

OpenStack RESTful API

Compatible or Consistent

2014.10.15

Ken'ichi Ohmichi
NEC Corporation

Agenda

- Who am I
- OpenStack
- OpenStack RESTful API
- Nova RESTful API
- Issues of Nova RESTful API
- Going Solutions
- Current Situation
- Summary



\$ who am i

- **Software developer**
- **Join OpenStack community for its quality assurance since 2012**
- **A core developer of**
 - **OpenStack Compute (Nova)**
 - **OpenStack QA (Quality Assurance)**

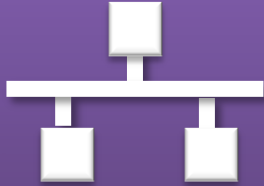
The background of the slide is a sunset or sunrise scene. A bright sun is positioned just below the horizon line, creating a strong orange and yellow glow that spreads across the sky. The sky transitions from a deep orange near the horizon to a clear blue at the top. The text 'OpenStack' is centered in the middle of the image, rendered in a white, italicized, sans-serif font with a thin black outline.

OpenStack

- **OpenStack components**

OpenStack

Neutron
Network



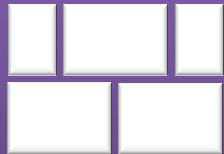
Nova
Compute



Glance
Image



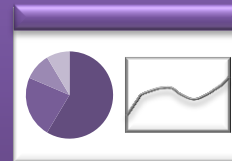
Cinder
Block storage



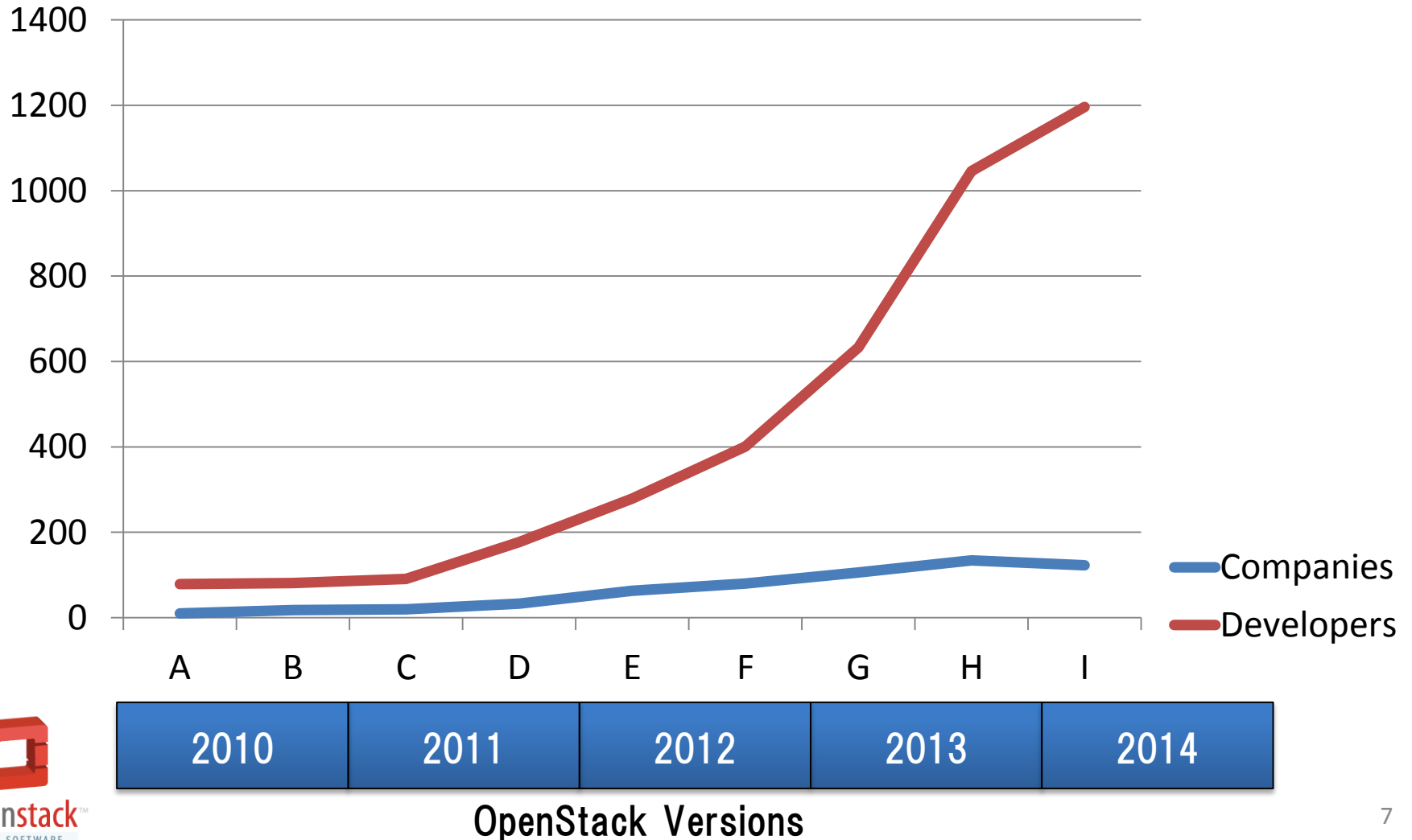
Keystone
Identity



Horizon
Dashboard



- OpenStack community
 - Rapid Growth



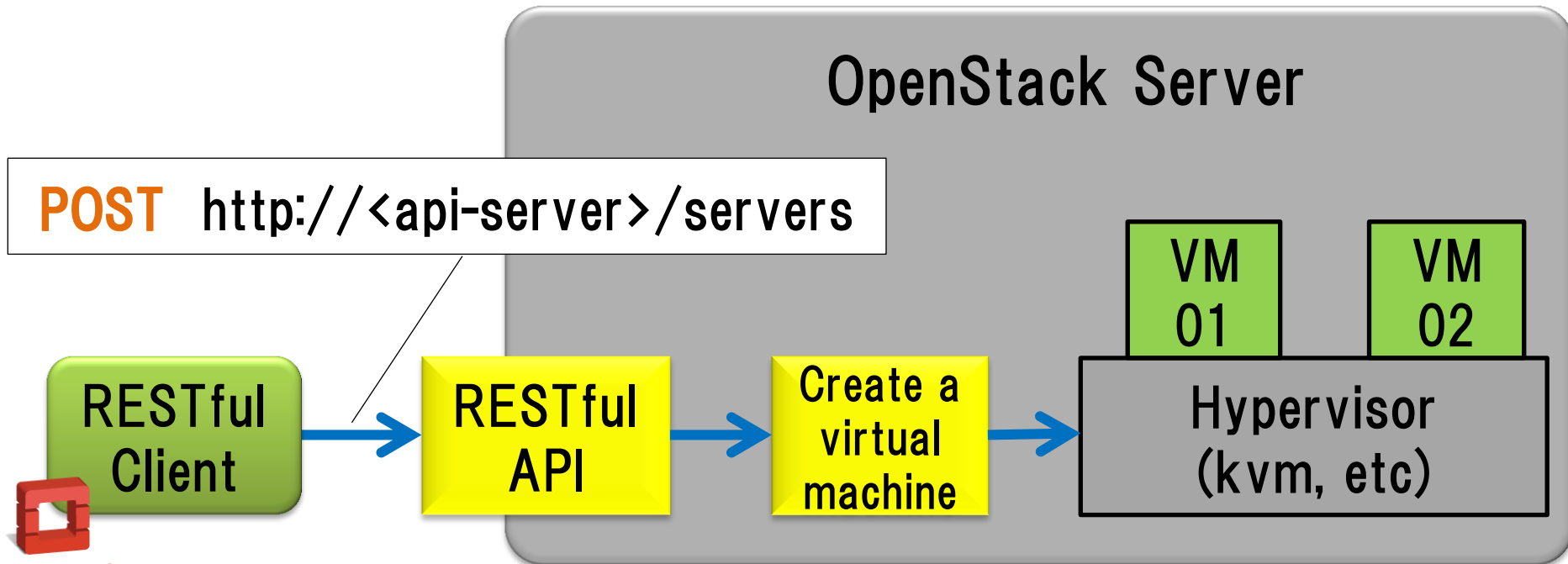
- **Community development**
 - Importance of code review
 - Can know the internal implementation
 - Can know the trend
 - Can get folks
 - Importance of communication
 - Daily : IRC or e-mail
 - Weekly: IRC meeting
 - Version: F2F at Design Summit



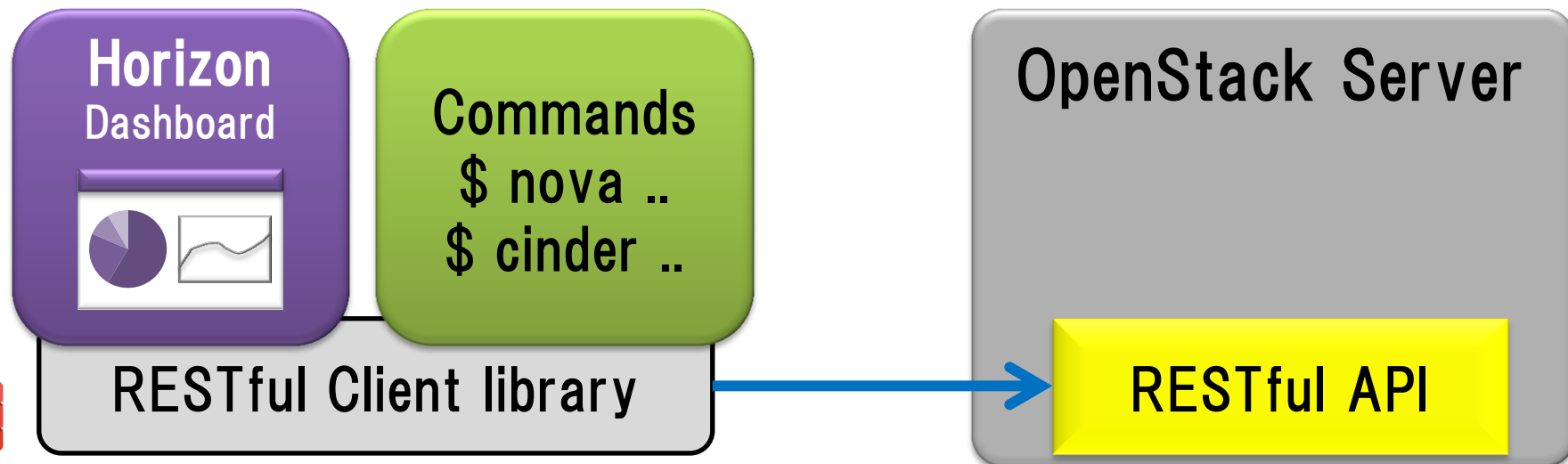
A scenic photograph of a sunrise or sunset. The sun is a bright, glowing orb on the horizon, partially obscured by a thick layer of white and orange-tinted clouds. The sky above is a deep blue, with a few wispy clouds. In the foreground, dark, silhouetted mountain peaks are visible against the lower part of the cloud layer. A small, colorful lens flare is visible in the upper center of the image.

