds rise!

February 25, 2016

Hassaan Ali hassaan.ali@xflowresearch.com

What are we going to do today?

- What is Orchestration and Why we do it?
- Heat Orchestration Template
- Architecture of Heat
- Heat Workflow
- Working with Heat
- Heat Use cases

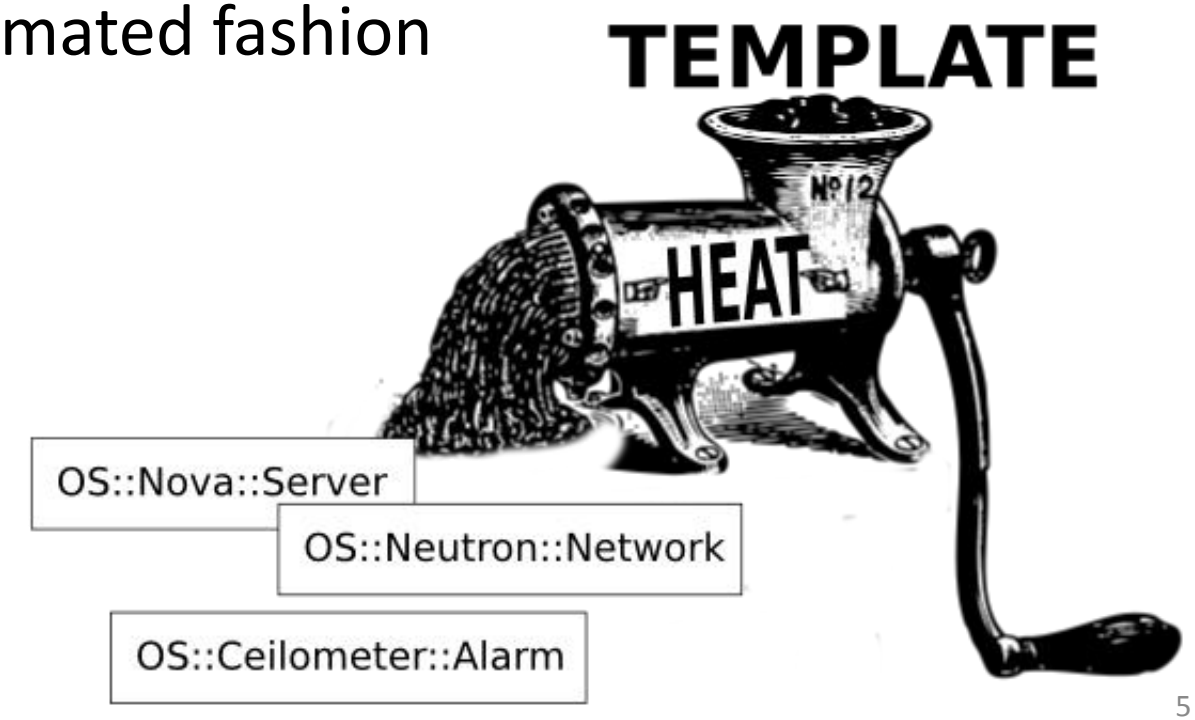
Orchestration

Orchestration (In Computing)

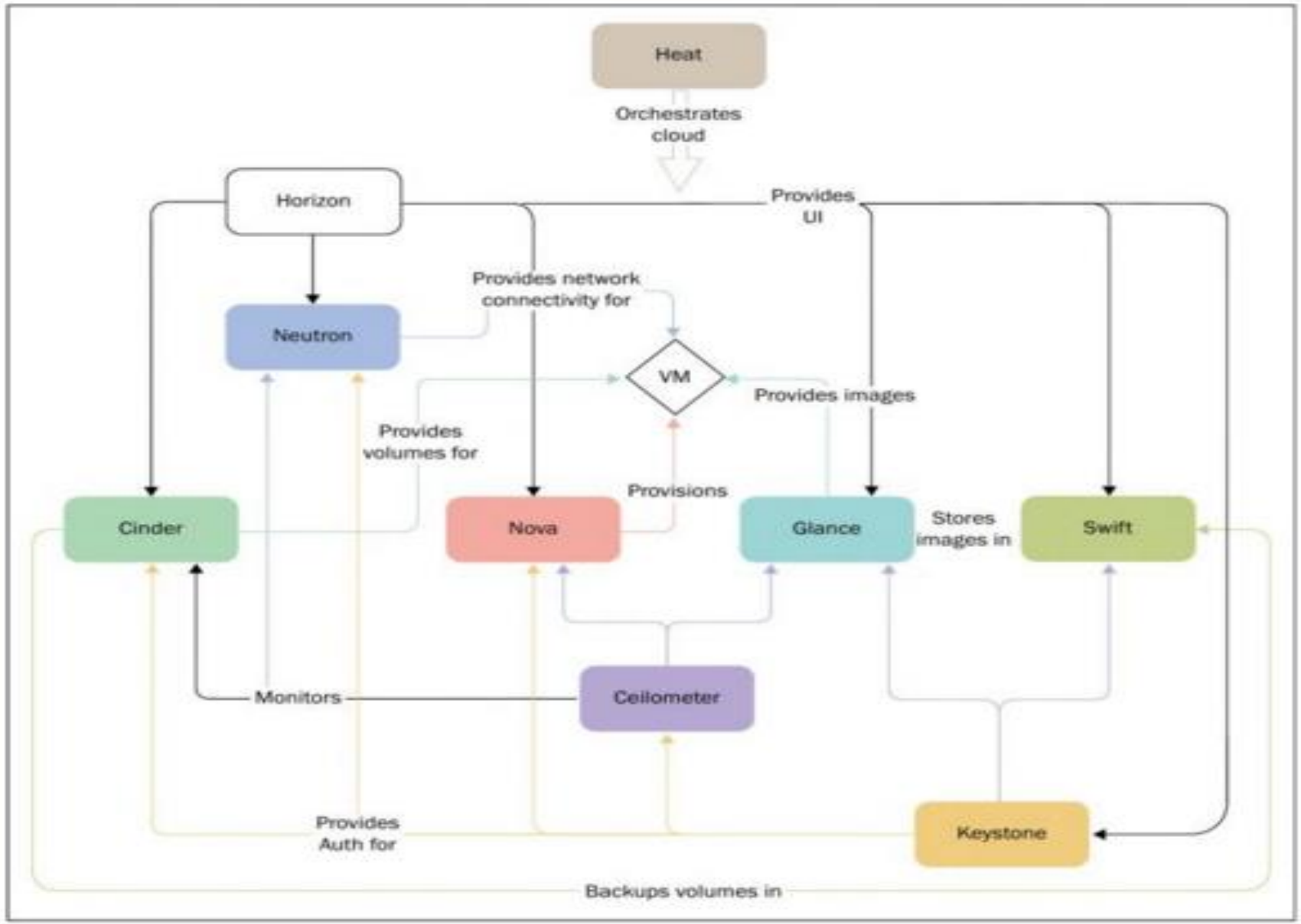
"Orchestration" is the "automated arrangement, coordination, and management of complex computer systems“ (Wikipedia)

Heat

- Heat provides a mechanism for orchestrating OpenStack resources through the use of modular templates
- Heat allows you to spin up multiple instances, logical networks, and other cloud services in an automated fashion



