Ironic Deploy Templates: Bespoke Bare Metal

Mark Goddard | mgoddard

Open Infrastructure Summit | Denver 2019

Intro

- Work for StackHPC in Bristol, UK
- Working on OpenStack for HPC since 2014
    - Previously at Cray
- Core in Ironic, Kayobe & Kolla
- Kolla Project Team Lead (PTL) for Train cycle
- Started the Kayobe project
- Job title: *YAML wrangler?*

StackHPC

- Mostly based in Bristol, UK
    - Some remote
- Software consultancy
- Focus on OpenStack for Scientific Computing
- https://stackhpc.com

Get on with it!

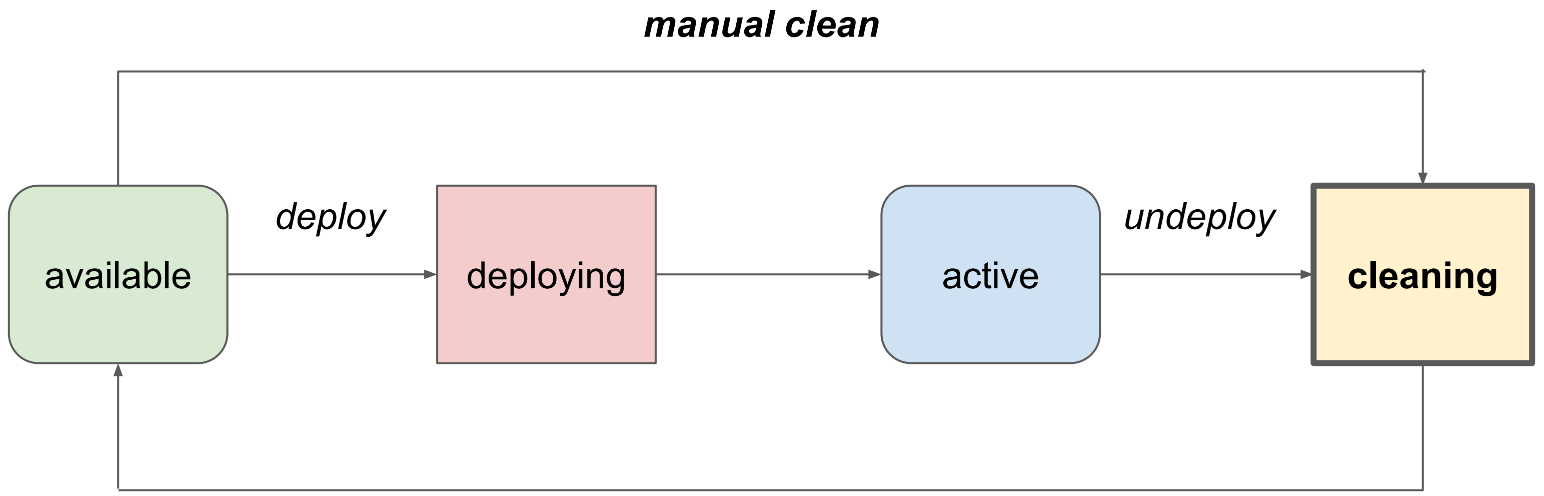
- BIOS & RAID today
- Goals
- Demo (part 1)
- Recap: Scheduling in Nova
- Ironic deployment steps
- Ironic deployment templates
- Demo (part 2)
- Tomorrow’s RAID
- Future Challenges

BIOS & RAID today

Ironic Node Cleaning

- Perform actions to sanitise hardware
- Mature feature
- Typical use case - shredding disks
- Executes a prioritised list of ***clean steps***
- Two modes
  - Automatic - when nodes are deprovisioned
  - Manual - on demand

Simplified Node State Machine



Full diagram, for the brave: https://docs.openstack.org/ironic/latest/contributor/states.html

