ZStack

Architecture Overview

Mission Statement

Help EVERY company build reliable, flexible, and maintainable cloud.

Design Goals:

1. can be sold to more than 100,000 customers.
2. even one machine can create a cloud.
3. as simple as install and run.
4. robust for long-term operation.
5. be the portal of future enterprise software.

Performance Data

VM Creation (by 100 concurrent CreateVm API requests):

VM Number	Time Cost
1	0.51 secs
10	1.55 secs
100	11.33 secs
1000	103 secs
10000	23 minutes

Installation:

It costs 5 minutes using our all-in-one script, or 30 minutes if you want to install it manually (30 minutes include the time you read the document).

Agenda

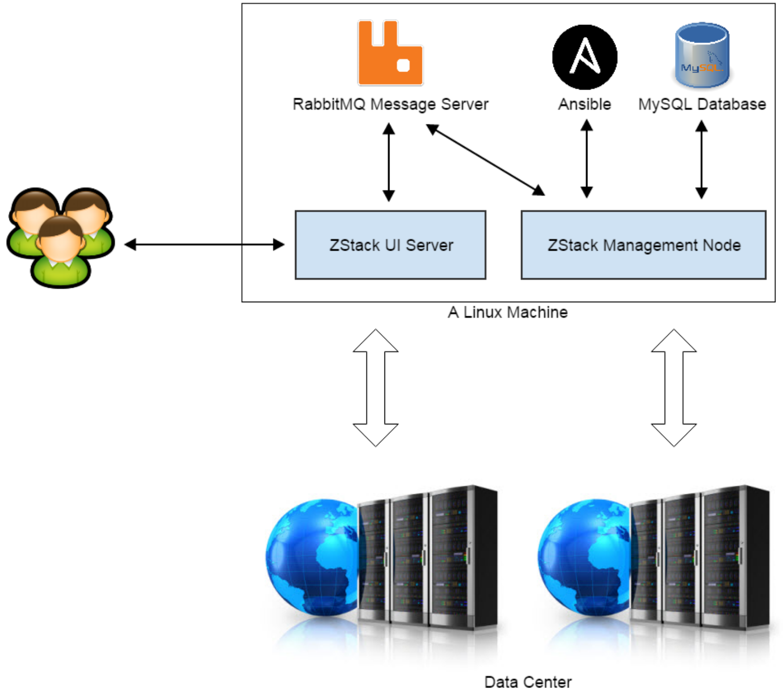
1. Architecture Highlights
2. Deployment Topology
3. Architecture Inside
4. Testing
5. Maturity
6. Q & A
7. Resource Links

Architecture Highlights

1. Huge scalable
    a. asynchronous architecture
    b. stateless services
    c. lock-free architecture
2. Extremely flexible and extendable
    a. in-process microservices
    b. versatile plugin system
    c. workflow engine
    d. tag system
    e. cascade framework
3. Easy to deploy and maintain
    a. full automation by Ansible
    b. comprehensive query API

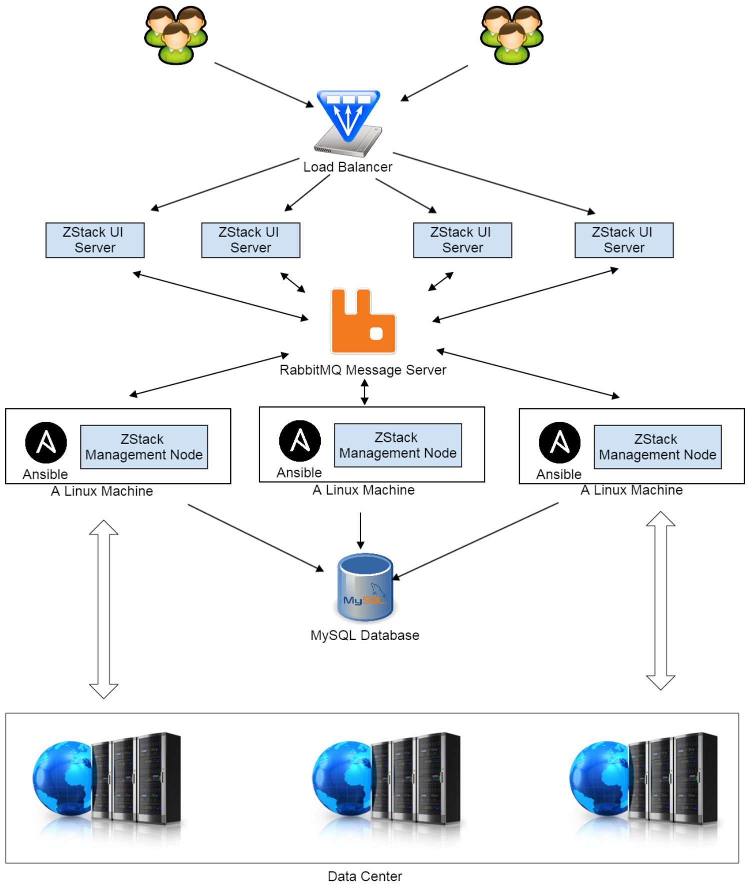
Deployment Topology -- Single Management Node

- All software can be installed on a single Linux machine -- management node
- A single management node is capable of managing millions of VMs and serving tens of thousands of concurrent APIs, as long as the hardware is powerful enough (e.g. powerful CPU, big memory)

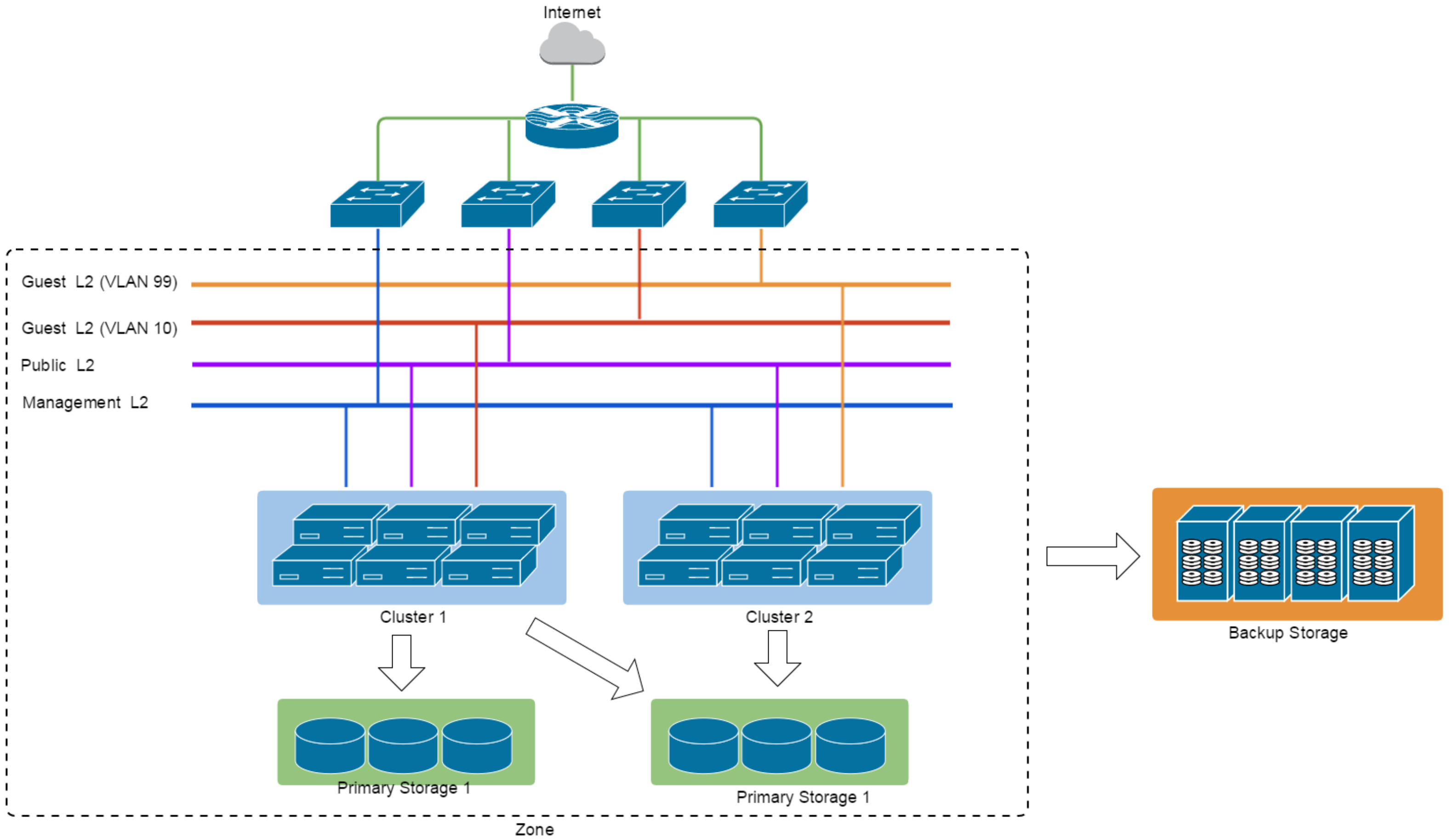


Deployment Topology -- Multi Management Nodes

- Multiple management nodes consist of a management cluster for HA purpose or super large data centers (e.g. millions of physical servers)
- Management nodes are stateless and play equal roles.
- A management node can join, leave, or fail at anytime. Other management nodes will take over automatically.
- The system complexity of a management cluster of one hundred nodes is almost the same to a cluster of two nodes, because nodes are stateless.