Openstack Logical Architecture



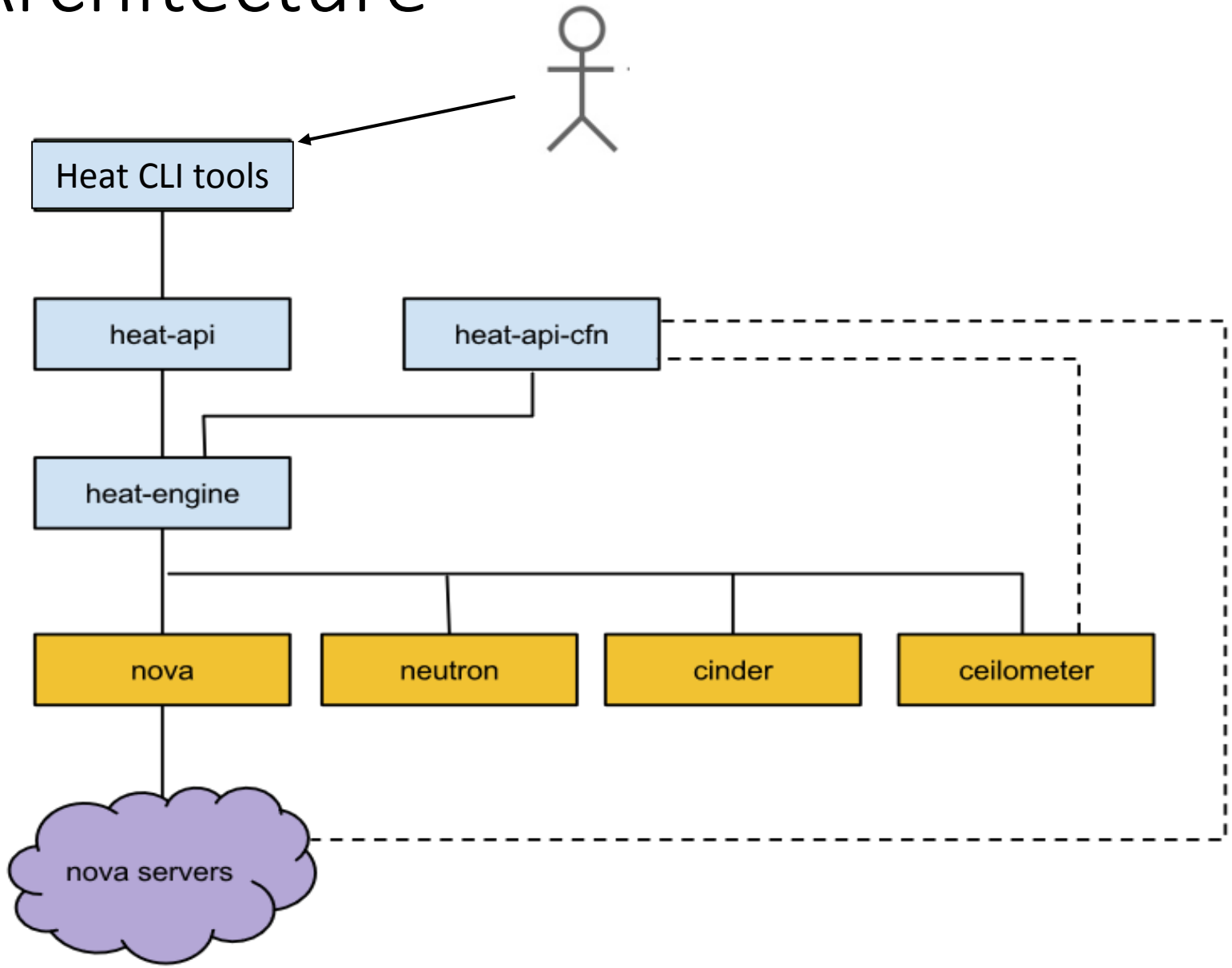
Heat Basics - Template

- Templates aka HOT are written in YAML syntax
- Templates define a stack
- Stack – group of connected cloud resources (VMs, Volumes, networks etc.)
- Heat templates have the same structure & abstractions as AWS CloudFormation templates

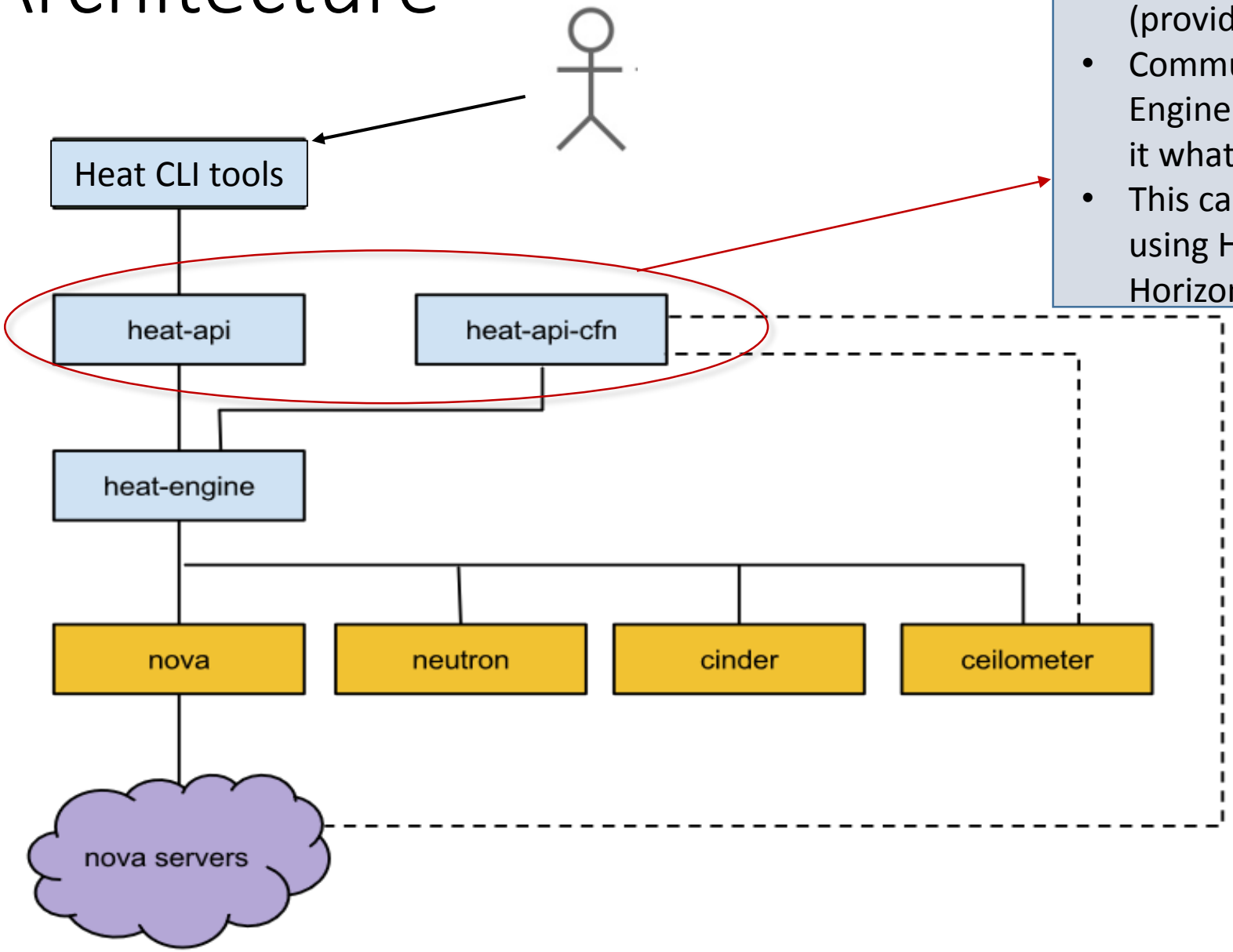
HOT – Basic Format

```
heat_template_version: 2014-10-16 #compulsory

description: This is not compulsory

parameters:
#paramterize configurations inside a heat template

resources:
#all things that can be accessed through Openstack API say network, volumes and instances etc.

outputs: #not compulsory
#If you want to give output to user upon execution of template
```