Clean Steps

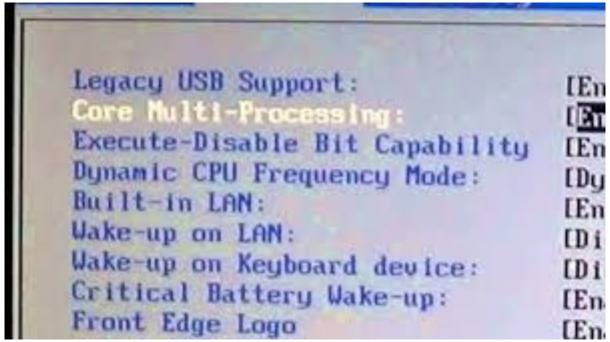
- Each step has:
  - Interface
    - Deploy, power, management, bios, raid
  - Step
    - Method (function) name on the driver interface
  - Arguments
    - Dictionary of keyword arguments
  - Priority
    - Order of execution (higher runs earlier)

BIOS

- Added BIOS driver interface in Rocky
- Provides two clean steps
  - Apply BIOS settings (list of names & values)
  - Factory reset

Example: Disable Hyperthreading

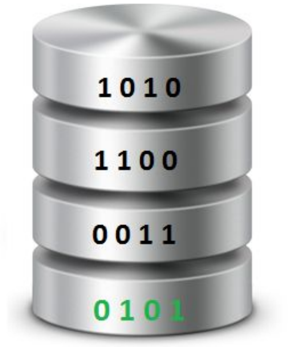
```
{
  “interface”: “bios”,
  “step”: “apply_configuration”,
  “args”: {
    “settings”: [
      {
        “name”: “LogicalProc”,
        “value”: “Disabled”
      }
    ]
  }
}
```



```
Legacy USB Support:                [En
Core Multi-Processing:             [En
Execute-Disable Bit Capability     [En
Dynamic CPU Frequency Mode:        [Dy
Built-in LAN:                      [En
Wake-up on LAN:                    [Di
Wake-up on Keyboard device:        [Di
Critical Battery Wake-up:          [En
Front Edge Logo                    [En
```

RAID

- Added RAID interface in Mitaka
- Provides two clean steps
  - Create configuration
  - Delete configuration
- Target RAID configuration set via a separate API



Example: create RAID configuration

```
{
  “interface”: “raid”,
  “step”: “create_configuration”,
  “args”: {
    “create_root_volume”: true,
    “create_nonroot_volumes”: true
  }
}
```

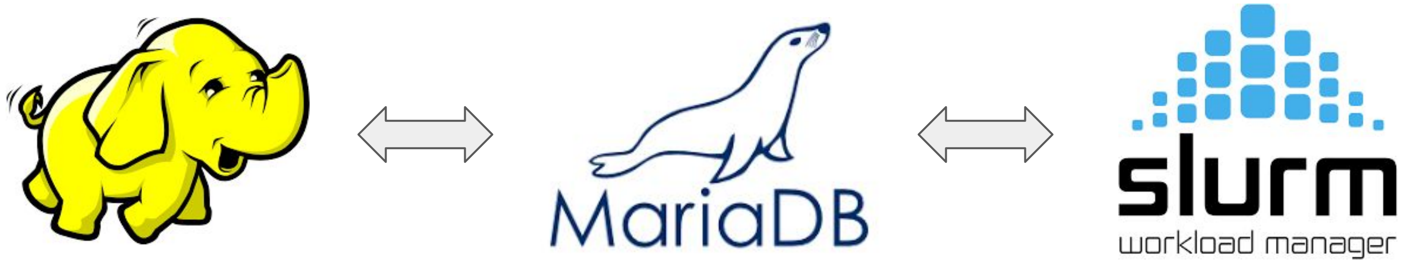
Limitations

- BIOS & RAID not integrated into deployment
    - Good for homogeneous nodes, not for customisation
- Cleaning not available to users of Nova
- RAID configuration applied in a separate API call

Goals

Use Cases

- Allow a pool of hardware to be applied to various tasks
- Use an optimal configuration for each task
- Examples
  - A Hadoop node with Just a Bunch of Disks (JBOD)
  - A database server with mirrored & striped disks (RAID 10)
  - A High Performance Computing (HPC) compute node, with tuned BIOS parameters