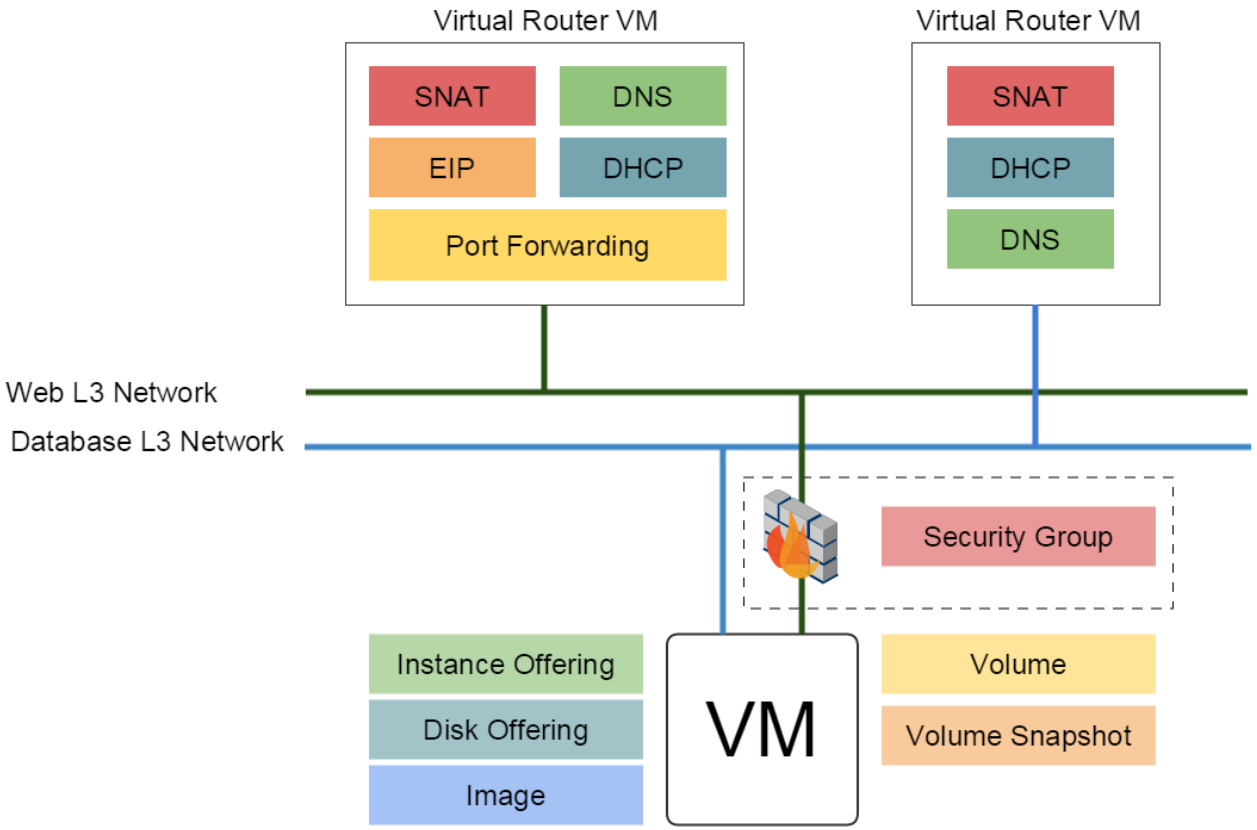
Deployment Topology: Administrator View



Deployment topology: admin view

- A cluster is a logic group of hosts
- A primary storage can be attached to multiple clusters.
- A L2 network can be attached to multiple clusters.
- A zone is a logic group of all resources.
- A backup storage can be attached to multiple zones.

Deployment Topology: User View

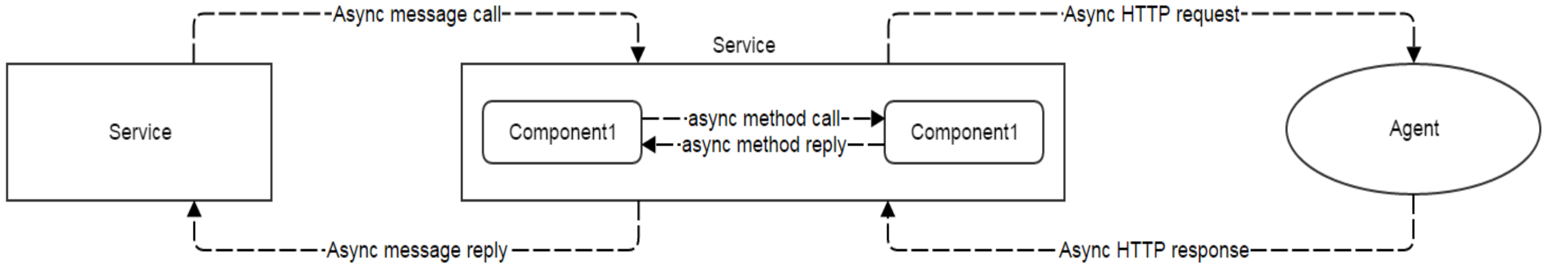


Deployment topology: user view

- A VM can have multiple L3 networks and volumes.
- Every VM nic can join security groups.
- Every L3 network has a dedicated networking node -- virtual router VM, which provides network services.

Architecture Inside -- Huge Scalable

Asynchronous Architecture

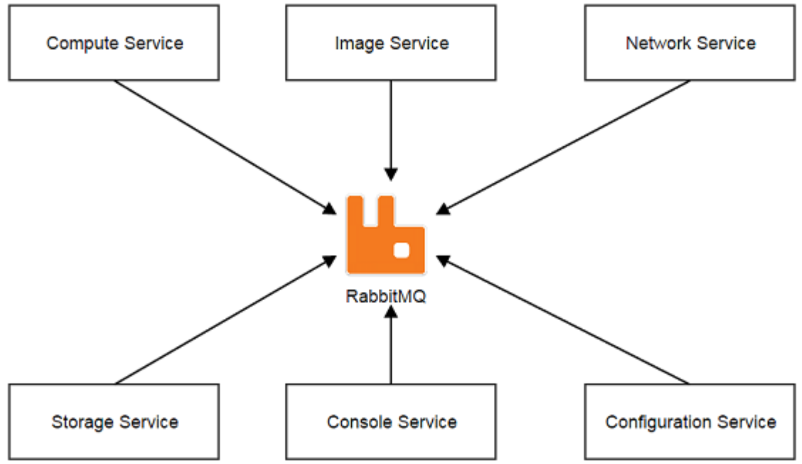


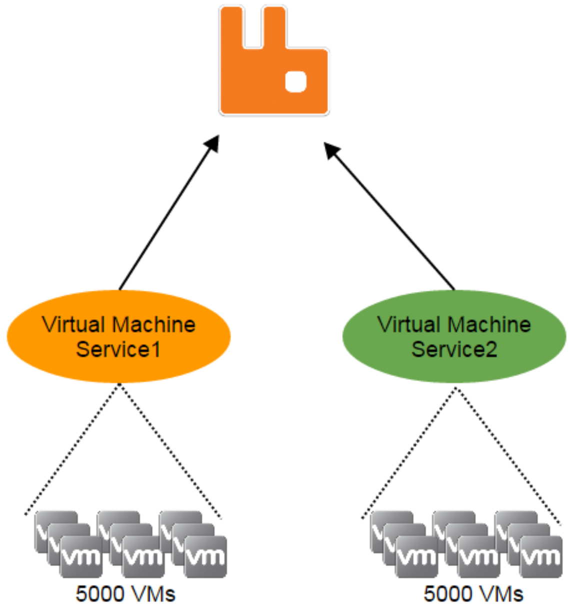
A single ZStack management node can:

