ceilometer + heat = Alarming

Julien Danjou
jd__@Freenode // @juldanjou
julien@danjou.info

Nick Barcet
nijaba@Freenode // @nijaba
nick@enovance.com

Eoghan Glynn
eglynn@Freenode
eglynn@redhat.com

Speakers

Nick Barcet co-founded the Ceilometer project at the Folsom summit and led the project through incubation

Julien Danjou has been a core Ceilometer contributor from the outset, taking over the PTL reins for Havana

Eoghan Glynn drove the addition of the Alarming feature to Ceilometer over the Havana cycle

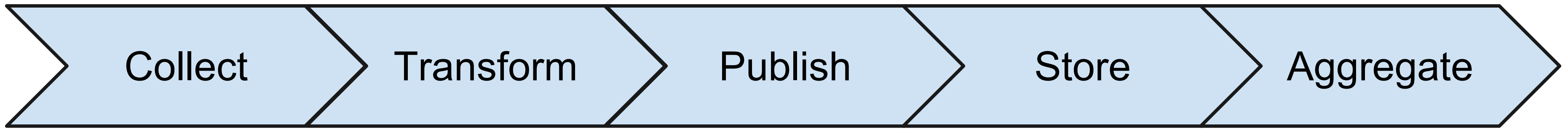
Two seemingly disjoint projects intersect

- **Heat** is a template-driven orchestration engine
    - automates complex deployments via declarative configuration

- **Ceilometer** is a metering infrastructure
    - collects data measuring resource usage and performance

- Appear on the surface to have minimal commonality ...

Ceilometer Workflow



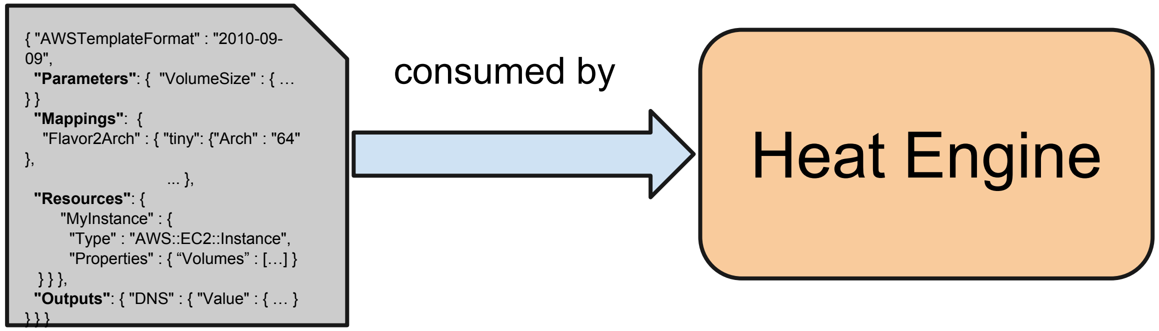
- Collect from OpenStack components
- Transform metering data if necessary
- Publish meters to any destination (including Ceilometer itself)
- Store received meters
- Aggregate samples via a REST API

Heat Workflow

`my_stack.template`

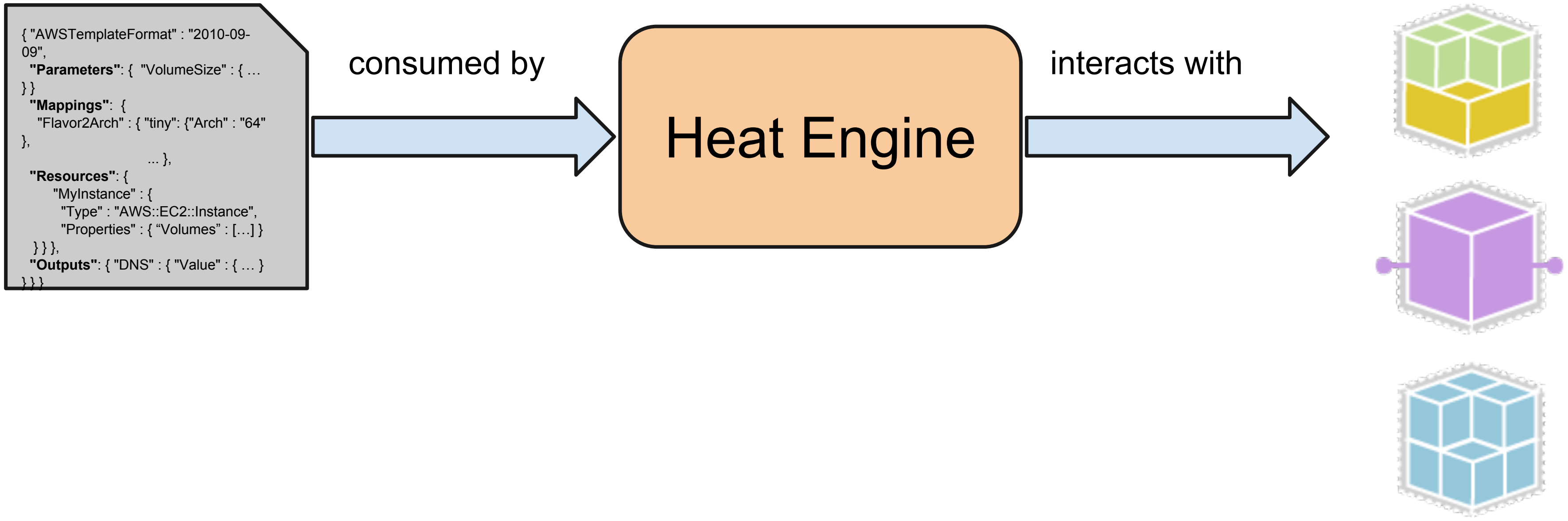
```
{ "AWSTemplateFormat" : "2010-09-09",
  "Parameters": {  "VolumeSize" : { … } }
  "Mappings":  {
   "Flavor2Arch" : { "tiny": {"Arch" : "64" },
                              ... },
  "Resources": {
       "MyInstance" : {
         "Type" : "AWS::EC2::Instance",
         "Properties" : { “Volumes” : […] }
   } } },
  "Outputs": { "DNS" : { "Value" : { … } } } }
```

