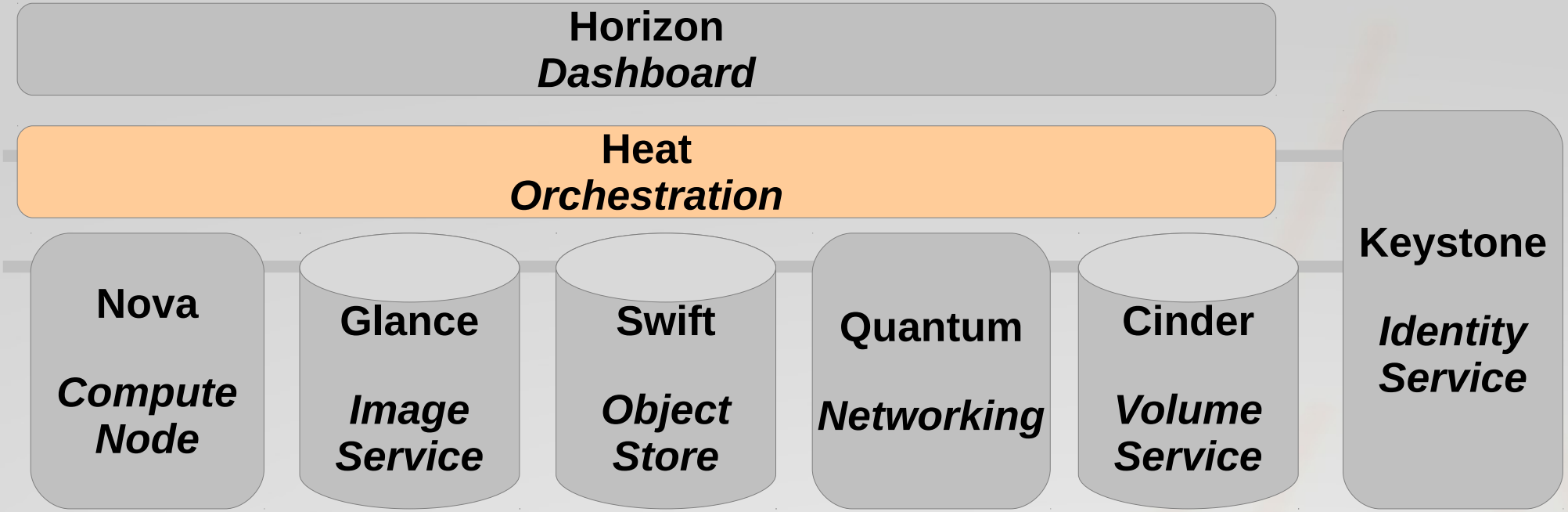
# Introduction to Heat API

## Orchestration for Openstack


Steven Hardy (shardy@redhat.com)