OpenStack RESTful API

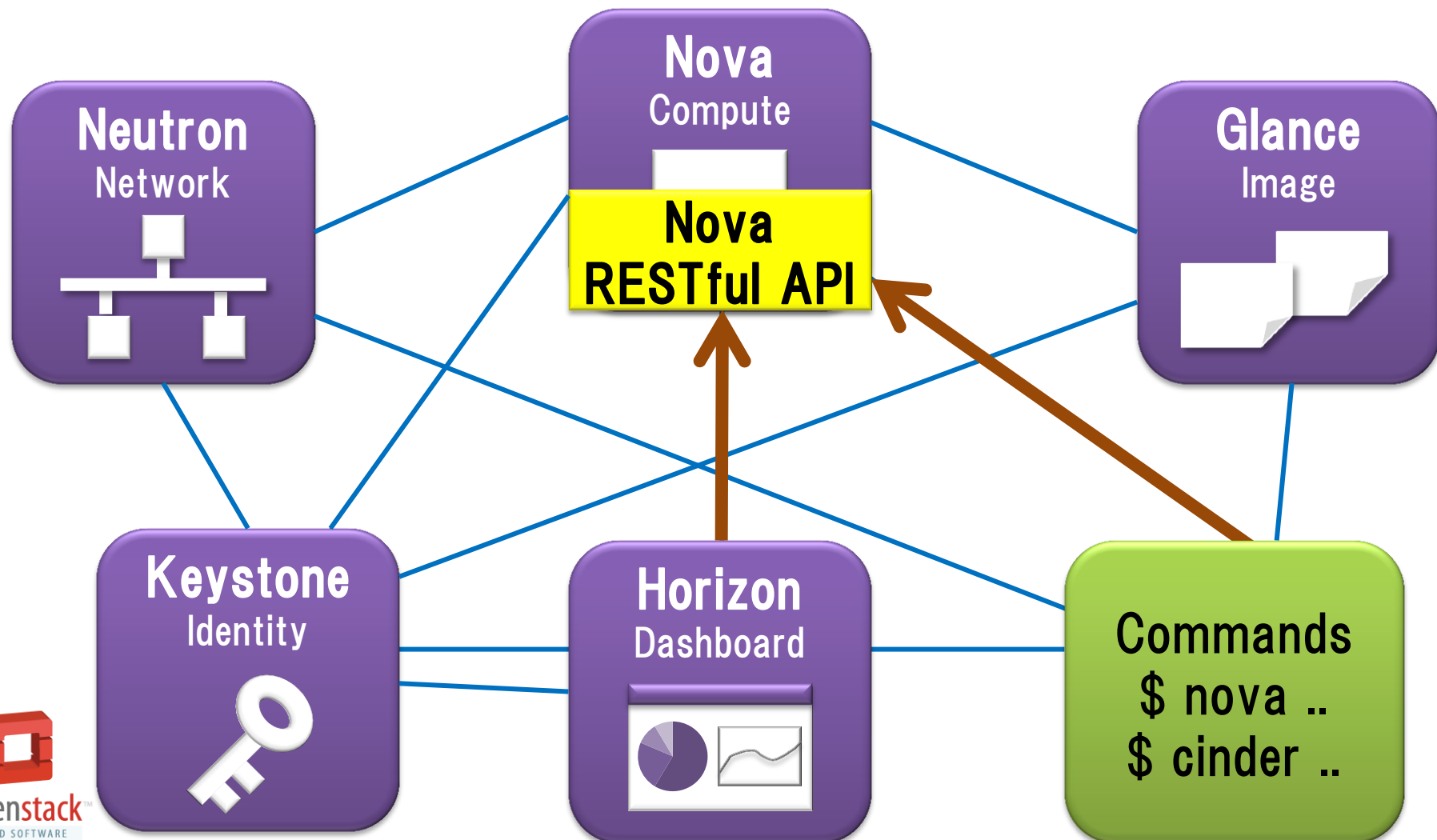
- RESTful API
 - Scalable and stateless
 - Simple and consistent interfaces
- OpenStack provides many features through RESTful API



- Many RESTful client libraries are available for OpenStack
 - Python
 - Java
 - .NET
 - ...
- Dashboard and commands request operations via RESTful APIs



- Each OpenStack component also is connected via RESTful APIs



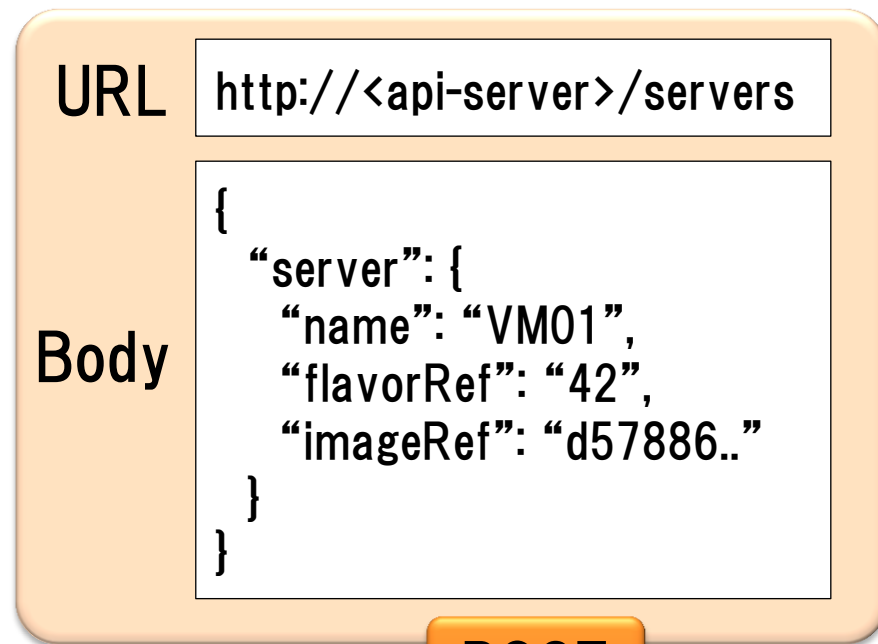
A full-page background image showing a bright sun setting or rising over a dark, choppy ocean. The sun is a large, glowing yellow orb in the center of the frame, casting a shimmering path of light across the water's surface. The sky is filled with soft, orange and yellow clouds, and the overall color palette is warm and golden.

Nova RESTful API

- **OpenStack core functions**
 - Create a VM (virtual machine)
 - Delete a VM
 - Attach a volume to a VM
 - ...
- **Scalability**
- **Stability**
- **Current API version is v2**
- **Nova v2 API contains 250+ APIs (Icehouse)**



- API usage
 - Create a VM (virtual machine)



POST

RESTful
Client

Nova
v2 API

Nova API server

VM01

Hypervisor
(kvm, etc)



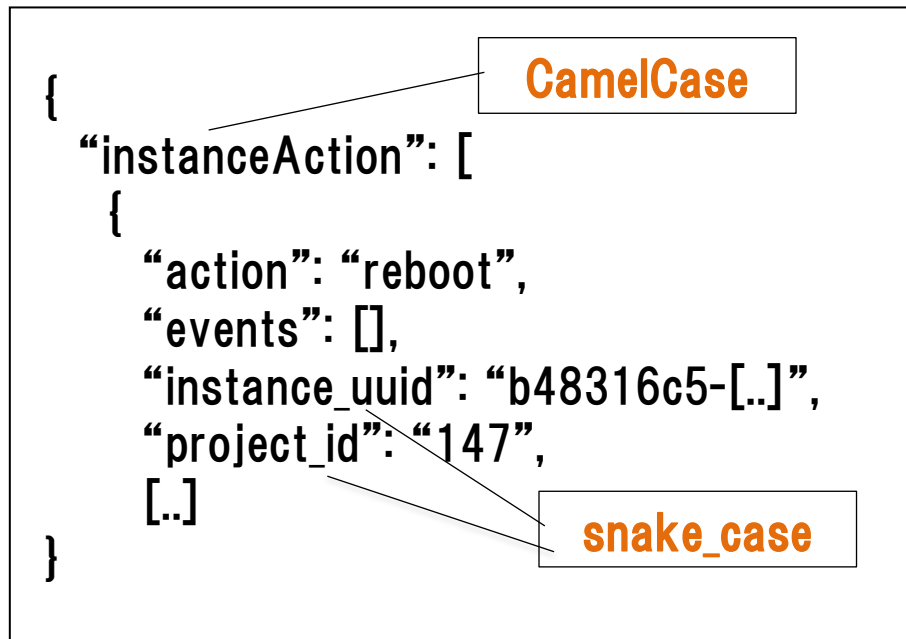
The background of the slide is a dramatic, low-key photograph of a stormy sky. Dark, heavy clouds fill most of the frame, with some lighter patches where light breaks through. In the foreground, the dark silhouettes of trees are visible against the bottom and right edges of the image.