1. serve tens of thousands of concurrent API requests
2. manage tens of thousands even hundreds of thousands of physical servers
3. manage millions of VMs

Architecture Inside -- Huge Scalable

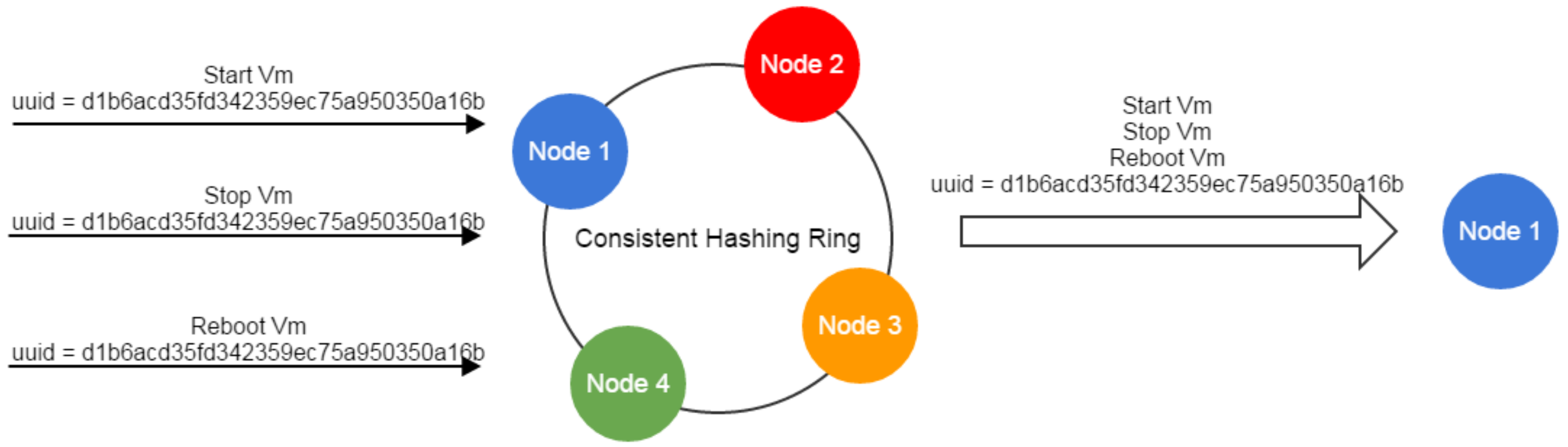
Stateless Services





Architecture Inside -- Huge Scalable

Stateless Services

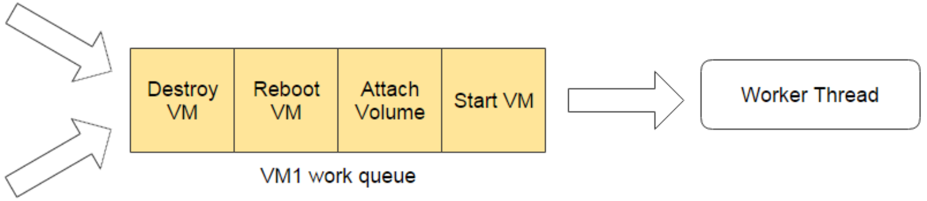

Start Vm
uuid = d1b6acd35fd342359ec75a950350a16b
Stop Vm
uuid = d1b6acd35fd342359ec75a950350a16b
Reboot Vm
uuid = d1b6acd35fd342359ec75a950350a16b
Node 1
Node 2
Node 3
Node 4
Consistent Hashing Ring
Start Vm
Stop Vm
Reboot Vm
uuid = d1b6acd35fd342359ec75a950350a16b
Node 1

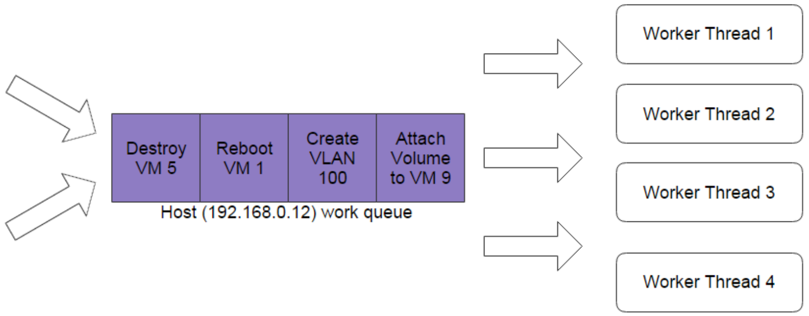
Administrators can deploy management nodes as many as they want. The consistent hashing ring guarantees messages to a resource are always routed to the same service instance.

Architecture Inside -- Huge Scalable

Lock-free Architecture

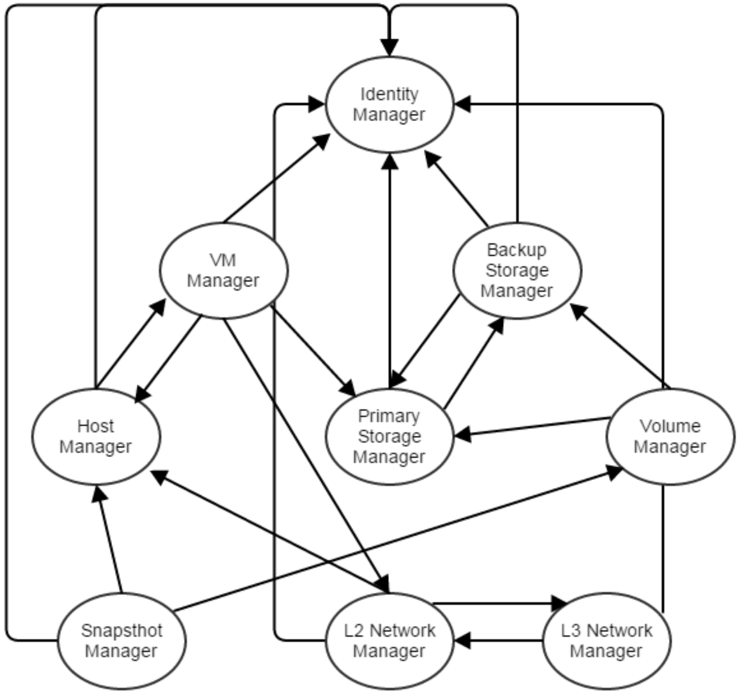
- No lock in ZStack business logic, concurrency and synchronization are controlled by queues.
- Requests to resources can be throttled. For example, you can configure a host to serve maximum 10 concurrent requests, and the requests are queued.



