



**Hewlett Packard**  
Enterprise

# **Helion OpenStack Carrier Grade 4.0**

## **TENANT USER'S GUIDE**

## Copyright Notice

© Copyright 2016 Hewlett Packard Enterprise Development LP

The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Links to third-party websites take you outside the Hewlett Packard Enterprise website. Hewlett Packard Enterprise has no control over and is not responsible for information outside the Hewlett Packard Enterprise website.

## Acknowledgements

Java® and Oracle® are registered trademarks of Oracle and/or its affiliates

<http://www.hpe.com/info/storagewarranty>

*Titanium Server*

*Cloud Administration, 16.10*

# Contents

<b>1 Overview .....</b>	<b>1</b>
Tenant User's Guide Overview .....	1
<b>2 Instance Management .....</b>	<b>3</b>
Managing Virtual Machines .....	3
The Virtual Machine Scheduler .....	4
Live Migration of Virtual Machines .....	6
Host Aggregates .....	7
Server Groups .....	9
Configuring Instances at Boot Time .....	11
Selecting Virtual Machine Flavors .....	13
Configuring PCI Addresses for Guest VNICs .....	14
Accessing the Metadata Server .....	14
Launching Virtual Machine Instances .....	15
Launching Virtual Machine Instances Using the CLI .....	17
Connecting Virtual Machines to External Networks .....	18
Scaling Virtual Machine Resources .....	19
The App Scale Helper Script .....	20
CPU Scaling .....	20
<b>3 Access and Security .....</b>	<b>25</b>
Access and Security for Tenant Network Users .....	25
Security Groups .....	26
<b>4 Tenant Network Management .....</b>	<b>29</b>
Tenant Networks .....	29
VLAN Transparent .....	31
Guest VLANs .....	33
Creating Tenant Networks .....	34
Setting Up Tenant Networks Using the CLI .....	35
<b>5 Volume and Image Management .....</b>	<b>39</b>
Volume and Image Management Overview .....	39
Creating Cached RAW Boot Images .....	39
Creating Cinder Volumes for Boot Images .....	40
Cinder Volume Backups .....	41

Backing Up Available Cinder Volumes .....	42
Backing Up In-Use Cinder Volumes .....	43
<b>6 Virtual Router Management .....</b>	<b>47</b>
Virtual Routers .....	47
Managed and Unmanaged Subnets .....	48
DiffServ-Based Priority Queuing and Scheduling .....	50
Quality of Service Policies .....	50
Virtual Router Administration .....	52
Adding Virtual Routers .....	53
Adding Virtual Routers Using the CLI .....	54
Adding a Gateway to a Virtual Router .....	54
Configuring SNAT on a Virtual Router .....	56
Configuring Port-based DNAT on a Virtual Router .....	57
Configuring Port-based DNAT Using the CLI .....	59
Adding and Maintaining Routes for a VXLAN Network .....	60
Address Filtering on Virtual Interfaces .....	61
<b>7 Stack Management .....</b>	<b>63</b>
Managing Stacks .....	63
Stacks .....	63
Templates .....	64
Parameters .....	65
Titanium Server Extensions to Heat .....	65
Launching a Stack .....	68
Customizing Images with User Data In a Heat Template .....	70
Resource Scaling (Autoscaling) .....	71
Sample Templates for Titanium Server .....	71
Supported Heat Resource Types .....	78
Further Reading .....	79
<b>8 Swift Object Storage Management .....</b>	<b>81</b>
Swift Object Storage .....	81
VM Configuration for Access to Swift Object Storage .....	82
Swift Upload File Size Considerations .....	82

# 1

## Overview

### Tenant User's Guide Overview

This guide is intended for non-admin users of tenants (or projects).

It provides information about the Titanium Server features and capabilities that non-admin tenant users have access to.

Titanium Server extended features and capabilities that can be accessed and configured by non-admin tenant users include:

- Instance management
- Access and security
- Tenant network features
- Volume and image management
- Virtual router management
- Stack management

Note that this guide only discusses areas that have been extended by Titanium Server; for full non-admin tenant users capabilities, refer to open-source OpenStack documentation.



# 2

## *Instance Management*

Managing Virtual Machines	3
The Virtual Machine Scheduler	4
Live Migration of Virtual Machines	6
Host Aggregates	7
Server Groups	9
Configuring Instances at Boot Time	11
Selecting Virtual Machine Flavors	13
Configuring PCI Addresses for Guest VNICS	14
Accessing the Metadata Server	14
Launching Virtual Machine Instances	15
Connecting Virtual Machines to External Networks	18
Scaling Virtual Machine Resources	19

### **Managing Virtual Machines**

Virtual machines and the applications that run on them are at the core of cloud-based communications solutions. Understanding how to manage them is of paramount importance.

This chapter elaborates on software features available exclusively on Titanium Server, which add to the already powerful set of management tools available from OpenStack. The exclusive features include virtual machine settings to optimize the use of virtual CPUs, improvements to the scheduling algorithms, better integration of NUMA architectures, and others.

## The Virtual Machine Scheduler

The virtual machine scheduler in Titanium Server is a modified version of the OpenStack Nova scheduler. It allows for better placement of virtual machines in order to exploit hardware and network resources, and for a better distribution of the system workload overall.

In addition to the scheduler actions taken based upon the virtual machine flavor in use, as discussed in *Virtual Machine Flavors*, the following are enhancements integrated into the Titanium Server Nova scheduler:

### Memory over-commit policy

There is no support for memory over-commit when scheduling virtual machines. When launching a virtual machine, the Nova scheduler reserves the requested memory block from the system in its entirety. The launch operation fails if not enough memory can be allocated from the compute cluster.

### CPU and memory platform utilization awareness

If platform CPU or memory usage on a compute host exceeds 80% of the platform allotment, new VMs are not scheduled on the host. This helps ensure that the platform services can support the added VM load. The threshold of 80% is not user-configurable.

### Network load balancing across processor nodes

When scheduling threads for new virtual CPUs, the Nova scheduler selects the target core taking into account the load average of the running AVS cores on each processor node. In a common scenario, the AVS is deployed on two cores, not necessarily part of the same processor. When a new virtual machine is launched, the Nova scheduler looks at the average load incurred by AVS cores on each processor, and then selects the processor with the lighter load as the target for the new virtual CPUs.