EMEA Openstack Day – 5th December 2012

# Heat Overview

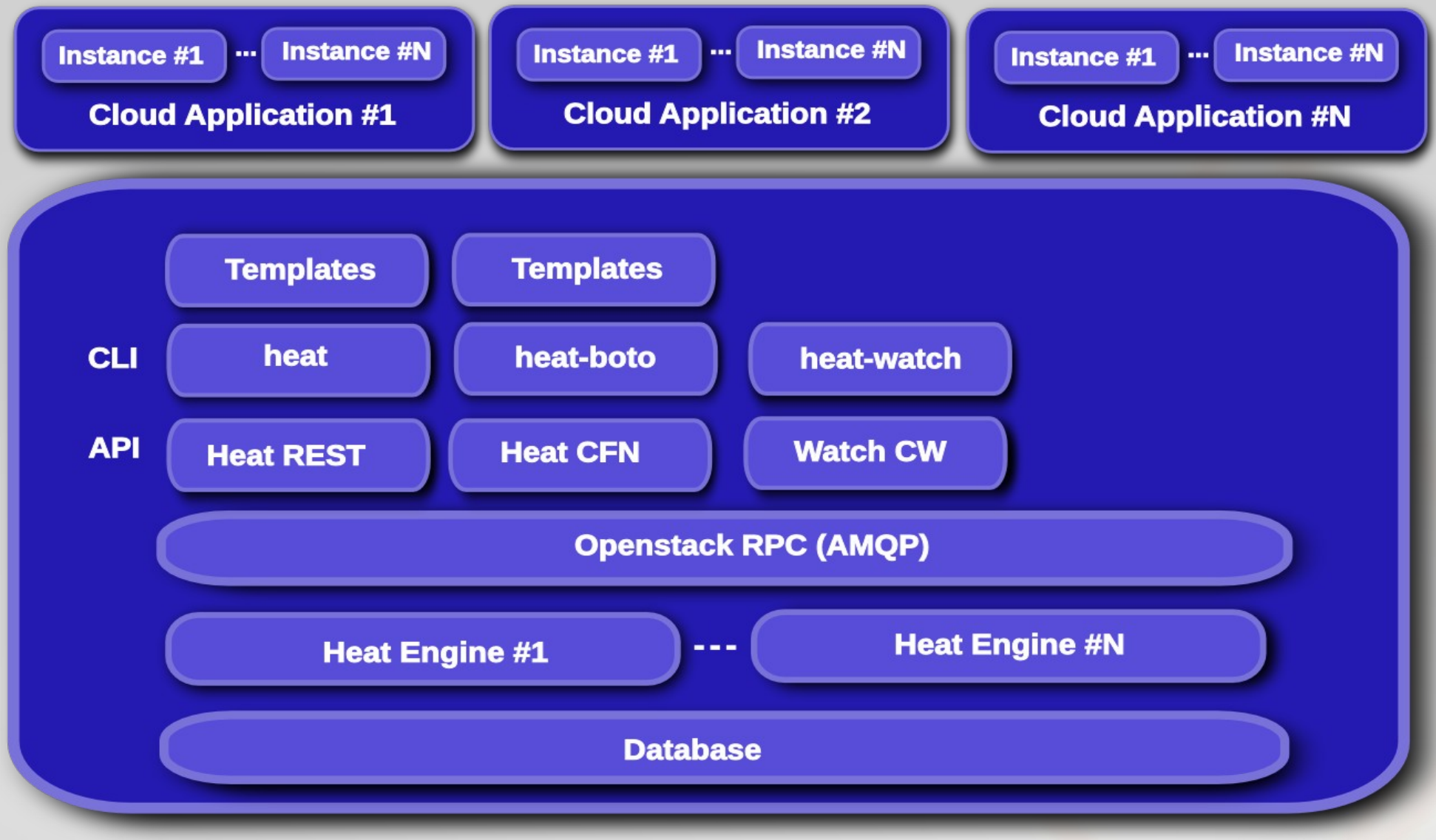



- Provides AWS Cloudformation and native ReST API
- Abstract configuration of services to single-template
- HA/Autoscaling/Monitoring features
- Openstack incubated project