Heat Workflow



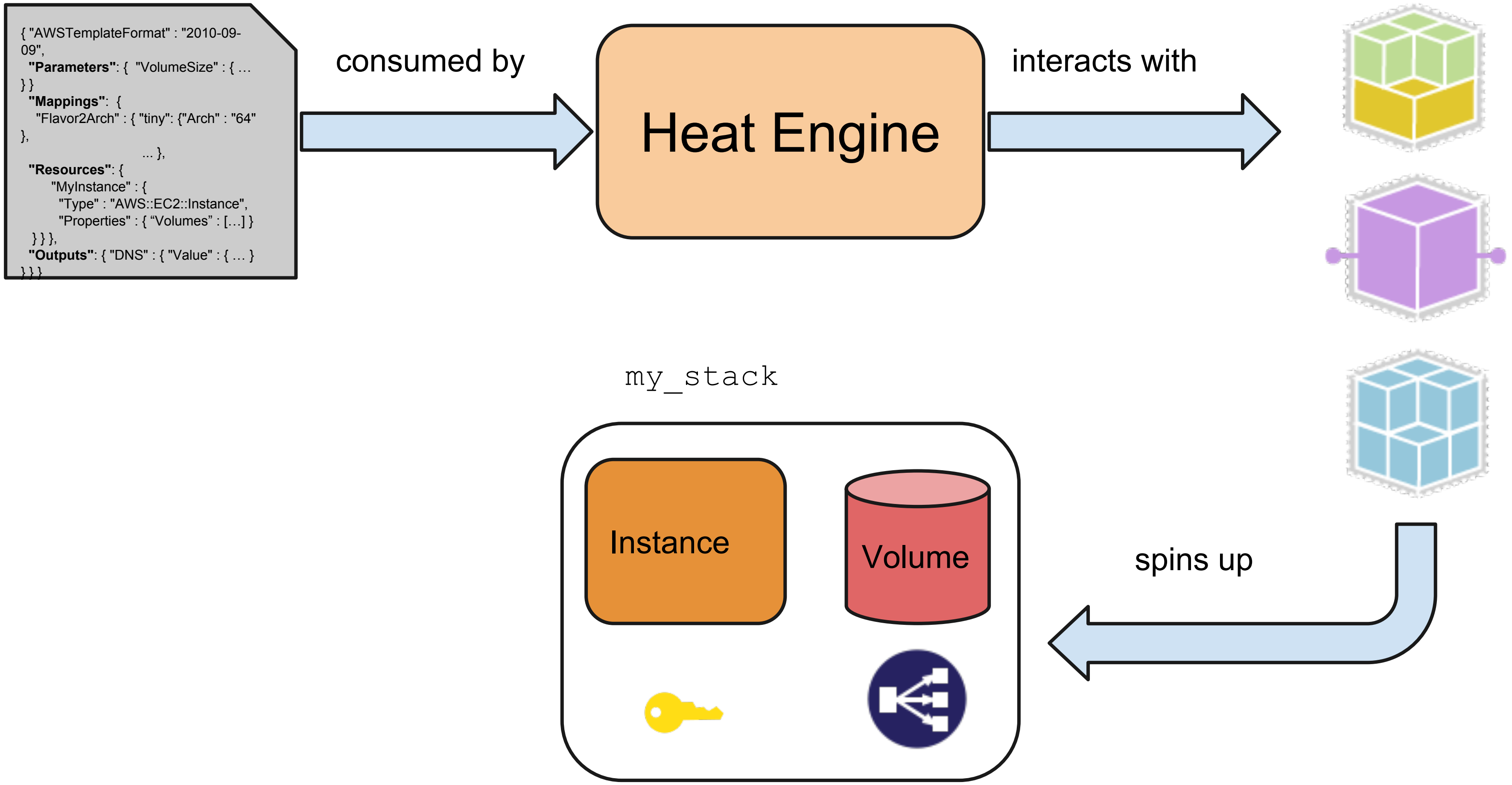
```
{ "AWSTemplateFormat" : "2010-09-
09",
  "Parameters": {  "VolumeSize" : { …
} }
  "Mappings":  {
    "Flavor2Arch" : { "tiny": {"Arch" : "64"
},
                                   ... },
  "Resources": {
        "MyInstance" : {
          "Type" : "AWS::EC2::Instance",
          "Properties" : { “Volumes” : […] }
   } } },
  "Outputs": { "DNS" : { "Value" : { … }
} } }
```

Heat Workflow



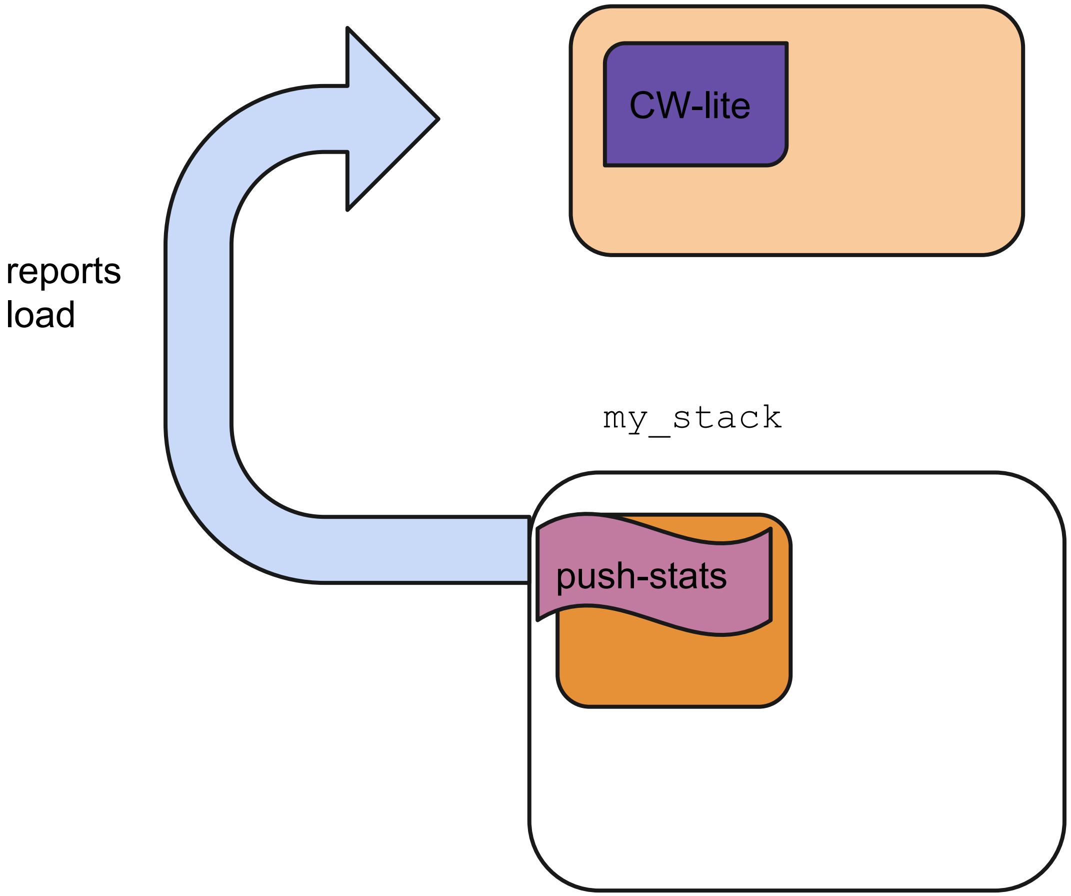
```
{ "AWSTemplateFormat" : "2010-09-
09",
  "Parameters": {  "VolumeSize" : { …
} }
  "Mappings":  {
    "Flavor2Arch" : { "tiny": {"Arch" : "64"
},
                              ... },
  "Resources": {
        "MyInstance" : {
          "Type" : "AWS::EC2::Instance",
          "Properties" : { “Volumes” : […] }
   } } },
  "Outputs": { "DNS" : { "Value" : { … }
} } }
```