This scheduling approach aims at balancing the switching load on the AVS across virtual machines as they are deployed, and minimizing the cross-NUMA inefficiencies for AVS switching.

### Provider networks access verification

When scheduling a virtual machine, the Nova scheduler verifies that the target compute node has data interfaces on all needed provider networks, as determined by the list of tenant networks the virtual NICs are attached to. This verification helps prevent unnecessary debugging steps incurred when the networking services on a guest application fail to operate due to the lack of proper network access.

### NUMA node pinning

Instances with certain extra specifications or image properties are scheduled with respect to NUMA nodes as well as compute nodes. In effect, each virtual NUMA node is mapped to a corresponding host NUMA node. To support instantiation, the host must be able to offer a set of NUMA nodes that satisfies the NUMA node topology and requirements of the instance.

The following extra specs or image properties cause the instance to be scheduled with respect to NUMA node availability:



- **CPU Policy**
- **Number of NUMA Nodes**
- **NUMA Node**
- **Vswitch NUMA Affinity**
- **Memory Page Size**
- **PCI Alias**

### **The Nova Scheduler and Server Group Policies**

Virtual machines launched as part of a Server Group are subject to scheduling actions determined by the selection of scheduling policy. See [Server Groups](#) on page 9 for details.

#### **Affinity policy**

The goal of this policy is to schedule all virtual machines in the Server Group to execute on the same host. The target compute node is selected by the Nova scheduler when the first instance in the Server Group is launched, in compliance with its corresponding flavor, network, and system requirements.

If the **Best Effort** flag of the Server Group is clear, then the scheduling policy is strictly enforced. Any new instance in the Server Group will fail to launch if any run-time requirements cannot be satisfied by the selected compute node.

If the **Best Effort** flag of the Server Group is set, then the scheduling policy is relaxed. New instances are scheduled to run on the selected compute node whenever possible, but are scheduled on a different host if otherwise necessary.

#### **Anti-affinity policy**

The goal of this policy is to schedule each virtual machine in the Server Group to execute on a different host. The target compute node for each instance is selected by the Nova scheduler in compliance with the corresponding flavor, network, and system requirements. As with the affinity policy, the Nova scheduler takes no special considerations regarding the nature of the target host, HT-enabled or not.

If the **Best Effort** flag of the Server Group is clear, then the scheduling policy is strictly enforced. Any new instance in the Server Group will fail to launch if its run-time requirements can not be satisfied by any newly selected compute node.

If the **Best Effort** flag of the Server Group is set, then the scheduling policy is relaxed. New instances are scheduled to run on a different compute node whenever possible, but are scheduled on any already selected host if otherwise necessary.

## Live Migration of Virtual Machines

Live migration occurs when a virtual machine is transferred to execute on a different compute node with minimal disruption of the guest applications. This can happen automatically, or upon request by the system administrator.



**NOTE:** Live migration is not currently supported for instances using the following:

- PCI passthrough
- SR-IOV

For storage considerations affecting live migration, see *VM Storage Settings for Migration, Resize, or Evacuation*.

While executing a live migration operation, Titanium Server manages the virtual machine's state in such a way that it appears unmodified on the migrated instance. This includes:

- system memory, both kernel and user space
- access to non-local storage resources (Cinder storage)
- all the virtual machine networking options (unmodified/virtio, AVP kernel driver, AVP DPDK Poll Mode Driver), and AVS

Automatic migration of virtual machines occurs whenever the administrator initiates a locking operation on a compute node. In this case, Titanium Server first live-migrates all virtual machine instances off of the compute node before administratively locking it.

The **admin** user can also initiate live migrations manually from the Instances page available by clicking the option **Instances** on the **Admin** side pane of the web administration interface. The **More** button of the selected instance provides the option **Live Migrate Instance**. When selected, the Live Migrate window is displayed as illustrated below:

Live Migrate

Current Host  
compute-1

New Host \*  
Auto schedule host

Block Migration

Description:  
Live migrate an instance to a specific host.

Cancel Live Migrate Instance

The following fields are available:

### Current Host

A read-only field displaying the compute node the selected instance is currently running on.

### New Host

The target compute node for the migration. The default value is to let Titanium Server auto-schedule the virtual machine following the current scheduling guidelines and constraints. Optionally, you can manually select the target compute node.

Note that the set of available target compute nodes for the migration is still subject to the scheduler constraints from the virtual machine flavor and other systems options that might be in place.

### Live Migration and Server Group Policies

Virtual machines launched as part of a Server Group are subject to additional live migration restrictions determined by the selection of scheduling policy. See [Server Groups](#) on page 9 for details.

#### Affinity policy

The goal of this policy is to schedule all virtual machines in the Server Group to execute on the same host.

If the **Best Effort** flag of the Server Group is clear, then the individual instances cannot be migrated since this would break the affinity policy.

Note that this means that a compute node running instances in a Server Group with affinity policy in strict mode cannot be locked. An alternative mode of operation is to always set the **Best Effort** flag and then manually migrate the instances to a common host.

If the **Best Effort** flag of the Server Group is set, then any individual instances can migrate to any other available host.

#### Anti-affinity policy

The goal of this policy is to schedule each virtual machine in the Server Group to execute on a different host.

If the **Best Effort** flag of the Server Group is clear, then the individual instances can migrate provided that there are suitable hosts where no other Server Group instance is running.

If the **Best Effort** flag of the Server Group is set, then any individual instances can migrate to any other available host.

## Host Aggregates

Host aggregates are collections of hosts that share common attributes for the purposes of VM scheduling.



---

**NOTE:** The information in this topic is preliminary and subject to change.

---

To view host aggregates, open the Host Aggregates page from the **Admin** menu.