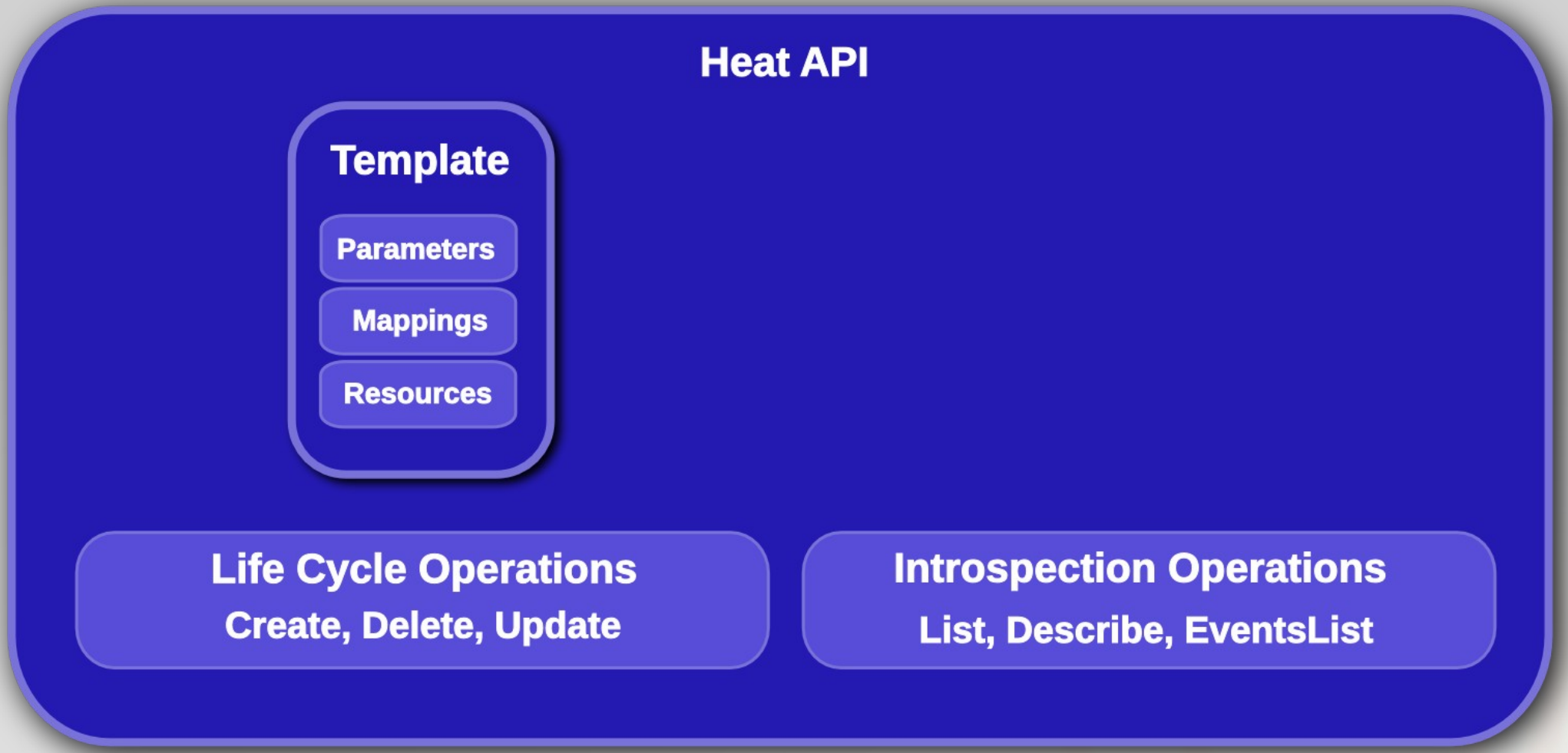
# Heat Overview

- Orchestration of Openstack deployments
- Integrates with all Openstack core projects
- Converts a JSON template into a cloud application
- Implements well known template and API (AWS Cloudformation, also YAML, ReST)
- Version your cloud applications like your software
- Repeatable deployments, fully automated

# Heat Overview



# Heat API


Heat API
Template
Parameters
Mappings
Resources
Life Cycle Operations
Create, Delete, Update
Introspection Operations
List, Describe, EventsList