Heat Workflow

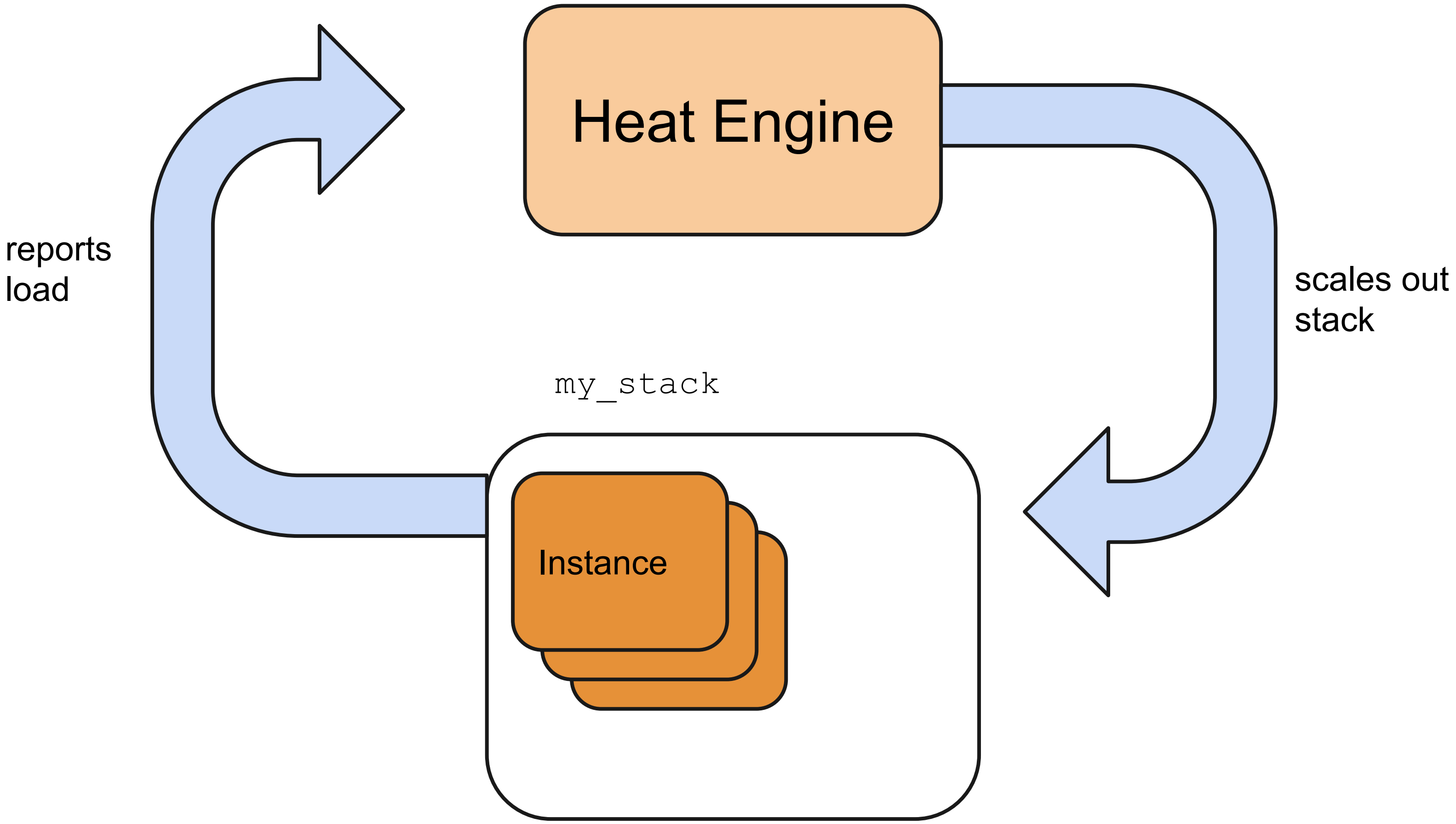


```
{ "AWSTemplateFormat" : "2010-09-
09",
  "Parameters": {  "VolumeSize" : { …
} }
  "Mappings":  {
    "Flavor2Arch" : { "tiny": {"Arch" : "64"
},
                              ... },
  "Resources": {
        "MyInstance" : {
          "Type" : "AWS::EC2::Instance",
          "Properties" : { “Volumes” : […] }
   } } },
  "Outputs": { "DNS" : { "Value" : { … }
} } }
```