Nova RESTful API

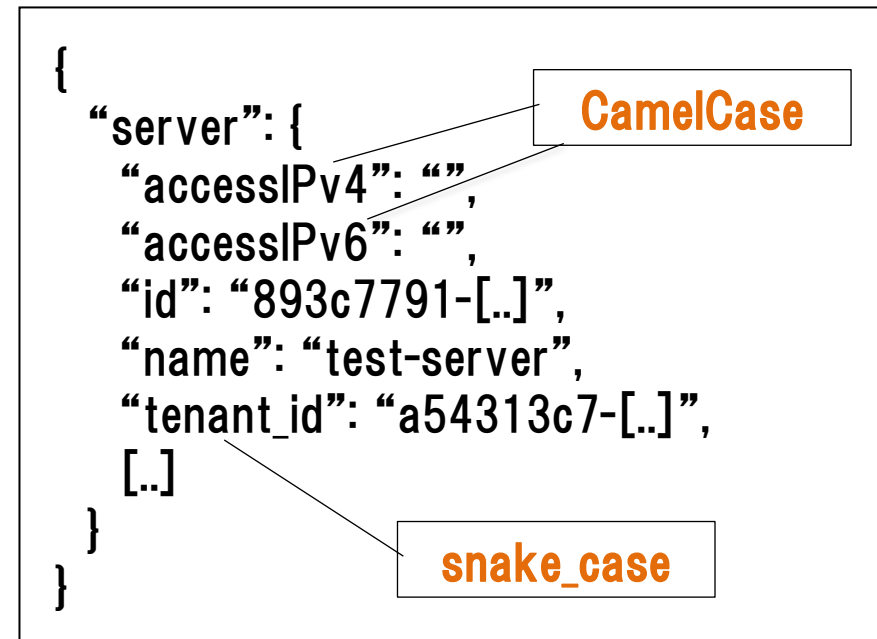
• Inconsistent interfaces

– Inconsistent naming

- “instance” or “server”, “project” or “tenant”
- CamelCase, or snake_case



“Show instance actions” API

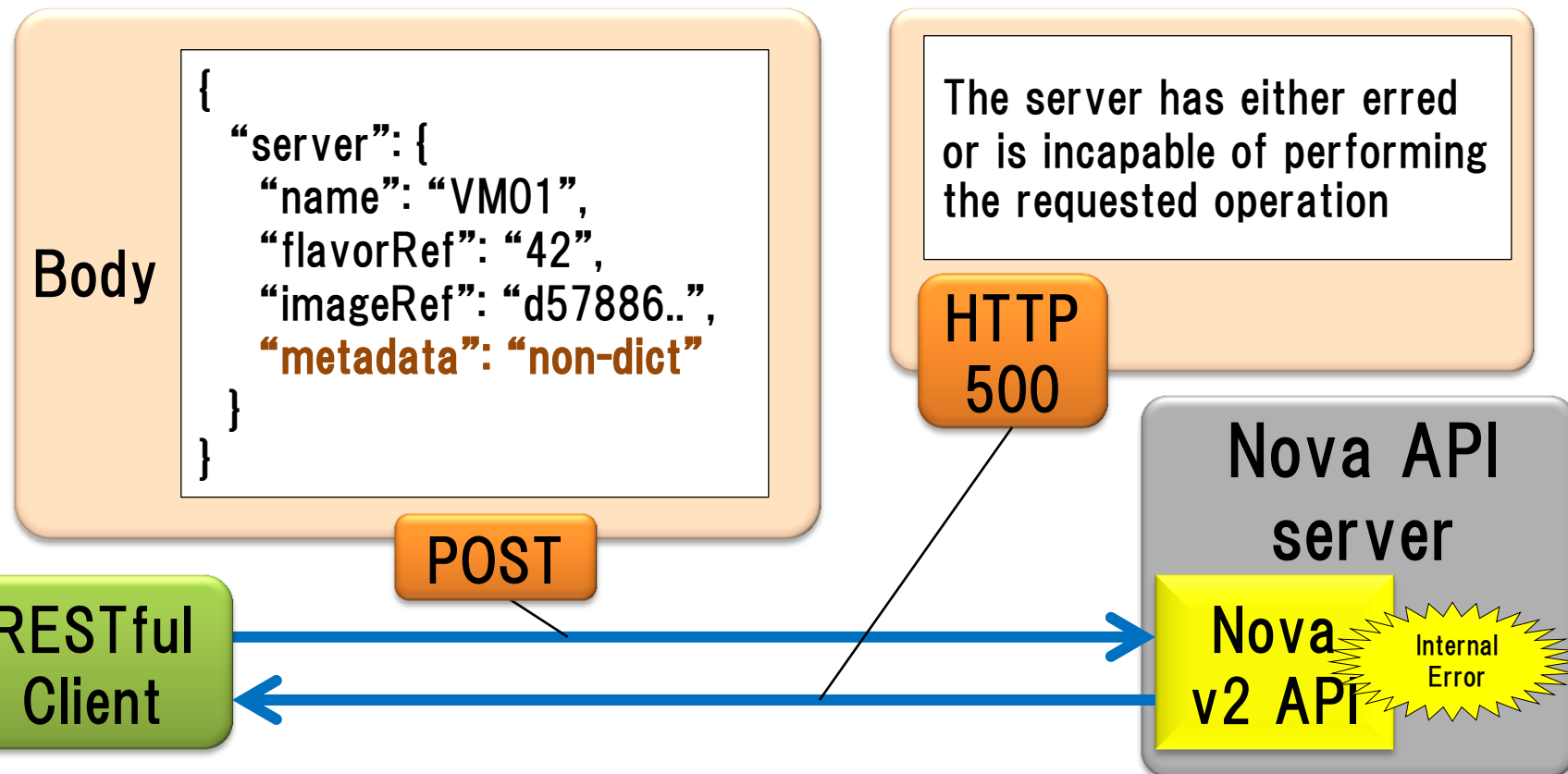


“Show server” API

- **Inconsistent interfaces**
 - **Return incorrect success codes**
 - **Should return 202(ACCEPTED) but some APIs return 201(CREATED) even if not complete to create a resource yet.**