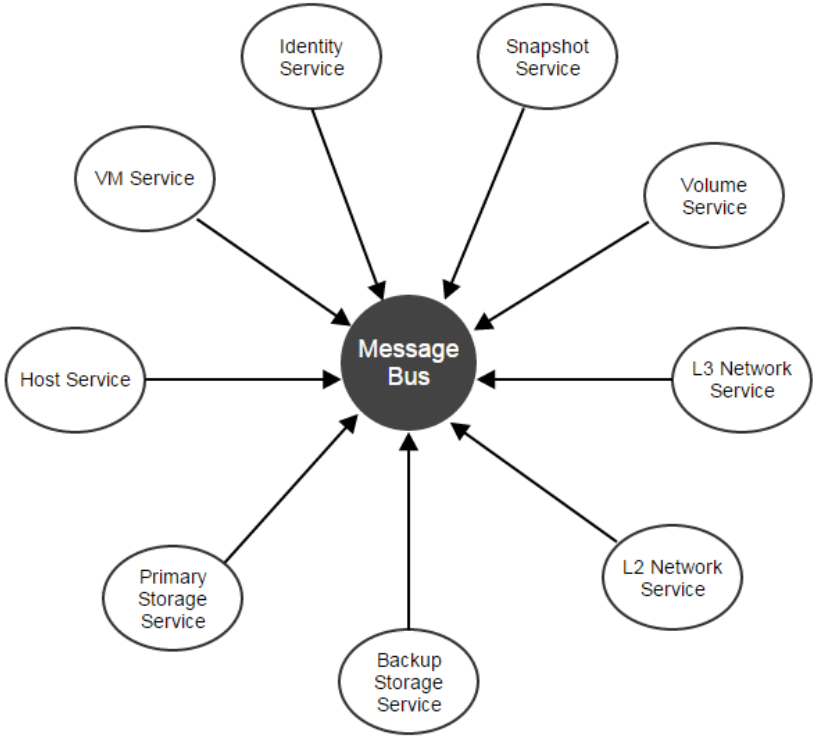


Architecture Inside -- Extremely flexible and extendable

In-process microservices



The organic growth of monolithic IaaS software



Microservices decoupling

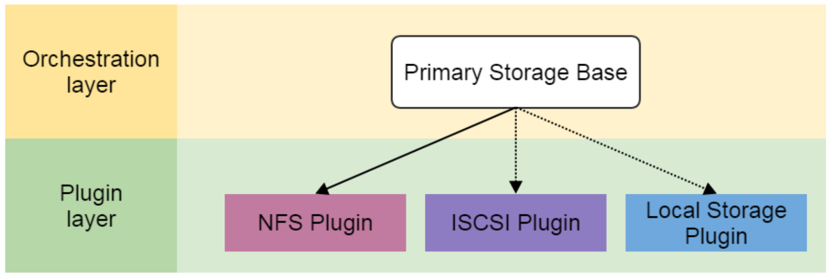
Architecture Inside -- Extremely flexible and extendable

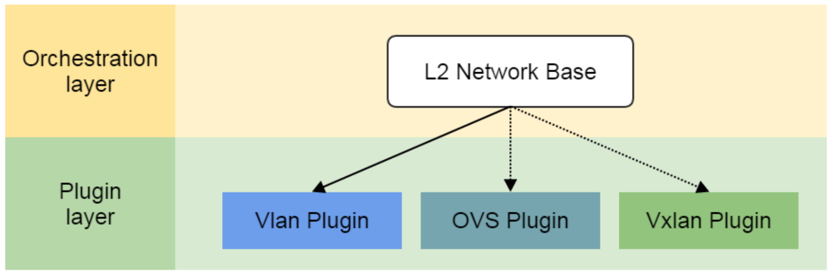
In-process microservices

- Services are independent and communicate through the message bus.
- Services are enclosed in the same process that acts as a container called a management node.
- Services in a management node may communicate with services in other nodes without knowing that. The destination service is selected by the consistent hashing ring mentioned in chapter stateless services.

Architecture Inside -- Extremely flexible and extendable

Versatile Plugin System



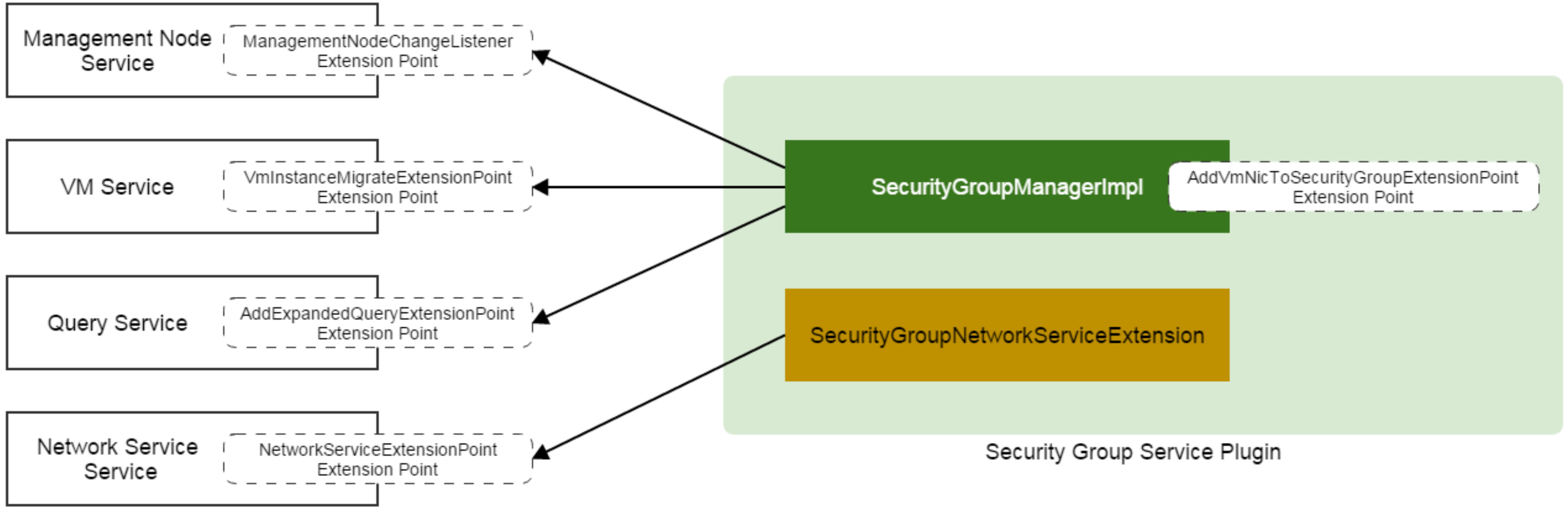


EVERY resource can be sub-typed:

- new sub-typed resource, for example, ISCSI plugin, has ZERO impact to the orchestration and sibling plugins.
- every sub-typed resource is a self-contained plugin.
- a sub-typed resource plugin can be added or removed anytime.

Architecture Inside -- Extremely flexible and extendable

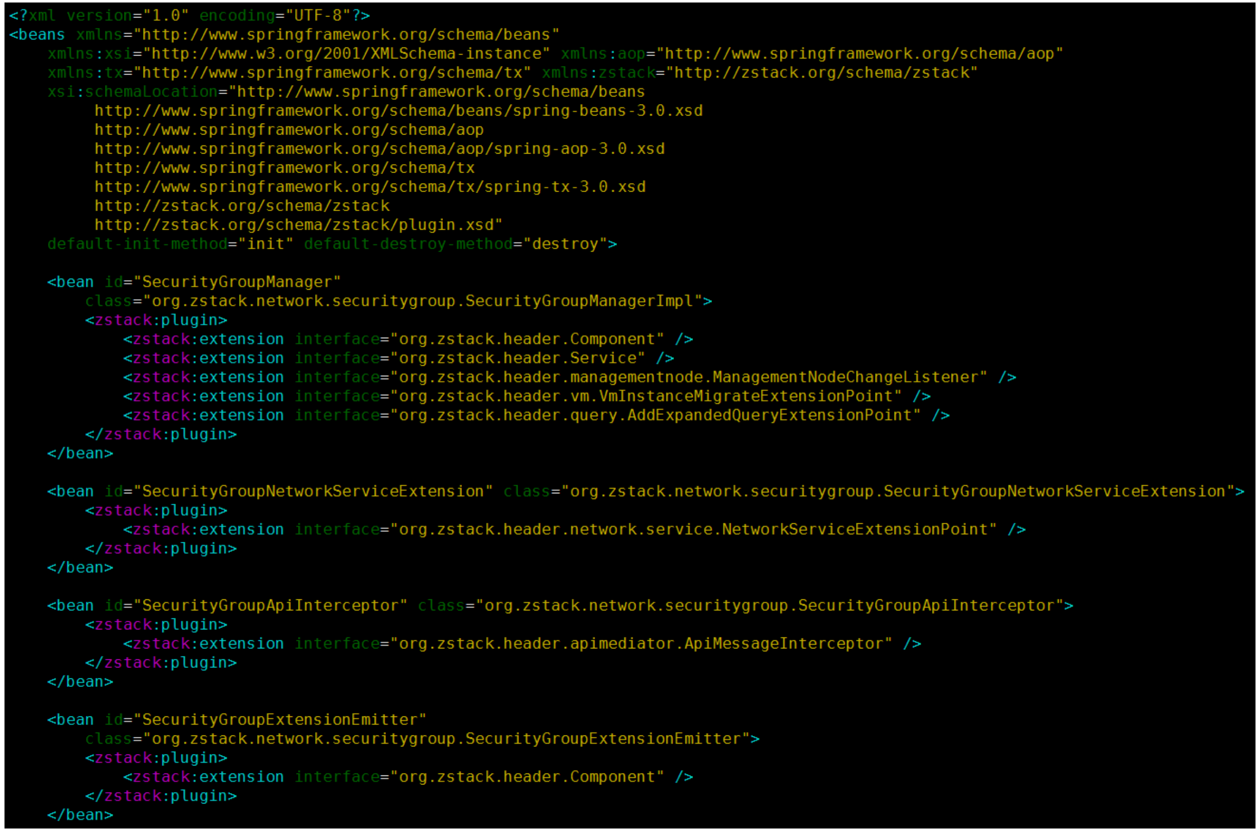
Versatile Plugin System


Management Node Service
ManagementNodeChangeListener Extension Point
VM Service
VmInstanceMigrateExtensionPoint Extension Point
Query Service
AddExpandedQueryExtensionPoint Extension Point
Network Service Service
NetworkServiceExtensionPoint Extension Point
SecurityGroupManagerImpl
AddVmNicToSecurityGroupExtensionPoint Extension Point
SecurityGroupNetworkServiceExtension
Security Group Service Plugin