## Host Aggregates

Q  
+ Create Host Aggregate ✖ Delete Host Aggregates

<input type="checkbox"/>	Name	Availability Zone	Hosts	Metadata	Actions
<input type="checkbox"/>	local_storage_image_hosts	-		storage = local_image	<div>Edit Host Aggregate</div> <div>▼</div>
<input type="checkbox"/>	local_storage_lvm_hosts	-	compute-0	storage = local_lvm	<div>Edit Host Aggregate</div> <div>▼</div>
<input type="checkbox"/>	provider_providernet-a	-	compute-0	provider:physical_network = providernet-a	<div>Edit Host Aggregate</div> <div>▼</div>
<input type="checkbox"/>	provider_providernet-b	-	compute-0	provider:physical_network = providernet-b	<div>Edit Host Aggregate</div> <div>▼</div>
<input type="checkbox"/>	remote_storage_hosts	-		storage = remote	<div>Edit Host Aggregate</div> <div>▼</div>

Displaying 5 items

## Availability Zones

Q

Availability Zone Name	Hosts	Available
internal	controller-0 (Services Up)	Yes
nova		No

Displaying 2 items

When the Nova Scheduler selects a compute node to instantiate a VM, it can use host aggregates to narrow the selection. For example, if the VM requires local storage with image-based instance backing, the Nova scheduler selects a host from the **local\_storage\_image\_hosts** host aggregate. Alternatively, if the VM requires LVM-based instance backing, the scheduler selects from the **local\_storage\_lvm\_hosts** host aggregate. This ensures that the instance is instantiated on a host that meets the requirements of the VM.

The Nova scheduler does not always use host aggregates. For example, if a VM does not specify the instance backing type, the Nova scheduler can instantiate it on any resource.

Some host aggregates are managed automatically by Titanium Server.



**CAUTION:** Do not make manual changes to host aggregates that are managed automatically.

- The **local\_storage\_image\_hosts** and **local\_storage\_lvm\_hosts** memberships are updated automatically whenever the instance backing type is changed on a compute host. For more information, see the instructions for configuring a compute node in the *Titanium Server Software Installation Guide*.



**NOTE:** The **remote\_storage\_hosts** aggregate is currently not used.


You can use host aggregates to meet special requirements. For example, you can create a pool of compute hosts to offer dedicated resources such as pinned NUMA nodes or specific huge page sizes, while grouping the remaining compute hosts to offer shared resources.

## Server Groups

*Server Groups* is a mechanism to group virtual machines to which a common set of attributes is applied.

Select the **Server Groups** option from either the **Project** or the **Admin** tabs on the Web administration interface to display the Server Groups window:

Server Groups

	Project	Group Name	Policies	Members	Metadata	Actions
No items to display.						
Displaying 0 items						

Click **Create Server Group** to create a new server group:

**Create Server Group** ×

**Project \***

**Server Group Name \***

**Policy**

**Best Effort**  
☒

**Max Group Size (Instances)**

**Description:**  
From here you can create a new server group

The following parameters can be defined for the new server group:

### Project

Identifies the tenant with which the new service group should be associated. This field is only available as part of the Admin operations; the project identity is implicit within the context of a specific tenant.

### Server Group Name

The name for the new server group.

### Policy

The following scheduling policy options are available:

### affinity

When this option is selected, new instances launched as part of this server group are scheduled to run on the same compute node.

### anti-affinity

When this option is selected, new instances launched as part of this server group are scheduled to run on different compute nodes. For example, you can use the anti-affinity option when you have two virtual machines that run in 1:1 HA protection mode, and you want them to be protected against hardware failures.

For additional considerations on scheduling and live migration for each of these policies, see [The Virtual Machine Scheduler](#) on page 4 and [Live Migration of Virtual Machines](#) on page 6.

### Best Effort

When selected, the policy in place, affinity or anti-affinity, is enforced whenever possible, on a best-effort basis. If for any reason the policy cannot be enforced, the deployment of the new instance proceeds using the regular scheduling.

When cleared, the policy in place must be executable for the new instance to be scheduled. Launching of a new virtual machine fails if there are no resources to comply with the selected policy.

### Max Group Size (Instances)

Determines the maximum number of instances that can co-exist as part of this server group.

The new server group is displayed as follows:

Server Groups						
			Filter	Q	+ Create Server Group	✕ Delete Server Groups
	Project	Group Name	Policies	Members	Metadata	Actions
<input type="checkbox"/>	admin	server-group-0	anti-affinity		wrs-sg:best_effort:true, wrs-sg:group_size:10	Delete Server Group
Displaying 1 item						

Once a server group is defined, you can add virtual machines to it at launch time. This is done by selecting the desired server group from the **Server Group** tab on the launch window.

## Server Groups and the CLI

Server groups can be created and deleted using the CLI as illustrated in the following examples:

```
~(keystone_admin)$ nova server-group-create --policy affinity \
--metadata wrs-sg:best_effort=true --metadata wrs-sg:group_size=2 ht-group-test
+-----+-----+-----+-----+-----+
| Id           | Name           | Policies           | Members | Metadata |
+-----+-----+-----+-----+-----+
| 63752c24-... | ht-group-test | [u'affinity...    | []      | {u'best... |
+-----+-----+-----+-----+-----+
~(keystone_admin)$ nova server-group-delete 63752c24-...
Server group 63752c24-... has been successfully deleted.
```

## Configuring Instances at Boot Time

Boot configuration user data can be passed to a virtual machine during startup.

For example, an Element Management System (EMS) may be used in cloud environments for VM configuration, but the VM may require some bootstrap information to successfully communicate with the EMS.

Titanium Server provides three mechanisms to accomplish this:

### *User Data*

This is a mechanism for passing a local file to an instance when it is launched. This method is typically used to pass a shell script or configuration file.

To send user data when calling nova boot, use the `--user-data /path/to/filename` option, or use the Heat service and set the `user_data` property and `user_data_format` to **RAW**.

On initialization, the VM queries the metadata service through either the EC2 compatibility API. For example:

```
$ curl http://169.254.169.254/openstack/2012-08-10/user_data
```

or the OpenStack metadata API. For example:

```
$ curl http://169.254.169.254/2009-04-04/user-data
```

In either case, text is returned to the VM and can be used for bootstrap configuration.

Access to the metadata server at 169.254.169.254 is provided by a virtual router attached to the tenant network on which the access request is made. Virtual routers are automatically configured as proxies to the metadata service.

The following requirements must be satisfied in order for a guest application to access the metadata service:

- There must be a route table entry to route traffic to the 169.254.169.254 address using a Neutron router, or using a suitable static route to the 169.254.169.254 address.
- The metadata server must know about the virtual machine instance associated with the MAC and IP addresses of the virtual Ethernet interface issuing the metadata service request. This is necessary for the metadata server to validate the request and identify the correct data to return to the VM.

On system managed subnets, the Neutron service has all address information associated with the virtual machines in its database.