Custom deploy-time Configuration

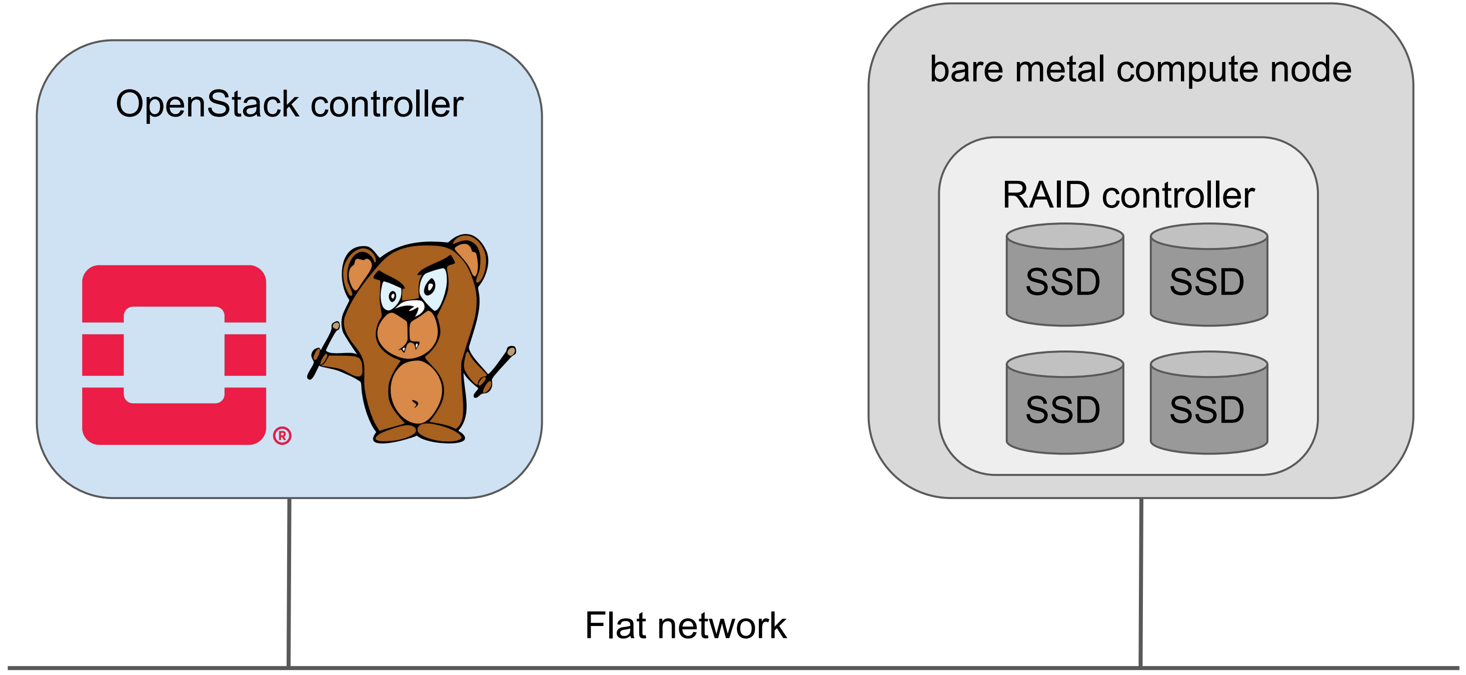
- BIOS
  - e.g. Enable Hyperthreading
- RAID
  - Hardware or software
  - e.g. Mirrored disks
- System performance profile
  - e.g. tuned for HPC
- *<your use case here>*

Make it Cloudy

- Works for non-admin users
- Abstracted from hardware specifics
- Operator controlled
  - What can be configured
  - Who can configure it
- Use existing interfaces

Demo (part 1)

Demo Setup


OpenStack controller
bare metal compute node
RAID controller
SSD
SSD
SSD
SSD
Flat network