Every plugin is just a JAR file plus configuration files. Adding or removing plugins will not impact the system, which guarantees the stability of the software.

Architecture Inside -- Extremely flexible and extendable

Versatile Plugin System

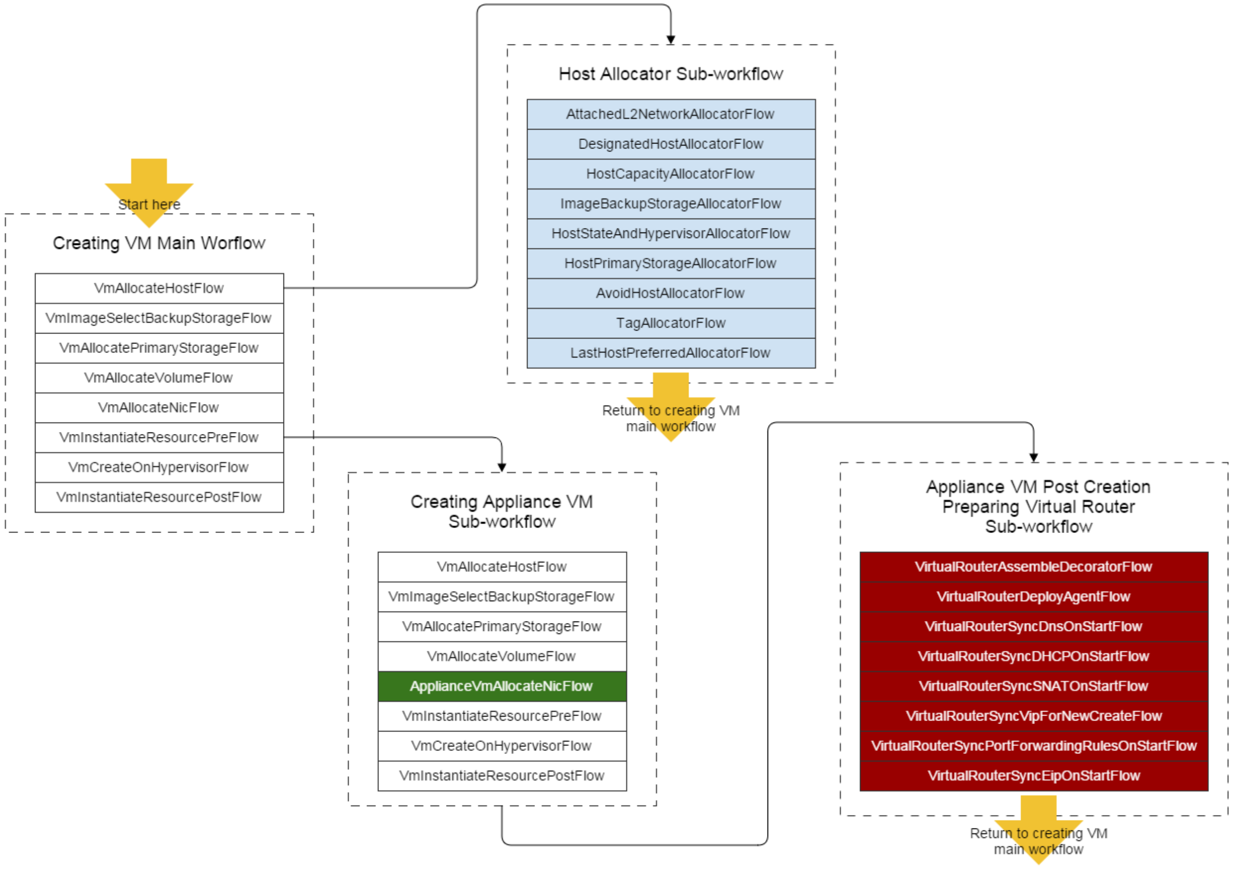


```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx" xmlns:zstack="http://zstack.org/schema/zstack"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
         http://www.springframework.org/schema/aop
         http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
         http://www.springframework.org/schema/tx
         http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
         http://zstack.org/schema/zstack
         http://zstack.org/schema/zstack/plugin.xsd"
    default-init-method="init" default-destroy-method="destroy">

    <bean id="SecurityGroupManager"
        class="org.zstack.network.securitygroup.SecurityGroupManagerImpl">
        <zstack:plugin>
            <zstack:extension interface="org.zstack.header.Component" />
            <zstack:extension interface="org.zstack.header.Service" />
            <zstack:extension interface="org.zstack.header.managementnode.ManagementNodeChangeListener" />
            <zstack:extension interface="org.zstack.header.vm.VmInstanceMigrateExtensionPoint" />
            <zstack:extension interface="org.zstack.header.query.AddExpandedQueryExtensionPoint" />
        </zstack:plugin>
    </bean>

    <bean id="SecurityGroupNetworkServiceExtension" class="org.zstack.network.securitygroup.SecurityGroupNetworkServiceExtension">
        <zstack:plugin>
            <zstack:extension interface="org.zstack.header.network.service.NetworkServiceExtensionPoint" />
        </zstack:plugin>
    </bean>

    <bean id="SecurityGroupApiInterceptor" class="org.zstack.network.securitygroup.SecurityGroupApiInterceptor">
        <zstack:plugin>
            <zstack:extension interface="org.zstack.header.apimediator.ApiMessageInterceptor" />
        </zstack:plugin>
    </bean>

    <bean id="SecurityGroupExtensionEmitter"
        class="org.zstack.network.securitygroup.SecurityGroupExtensionEmitter">
        <zstack:plugin>
            <zstack:extension interface="org.zstack.header.Component" />
        </zstack:plugin>
    </bean>
```