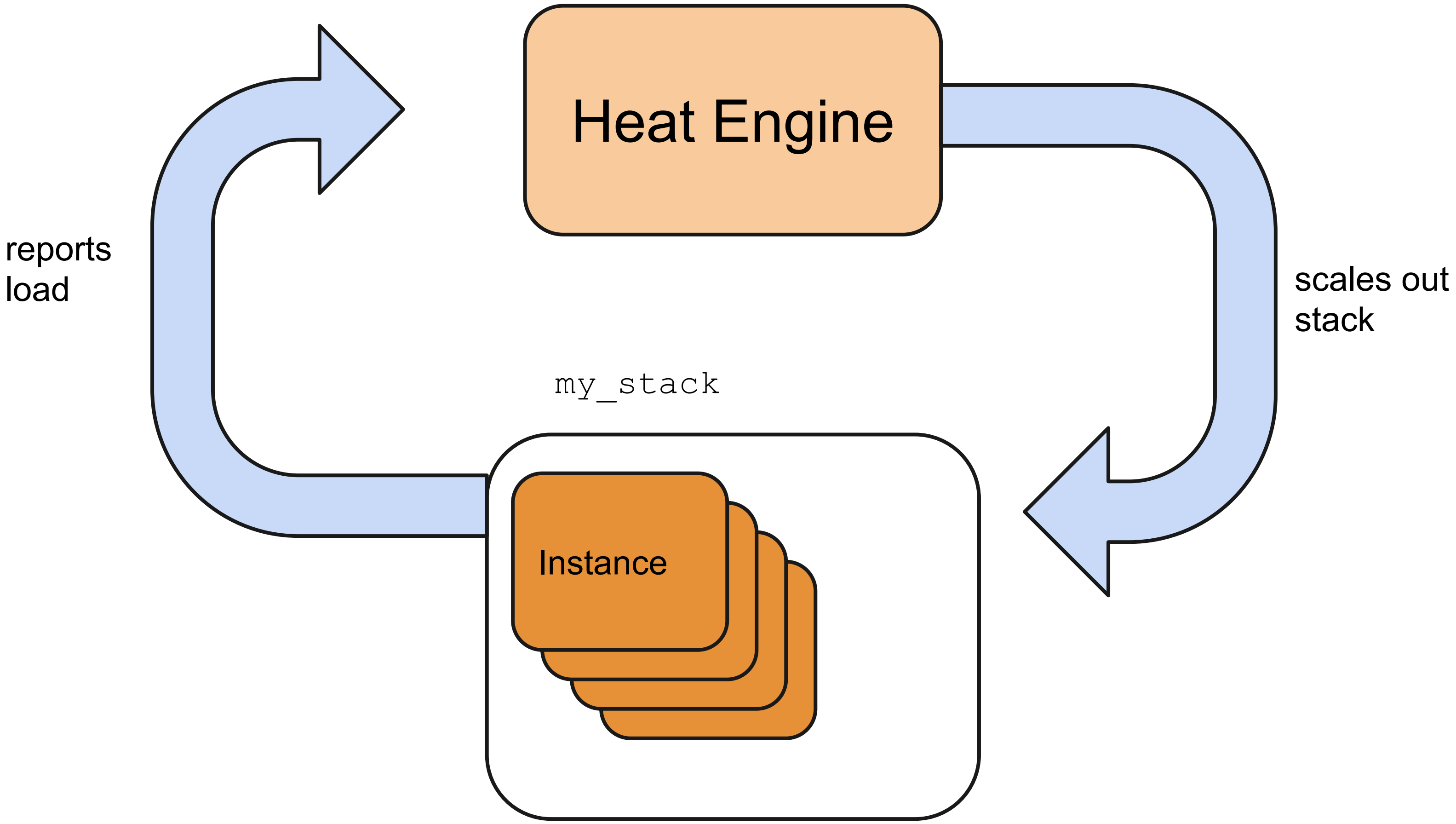
Heat Autoscaling v1.0



Heat Autoscaling v1.0



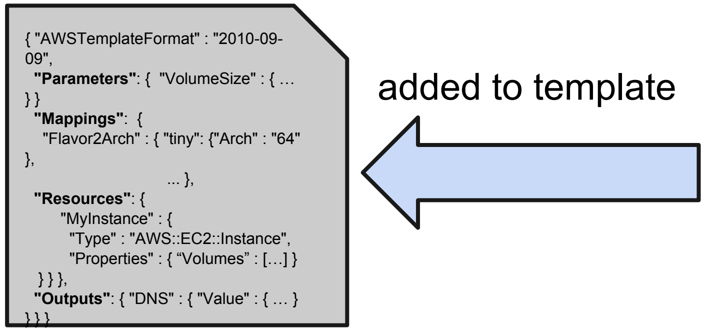
Heat Autoscaling v1.0



Ceilometer to the rescue!

- compute agent already collects most relevant stats from *outside* the instance
- API service exposes aggregation over the evaluation window
- define new API exposing alarm lifecycle
- provide new service to evaluate alarms against their defined rules
- additional service driving asynchronous notifications when alarms fire

How it all hangs together



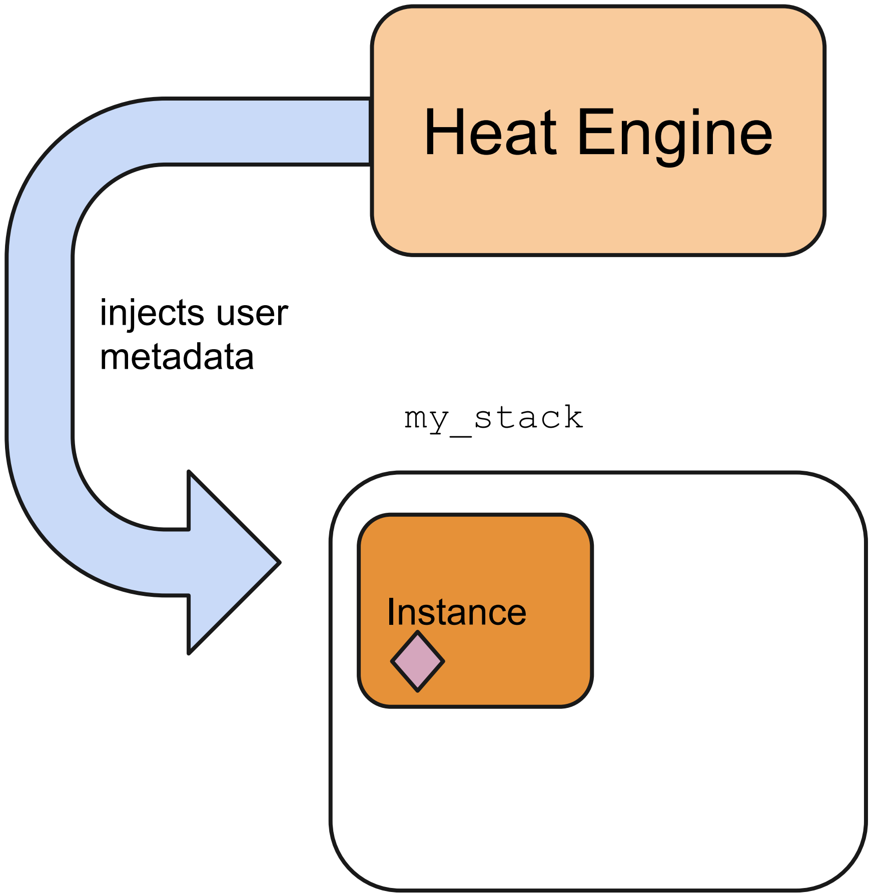
```
{ "AWSTemplateFormat" : "2010-09-
09",
  "Parameters": {  "VolumeSize" : { ...
} }
  "Mappings":  {
    "Flavor2Arch" : { "tiny": {"Arch" : "64"
},
                              ... },
  "Resources": {
       "MyInstance" : {
         "Type" : "AWS::EC2::Instance",
         "Properties" : { "Volumes" : [...] }
   } } },
  "Outputs": { "DNS" : { "Value" : { ... }
} } }
```