Demo Setup - Specifics

- OpenStack Stein pre-release
- Deployed via Kayobe
- Some patches applied to Ironic for RAID configuration
- Both nodes are Dell R630
- Compute node has 4 x 372GB SSDs with a RAID controller

Recap: Scheduling in Nova

Placement

- Added in Newton
- Extracted from Nova in Stein
- API for tracking resource inventory & consumption
- Quantitative & qualitative scheduling

Placement Concepts

A ***Resource Provider*** provides an ***Inventory*** of resources of different ***Resource Classes***

A ***Resource Provider*** may be tagged with one or more ***Traits***

A ***Consumer*** may have an ***Allocation*** that consumes some of a ***Resource Provider****’s* ***Inventory***

Placement Concepts (for VMs)

A ***Compute Node*** provides an ***Inventory*** of ***vCPU***, ***Disk*** & ***Memory*** resources

A ***Compute Node*** may be tagged with one or more ***Traits***

An ***Instance*** may have an ***Allocation*** that consumes some of a ***Compute Node***'s ***Inventory***

Placement Example (for VMs)

GET /resource_providers/{uuid}/inventories

```
{
    "inventories": {
        "DISK_GB": {
            "allocation_ratio": 1.0, "max_unit": 35, "min_unit": 1,
            "reserved": 0, "step_size": 1, "total": 35
        },
        "MEMORY_MB": {
            "allocation_ratio": 1.5, "max_unit": 5825, "min_unit": 1,
            "reserved": 512, "step_size": 1, "total": 5825
        },
        "VCPU": {
            "allocation_ratio": 16.0, "max_unit": 4, "min_unit": 1,
            "reserved": 0, "step_size": 1, "total": 4
        }
    },
    "resource_provider_generation": 7
}
```

Why this doesn’t work for Ironic

- Bare metal nodes are indivisible units
- Cannot be shared (by instances)
- Cannot be overcommitted
- Either in use or not

Placement Concepts (for Ironic)

A ***Bare Metal Node*** provides an ***Inventory*** of ***one unit of a custom resource***

A ***Bare Metal Node*** may be tagged with one or more ***Traits***

An ***Instance*** may have an ***Allocation*** that consumes all of a ***Bare Metal Node****'s* ***Inventory***

Placement Example (for Ironic)

```
GET /resource_providers/{uuid}/inventories
{
    "inventories": {
        "CUSTOM_GOLD": {
            "allocation_ratio": 1.0,
            "max_unit": 1,
            "min_unit": 1,
            "reserved": 0,
            "step_size": 1,
            "total": 1
        }
    },
    "resource_provider_generation": 1
}
```

Bare Metal Flavors

```
openstack flavor show bare-metal-gold -f json \
-c name -c ram -c properties -c vcpus -c disk
{
  "name": "bare-metal-gold",
  "vcpus": 4,
  "ram": 4096,
  "disk": 1024,
  "properties": "resources:CUSTOM_GOLD='1',
                 resources:DISK_GB='0',
                 resources:MEMORY_MB='0',
                 resources:VCPU='0'"
}
```

Traits

```
GET /resource_providers/{uuid}/traits
{
    "resource_provider_generation": 1,
    "traits": [
        "CUSTOM_HW_FPGA_CLASS1",
        "CUSTOM_HW_FPGA_CLASS3"
    ]
}
```

Ironic Node Resource Class and Traits

A bare metal node has a resource class and zero or more traits:

```
GET /nodes/{uuid}?fields=name,resource_class,traits
{
  "Name": "gold-node-1",
  "Resource Class": "GOLD",
  "Traits": [
    “CUSTOM_RAID0”,
    “CUSTOM_RAID1”,
  ]
}
```

Requesting Traits

```
openstack flavor show bare-metal-gold -f json -c name -c properties
{
  "name": "bare-metal-gold",
  "properties": "resources:CUSTOM_GOLD='1',
                 resources:DISK_GB='0',
                 resources:MEMORY_MB='0',
                 resources:VCPU='0',
                 trait:CUSTOM_RAID0='required'"
}
```