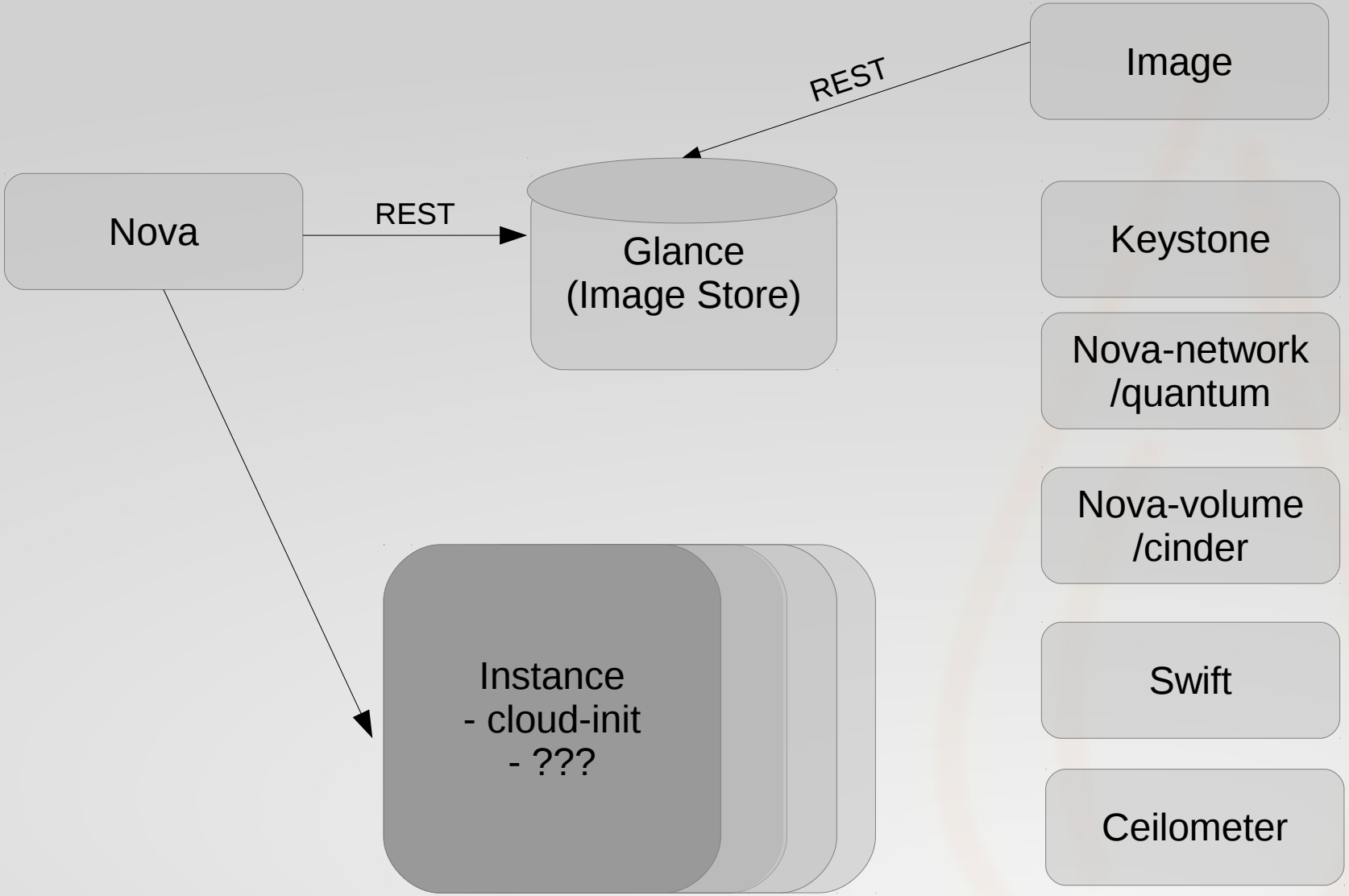
# Heat API : Key features

- Compatibility with AWS Cloudformation (template/API)
- Also superset of OS native resources & ReST API
- Fully open community project (come get involved!)
- Incubated project, aiming for $core
- Implements HA (service/instance/stack)
- Implements Instance Autoscaling
- Watch/Monitoring API (will move to Ceilometer)
- Pluggable resource implementations
- Watch this space!

# Nova Instance Lifecycle

- Base OS image stored in glance
- Deployment-time configuration/customization
- Cloud-init (nova user/metadata)
- Puppet/Chef/Scripts/???
- Potentially complex
- Everyone rolling-their-own solutions
- High maintenance overhead