Heat Architecture



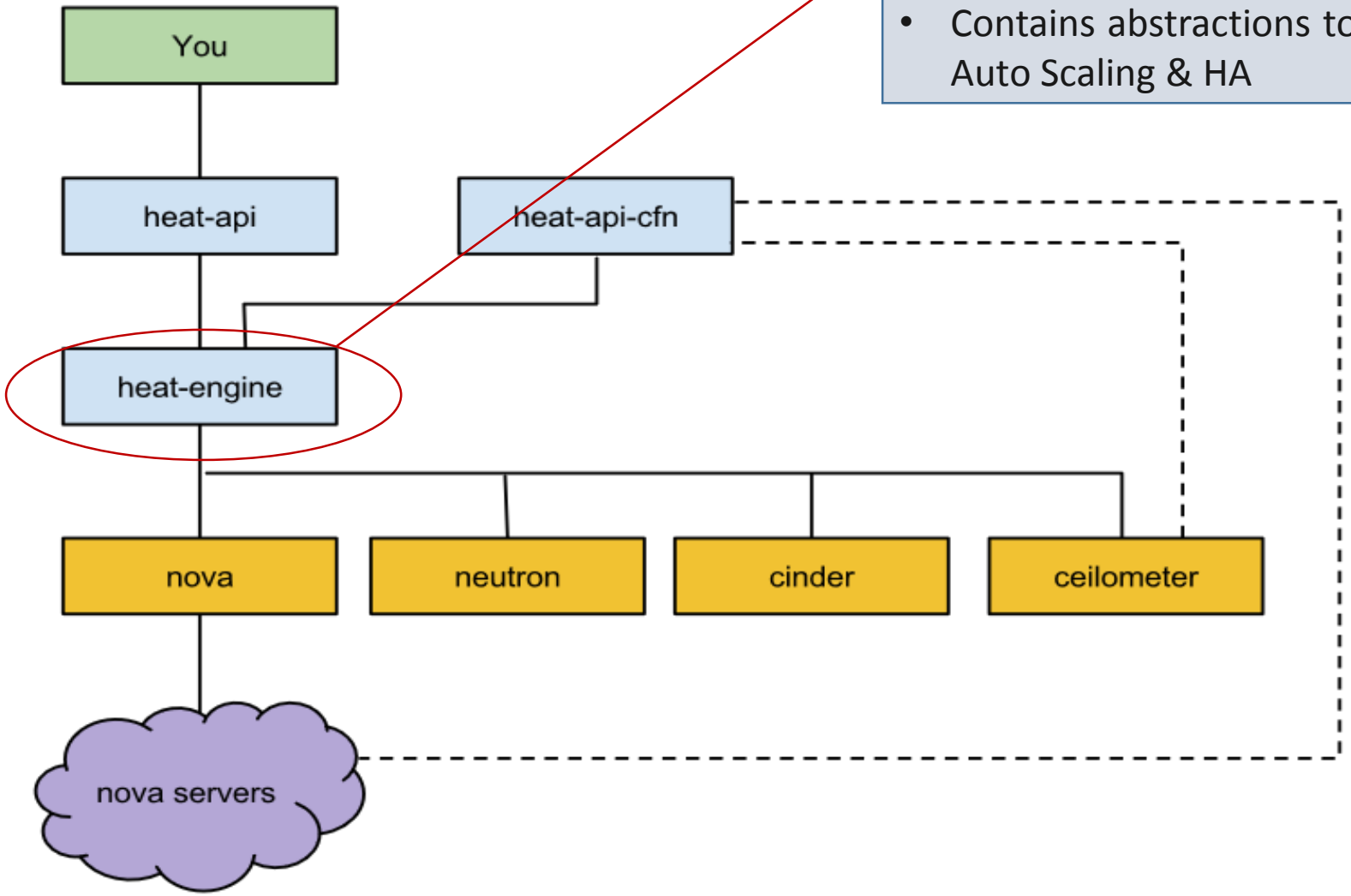
Heat Architecture



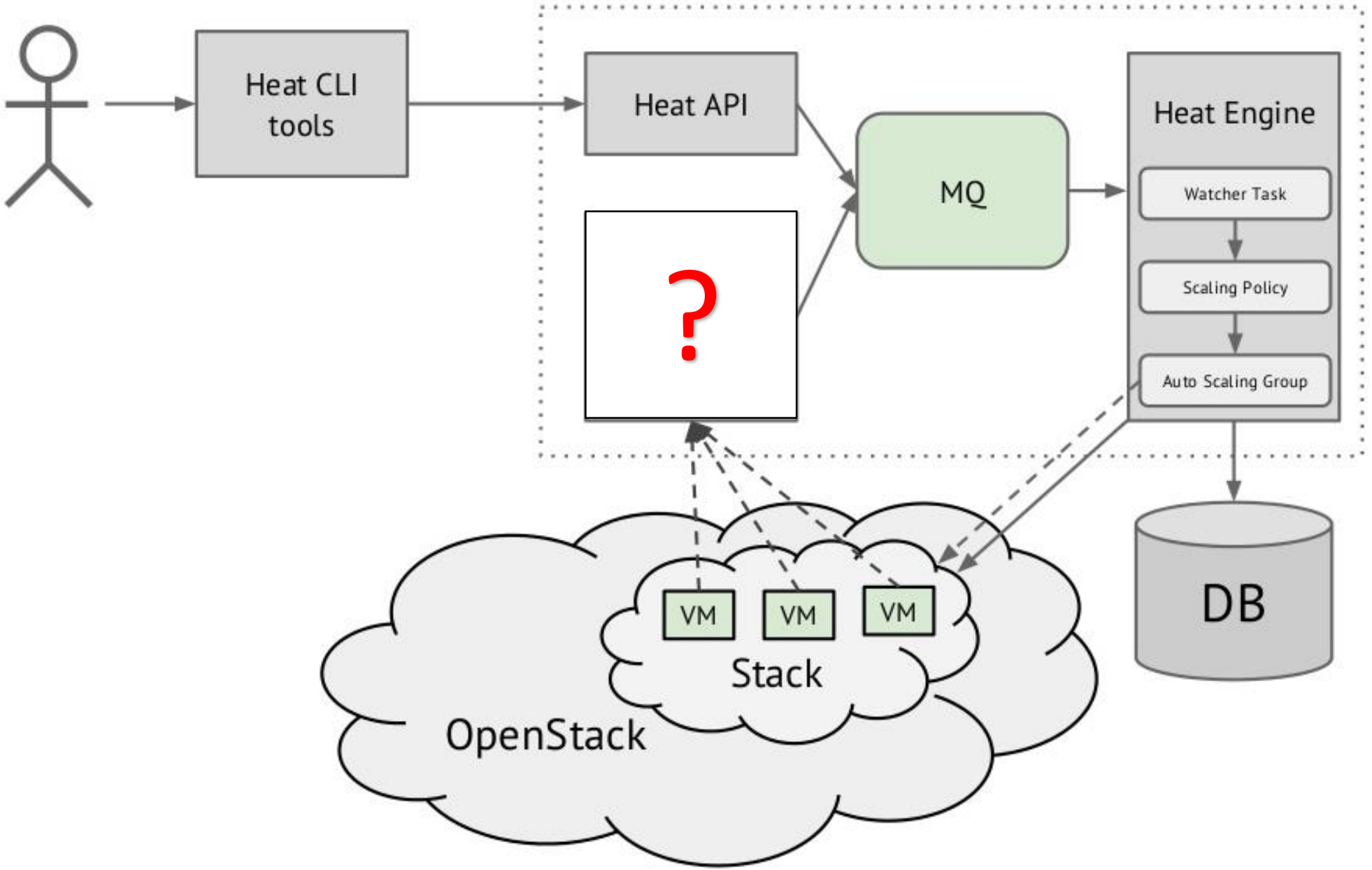
- Heat-api (Openstack native REST API) or heat-api-cfn (provides AWS Query API)
- Communicates with Heat Engine via AMQP and tells it what actions to perform
- This can be initiated from using Heat CLI tools or Horizon Dashboard