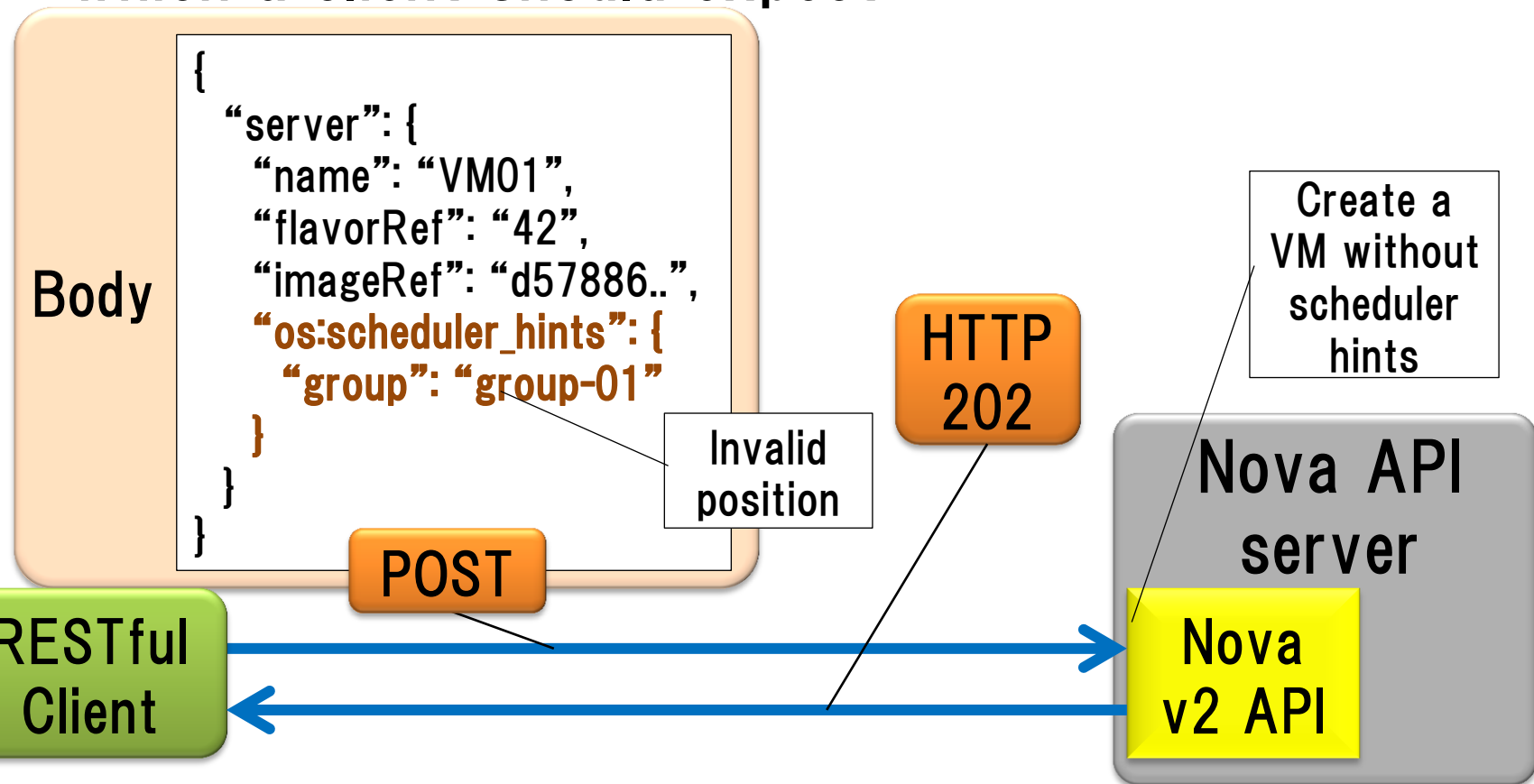
- Input validation lacks

- Cause internal errors, output a lot of error logs in server, and return useless error messages
 - Ex) Pass non-dict as metadata to “create a VM”



- Input validation lacks

- Ignore unexpected data which a client sends, and the request operates without some details which a client should expect



- **Problems of input validation lacks**
 - Clients cannot understand what happens and what they should do
 - Error response message doesn't contain any hints:

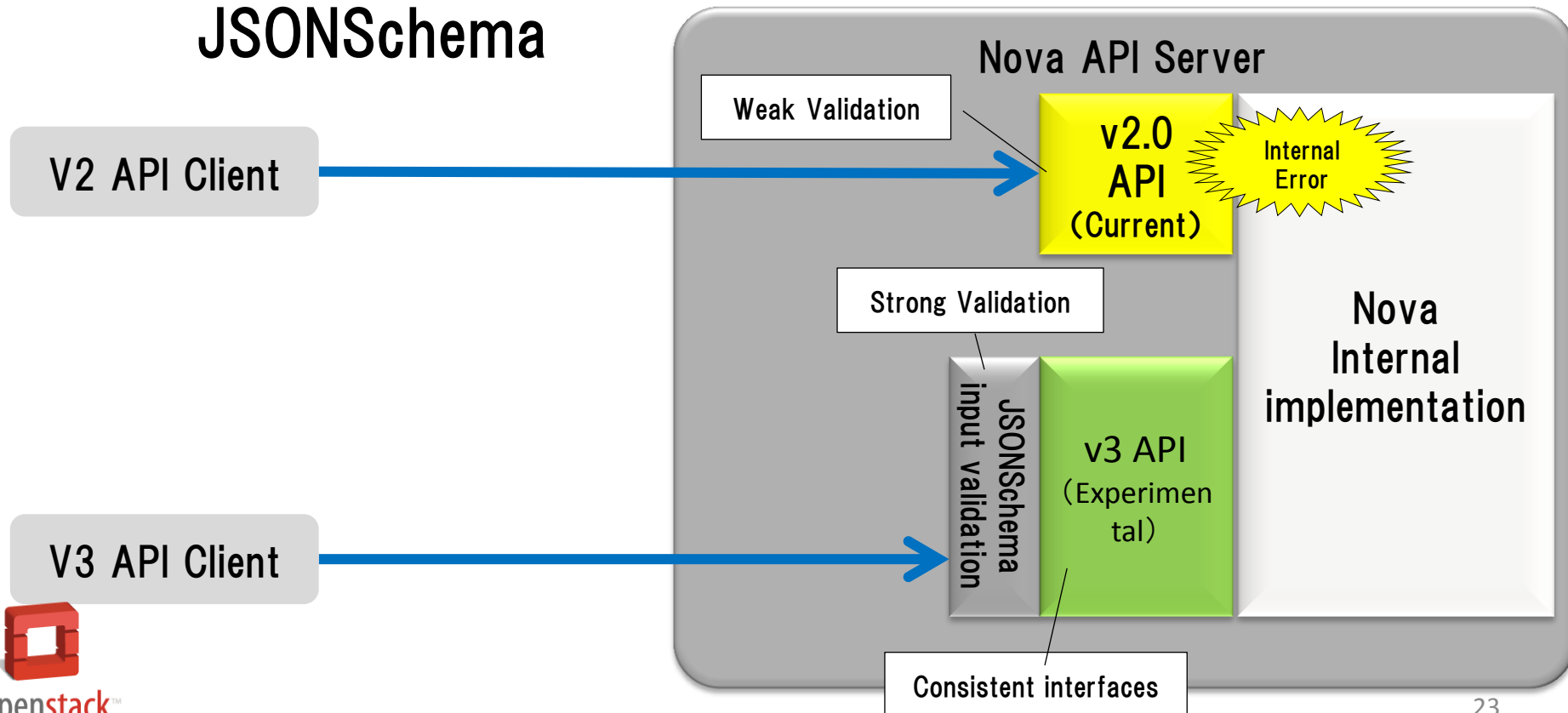
The server has either erred or is incapable of performing the requested operation
 - Cloud operators should investigate internal errors and explain the reasons to clients
 - Database inconsistencies happened sometimes and cloud operators should fix them by hands





Possible Solutions

- Nova v3 API (Havana -> Juno)
 - Re-define consistent success codes
 - Re-define consistent naming
 - Comprehensive input validation with JSONSchema



- JSONSchema

- Common data format definition library
- Clear, human- and machine-readable data format definition

```
{  
  "get_console_output": {  
    "length": 100  
  }  
}
```

Correct request data

Pass

```
{  
  "type": "object",  
  "properties": {  
    "get_console_output": {  
      "type": "object",  
      "properties": {  
        "length": {  
          "type": "integer",  
          "minimum": -1  
        }  
      }  
    }  
  },  
  "additionalProperties": False,  
  "required": ["get_console_output"]  
}
```

Data format definition with JSONSchema



- JSONSchema

- Contains basic validation features in nature
- Ex) minimum

```
{  
  "get_console_output": {  
    "length": -100  
  }  
}
```

Invalid request data

Fail

```
{  
  "type": "object",  
  "properties": {  
    "get_console_output": {  
      "type": "object",  
      "properties": {  
        "length": {  
          "type": "integer",  
          "minimum": -1  
        }  
      }  
    }  
  },  
  "additionalProperties": False,  
  "required": ["get_console_output"]  
}
```

Data format definition with JSONSchema



- JSONSchema

- Can deny undefined parameters
- Ex) `additionalProperties: False`

```
{  
  "get_console_output": {  
    "length": 100  
  },  
  "undefined": 100  
}
```

Invalid request data

Fail

```
{  
  "type": "object",  
  "properties": {  
    "get_console_output": {  
      "type": "object",  
      "properties": {  
        "length": {  
          "type": "integer",  
          "minimum": -1  
        }  
      }  
    }  
  },  
  "additionalProperties": False,  
  "required": ["get_console_output"]  
}
```

Data format definition with JSONSchema



- **Use JSONSchema for input validation**
 - Strong validation, then avoid internal errors
 - Return useful error message if a client sends invalid format data

**The server has either
erred or is incapable of
performing the requested
operation**

before JSONSchema validation

**Invalid input for
field/attribute name.
Value: 1.
1 is not of type 'string'**

after JSONSchema validation

- Use JSONSchema for input validation
 - Clean up logic code by separating validation code
 - Validation code also can be reduced

```
try:
    min_count = int(str(min_count))
except ValueError:
    msg = _('min_count must be an integer value')
    raise exc.HTTPBadRequest(explanation=msg)
if min_count < 1:
    msg = _('min_count must be > 0')
    raise exc.HTTPBadRequest(explanation=msg)
```

Validation code of v2 API

```
'min_count': {
    'type': 'integer', 'minimum': 1
}
```

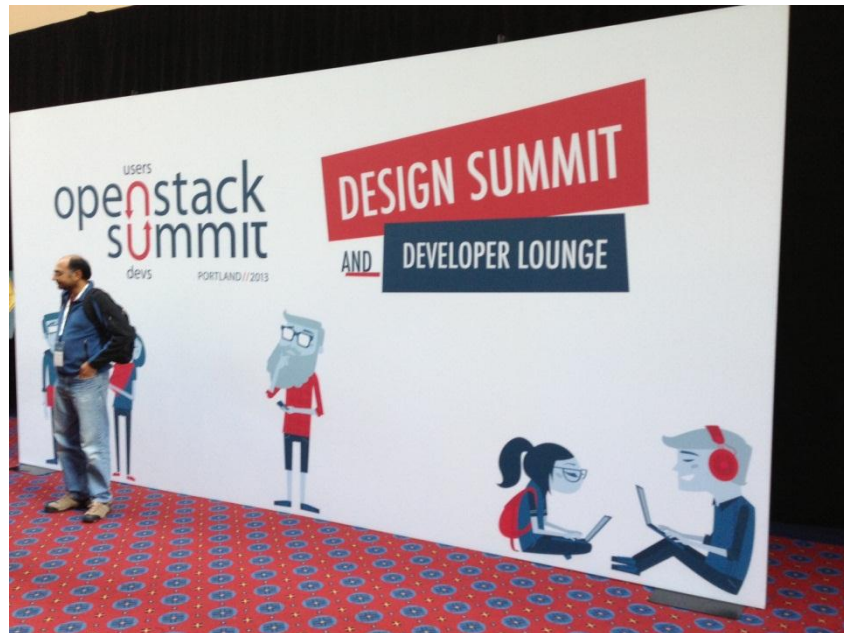
Validation code of v3 API





Current Situation

- **Input validation development status**
 - 2013.03 The prototype proposal
 - 2013.04 The first proposal in Portland summit
 - 2013.07 The framework part was approved, but merger failed
 - 2013.08 Do not merge * 2



- **Input validation development status**
 - 2013.08 Investigated input validations of all API parameters and proposed necessary features.
 - 2013.11 The second proposal in Hong Kong summit. We have gotten a consensus.
 - 2014.09 All 38 patches are merged.