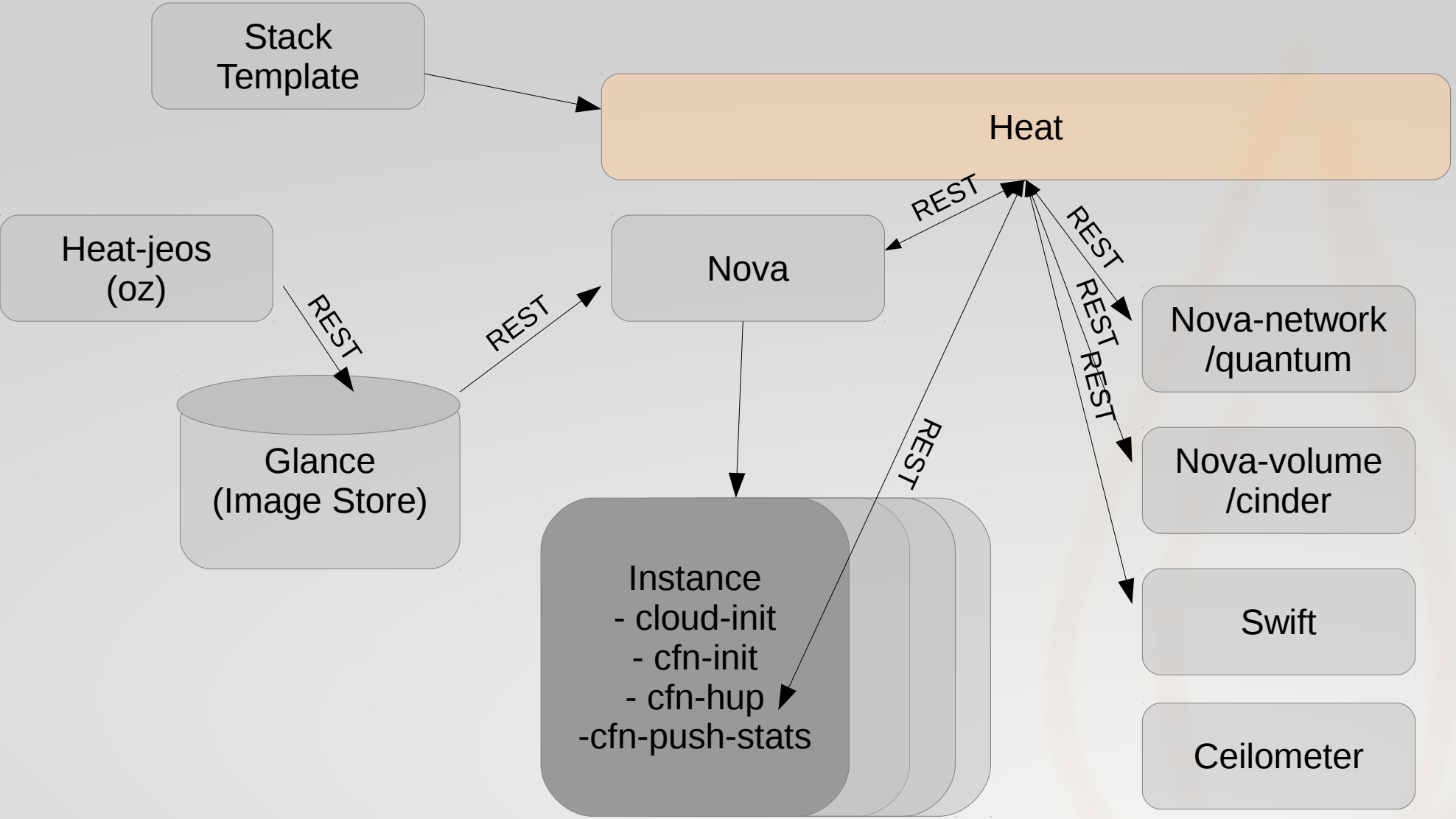
# Nova Instance Lifecycle



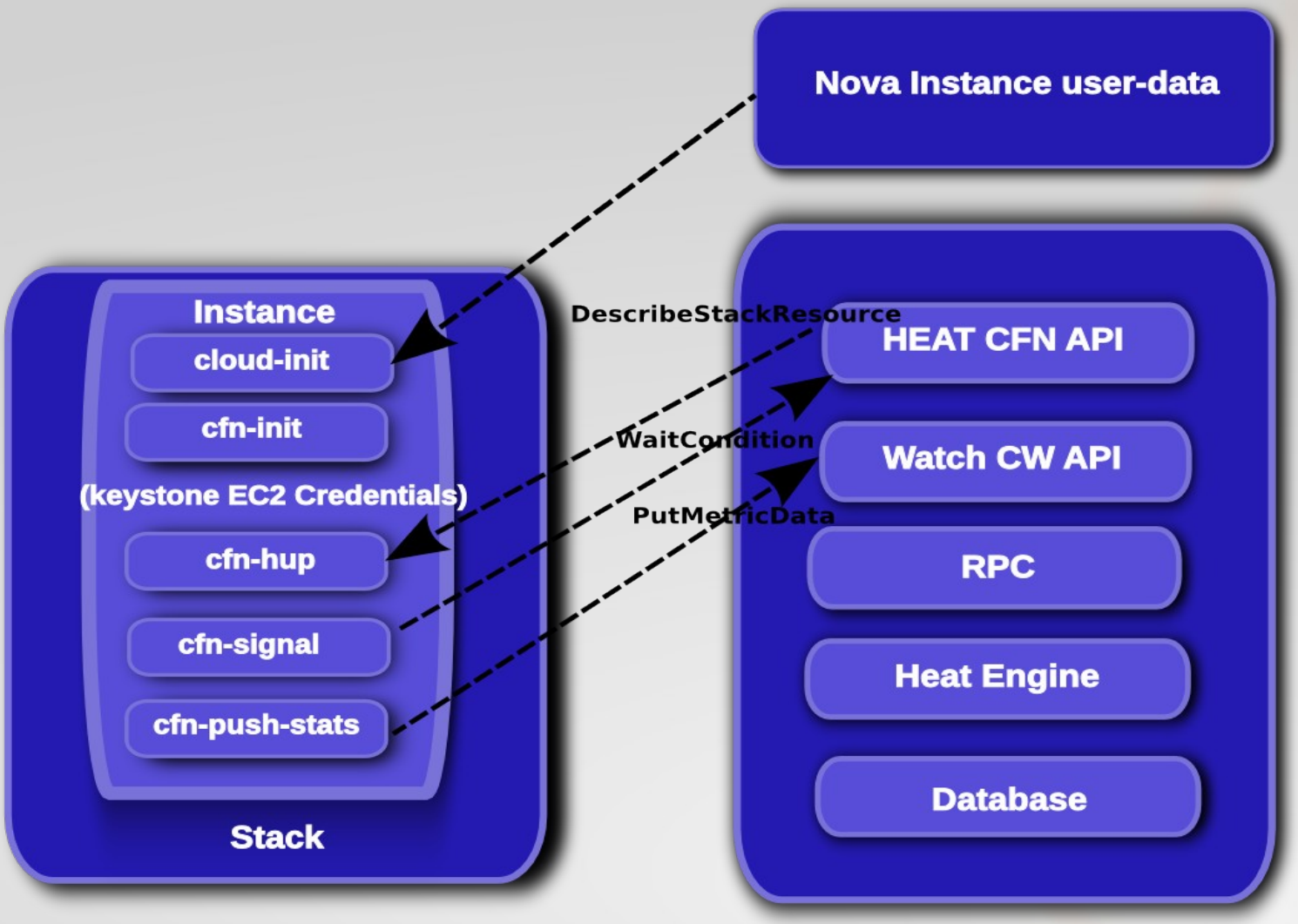
# Heat Stack lifecycle

- Base OS image stored in glance
- Deploy instances & configure services based on template contents (resources)
- Deployment-time configuration/customization
- Very flexible, but much reduced complexity
- Cloud-init (nova user/metadata), plus cfn-init
- Puppet/Chef can still be used if you want!
- Fully integrated single-service solution