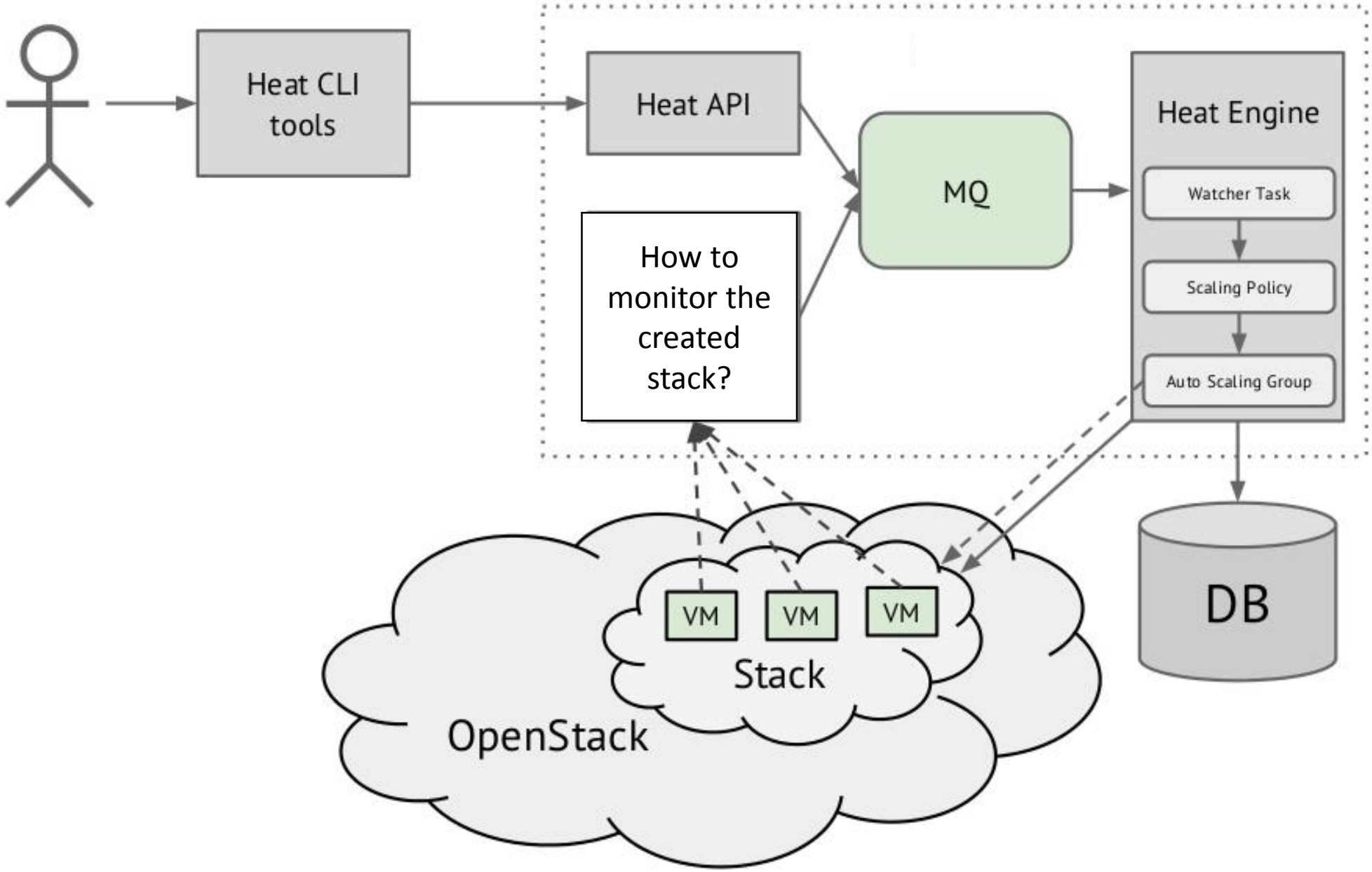
Heat Architecture

- Core part – Does all the orchestration work
- Layer on which resource integration is implemented
- Contains abstractions to use Auto Scaling & HA



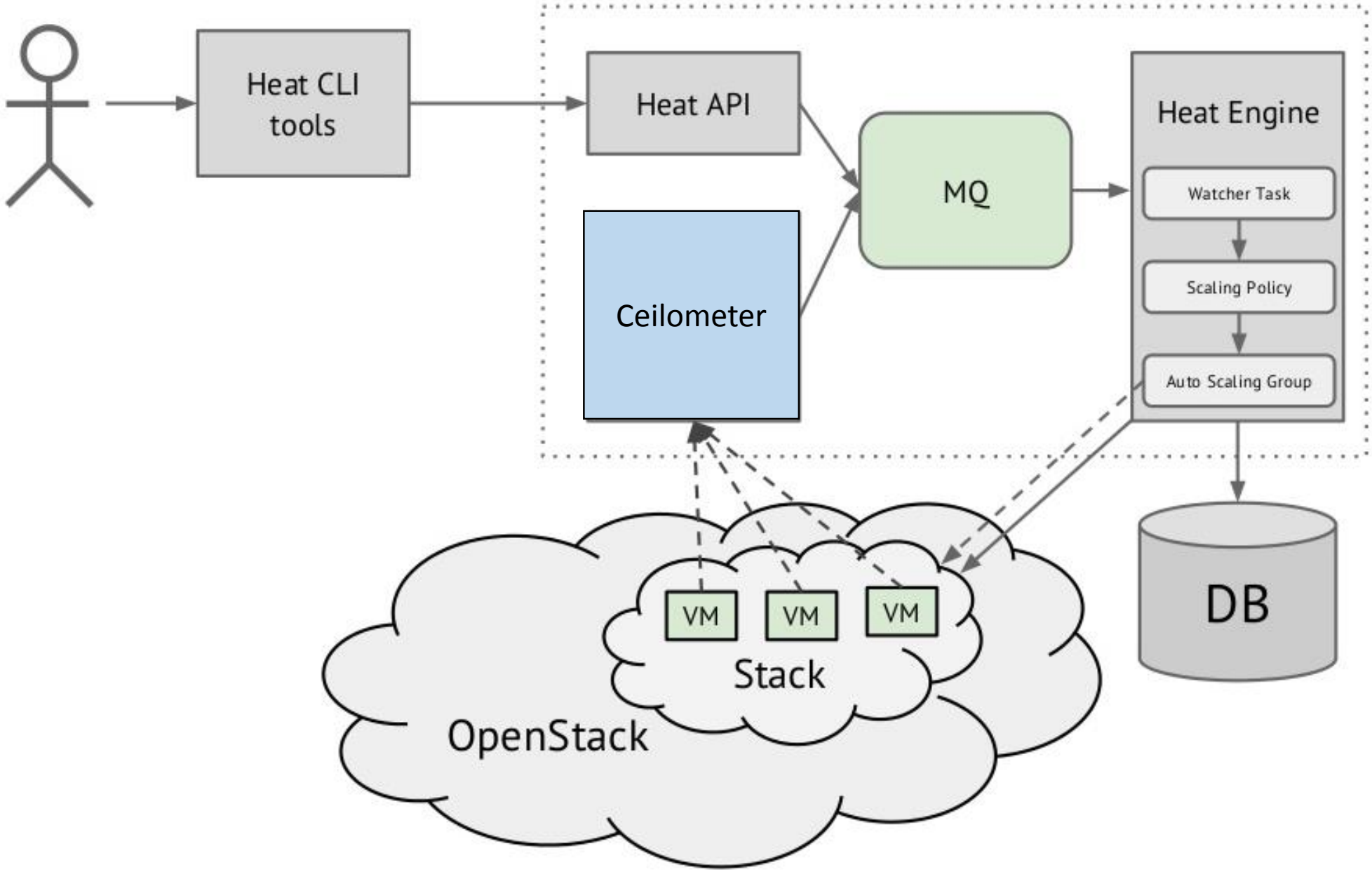
Heat Orchestration Workflow




Heat CLI tools
Heat API
MQ
Heat Engine
Watcher Task
Scaling Policy
Auto Scaling Group
How to monitor the created stack?
DB
VM
VM
VM
Stack
OpenStack

Ceilometer

- **Not a part of Heat** – A separate Openstack component
- Provides a framework for metering and monitoring in Openstack cloud
- Meters and Monitors all resources
- Generates Alarms to create more resources
- An alarm allows Ceilometer to POST to a URL when a metric matches certain values


Heat CLI tools
Heat API
MQ
Heat Engine
Watcher Task
Scaling Policy
Auto Scaling Group
Ceilometer
DB
VM
VM
VM
Stack
OpenStack

Creating Single Instance with HOT

```
heat_template_version: 2014-10-16

description: Simple template to deploy a single compute instance with a predefined image

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: TestIMG
      flavor: m2.small
      networks:
        -network: net04
```

Works fine but we want to write portable, reusable templates that do not hardcode information about your local environment