Architecture Inside -- Extremely flexible and extendable

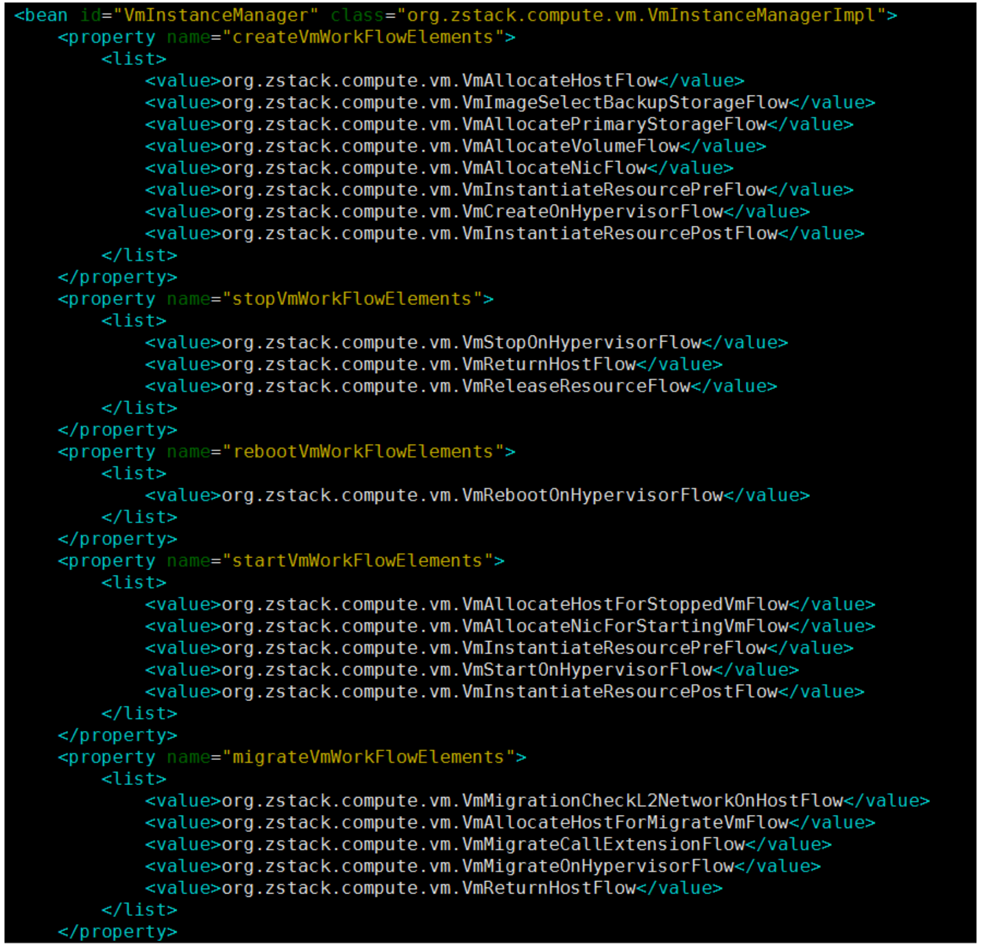
Workflow Engine

- Flows can be configured either by XML files or programmatic ways.
- Every flow can be rolled back on errors.
- A workflow can contain sub-flow to decouple the business logic further.


Start here
Creating VM Main Worflow
VmAllocateHostFlow
VmImageSelectBackupStorageFlow
VmAllocatePrimaryStorageFlow
VmAllocateVolumeFlow
VmAllocateNicFlow
VmInstantiateResourcePreFlow
VmCreateOnHypervisorFlow
VmInstantiateResourcePostFlow
Host Allocator Sub-workflow
AttachedL2NetworkAllocatorFlow
DesignatedHostAllocatorFlow
HostCapacityAllocatorFlow
ImageBackupStorageAllocatorFlow
HostStateAndHypervisorAllocatorFlow
HostPrimaryStorageAllocatorFlow
AvoidHostAllocatorFlow
TagAllocatorFlow
LastHostPreferredAllocatorFlow
Return to creating VM main workflow
Creating Appliance VM Sub-workflow
VmAllocateHostFlow
VmImageSelectBackupStorageFlow
VmAllocatePrimaryStorageFlow
VmAllocateVolumeFlow
ApplianceVmAllocateNicFlow
VmInstantiateResourcePreFlow
VmCreateOnHypervisorFlow
VmInstantiateResourcePostFlow
Appliance VM Post Creation Preparing Virtual Router Sub-workflow
VirtualRouterAssembleDecoratorFlow
VirtualRouterDeployAgentFlow
VirtualRouterSyncDnsOnStartFlow
VirtualRouterSyncDHCPOnStartFlow
VirtualRouterSyncSNATOnStartFlow
VirtualRouterSyncVipForNewCreateFlow
VirtualRouterSyncPortForwardingRulesOnStartFlow
VirtualRouterSyncEipOnStartFlow
Return to creating VM main workflow

Architecture Inside -- Extremely flexible and extendable

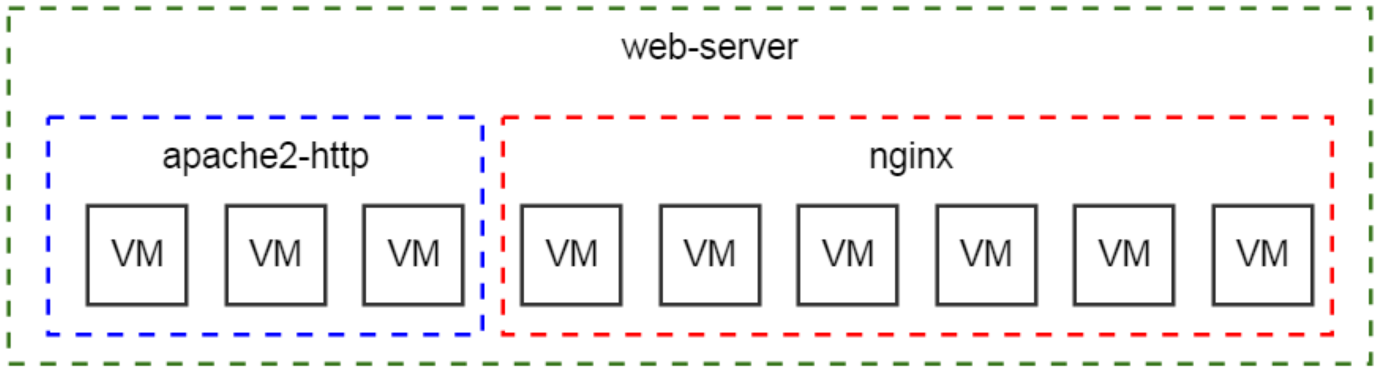
Workflow Engine



```
<bean id="VmInstanceManager" class="org.zstack.compute.vm.VmInstanceManagerImpl">
    <property name="createVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmAllocateHostFlow</value>
            <value>org.zstack.compute.vm.VmImageSelectBackupStorageFlow</value>
            <value>org.zstack.compute.vm.VmAllocatePrimaryStorageFlow</value>
            <value>org.zstack.compute.vm.VmAllocateVolumeFlow</value>
            <value>org.zstack.compute.vm.VmAllocateNicFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePreFlow</value>
            <value>org.zstack.compute.vm.VmCreateOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePostFlow</value>
        </list>
    </property>
    <property name="stopVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmStopOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmReturnHostFlow</value>
            <value>org.zstack.compute.vm.VmReleaseResourceFlow</value>
        </list>
    </property>
    <property name="rebootVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmRebootOnHypervisorFlow</value>
        </list>
    </property>
    <property name="startVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmAllocateHostForStoppedVmFlow</value>
            <value>org.zstack.compute.vm.VmAllocateNicForStartingVmFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePreFlow</value>
            <value>org.zstack.compute.vm.VmStartOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePostFlow</value>
        </list>
    </property>
    <property name="migrateVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmMigrationCheckL2NetworkOnHostFlow</value>
            <value>org.zstack.compute.vm.VmAllocateHostForMigrateVmFlow</value>
            <value>org.zstack.compute.vm.VmMigrateCallExtensionFlow</value>
            <value>org.zstack.compute.vm.VmMigrateOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmReturnHostFlow</value>
        </list>
    </property>
```

Architecture Inside -- Extremely flexible and extendable

Tag System



User tags can:

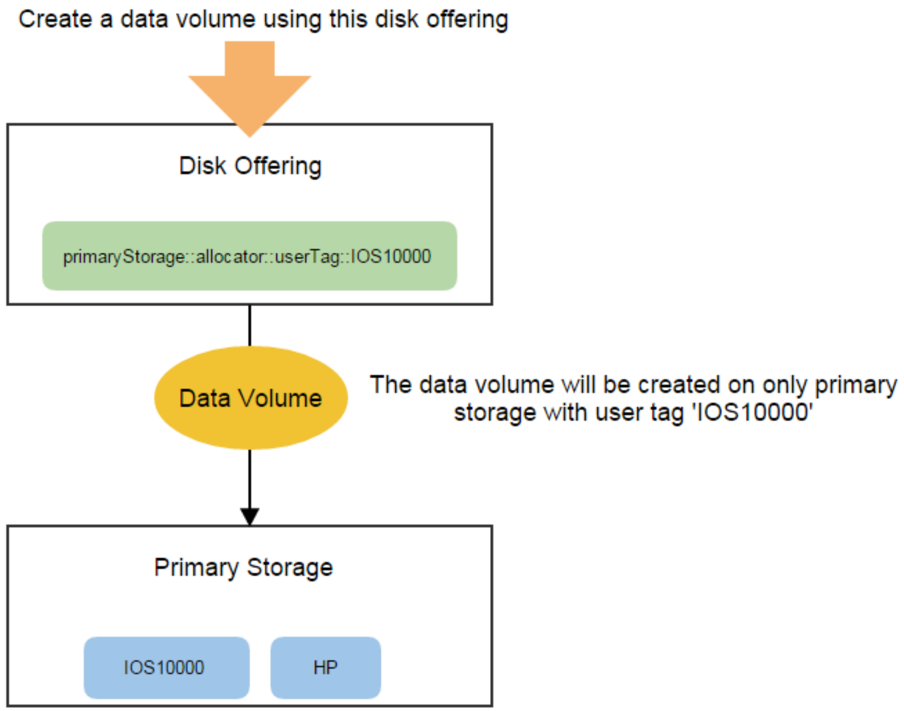
- help user to group their resources.
- help user search resources with specific tags.
- cooperate with system tags for specific business logic.