On unmanaged subnets, you must tell the Neutron service the IP address of the network interface issuing the metadata service requests. This can only be done using the command line or API interfaces when instantiating the virtual machine. For example:

```
$ nova boot \  
--nic net-id=net-uuid,vif-model=avp,v4-fixed-ip=172.18.0.1 \  
my-new-vm
```

In this simple example, the option **v4-fixed-ip** tells Neutron the IP address for this interface. The interface's MAC address is automatically generated by Nova.

#### *cloud-init*

This is an open-source package available from <https://cloudinit.readthedocs.org/en/latest/> that supports a variety of guests. It expects a particular file format for user data, retrieves the user data from the metadata server, and takes action based on the contents of the data.

Two commonly used input formats are:

#### *shell scripts*

You can configure an instance at boot time by passing a shell script as user data. The script file must begin with

```
#!
```

for **cloud-init** to recognize it as a shell script.

#### *Cloud config files*

A configuration format based on YAML that you can use to configure a large number of options on a system. For example, to set the hostname:

```
#cloud-config
hostname: mynode
fqdn: mynode.example.com
manage_etc_hosts: true
```

See <https://cloudinit.readthedocs.org/en/latest> for a complete list of capabilities.

The following requirements must be satisfied in order for a guest application to access the metadata service:

- There must be a route table entry to route traffic to the 169.254.169.254 address using a Neutron router, or using a suitable static route to the 169.254.169.254 address.
- The metadata server must know about the virtual machine instance associated with the MAC and IP addresses of the virtual Ethernet interface issuing the metadata service request. This is necessary for the metadata server to validate the request and identify the correct data to return to the VM.

#### *Config drive*

Titanium Server can be configured to use a special-purpose configuration drive (abbreviated *config drive*) to store metadata (including injected files). Metadata is written to the drive, which is attached to the instance when it boots. The instance can retrieve information normally available through the metadata service by reading from the mounted drive.

The config drive can be enabled by using the **--config-drive=true** option with nova boot.

The following example enables the config drive and passes user data, injecting two files and two key/value metadata pairs. These can be read from the config drive.

```
$ nova boot --config-drive=true --image my-image-name --key-name mykey \  
--flavor 1 --user-data ./my-user-data.txt myinstance \  
--file /etc/network/interfaces=/home/myuser/instance-interfaces --file known_hosts=  
home/myuser/.ssh/known_hosts \  
--meta role=webserver --meta essential=false
```



From within the instance, the config drive volume is labeled **config-2**. You can mount the config drive as the **/dev/disk/by-label/config-2** device if your guest OS supports accessing disks by label. For example:

```
# mkdir -p /mnt/config
# mount /dev/disk/by-label/config-2 /mnt/config
```

The contents of the config drive depend on the options passed to nova boot. The contents of the config drive for the example above are:

```
ec2/2009-04-04/meta-data.json
ec2/2009-04-04/user-data
ec2/latest/meta-data.json
ec2/latest/user-data
openstack/2012-08-10/meta_data.json
openstack/2012-08-10/user_data
openstack/content
openstack/content/0000
openstack/content/0001
openstack/latest/meta_data.json
openstack/latest/user_data
```

For file format details and full details on config-drive see [http://docs.openstack.org/user-guide/cli\\_config\\_drive.html](http://docs.openstack.org/user-guide/cli_config_drive.html).



**CAUTION:** If a VM uses config-drive for user data or file injection, VM evacuations due to a compute node failure and VM live migrations to another compute node will cause the config drive to be rebuilt on the new compute node and metadata to be populated, but user data and injected files are not populated in the evacuated or live-migrated config drive of the VM.

For a VM using **config-file** with file injection, it is recommended to copy the injected files to the root disk of the VM on initial boot, and to set a flag to prevent the use of injected files on subsequent boots.

#### Related Links

[Customizing Images with User Data In a Heat Template](#) on page 70

You can provide bootstrap configuration for an instance at launch by including user data in a Heat template.

## Selecting Virtual Machine Flavors

Flavors are created by the system administrator.

A flavor is a list of attributes applied to a virtual machine when a new instance is created. It specifies resources to be allocated by the system, such as size of RAM, number of cores, storageresources, and so on.

Existing flavors can be selected by tenant users.

## Configuring PCI Addresses for Guest VNICs

You can configure virtual PCI addresses for guest VNICs.

The CLI command **nova boot** provides support for configuring virtual guest NIC devices. The format for the argument is:

```
<domain>:<bus>:<slot>.<function>
```

This enables you to specify the bus and the slot and allows you to address a wide range of devices within a guest. If a bus number greater than 0 is specified, the corresponding PCI bridges will be configured in the guest.

Limitations:

- It is not possible to specify a domain value other than 0 and a function value other than 0.
- It is not possible to address a bus value higher than 8.
- Slots 0 and 1 are reserved for PCI bus 0.
- Slot 0 is reserved for any PCI bus greater than 0.
- For AVP NICs, use of a bus other than 0 is not currently supported.
- This feature only supports PCI devices of type NIC. Other PCI devices that can be specified with a flavor are not supported.
- Configure virtual PCI slots using the **vif-pci-address** parameter of the **nova boot** command, as shown below.

```
$ nova boot --flavor=small.prefer \  
--nic net-id=$NET0ID,vif-model=avp,vif-pci-address=0000:00:09.0 \  
--nic net-id=$NET1ID,vif-model=virtio,vif-pci-address=0000:02:01.0 \  
--nic net-id=$NET2ID,vif-model=virtio,vif-pci-address=0000:02:02.0 \  
--image=cust-guest wr1-2
```

## Accessing the Metadata Server

When an instance is launched, it has the option to retrieve instance-specific and user data from a metadata server, which can be accessed using the well-known address 169.254.169.254.

The method used to access the server depends on your deployment scenario.

- If the VM is attached to a tenant network with a virtual router then the metadata server is reachable using the virtual router as the route gateway.
- If the VM instance is attached to multiple tenant networks, each with access to a virtual router, then any of the virtual routers provides access to the metadata server. However, installing

multiple default routes on the VM might impact the VM's ability to route packets back to external networks.

- If the VM is attached to a tenant network that does not have a virtual router then the a route to the metadata server can be retrieved from the Neutron DHCP service, provided that this service is enabled. If DHCP is enabled on a tenant network, then the VM can use DHCP option 121, Classless Static Route, to retrieve the metadata server static route information, which uses the DHCP server's address as the gateway.