Templates: Parameters

```yaml
heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance using parameters passed by the user

parameters:
  image:
    type: string
    label: Image name or ID
    description: Image to be used for compute instance
    default: TestVM
  flavor:
    type: string
    label: Flavor
    description: Type of instance (flavor) to be used
    default: m1.small
    hidden: true
  private_network:
    type: string
    label: Private network name or ID
    description: Network to attach instance to.
    default: net04

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks:
        - network: { get_param: private_network }
```

To create a heat stack, parameters can be specified:

```
heat stack-create -f test2.yaml -P "image=TestIMG; flavor=m1.medium; private_network=net04" mystack
```

Adding a script to run at Instance Launch

```
user_data_format: RAW
 user_data: |
   #!/bin/sh
   while [ 1 ] ; do echo $((13**99)) 1>/dev/null 2>&1; done &
```

Environment Files

- An environment file is a YAML file with a parameters section containing values for parameters declared in your template:

```
parameters:
  image: fedora-20-x86-64_updated
```

- To create a heat stack in CLI using environment file:

```
heat stack-create -f my_stack.yaml -e my_env.yaml mystack
```

List User’s Stacks

```
[root@node-2 ~]# heat stack-list
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| b47b9e32-f7bb-42a2-b9c8-0608007b1163 | test       | CREATE_COMPLETE | 2016-02-16T06:38:12Z |
| fe76ad33-bf8a-4212-81e1-c31269f81f00 | mystack    | CREATE_COMPLETE | 2016-02-18T06:54:33Z |
| c50e7d74-8e4a-45e8-ac67-ca1ed89fbf91 | es         | UPDATE_COMPLETE | 2016-02-22T04:47:23Z |
+--------------------------------------+------------+-----------------+----------------------+
```

Templates: Outputs

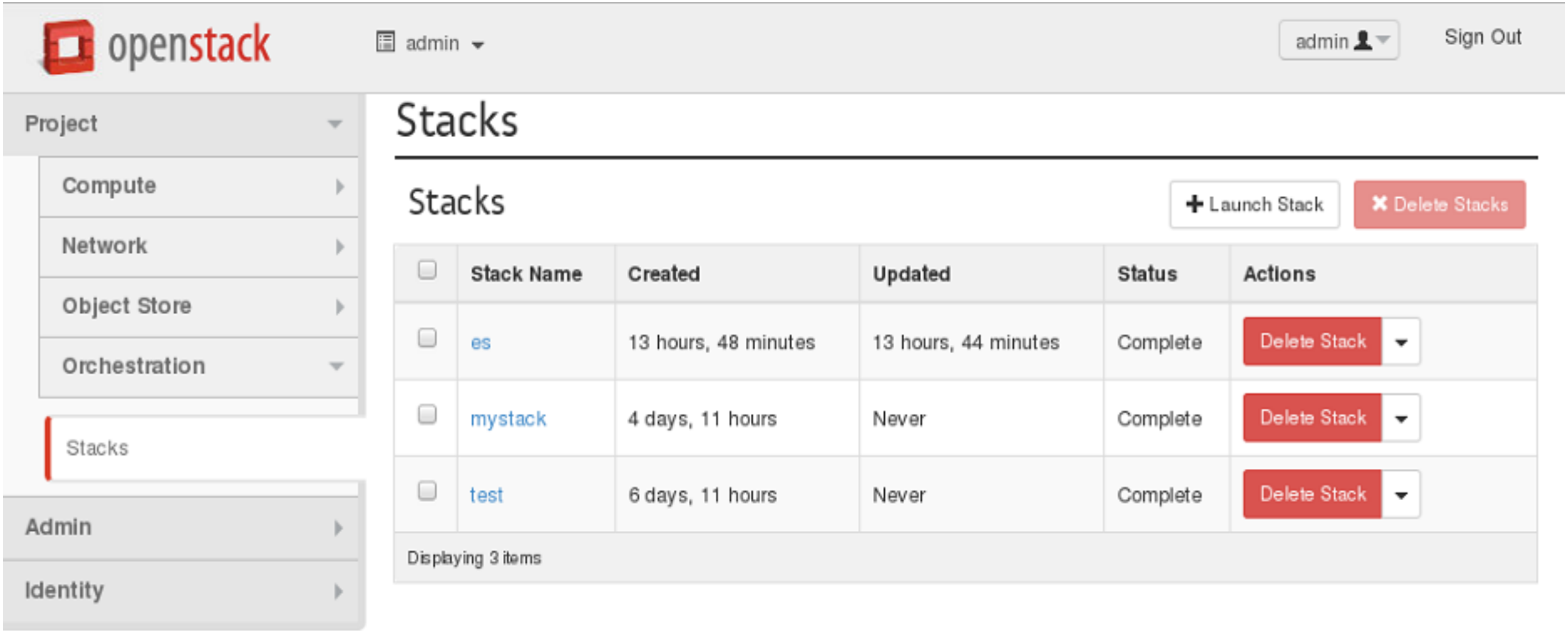
- Sometimes we want to extract information about a stack:

```
outputs:
  instance_name:
    description: Name of the instance
    value: { get_attr: [my_instance, name] }
  instance_ip:
    description: IP address of the instance
    value: { get_attr: [my_instance, first_address] }
```

- These outputs can be retrieved via **heat output-list** and **heat output-show** commands:

```
[root@node-2 ~]# heat output-list mystack
+---------------+-----------------------------+
| output_key    | description                 |
+---------------+-----------------------------+
| instance_ip   | IP address of the instance  |
| instance_name | Name of the instance        |
+---------------+-----------------------------+
[root@node-2 ~]# heat output-show mystack instance_ip
"192.168.4.50"
```

Using Heat from Dashboard



Select Template

Template Source *

File

Template File

Browse... No file selected.

Environment Source

File

Environment File

Browse... No file selected.

Description:

Use one of the available template source options to specify the template to be used in creating this stack.

Cancel Next

Launch Stack

Launch Stack

Stack Name: *

Creation Timeout (minutes): *

60

Rollback On Failure:

Password for user "lars": *

db_user:

wordpress

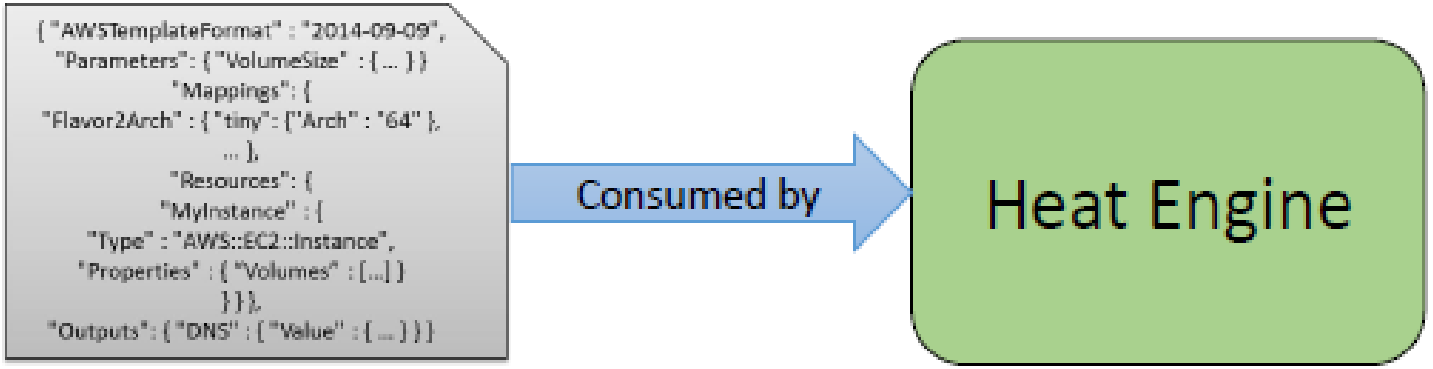
mysql_root_password:

secret

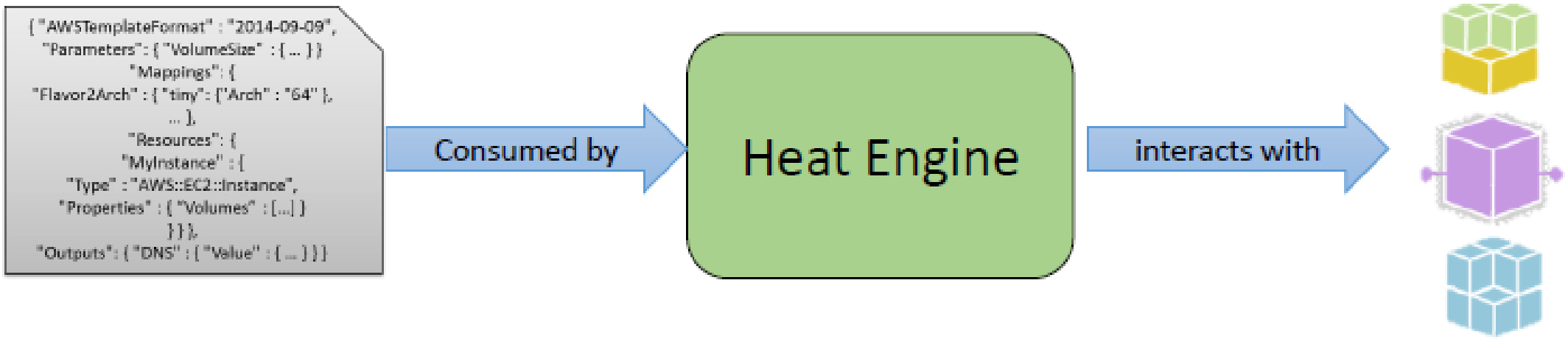
Description:

Create a new stack with the provided values.