- alarms bounding busy/idleness of instances
- membership of autoscale group represented via user metadata
- alarm actions refer to scale up/down policies
- action URLs are pre-signed
- policies define adjustment step size & cooldown period

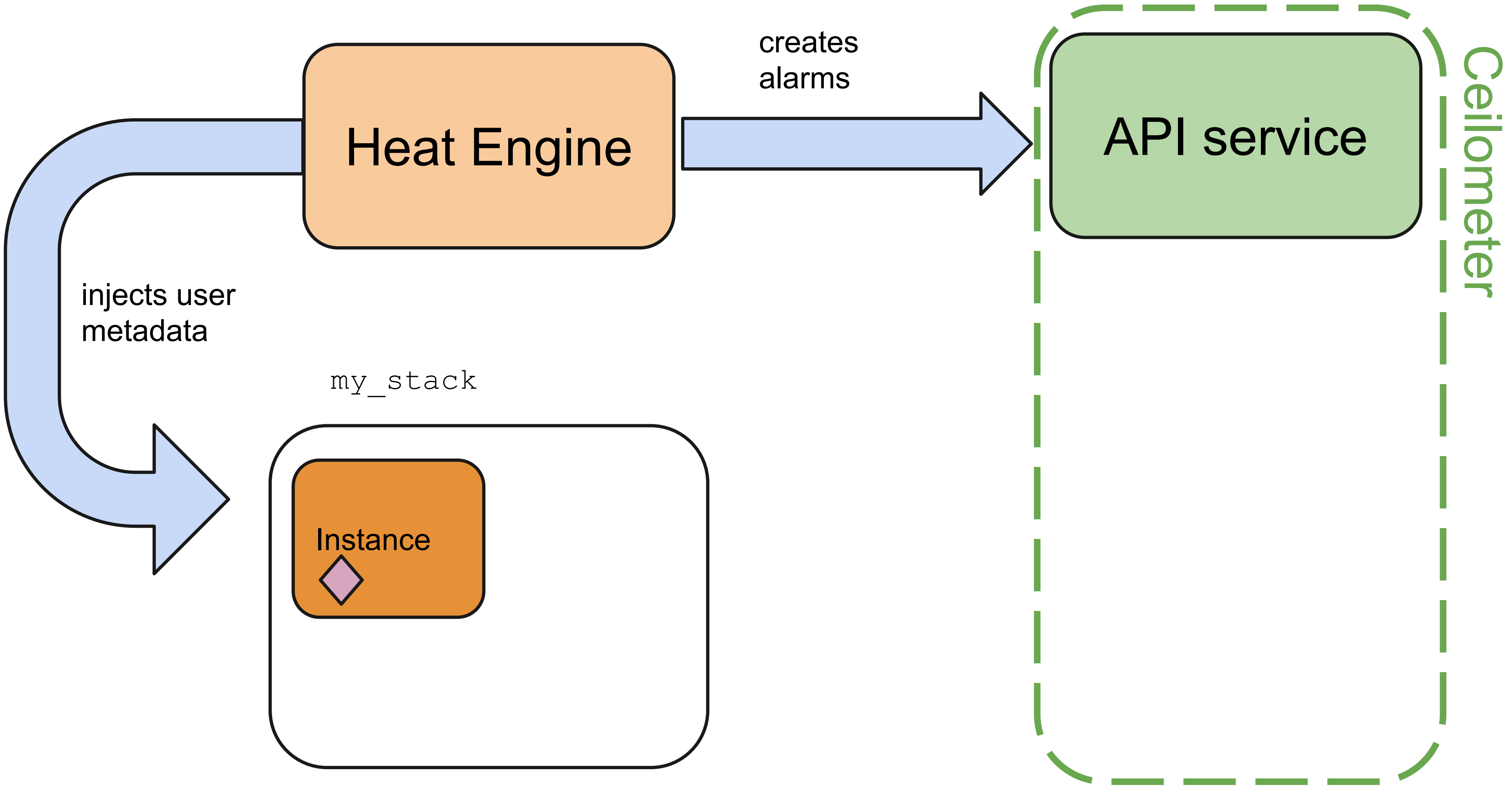
How it all hangs together

```
"CPUAlarmHigh": {
  "Type": "OS::Metering::Alarm",
  "Properties": {
    "meter_name": "cpu_util", threshold: "75"
    "evaluation_periods": "5", "period": "60",
    "statistic": "avg", "comparison_operator": "gt",
    "description": "Scale-up if CPU > 75% for 300s",
    "alarm_actions":[…"ScaleUpPolicy", "AlarmUrl"…],
    "matching_metadata": {
      "metadata.user_metadata.server_group":
      "MyWebServerGroup"
}}}
```