Note that when using DHCP option 121, the answer from the DHCP server will also include other applicable static routes. They include:

- any static routes configured when the IP subnet was created
- default route to the subnet gateway IP address, if one is configured on the subnet
- on-link routes for all other subnets on the same network and same guest VLAN

The DHCP service only responds to option 121 is there are no virtual routers on the network.

The following requirements must be satisfied in order for a guest application to access the metadata service:

- There is a route table entry to route traffic destined to the 169.254.169.254 address via a Neutron router, or via a suitable static route to the 169.254.169.254 address.
- The metadata server knows about the virtual machine instance associated with the MAC and IP addresses of the virtual Ethernet interface issuing the metadata service request. This is necessary for the metadata server to be able to validate the request, and to identify the virtual machine's specific data to be returned.

On system managed subnets, the Neutron service has all address information associated with the virtual machines in its database.

On unmanaged subnets, you must tell the Neutron service the IP address of the network interface issuing the metadata service requests. This can only be done using the command line or API interfaces when instantiating the virtual machine, as illustrated below:

```
$ nova boot \  
--nic net-id=net-uuid,vif-model=avp,v4-fixed-ip=172.18.0.1 \  
my-new-vm
```

In this simplified example, the option **v4-fixed-ip** tells Neutron what the IP address for this interface should be. The interface's MAC address is automatically generated by Nova.

## Launching Virtual Machine Instances

You can launch a virtual machine from the web administration interface or the CLI.

You can launch a virtual machine from the CLI using the nova boot command. For examples, see [Launching Virtual Machine Instances Using the CLI](#) on page 17.

Note that not all of the options described in the following steps are required. For example, it is possible to launch an instance without defining any post-creation actions. Required tabs and fields are marked with an asterisk (\*) in the web interface.

## Procedure

1. Logged in as a user, navigate to **Project > Compute > Instances** and click **Launch Instance**.

2. Specify details about the instance.

In the Launch Instance window, select the Details tab.

These options define basic run-time attributes such as name, number of vCPUs, memory and storage, as well as the type of boot source to use (volume or image). For more information, see <http://docs.openstack.org/>.

Ensure that the storage space on the hosts is configured appropriately. For more information, see *Titanium Server System Administration: Storage Planning*.



---

**NOTE:** For fast instance booting, use a prepared Cinder volume, and set the **Instance Boot Source** to **Boot from Volume**. For more information, see [Creating Cinder Volumes for Boot Images](#) on page 40.

---

3. Specify a server group.

In the Launch Instance window, select the Server Group tab.

Server groups is a mechanism to group virtual machines to which a common set of attributes is applied, such as affinity and anti-affinity behavior. At least one server group must exist to use this option. For more information, see [Server Groups](#) on page 9.

4. Establish access security for the instance.

a) In the Launch Instance window, select the Access & Security tab and associate the available tenant's keypair with this instance. A keypair contains SSH credentials that will be added to the image when you launch it. For information, see the *Titanium Server Tenant User's Guide: Access and Security for Tenant Network Users* on page 25.

b) Select a security group. Security groups are sets of firewall policies that determine which incoming network traffic is forwarded to the instance. For more information, see [Security Groups](#) on page 26.

5. Specify which networks the image will use.

In the Launch Instance window, select the Networking tab.

You must choose at least one tenant network over which the instance will communicate. Click the blue + buttons on the Available Networks list to move the tenant networks to the Selected Networks list. For each interface, you can specify the driver type from a drop-down menu.



---

**NOTE:** The order in which the networks are moved determines the order in which the virtual Ethernet interfaces eth0, eth1, eth2, and so on are assigned.

---

For more information on tenant networking concepts, see [Tenant Networks](#) on page 29.

6. Specify any post-creation actions to take.

In the Launch Instance window, select the Post-Creation tab.

You can load *user data* from file or add them via the web interface to perform post creation setup actions such as implementing a bridge on the instance. The following example implements a bridge.

```
#wrs-config  
FUNCTIONS="bridge"  
BRIDGE_PORTS="eth1,eth2.5"
```



**CAUTION:** The first line:

```
#wrs-config
```

is a mandatory comment. Without it, the rest of the input is ignored.

#### 7. Set any advanced options.

In the Launch Instance window, select the Advanced tab.

You have the following disk partitioning options:

##### Automatic

The entire disk will be treated as a single partition and automatically resized.

##### Manual

Results in faster build times but requires manual partitioning.

You can set the minimum number of instances to launch.

Enabling the **Configuration Drive** option will cause Titanium to write metadata to a special configuration drive that attaches to the instance when it boots, making the metadata available. For more information on configuration drives, see <http://docs.openstack.org/user-guide/cli-config-drive.html>.

#### 8. Click **Launch** to start the instance.

## Launching Virtual Machine Instances Using the CLI

You can use the CLI to launch virtual machine instances.

You use the CLI command **nova boot**, with various options, to launch virtual machine instances. These options include standard Open Stack options as well as Titanium Server specific options.

### Prerequisites

To launch an instance, you must first set up required resources, such as tenants and tenant networks with IP subnets, management IP subnets, virtual routers, and so forth. For more information, see *OpenStack Accounts*.

- Launch instances on the tenants as described in the following examples.
  - The following example launches an instance named **tenant1-kernel-bridging**, using a Linux bridge.

```
~(keystone_user1)$ nova boot --poll --key_name=tenant1-controller-0 \  
--block_device_mapping vda=${vol UUID}:::0 \  
--user-data=/usr/share/userdata/linux-bridge.txt \  
--flavor=100 \  
--nic net-id=${tenant1_mgmt_net UUID},vif-model=virtio \  
--nic net-id=${tenant1_net_UUID},vif-model=avp \  

```

```
--nic net-id=${internal_net_UUID},vif-model=avp \
tenant1-kernel-bridging
```



**CAUTION:** The order in which the network interfaces are selected determines how they are enumerated for the instance. This example uses the following order: **eth0** (**tenant1-mgmt-net**) of the virtio type, and **eth1** (**tenant1-net**) and **eth2** (**internal-net**) of the avp type..