How Ironic Knows Which Traits Were Requested

```
GET /nodes/{uuid}?fields=name,instance_info,traits
{
  "Name": "gold-node-1",
  "Instance Info": {
    "traits": [
      "CUSTOM_RAID0"
    ]
  }
  "Traits": [
    "CUSTOM_RAID0",
    "CUSTOM_RAID1",
  ]
}
```

Scheduling Summary

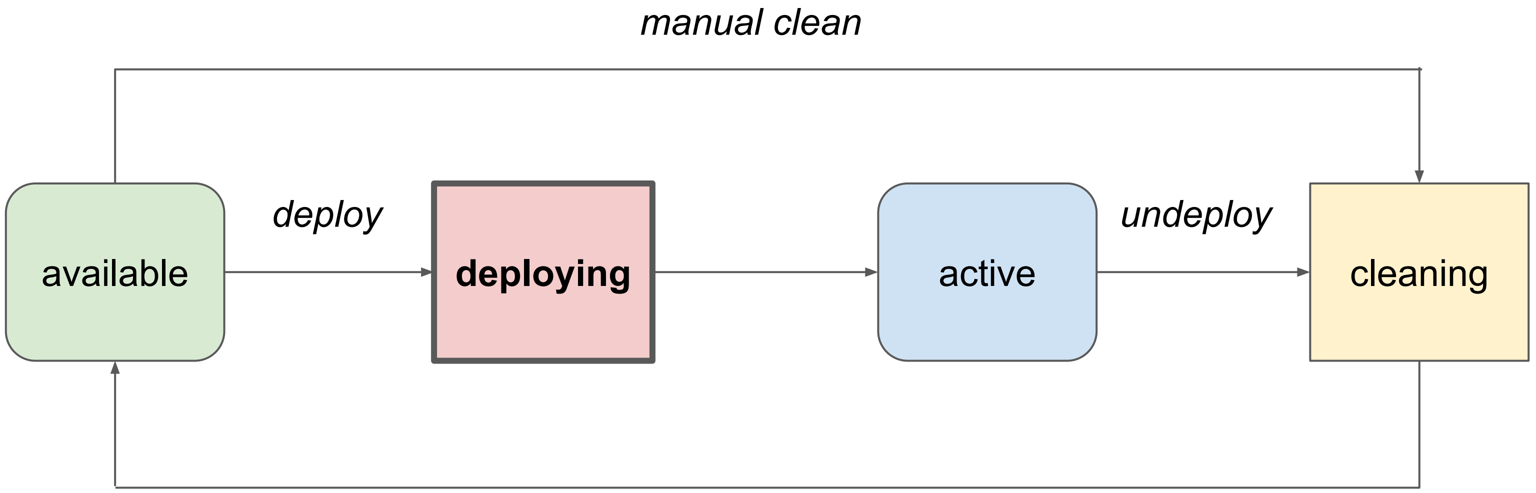
- Resource classes
  - Bare metal nodes are assigned a custom ***resource class***
  - A flavor requests one unit of the node’s ***resource class***
- Traits
  - Bare metal nodes may be assigned zero or more ***traits***
  - A flavor requests a subset of the node’s ***traits***
  - An image may also request a subset of the node’s ***traits***
  - Ironic is informed which ***traits*** were requested by an instance

Deployment Steps

Deployment Steps

- Added in Rocky
- Based on clean step model
- Define steps to execute during deployment
- Each step has:
  - Interface
    - Deploy, power, management, bios, raid
  - Step
    - Function name
  - Arguments
    - Name/values
  - Priority
    - Order of execution

Simplified Node State Machine



Full diagram, for the brave: https://docs.openstack.org/ironic/latest/contributor/states.html

The Mega Step

- Entire deployment process moved to a single step in Rocky
- Drivers can currently add steps before or after this
- Plan to split this into multiple ***core*** steps in Train

Limitations

- Steps are static for a given set of driver interfaces
- Cannot execute steps on the deployment agent
- Mega step limits ordering

Deployment Templates

Deployment Templates

- Added in Stein
- Adds an API to register ***deployment templates***
- A deployment template has
  - A name, which must be a valid ***trait***
  - A list of deployment steps