# Heat Stack lifecycle



# Heat Stack lifecycle



# Why Heat/orchestration?

- Orchestration makes things repeatable
- Much easier to provide “on demand” infrastructure
- Much less scripting, single template system
- Leverage the power of openstack without learning fine detail of every service (learn one set of APIs/tools)
- Portability/migration
- Version/tag/branch your infrastructure like code!
- Composed templates/modularity
- Pluggable, user-modifiable resource implementations
- Open(stack) community project :)

# Template Parameters

## Definition:

```
"Parameters" : {
  "InstanceType" : {
    "Description" : "WebServer EC2 instance type",
    "Type" : "String",
    "Default" : "m1.large",
    "AllowedValues" : [ "t1.micro", "m1.small", "m1.large",
      "m1.xlarge", "m2.xlarge","m2.2xlarge", "m2.4xlarge",
      "c1.medium", "c1.xlarge", "cc1.4xlarge" ],
    "ConstraintDescription" : "must be a valid instance type."
  },
}
```

## Using the Parameter:

```
{ “Ref” : “InstanceType” }
```

# Template Mappings

## Definition:

```
“Mappings”: {
 "DistroArch2Inst": {
  "F16" : { "32" : "F16-i386-cfntools", "64" : "F16-x86_64-cfntools" },
  "F17" : { "32" : "F17-i386-cfntools", "64" : "F17-x86_64-cfntools" },
  "U10" : { "32" : "U10-i386-cfntools", "64" : "U10-x86_64-cfntools" }
 }
}
```

## Using the Mapping:

```
“ImageId”: {
 "Fn::FindInMap" : [
  "DistroArch2Inst", { "Ref" : "Distribution" }, { “Ref” : “Arch” }
 ]
}
```

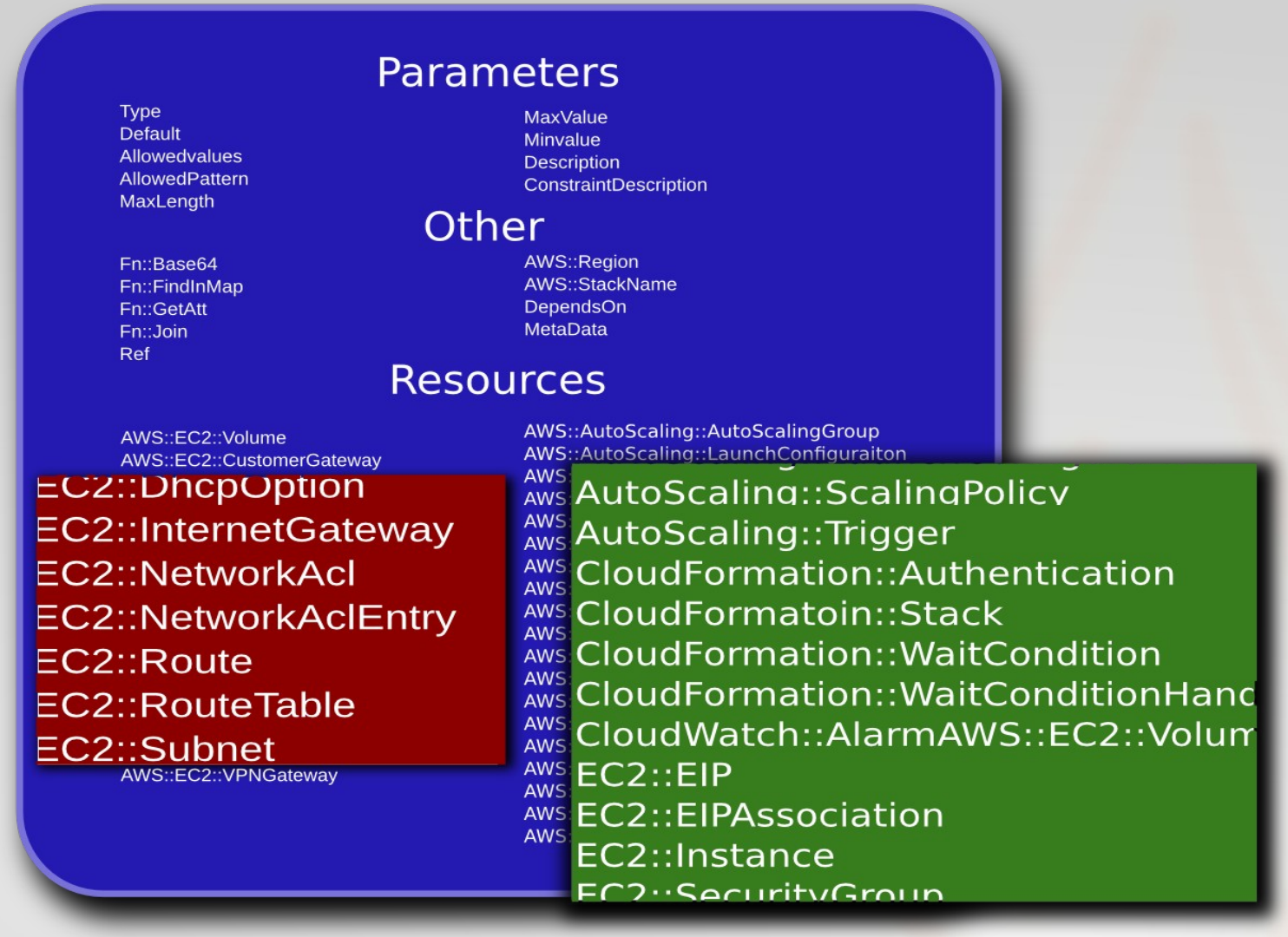
# Template Resources

```
Resources {
  “WikiDatabase”: {
    "Type" : "AWS::EC2::Instance",
    .. bunch of stuff ...
  },

  "DatabaseIPAddress" : {
    "Type" : "AWS::EC2::EIP"
  },

  "DatabaseIPAssoc" : {
    "Type" : "AWS::EC2::EIPAssociation",
    "Properties" : {
      "InstanceId" : { "Ref" : "WikiDatabase" },
      "EIP" : { "Ref" : "DatabaseIPAddress" }
    }
  }
}
```