- The following example launches an instance named **tenant1-dpdk-bridging**, using an AVP virtual network interface, and the DPDK-accelerated virtual switch.

```
~(keystone_user1)$ nova boot --poll --key_name=tenant1-controller-0 \
--block_device_mapping vda=${vol_UUID}:::0 \
--user-data=/usr/share/userdata/linux-dpdk-vswitch.txt \
--flavor=101 \
--nic net-id=${tenant1_mgmt_net_UUID},vif-model=virtio \
--nic net-id=${tenant1_net_UUID},vif-model=avp \
--nic net-id=${internal_net_UUID},vif-model=avp \
tenant1-dpdk-bridging
```

- The following example launches an instance named **tenant1-vnic**, using AVP virtual network interfaces and a Linux router, and assigns virtual NIC interfaces.

```
~(keystone_user1)$ nova boot --poll --key_name=tenant1-controller-0 \
--block_device_mapping vda=${vol_UUID}:::0 \
--user-data=/usr/share/userdata/tenant1-linux-router.txt \
--flavor=101 \
--nic net-id=${tenant1_mgmt_net_UUID},vif-model=virtio vif-pci-
address=0000:00:09.0\
--nic net-id=${tenant1_net_UUID},vif-model=avp vif-pci-address=0000:02:01.0\
--nic net-id=${internal_net_UUID},vif-model=avp vif-pci-address=0000:02:02.0 \
tenant1-vnic
```

The instances are now running. Each one establishes a bridge or router between the last two interfaces in the command line.

## Connecting Virtual Machines to External Networks

You can provide VMs with external connectivity using one of several methods.

To connect VMs to external networks, you require an internal tenant network for the VMs, an external tenant network connected to an edge router, and a virtual router attached to both networks.

You can use floating IP addresses, source network address translation (SNAT), or port-based destination network address translation (DNAT) to expose VMs to an external network. Alternatively, you can use an external NAT device, or use a network address space for the VMs that is directly visible from the external network.

You can associate a floating IP address with a VM using a pool of available floating IP addresses configured by the system administrator. When a host on an external network initiates a connection to a public floating IP address, the virtual router updates a connection table to enable bidirectional traffic flow between the host and the VM. This approach requires a range of dedicated externally visible addresses.

When SNAT is enabled, the virtual router uses a port-based network address translation table. Only one externally visible IP address is required; ports on this IP address are mapped to internal

IP addresses and ports used by VMs. However, unless DNAT is also used, the table is updated only when a VM initiates connection to an external host. Until this happens, the VM is inaccessible from the external network. This is true even if the private IP address of the VM is used directly by incoming traffic.

In some cases, the use of virtual routers on the tenant networks can result in multiple default routes for a VM. If this happens, you can establish alternative VM access to the metadata server; see [#unique\\_23/unique\\_23\\_Connect\\_42\\_accessing\\_metadata\\_server](#) on page 49 for more information.

When port-based DNAT (port forwarding) entries are added to the SNAT network address translation table, connections to VMs can be initiated externally.

You can use floating IP addresses, SNAT, and DNAT in any combination. The following table presents an overview.

	Floating IP disabled	Floating IP enabled
SNAT disabled	VMs are directly accessible from the external network, provided the tenant network address space is directly visible from the external network, or NAT is implemented using a separate device.	VMs cannot initiate connections to the external network unless they have assigned floating IP addresses. External hosts can initiate connections to VMs that have assigned floating IP addresses.
SNAT enabled	VMs can initiate connections to the external network. External hosts cannot initiate connections to the VMs unless port-based DNAT is also used.	VMs can initiate connections to the external network. External hosts can initiate connections to VMs that have assigned floating IP addresses.

To enable or disable SNAT, see [Configuring SNAT on a Virtual Router](#) on page 56. To configure port-based DNAT, see [Configuring Port-based DNAT on a Virtual Router](#) on page 57.

To configure floating IP addresses for VMs, refer to the public OpenStack documentation.

## Scaling Virtual Machine Resources

You can scale the resources of individual instances up or down.

Currently, the CPU resources for an instance are scalable.

For an instance to be scalable, the following requirements must be satisfied:

- The image used to launch the instance must support scaling.

The example image provided with Titanium Server supports scaling. You can also build your own image, incorporating the required libraries and services. For more about building your own images, or about the technical requirements for scaling support, refer to the documentation included with the Titanium Server Software Development Kit.

- The flavor used to launch the instance must be configured with maximum and minimum scaling limits (the *scaling range*) for the resource.

When scaling a VM, use the credentials of the user that launched the VM. This ensures that quotas are correctly managed.

Depending on the resource being scaled, the scaling behavior applied by Titanium Server may be adjustable. For example, you can control which CPUs are released when the CPU resources for an instance are scaled down. To adjust scaling behavior, use the App Scale Helper script.

Normally, scaling is performed under the direction of Heat orchestration. For more about Heat autoscaling, see [Resource Scaling \(Autoscaling\)](#) on page 71. If required, you can scale resources manually from the command line using the **nova scale** command, with the following syntax:

```
~(keystone_admin)$ nova scale instance_id resource {up | down}
```

For example, to reduce the number of CPUs allotted to an instance, you can use this command:

```
~(keystone_admin)$ nova scale instance_id cpu down
```



---

**NOTE:** To scale up the resources for an instance, sufficient resources are required on the host. If all available resources are already allocated to other instances, you may need to free up resources manually.

---

## The App Scale Helper Script

The App Scale Helper script supports resource scaling customizations.

This is an optional script running on the guest. If present, it can modify aspects of scaling behavior. For example, it can control which CPU is taken offline during a scale-down operation, overriding the default selection.

It can also call other scripts or programs. You can use this to coordinate or manage processing work for the vCPUs as you scale them.

For more about the App Scale Helper script, refer to the documentation for the *Titanium Server Software Development Kit*.

## CPU Scaling

Titanium Server supports CPU up/down scaling for instances.

You can enable CPU scaling for an instance by setting a scaling range (see [Setting the CPU Scaling Range](#) on page 21). In addition, the **CPU Policy** must be set to **Dedicated**, and if the **CPU Thread Policy** is also present, it must be set to **Isolate** (see *Specifying Host CPU Threading Policies for a VM*).

When an instance is first started, all available vCPUs are brought online. If the instance is restarted, the number of online vCPUs is set to match the number when the instance was stopped. This is controlled by a helper script that automatically takes the required number of vCPUs offline.

When CPU resources are scaled up, a vCPU is brought online from the pool of available vCPUs for the instance, subject to the maximum allowed by the flavor. The lowest-numbered offline vCPU is selected.



When CPU resources are scaled down, a vCPU is taken offline, subject to the minimum allowed by the flavor. By default, the highest-numbered online vCPU is selected. This can be overridden using the App Scale Helper script.