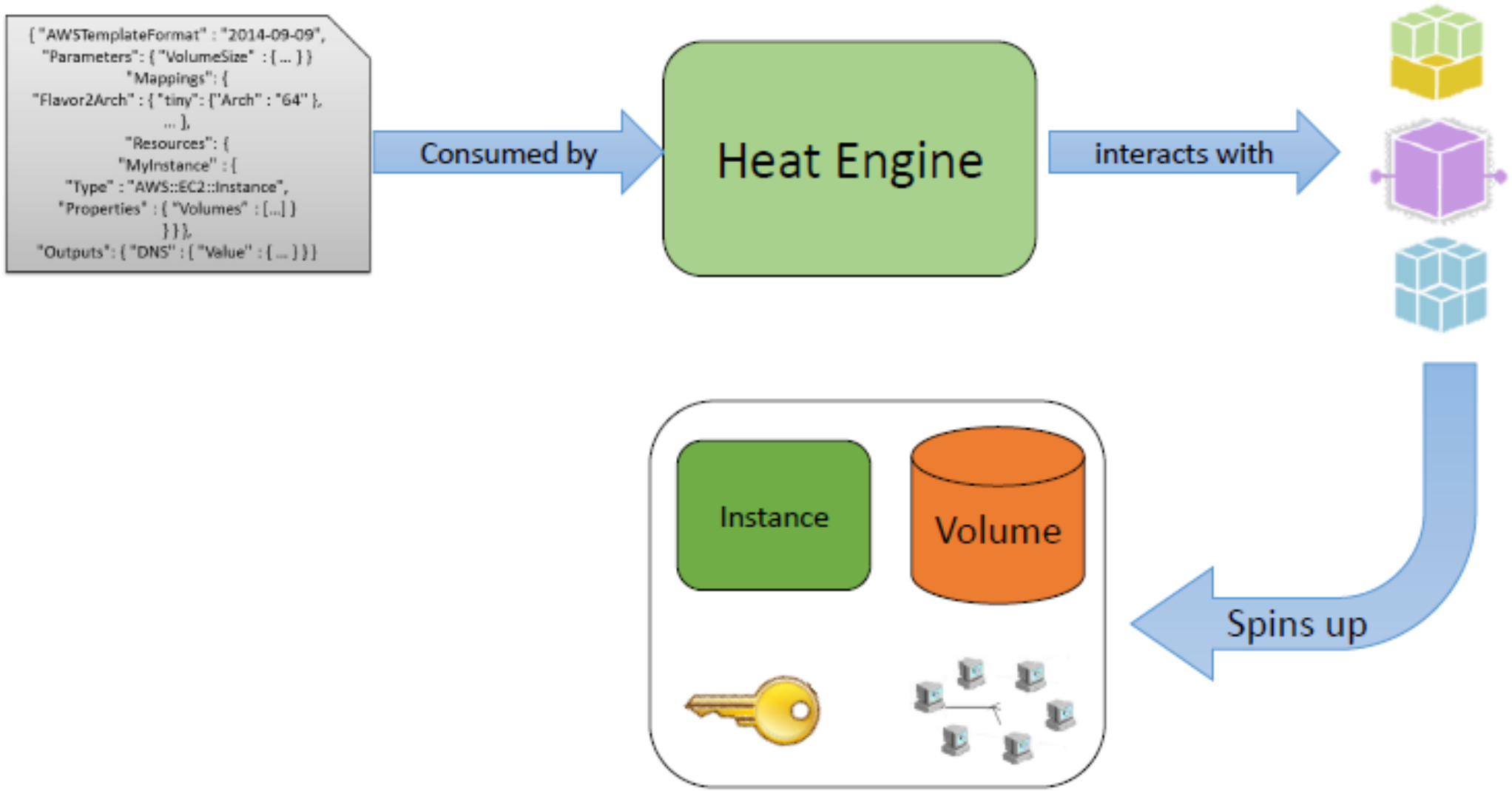
Heat Use Case: Stack Deployment



Heat Use Case: Stack Deployment



Heat Use Case: Stack Deployment



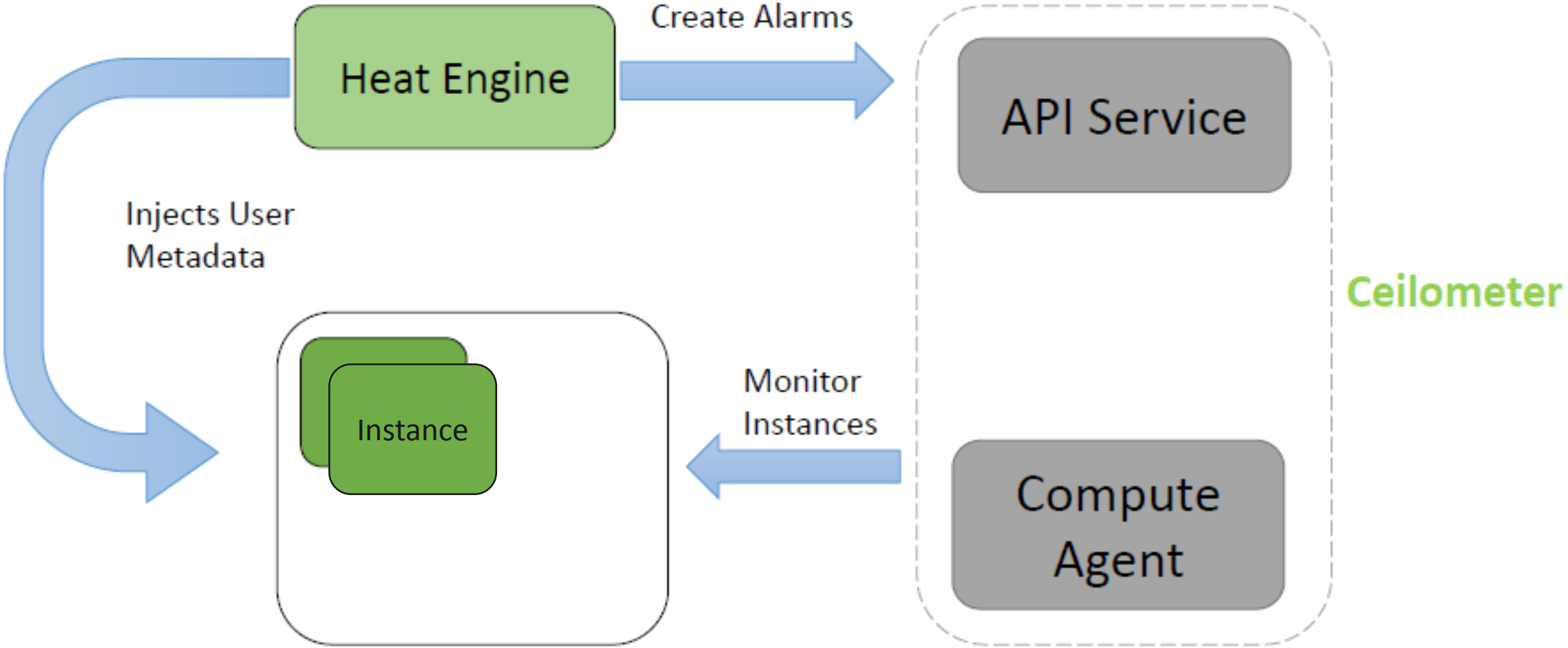
Heat Use Case: Auto Scaling

- There are number of resources associated with auto scaling:
  - AUTO SCALING GROUP- a group that can scale an arbitrary set of heat resources.
  - SCALING POLICY- defines an action that Heat can take on an Auto Scaling Group. Does nothing by itself; creates an endpoint for external triggers.
  - CEILOMETER ALARM – generate an alarm when a defined threshold is reached and report it to the Heat Engine in order to take action

Auto Scaling Group

```
heat_template_version: 2014-10-16
description: A simple auto scaling group.
resources:
  group:
    type: OS::Heat::AutoScalingGroup
    properties:
      cooldown: 60
      desired_capacity: 2
      max_size: 6
      min_size: 1
      resource:
        type: OS::Nova::Server::Cirros
```

Heat Use Case: Auto Scaling

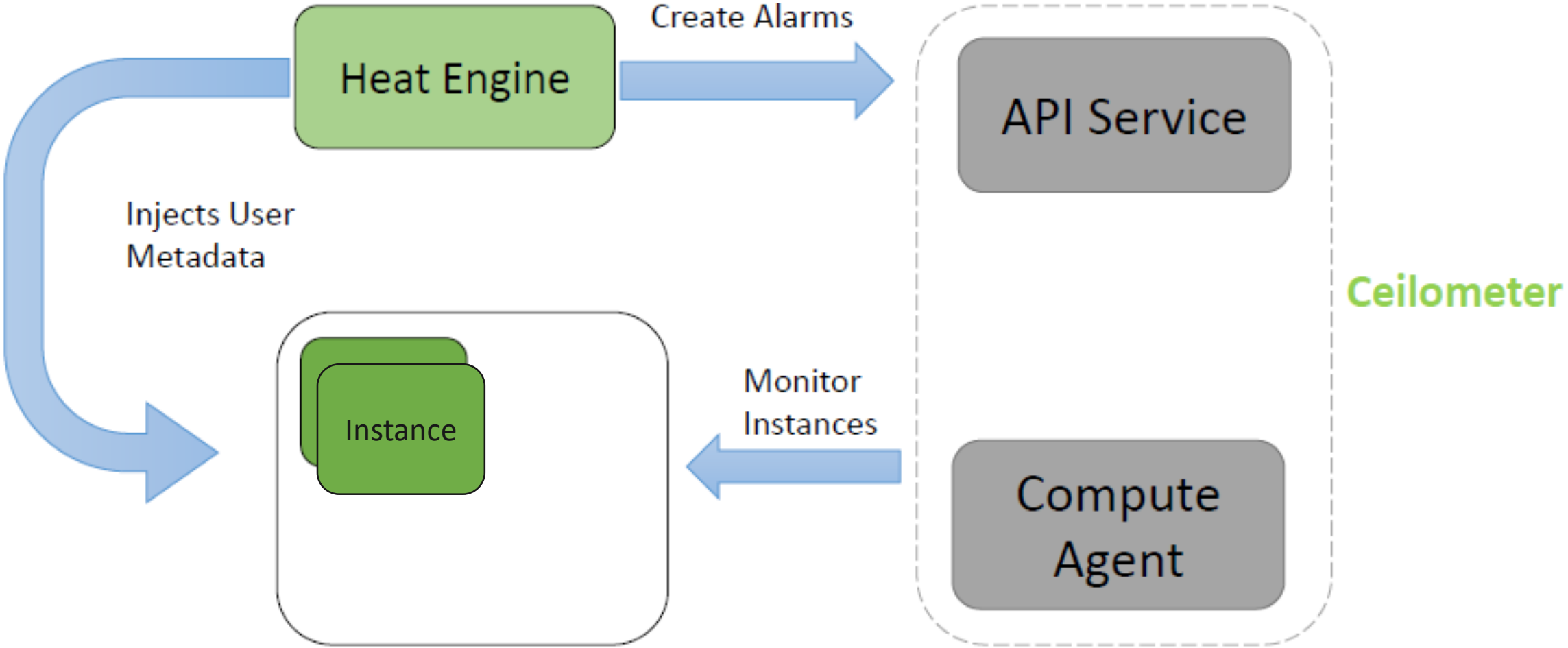


Scaling Policy

```
scaleup_policy:
    type: OS::Heat::ScalingPolicy
    properties:
      adjustment_type: change_in_capacity
      auto_scaling_group_id: { get_resource: group }
      cooldown: 60
      scaling_adjustment: 1
```