# Template Resources (YAML)

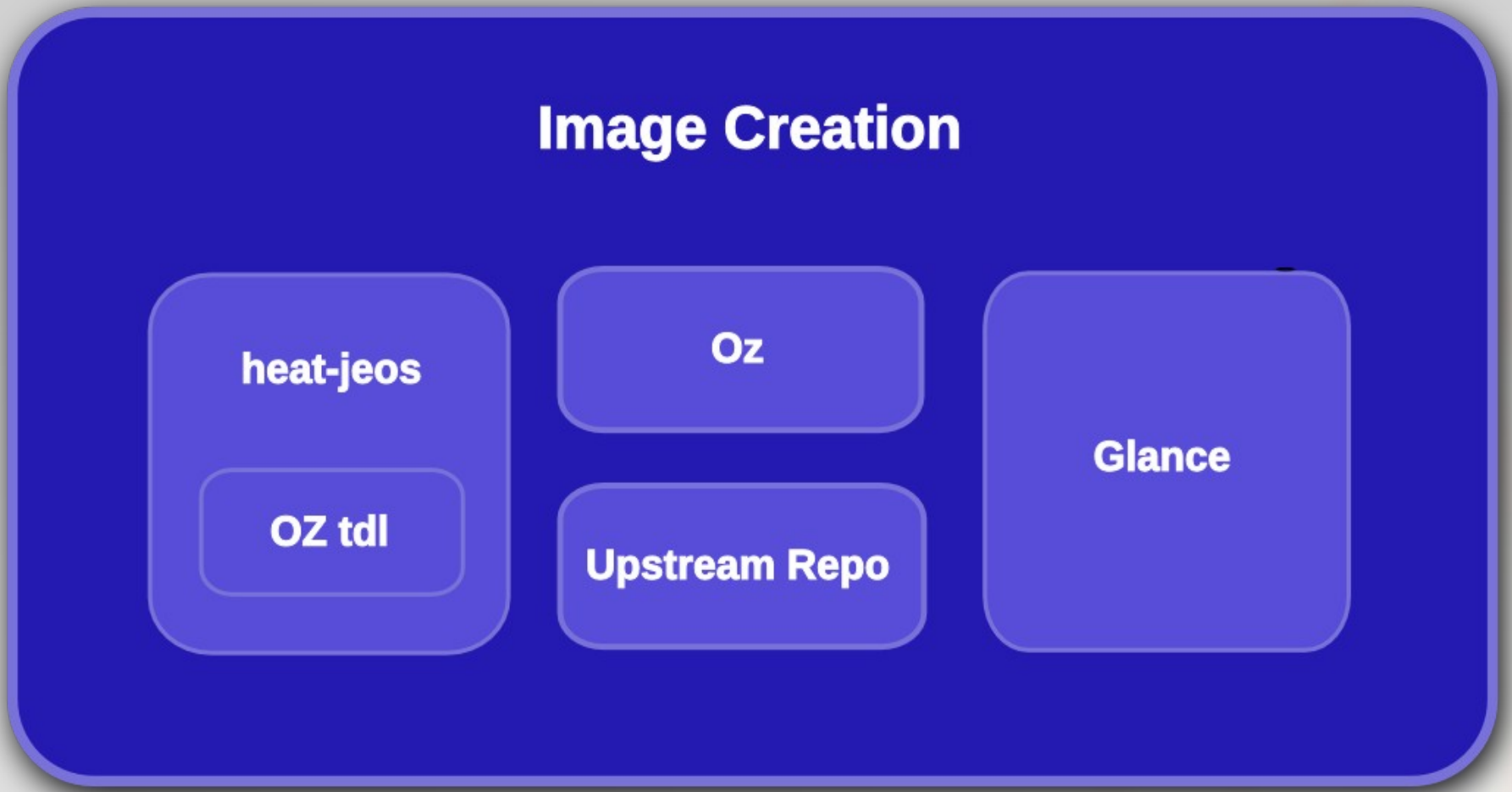
```
Resources:
  WebServer:
    Type: AWS::EC2::Instance
    ..bunch of stuff..

  IPAddress: {Type: 'AWS::EC2::EIP'}

  IPAssoc:
    Type: AWS::EC2::EIPAssociation
    Properties:
      InstanceId: {Ref: WebServer}
      EIP: {Ref: IPAddress}
```