Example

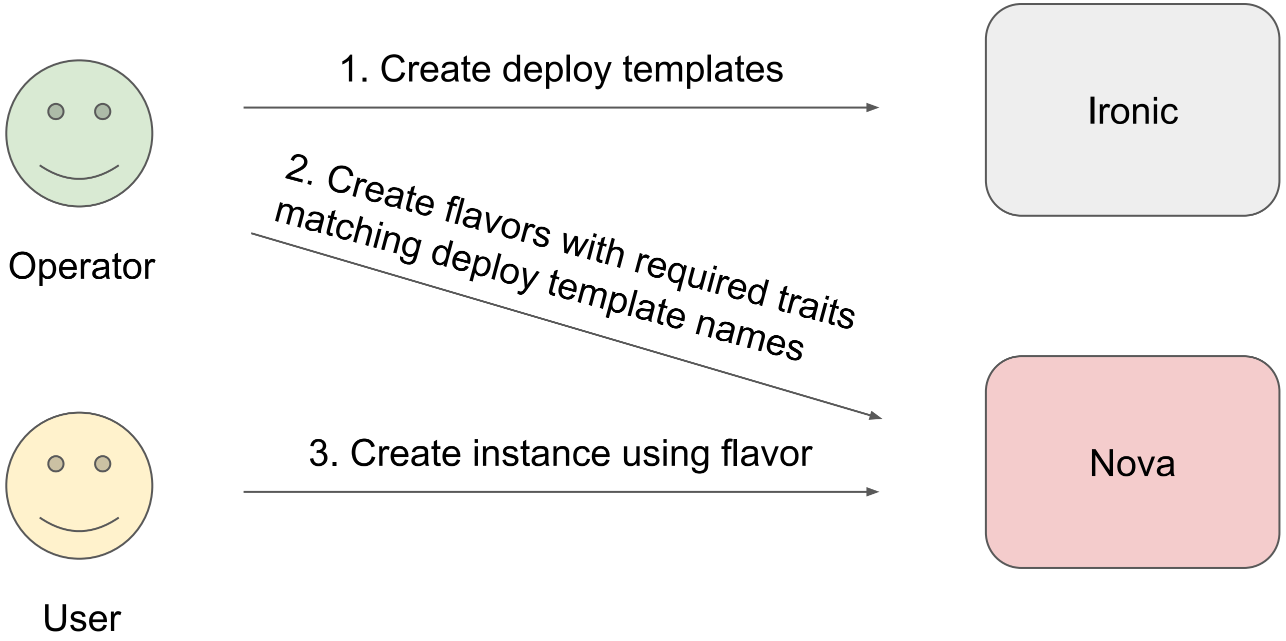
POST /v1/deploy_templates

```
{
    "name": "CUSTOM_HYPERTHREADING_ON",
    "steps": [
        {
            "interface": "bios",
            "step": "apply_configuration",
            "args": {
                "settings": [
                    {
                        "name": "LogicalProc",
                        "value": "Enabled"
                    }
                ]
            },
            "priority": 150
        }
    ]
}
```

End to End

1. Operator creates deploy templates via Ironic API to execute deploy steps for allowed actions
2. Operator creates Nova flavors or Glance images with required traits that reference deploy templates
3. User creates bare metal instances using allowed flavors

End to End (2)