Architecture Inside -- Extremely flexible and extendable

Tag System

System tags can:

- cooperate with plugins to change system behaviors.
- add new properties to resources without changing their database schema.

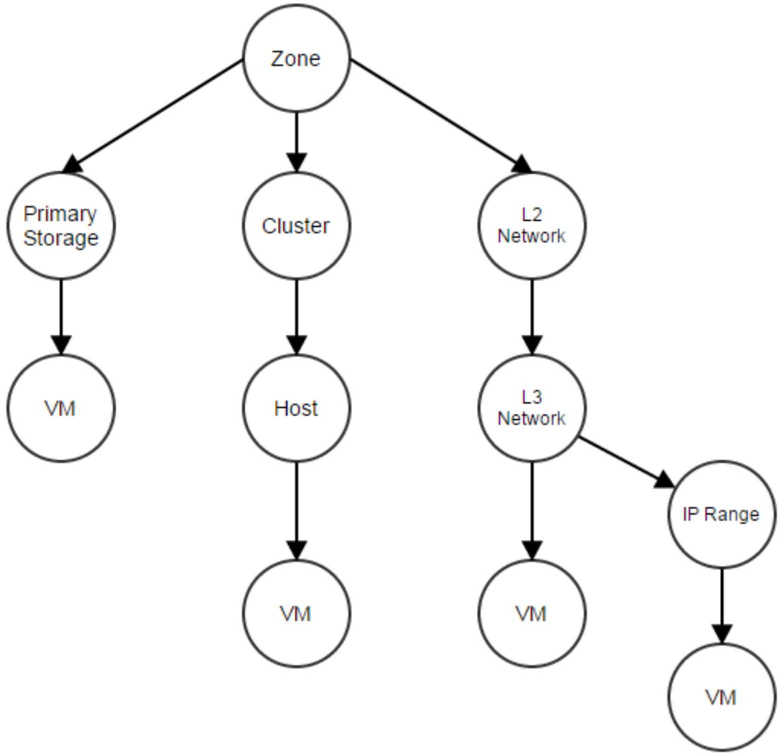

Create a data volume using this disk offering
Disk Offering
primaryStorage::allocator::userTag::IOS10000
Data Volume
The data volume will be created on only primary storage with user tag 'IOS10000'
Primary Storage
IOS10000
HP

Architecture Inside -- Extremely flexible and extendable

Cascade Framework

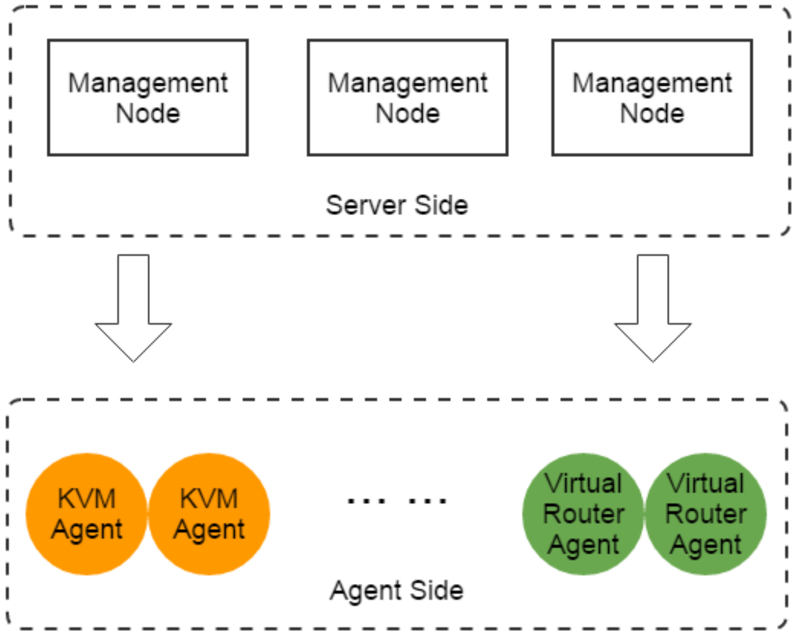
Cascade Framework can help:

- spread an operation from one resource to other resources. For example, spread deleting operation from a zone to all descendant resources.
- resources can choose to join cascade framework by writing a small plugin.
- joining or quitting cascade framework won't affect other resources



Architecture Inside -- Easy to deploy and maintain

Full Automation By Ansible



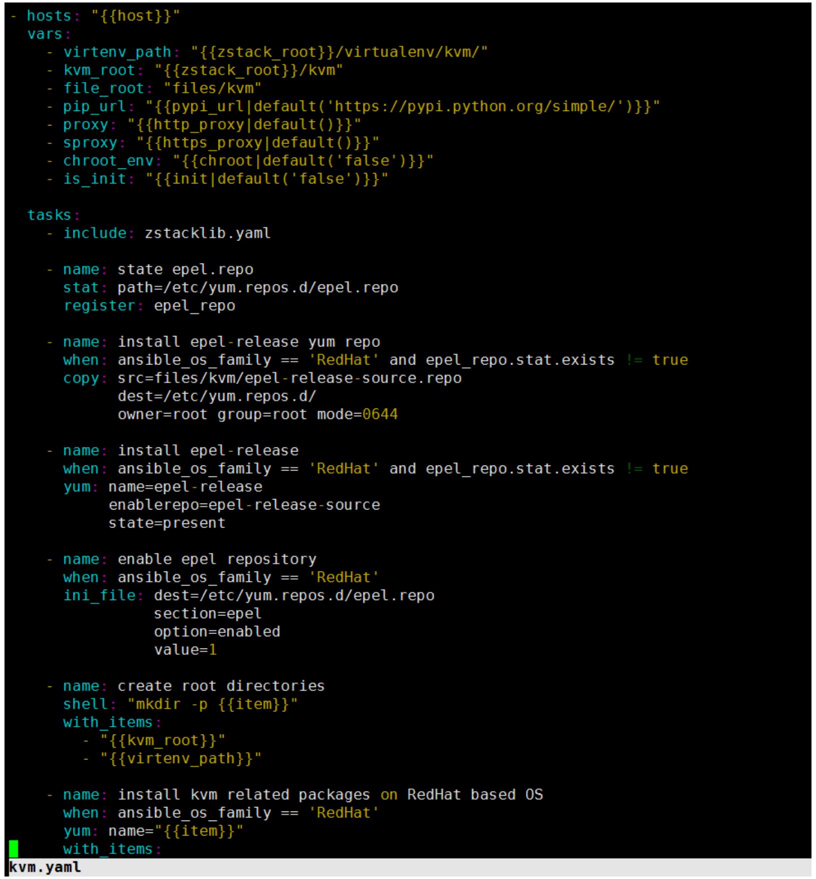
ZStack uses typical server-agent model that several agents need to be deployed on remote machines.