This template does nothing by itself unless an alarm is triggered

Heat Use Case: Auto Scaling

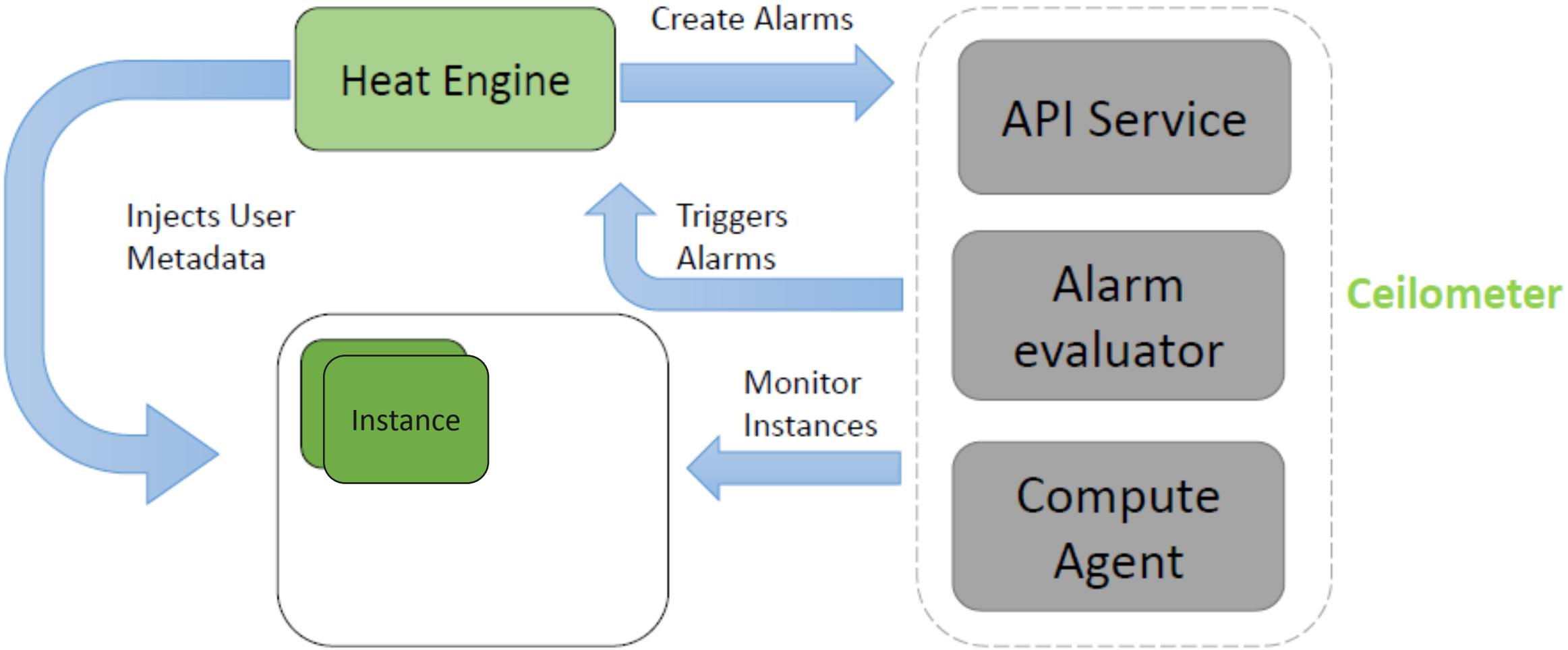


Ceilometer Alarm

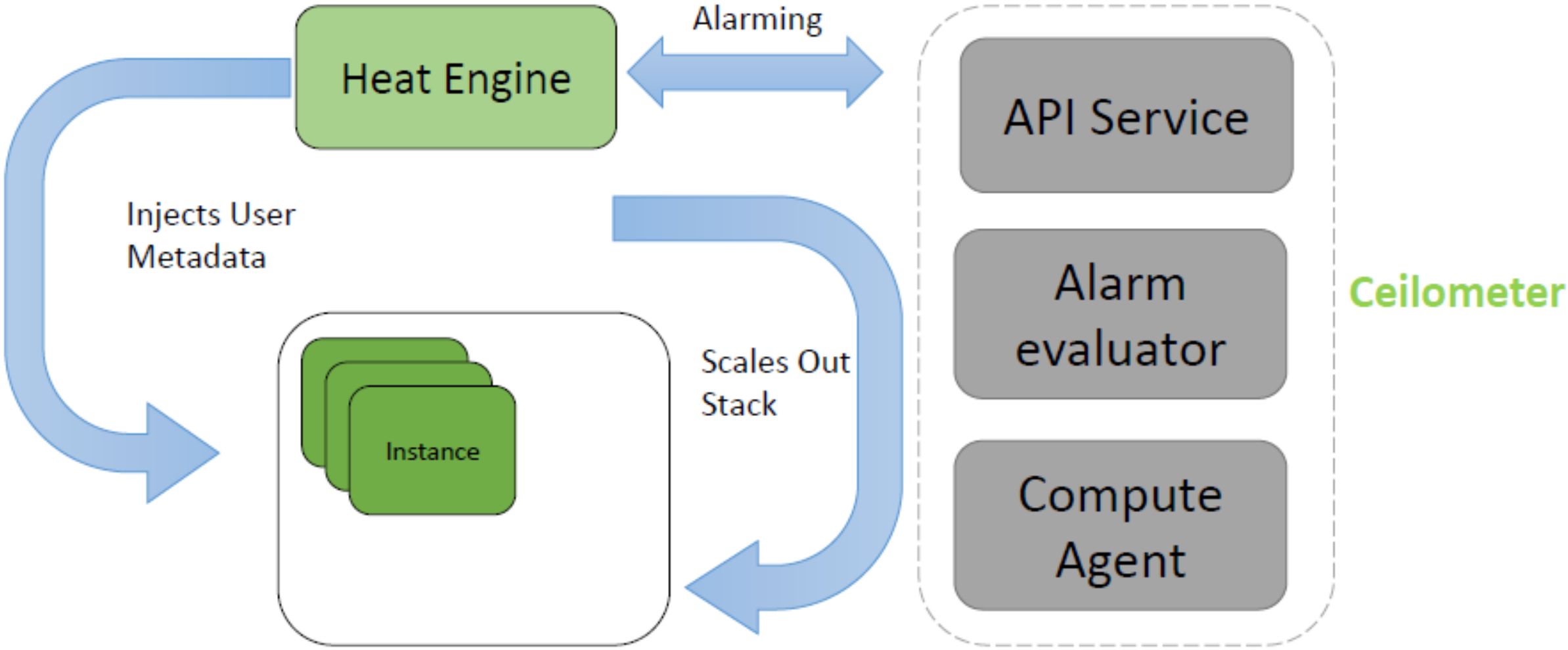
```
cpu_alarm_high:
  type: OS::Ceilometer::Alarm
  properties:
    meter_name: cpu_util
    statistic: avg
    period: 60
    evaluation_periods: 1
    threshold: 50
    alarm_actions:
      - {get_attr: [scaleup_policy, alarm_url]}
    comparison_operator: gt
```

This resource will notify the scaling policy resource. The scaling policy resource will increase the number of the resources defined in the scaling group.

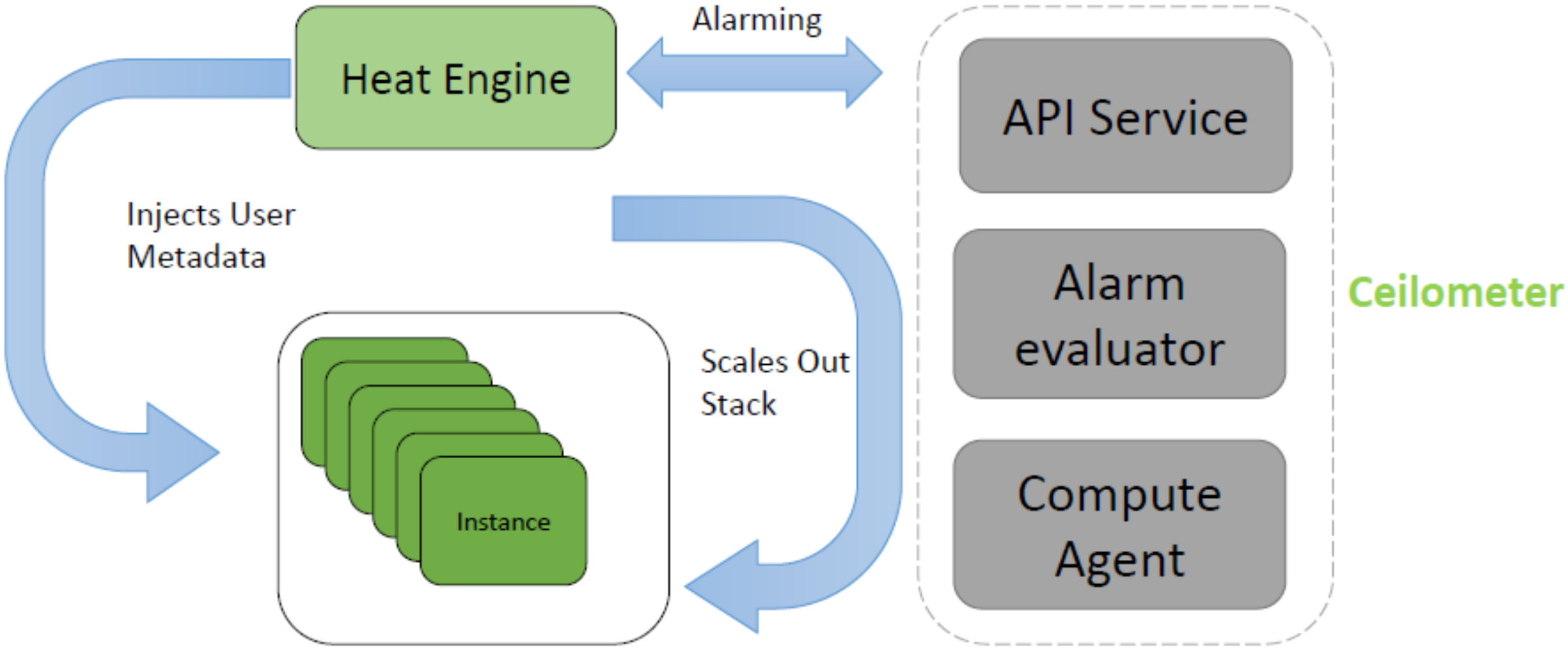
Heat Use Case: Auto Scaling



Heat Use Case: Auto Scaling



Heat Use Case: Auto Scaling



Useful Links

- http://docs.openstack.org/developer/heat/
- https://wiki.openstack.org/wiki/Heat
- https://github.com/openstack/heat
- https://github.com/openstack/heat-templates

Thank you!

Now you are ready to orchestrate your application in Openstack ☺

For questions/queries related to Heat or Orchestration in general, please shoot an email at: hassaan.ali@xflowresearch.com