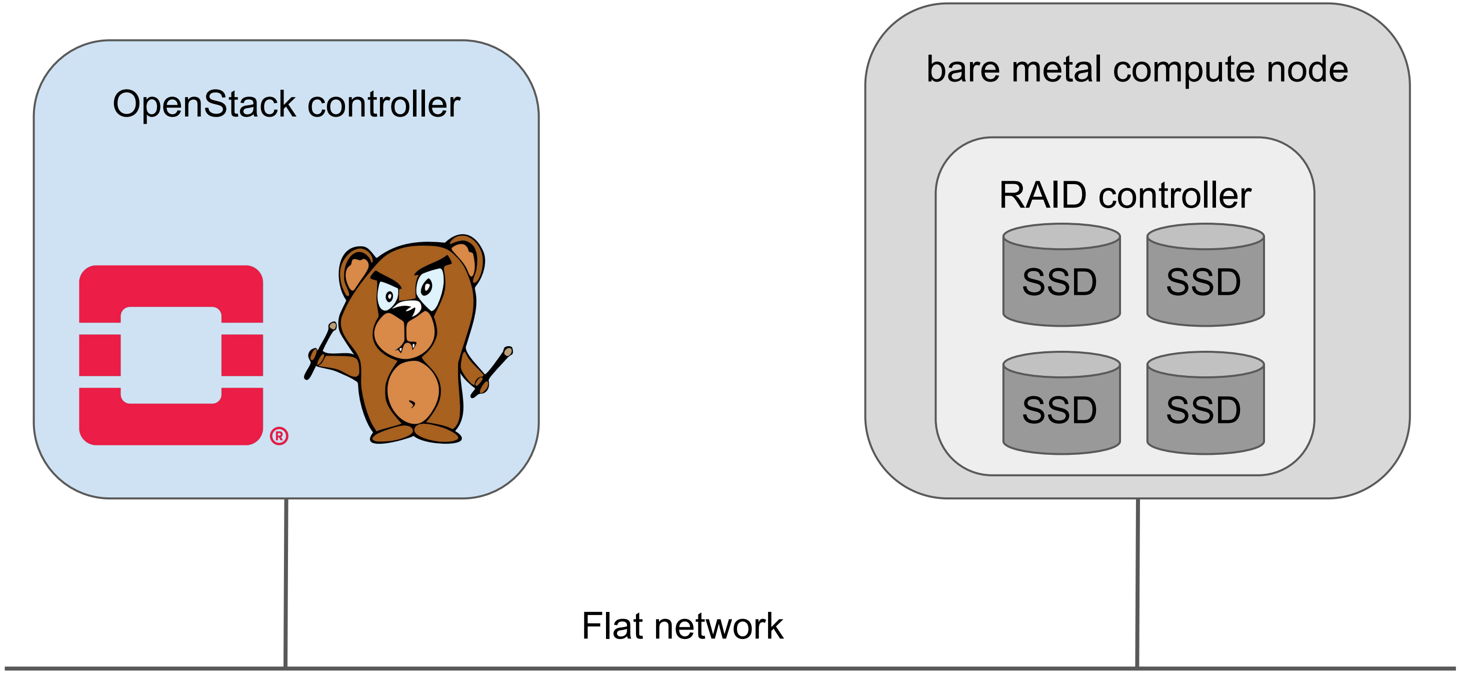

1. Create deploy templates
Ironic
2. Create flavors with required traits matching deploy template names
Operator
3. Create instance using flavor
Nova
User

Bespoke Bare Metal

- Finally, we have an API we can use to influence deployment
- https://docs.openstack.org/ironic/stein/admin/node-deployment.html#deploy-steps
- https://docs.openstack.org/ironic/stein/admin/node-deployment.html#deploy-templates

Demo (part 2)

Demo Setup Reminder



Tomorrow’s RAID

Patchy McPatchface

- In Stein we lack deploy step implementations to make this useful
- Demo required patches to Ironic
- Adds an ‘apply_configuration’ deploy step to the RAID interface
- Accepts RAID configuration as an argument
  - No separate API call as in cleaning
- Changes to avoid unnecessary reboots
- Should be included in Train release

Future Challenges

Train & Beyond

- Splitting out the core deploy mega step
- Running deploy steps in the agent
- Deploy step implementations
  - BIOS
  - RAID
- Executing a step multiple times
  - How to combine multiple executions of the same step?
  - e.g. multiple BIOS settings in different templates

Train & Beyond (2)

- Deploying with an explicit deploy step list
  - Similar to manual cleaning
- Executing deploy steps on active nodes
  - BIOS tweaks etc.
  - https://storyboard.openstack.org/#!/story/2005129
- Avoiding unnecessary reboots
- Vendor-agnostic templates

Thanks for listening

Questions?