# Heat Resource Types



# Heat Image Contents

# Heat Image Creation



# Demonstration of High Availability

```
“Resources”: {
  "WebServerRestartPolicy" : {
      "Type" : "HEAT::HA::Restarter"
      "Properties" : {
        “InstanceId" : { "Ref" :"WikiServer" }
      }
  },
  "HttpFailureAlarm": {
    "Type": "AWS::CloudWatch::Alarm",
    "Properties": {
      "AlarmDescription": "Restart the
       WikiDatabase if httpd fails > 3 times
       In 5 minutes",
          "MetricName": "ServiceFailure",
          "Namespace": "system/linux",
          "Statistic": "SampleCount",
          "Period": "300",
          "EvaluationPeriods": "1",
          "Threshold": "2",
          "AlarmActions": [ { "Ref":
  "WebServerRestartPolicy" } ],
          "ComparisonOperator":
  "GreaterThanThreshold"
    }
  },
  “WebServer”: {
    "Type": "AWS::EC2::Instance",]
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "config" : {
            "files" : {
     "/etc/cfn/notify-on-httpd-restarted" : {
       "content" : { "Fn::Join" : ["", [
       "#!/bin/sh\n",
       "/opt/aws/bin/cfn-push-stats --watch ",
         { "Ref" : "HttpFailureAlarm" },
         " --service-failure\n"
        ]]},
     },
     "/tmp/cfn-hup-crontab.txt" : {
       "content" : { "Fn::Join" : ["", [
       "MAIL=\"\"\n",
       "\n",
       "* * * * * /opt/aws/bin/cfn-hup -f\n"
       ]]},
   },
   "/etc/cfn/hooks.conf" : {
       "content": { "Fn::Join" : ["", [
       "[cfn-http-restarted]\n",
       "triggers=service.restarted\n",
       "path=Resources.WebServer.Metadata\n",
       "action=/etc/cfn/notify-on-httpd-
restarted\n",
        "runas=root\n"
     ]]},
   }
}}}}
... more instance stuff ...
}}
```

# Demonstration of Autoscaling

```
“Resources”: {
"WebServerGroup" : {
  "Type" :"AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
  "AvailabilityZones" : { "Fn::GetAZs" : ""},
  "LaunchConfigurationName" : { "Ref" : "LaunchConfig" },
  "MinSize" : "1",
  "MaxSize" : "3",
  "LoadBalancerNames" : [ { "Ref" : "ElasticLoadBalancer" } ]
      }
    },

 "WebServerScaleUpPolicy" : {
   "Type" : "AWS::AutoScaling::ScalingPolicy",
      "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "WebServerGroup" },
        "Cooldown" : "60",
        "ScalingAdjustment" : "1"
      }
    },

    "WebServerScaleDownPolicy" : {
      "Type" : "AWS::AutoScaling::ScalingPolicy",
      "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "WebServerGroup" },
        "Cooldown" : "60",
        "ScalingAdjustment" : "-1"
      }
    },
```

```
"MEMAlarmHigh": {
  "Type": "AWS::CloudWatch::Alarm",
  "Properties": {
     "AlarmDescription": "Scale-up",
     "MetricName": "MemoryUtilization",
     "Namespace": "system/linux",
     "Statistic": "Average",
     "Period": "60",
     "EvaluationPeriods": "1",
     "Threshold": "50",
     "AlarmActions": [ { "Ref":
       "WebServerScaleUpPolicy" } ],
     "Dimensions": [
       {
         "Name": "AutoScalingGroupName",
         "Value": { "Ref": "WebServerGroup" }
       }
     ],
     "ComparisonOperator": "GreaterThanThreshold"
   }
 },
```

# In Closing

- Users and developers wanted!
  - Connect with the community via IRC on #heat@freenode
  - Check out the repository:

    https://github.com/openstack/heat
  - Read the Documentation:

    http://wiki.openstack.org/Heat/
- Heat simple but powerful method for orchestrating OpenStack environments

# Questions?