Complete!!!



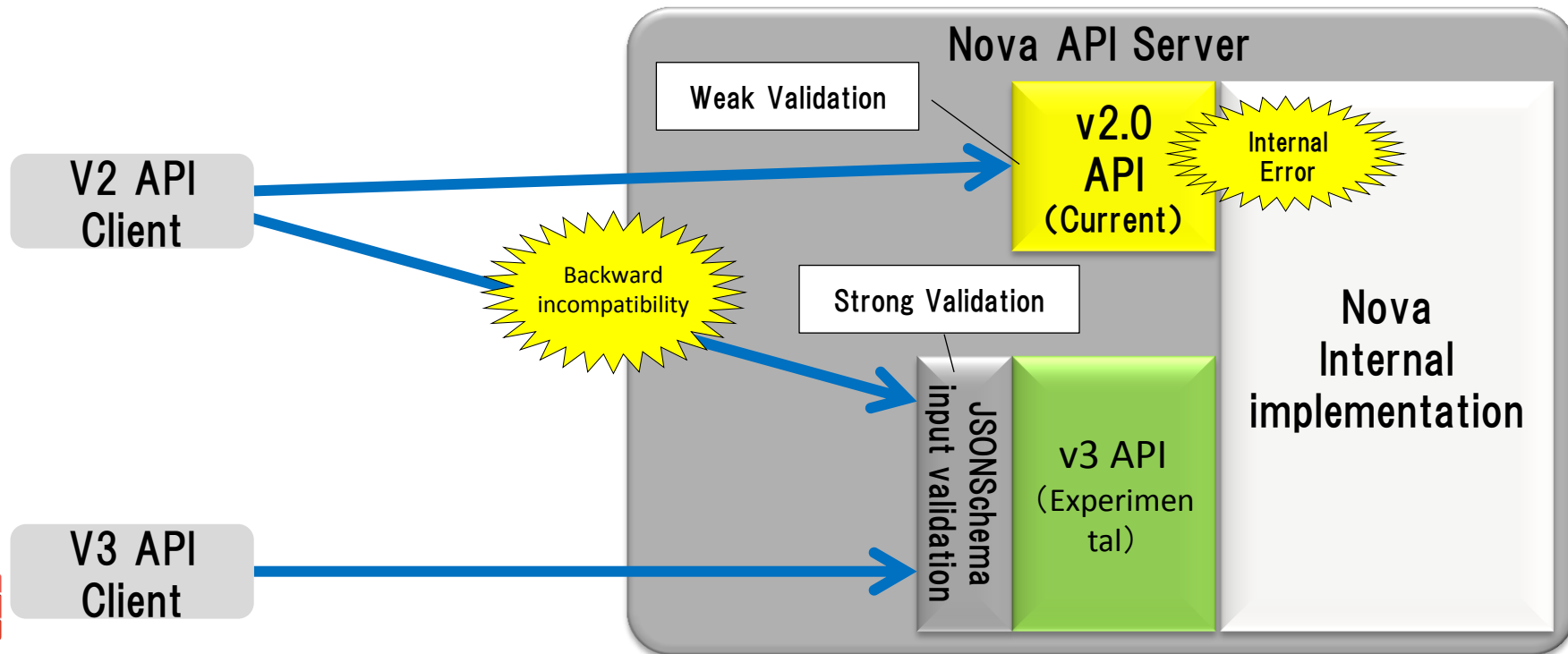
// DEVELOPERS // USERS // OPS

香港 HONG KONG

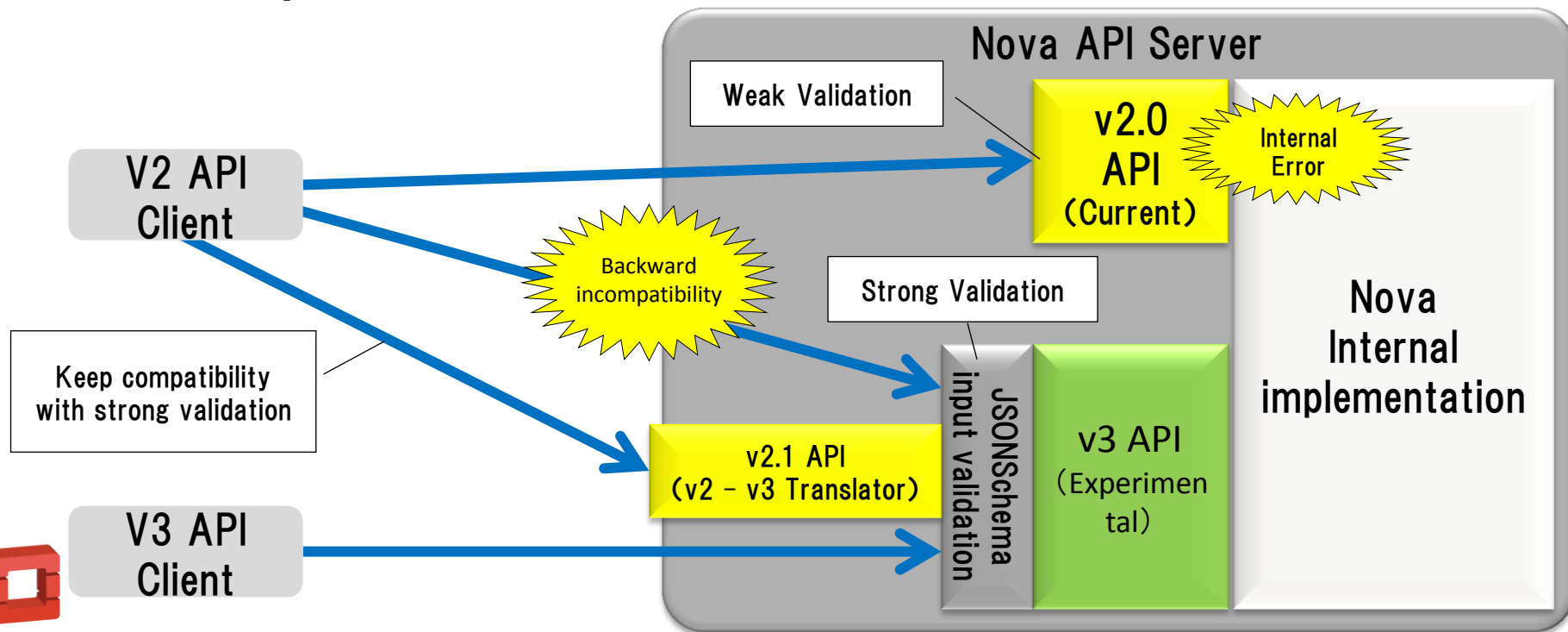


openstack™
CLOUD SOFTWARE

- Stopped v3 API development
 - Backward incompatibility against v2 API
 - Large maintenance cost due to 250+ APIs for each API version
 - Huge mail thread to keep/drop v3 API

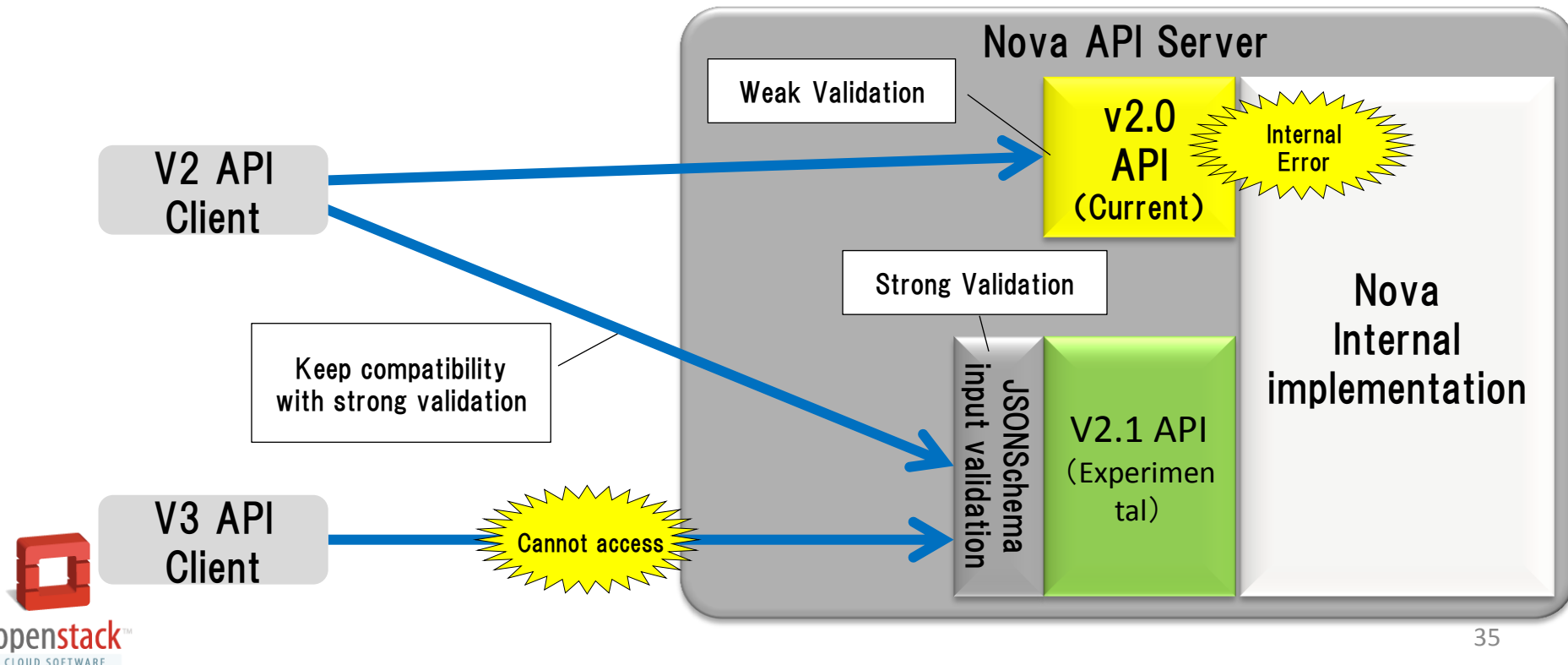


- Proposed v2.1 API for both APIs
 - Translate v2 request/response and use v3 implementation
 - Strong API validation
 - Keep maintenance cost low



- **Problem of v2.1 API for both APIs**
 - Could not be sure the interfaces of v3 API is the best for API consistency
 - Avoid many backwards incompatibilities
 - Need to build a consensus of API consistency for whole OpenStack projects before making backwards incompatible changes

- Implementing v2.1 API for v2 API only
 - Revert (inconsistent) v2 interfaces by removing (consistent) v3 interfaces on v3 code
 - Strong API validation



- **V2.1 API development status**
 - Built a consensus and implementing v2.1 API in Juno cycle
 - But some works still remain and need to implement them in Kilo cycle
 - Hope v2.1 API will be complete in Kilo cycle and the API will be CURRENT(not EXPERIMENTAL)

• Future Works

– Microversioning

- Idea for making consistent interfaces by small steps
- Change interfaces by enough considering on each step and provide old behaviors during the deprecated period
 - Example of microversions (just one idea)
 - 3.0.0: Rename all CamelCase parameters to snake_case
 - 4.0.0: Change status codes of all sync creations to HTTP201
 - 5.0.0: Change status codes of all sync deletions to HTTP204
 - ...
- Clients can choose microversion by writing it on an accept header
 - Example:
Accept: application/vnd.openstack-org.compute-v3.0.0+json
- Still discussing this for the design



- **Future Works**

- **JSON-Home**

- Current API feature discovery way is Nova-specific, not common way
 - Idea for discovering API features with common way
 - OpenStack Identification (Keystone) has already implemented this feature
 - I will implement this feature for v2.1 API



Summary

- **Nova v2.1 API will be available after Kilo**
 - Current v2 compatible interfaces
 - Strong input validation
 - JSON-Home as a common feature discovery (my hope)
- **Nova v3 API has been disappeared**
 - But the part of v3 consistent interfaces will be appeared by microversions (my hope)
- **Join OpenStack community**
 - Code review is important
 - Communication is important





Thanks