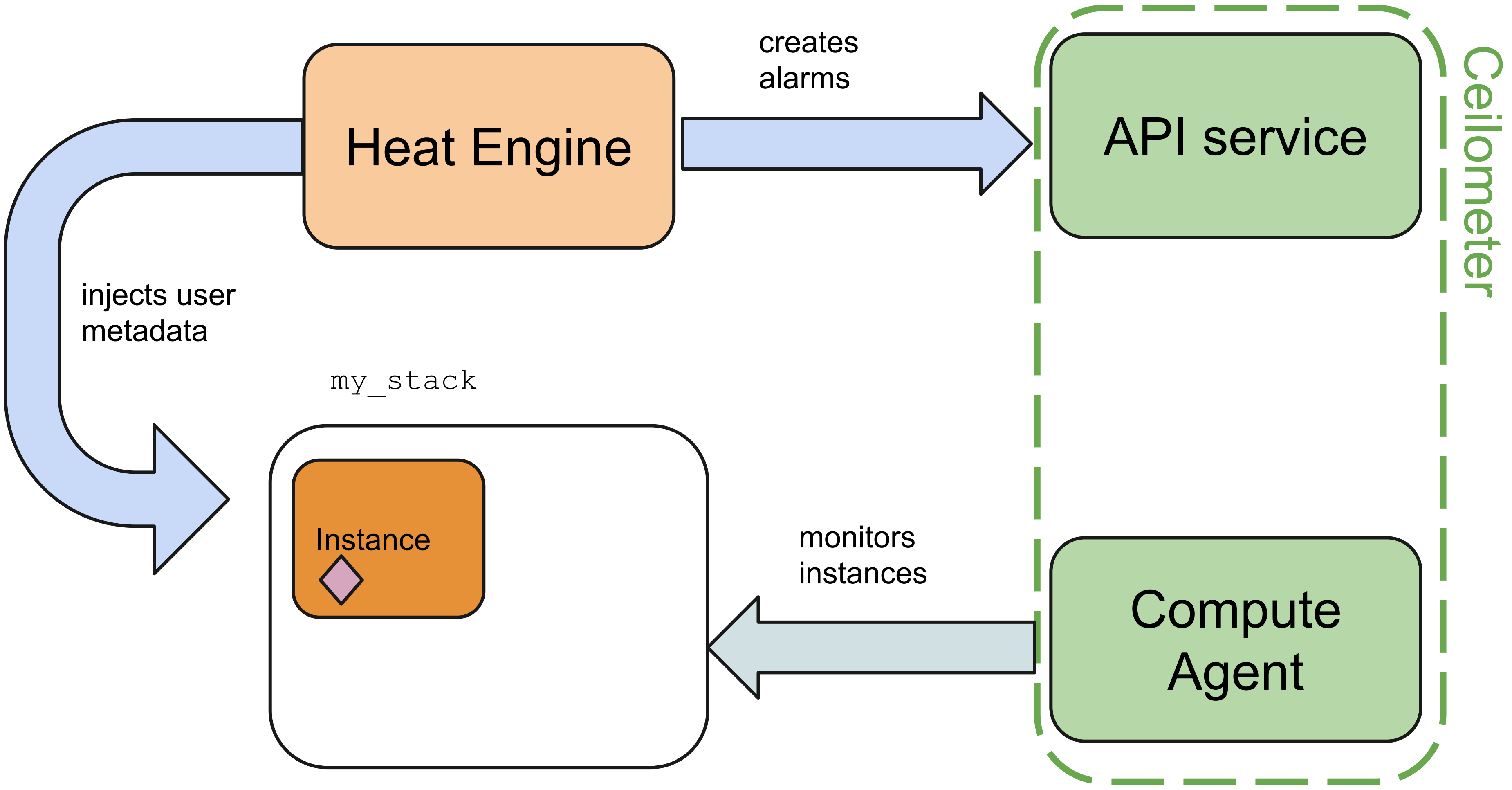
How it all hangs together



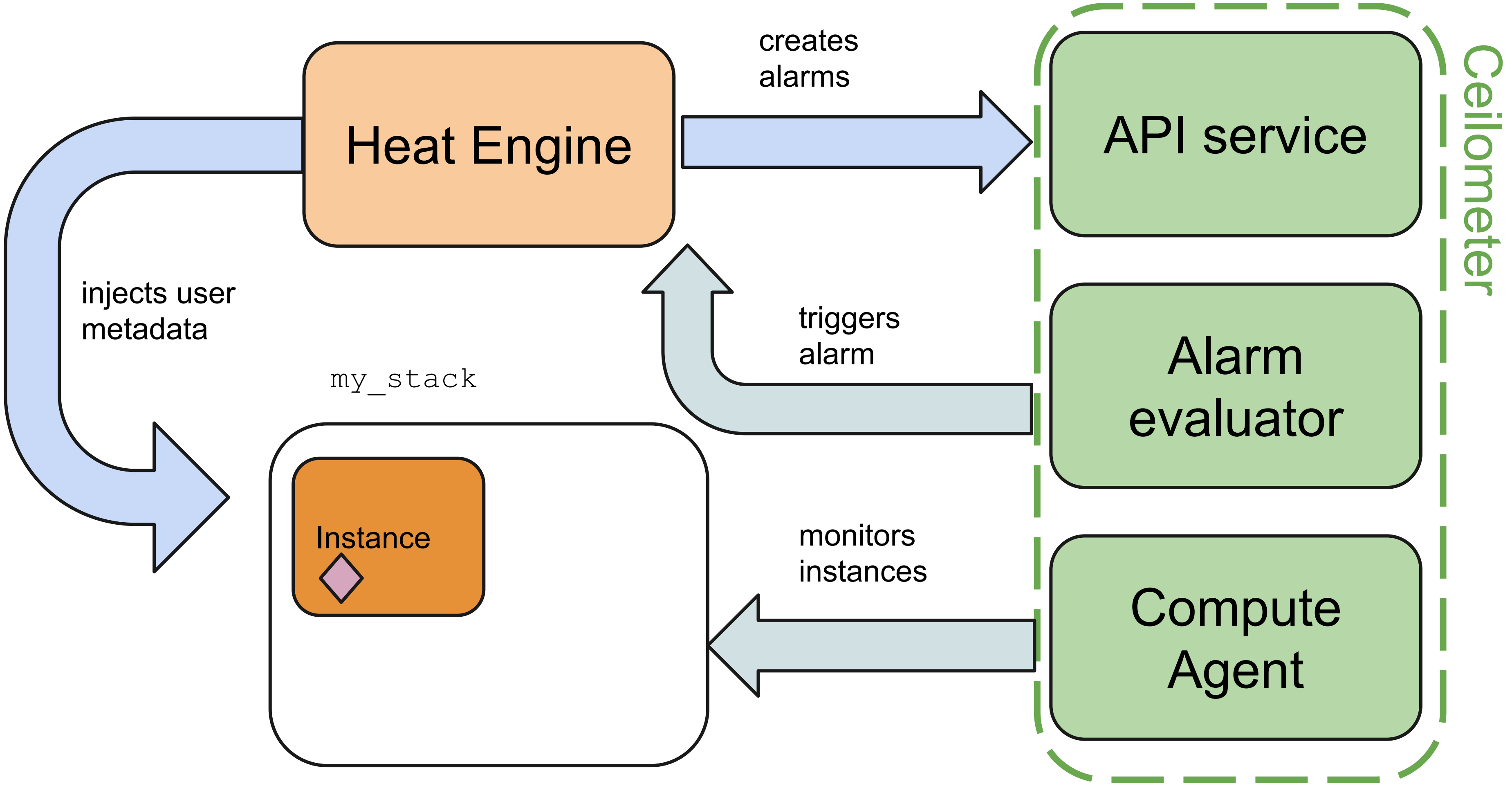
How it all hangs together



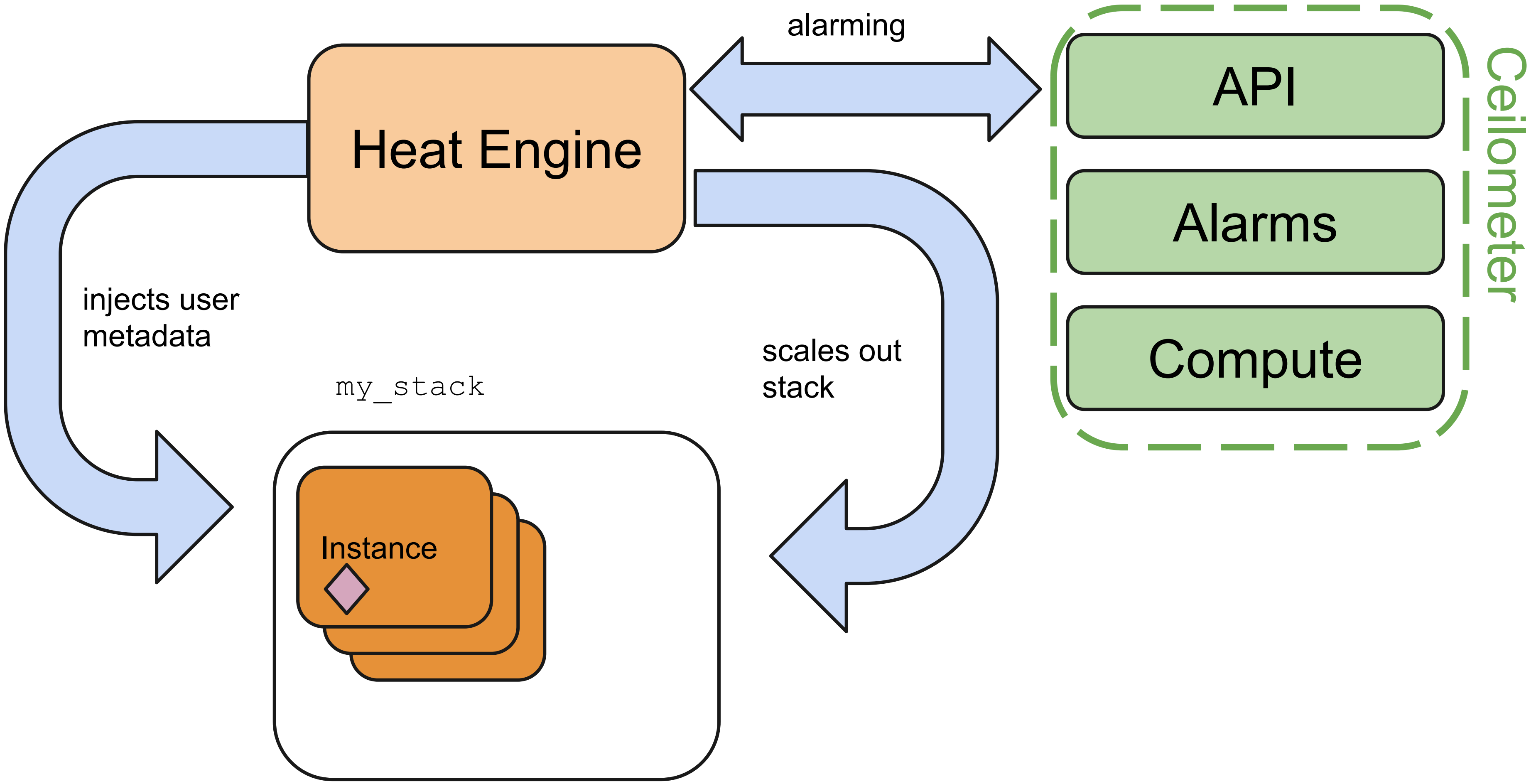
How it all hangs together



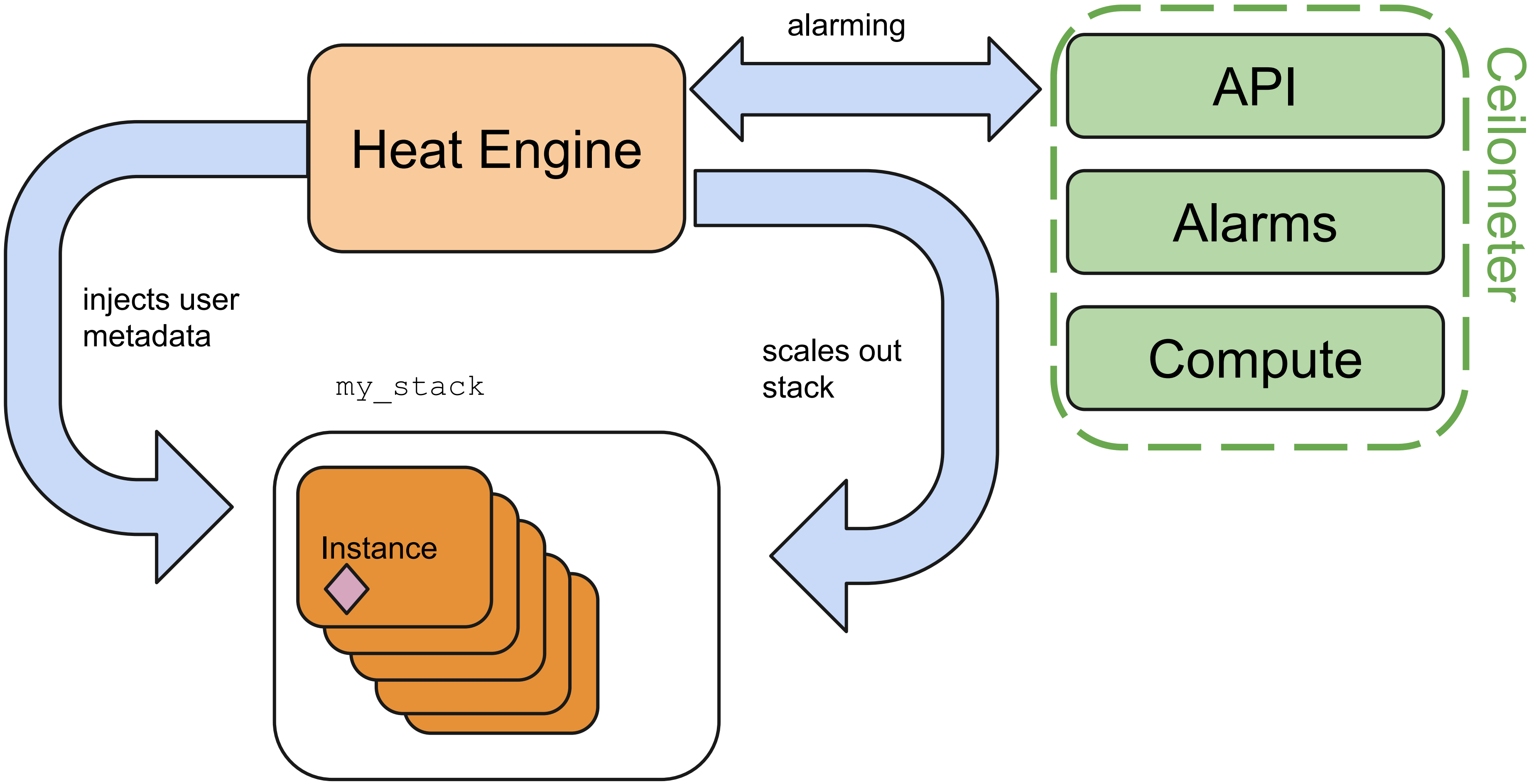
How it all hangs together



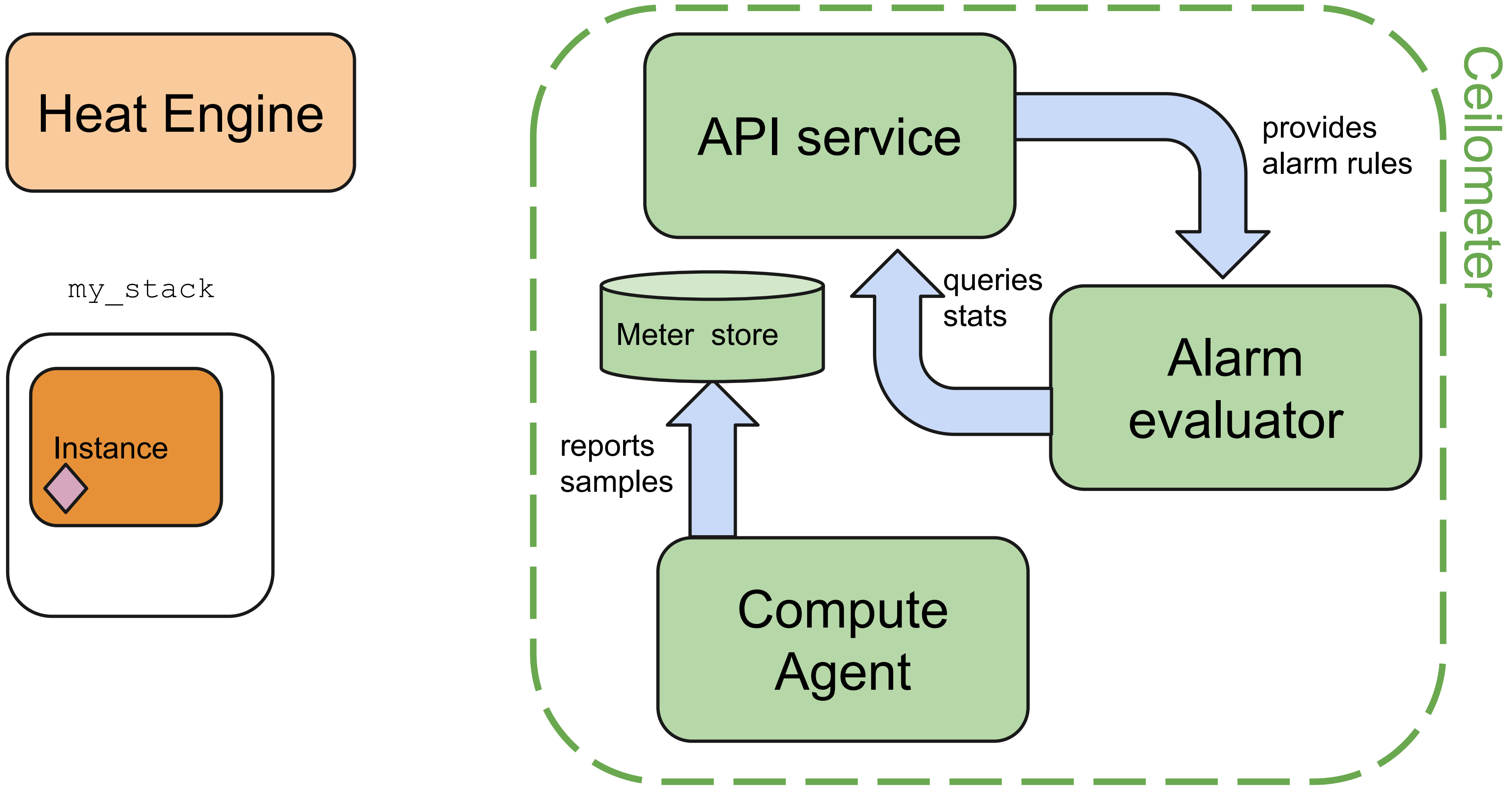
How it all hangs together



How it all hangs together



How it all hangs together



Lessons learned

Keys to successful intra-project interactions:

- buy-in from stakeholders on both sides
- early validation and proof-points
- protect consuming project from churn during the development cycle
- split deliverables into bite-sized separately consumable chunks

Future directions

- expand metering coverage to also capture:
  - memory utilization %
  - LBaaS statistics
  - network & disk I/O *rates*

- add combination alarm support to Heat templates
  - allow thresholds over multiple metrics to be modeled

- exclude low-quality datapoints
  - avoid scaling when only outliers have reported metrics

Future directions

- monitor baremetal via IPMI or SNMP
    - autoscale groups of hosts managed as Ironic instances
- constrain alarms for time-of-day or day-of-week
    - e.g. set the bar higher on weekends, lower on weekdays
- decouple autoscaling usage from Heat templates
- authenticate webhook calls with keystone trusts
    - avoid ec2-signer use without keystone EC2 tokens ext

Further questions?

- Chat on Freenode:
  - #openstack-metering
  - #heat
- Mail the dev list:
  - openstack-dev@lists.openstack.org
- Harangue us via Launchpad:
  - https://launchpad.net/ceilometer/+filebug