For more about CPU scaling, refer to the documentation for the Titanium Server Software Development Kit.



**NOTE:** If there are insufficient resources to launch a scalable VM with the expected allotment when the VM is started or restarted, the launch fails. The VM is *not* automatically scaled down to compensate.

## Setting the CPU Scaling Range

You can define the maximum and minimum CPU resources available to an instance by using a flavor.

### Prerequisites

For CPU scaling to take effect, the **CPU Policy** for the VM must be set to **Dedicated**. For more information, see *Specifying Host CPU Threading Policies for a VM*.

### Procedure

1. Display the Flavors list.

Select the **Admin** menu in the web administration interface, and then in the **System Panel** section of the menu, click **Flavors**.

Flavors Filter  + Create Flavor ✕ Delete Flavors

<input type="checkbox"/>	Flavor Name	vCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	ID	Public	Metadata	Actions
<input type="checkbox"/>	example-guest.tiny	1	512MB	0GB	0GB	0MB	06bf90f1-e996-4c12-80f5-18cee762d38b	Yes	Yes	Edit Flavor <input type="button" value="v"/>
<input type="checkbox"/>	example-guest.small	2	1024MB	0GB	0GB	0MB	0630a9eb-e0c2-496f-8158-5dbabdba3109	Yes	Yes	Edit Flavor <input type="button" value="v"/>

Displaying 2 items

2. Optional: Select a suitable flavor.

You can edit it to specify the maximum and minimum vCPUs.

3. Specify the maximum number of vCPUs for the instance.

In the Flavors list, locate the flavor, and then click **Edit** to open the Edit Flavor dialog box.

In the **vCPU** field, enter the maximum number of vCPUs.

When you are finished editing, click **Save** to return to the Flavors list.

4. Specify the minimum number of vCPUs for the instance.

You can specify the minimum number of vCPUs by adding an Extra Spec for the flavor.

- a) In the Flavors list, click the **Flavor Name** for the flavor to open the Flavor Details dialog box.

## Flavor Detail: example-guest.cpus 12/9/2014, 10:43:24 AM

Overview
Extra Specs

### Info

**Flavor ID**  
7da443c3-6967-4d0f-b65a-5a3264d67b40

**Public**  
True

**Disabled**  
False

**VCPU Model**  
QEMU Virtual Processor

**VCPUs**  
4

**Memory MB**  
512

**Disk GB**  
0

**Ephemeral Disk GB**  
0

**Swap Space MB**  
No Swap Space defined.

**RxTx Factor**  
1.0

- b) On the **Extra Specs** tab, click **Create**.

Overview
Extra Specs

## Extra Specs

+ Create
Delete extra specs

<input type="checkbox"/>	Name	Key	Value	Actions
<input type="checkbox"/>	CPU Policy	hw:cpu_policy	dedicated	Edit ▼
<input type="checkbox"/>	Mem Page Size	hw:mem_page_size	2048	Edit ▼

Displaying 2 items

- c) In the **Create Flavor Extra Spec** dialog, select **Minimum Number of CPUs** from the **Extra Spec** drop-down menu.

Create Flavor Extra Spec

Extra Spec \*

Custom Extra Spec

Description:

Create a new "extra spec" key-value pair for a flavor.

Key ?

Value ?

Cancel Create

- d) In the **Key** field, enter `wrs:min_vcpus`.
- e) In the **Value** field, enter the minimum allowed number of vCPUs for the flavor.
- f) Click **Create**.



# 3

## *Access and Security*

[Access and Security for Tenant Network Users](#) 25

[Security Groups](#) 26

### **Access and Security for Tenant Network Users**

The Access and Security window provides controls for network protocol and port security, SSH-based access to virtual machines, API access to services, and floating IP address allocations.

The Security Groups tab enables you to select and apply a security group when launching an instance.

The Key Pairs tab enables you to select a key pair for the instance. A key pair contains SSH credentials that will be added to the image when you launch it. Key Pairs provide for secure login to guest images.



---

**NOTE:** Key pairs are shown based on the logged-in user, not the selected project. For example, an **admin** user examining the project **tenant1** cannot see key pairs for users associated with **tenant1**.

---

You can use floating IP addresses to expose VMs to an external network.

## Security Groups

Security groups are tenant-specific sets of IP filter rules that are applied to the networking stack of a virtual machine.

Titanium Server does not enforce a default security group when launching a new virtual machine. Instead, the security groups are optional, and only enforced when explicitly assigned to a virtual machine, either at launch time, or while it is executing.

At launch time, select the desired security groups from the tab **Access & Security** on the Launch Instance window, as illustrated below.

The screenshot shows the 'Launch Instance' window with the 'Access & Security' tab selected. The window has a title bar with a close button (X). Below the title bar are five tabs: 'Details \*', 'Server Group', 'Access & Security' (active), 'Networking', and 'Post-Creation'. The 'Access & Security' tab contains the following fields and controls:

- Keypair:** A dropdown menu showing 'controller-0' and a '+' button to add more.
- Admin Pass:** A text input field.
- Confirm Admin Pass:** A text input field.
- Security Groups:** A section with a checkbox labeled 'default'.

To the right of these fields is a descriptive text: 'Control access to your instance via keypairs, security groups, and other mechanisms.' At the bottom right of the window are two buttons: 'Cancel' and 'Launch'.

To assign security groups to a running instance, display the Instances page, click the **More** button associated with the instance, and select the option **Edit Security Groups**. The Edit Instance window displays with the tab **Security Groups** showing the available security instance groups, as illustrated below.

### Edit Instance

×

Info \*

Security Groups

From here you can add and remove security groups to this project from the list of available security groups.

All Security Groups

Filter

Q

default

+

Instance Security Groups

Filter

Q

No security groups enabled.

Cancel

Save

You can then drag and drop security groups from the section **All Security Groups** to the section **Instance Security Groups**. The selected security groups are applied immediately once the configuration is saved.

In the Titanium Server implementation, the following limitations apply:

- the maximum number of security groups that can be applied to a single tenant network is 10
- the maximum number of rules that can be configured on a single tenant network is 100

These two limitations work always together, whichever is first reached.





# 4

## *Tenant Network Management*

Tenant Networks 29

Creating Tenant Networks 34

Setting Up Tenant Networks Using the CLI 35

### **Tenant Networks**

Tenant networks are logical networking entities visible to tenant users, and around which working network topologies are built.