Architecture Inside -- Easy to deploy and maintain

Full Automation By Ansible

Seamlessly integrated with Ansible:

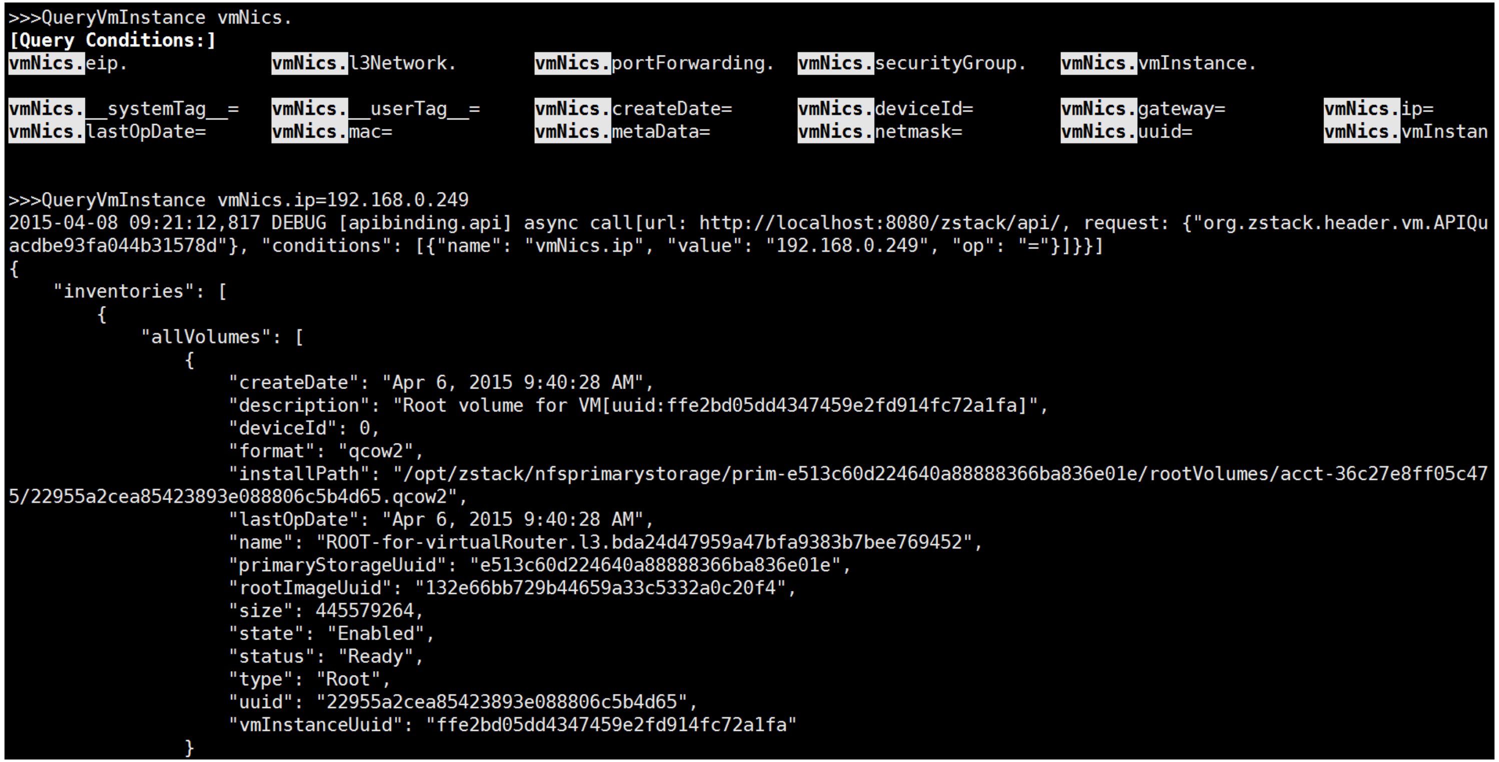
- Ansible is agent-less, SSH based.
- Administrators may not even notice the existence of agents.
- Advanced administrators can extend ZStack's Ansible YAML file to configure remote systems; for example, apply a critical security fix on all KVM hosts.
- Administrators only need to install the minimal system. Ansible takes care of all dependencies.



```
- hosts: "{{host}}"
  vars:
    - virtenv_path: "{{zstack_root}}/virtualenv/kvm/"
    - kvm_root: "{{zstack_root}}/kvm"
    - file_root: "files/kvm"
    - pip_url: "{{pypi_url|default('https://pypi.python.org/simple/')}}"
    - proxy: "{{http_proxy|default()}}"
    - sproxy: "{{https_proxy|default()}}"
    - chroot_env: "{{chroot|default('false')}}"
    - is_init: "{{init|default('false')}}"

  tasks:
    - include: zstacklib.yaml

    - name: state epel.repo
      stat: path=/etc/yum.repos.d/epel.repo
      register: epel_repo

    - name: install epel-release yum repo
      when: ansible_os_family == 'RedHat' and epel_repo.stat.exists != true
      copy: src=files/kvm/epel-release-source.repo
            dest=/etc/yum.repos.d/
            owner=root group=root mode=0644

    - name: install epel-release
      when: ansible_os_family == 'RedHat' and epel_repo.stat.exists != true
      yum: name=epel-release
           enablerepo=epel-release-source
           state=present

    - name: enable epel repository
      when: ansible_os_family == 'RedHat'
      ini_file: dest=/etc/yum.repos.d/epel.repo
                section=epel
                option=enabled
                value=1

    - name: create root directories
      shell: "mkdir -p {{item}}"
      with_items:
        - "{{kvm_root}}"
        - "{{virtenv_path}}"

    - name: install kvm related packages on RedHat based OS
      when: ansible_os_family == 'RedHat'
      yum: name="{{item}}"
      with_items:
kvm.yaml
```

Architecture Inside -- Easy to deploy and maintain

Comprehensive Query API



```
>>>QueryVmInstance vmNics.
[Query Conditions:]
vmNics.eip.             vmNics.l3Network.       vmNics.portForwarding.  vmNics.securityGroup.   vmNics.vmInstance.

vmNics.__systemTag__=   vmNics.__userTag__=     vmNics.createDate=      vmNics.deviceId=        vmNics.gateway=         vmNics.ip=
vmNics.lastOpDate=      vmNics.mac=             vmNics.metaData=        vmNics.netmask=         vmNics.uuid=            vmNics.vmInstan


>>>QueryVmInstance vmNics.ip=192.168.0.249
2015-04-08 09:21:12,817 DEBUG [apibinding.api] async call[url: http://localhost:8080/zstack/api/, request: {"org.zstack.header.vm.APIQu
acdbe93fa044b31578d"}, "conditions": [{"name": "vmNics.ip", "value": "192.168.0.249", "op": "="}]}}]
{
    "inventories": [
        {
            "allVolumes": [
                {
                    "createDate": "Apr 6, 2015 9:40:28 AM",
                    "description": "Root volume for VM[uuid:ffe2bd05dd4347459e2fd914fc72a1fa]",
                    "deviceId": 0,
                    "format": "qcow2",
                    "installPath": "/opt/zstack/nfsprimarystorage/prim-e513c60d224640a88888366ba836e01e/rootVolumes/acct-36c27e8ff05c47
5/22955a2cea85423893e088806c5b4d65.qcow2",
                    "lastOpDate": "Apr 6, 2015 9:40:28 AM",
                    "name": "ROOT-for-virtualRouter.l3.bda24d47959a47bfa9383b7bee769452",
                    "primaryStorageUuid": "e513c60d224640a88888366ba836e01e",
                    "rootImageUuid": "132e66bb729b44659a33c5332a0c20f4",
                    "size": 445579264,
                    "state": "Enabled",
                    "status": "Ready",
                    "type": "Root",
                    "uuid": "22955a2cea85423893e088806c5b4d65",
                    "vmInstanceUuid": "ffe2bd05dd4347459e2fd914fc72a1fa"
                }
```

Architecture Inside -- Easy to deploy and maintain

Comprehensive Query API

- every field of every resource can be queried.
- about 4 million query conditions, countless query combinations.
- except defining query API, developers doesn’t need to write any code. All queries are automatically generated by ZStack.
- can be used to create sophisticated UI. For example, create a UI view only showing VMs running on hosts with user tag ‘high-performance-server’.

Testing

- 643 integration testing cases based on simulator.
- 199 system testing cases based on real environment.
- 8 model-based testing cases that execute API in a random manner.

Time Cost:

Integration Testing	System Testing	Model-Based Testing
~ 10 hours	~ 3.5 hours	depending on success conditions, they can run several days.

Every ZStack component is rigorously tested; developers can write code with confidence.

Maturity

- The core orchestration is completed and stable.
- Ready for the private cloud deployment.
- Still lack some features for the public cloud.
- Both UI and command line tool are done.

The whole project consists of ~ 220K Java code , ~ 70K Python code, and ~40K Javascript code (UI).

Computing	Zone Cluster Host Hypervisor: KVM
Storage	Primary Storage: NFS Backup Storage: SFTP Volume Volume Snapshot Images
Networking	L2 network: non vlan, vlan L3 network Network services: DHCP, DNS, SNAT, EIP, Port Forwarding, Security Group
Others	Instance Offering Disk Offering Global Configurations Tags Multi Management Nodes Query API

Q & A

Resource Links

website:

http://zstack.org/