Tenant networks need support from the physical layers to work as intended. This means that the access L2 switches, providers' networks, and data interface definitions on the compute nodes, must all be properly configured. In particular, when using provider networks of the VLAN or VXLAN type, getting the proper configuration in place requires additional planning.

For provider networks of the VLAN type, consider the following guidelines:

- All ports on the access L2 switches must be statically configured to support all the VLANs defined on the provider networks they provide access to. The dynamic nature of the cloud might force the set of VLANs in use by a particular L2 switch to change at any moment.
- The set of VLANs used by each compute node is not fixed; it changes over time. The current VLAN set in use is determined by the configuration details of the tenant networks, and the scheduling on the compute nodes of the virtual machines that use them. This information is provided to the Neutron's AVS plugin, which then uses it to configure the AVS as required.

When a tenant creates a new network, the Neutron segmentation allocation strategy is to look first for an available segmentation identifier owned by the tenant. If none is available, the search continues over the available shared segmentation identifiers. The allocation process returns an error if no segmentation identifiers are available.

The VLAN ID assigned to a tenant network is fixed for as long as the tenant network is defined in the system. If for some reason the VLAN ID has to change, the tenant network must be deleted and recreated again.

- Configuring a tenant network to have access to external networks (not just providing local networking) requires the following elements:
  - A physical router, and the provider network's access L2 switch, must be part of the same Layer-2 network. Because this Layer 2 network uses a unique VLAN ID, this means also that the router's port used in the connection must be statically configured to support the corresponding VLAN ID.
  - The router must be configured to be part of the same IP subnet that the tenant network is intending to use.
  - When configuring the IP subnet, the tenant must use the router's port IP address as its external gateway.
  - The tenant network must have the **external** flag set. Only the **admin** user can set this flag when the tenant network is created.

For provider networks of the VXLAN type, consider the following guidelines:

- Layer 3 routers used to interconnect compute nodes must be multicast-enabled, as required by the VXLAN protocol.
- To minimize flooding of multicast packets, IGMP and MLD snooping is recommended on all Layer 2 switches. The AVS switch supports IGMP V1, V2 and V3, and MLD V1 and V2.
- To support IGMP and MDL snooping, Layer 3 routers must be configured for IGMP and MDL querying.
- To accommodate VXLAN encapsulation, the MTU values for Layer 2 switches and compute node data interfaces must allow for additional headers. For more information, see *Titanium Server Planning: The Ethernet MTU*.
- To participate in a VXLAN network, the data interfaces on the compute nodes must be configured with IP addresses, and with route table entries for the destination subnets or the local gateway. For more information, see *Titanium Server System Administration: Managing Data Interface Static IP Addresses Using the CLI*, and *Titanium Server System Administration: [Adding and Maintaining Routes for a VXLAN Network](#) on page 60*.

In some circumstances, tenant networks can be configured to use VLAN Transparent mode, in which VLAN tagged packets from the guest are encapsulated within a provider network segment (VLAN) without removing or modifying the guest VLAN tag. For more information, see *Titanium Server Tenant User's Guide: [VLAN Transparent](#) on page 31*. Alternately, guest VLAN-tagged traffic can be supported by Titanium Server tenants explicitly defining one or more VLAN-tagged IP subnets on a single tenant network. With this approach, the guest VLAN-tagged IP subnets have access to all of the services of the virtualized network infrastructure, such as DHCP, virtual routing, meta-data server, etc. For more information, see *Titanium Server Introduction: Titanium Server Overview*.

## VLAN Transparent

A *vlan transparent* tenant network is one that allows VLAN tagged packets to be encapsulated within a provider network segment without removing or modifying the guest VLAN tag. VLAN Transparent is provided in addition to VLAN-tagged Neutron subnets for guest VLANs.

VLAN Transparent must be supported on a provider network before a VLAN Transparent tenant network can be created on that provider network. The **VLAN Transparent** column in the list of defined provider networks on the page of the **Admin > Platform > Provider Networks** page indicates availability.

VLAN Priority attributes are propagated to the provider VLAN tag and then restored to the guest VLAN tag.

Provider Networks							
				Filter	Q	Create Provider Network	✖ Delete Provider Networks
<input type="checkbox"/>	Network Name	Status	Type	MTU	Segmentation Ranges	VLAN Transparent	Actions
<input type="checkbox"/>	group0-data0b	ACTIVE	vlan	1500	568-571, 572-575, 576-579, 580-583, 584-587, 588-591, 592-595, 596-599	True	Edit Provider Network ▾
<input type="checkbox"/>	group0-data1	ACTIVE	vlan	1500	664-695	False	Edit Provider Network ▾

This information is also available from the following command.

```
~(keystone_admin)$ neutron providernet-show provider1-vt
+-----+-----+
| Field | Value |
+-----+-----+
| description | 72a09e68-7170-493e-b9c6-d2bcb881f458 |
| id | 1500 |
| mtu | provider1-vt |
| name | { |
| ranges | { |
| | "name": "provider1-vt-r1-0", |
| | "tenant_id": "c17a85ac5283496da1152ce68b79e606", |
| | "maximum": 50, |
| | "minimum": 59, |
| | "shared": false, |
| | "id": "8b1fa184-80e0-4555-bd9f-bcf444384189", |
| | "description": null |
| | } |
| status | DOWN |
| type | vlan |
| vlan_transparent | True |
+-----+-----+
```

The summary information available by selecting a tenant network name from the **Provider Networks** tab of the **System > Networks** page indicates if a tenant network will request VLAN Transparent services from a provider network.

## Provider Network Overview

**Name**  
group0-data1  
**ID**  
67f9d397-1aef-4114-96a6-1c9f54e83dad  
**Type**  
vlan  
**MTU**  
1500  
**Description**  
None  
**VLAN Transparent**  
Yes  
**PCI PFs Configured**  
0  
**PCI PFs Used**  
0  
**PCI VFs Configured**  
0  
**PCI VFs Used**  
0

This information is also available from the following command.

```
~(keystone_admin)$ neutron net-show tenant1-vt
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True |
| id | 011ceale-422f-49d1-9d80-ba864bab361f |
| mtu | 1500 |
| name | tenant1-vt |
| provider:network_type | vlan |
| provider:physical_network | provider1-vt |
| provider:segmentation_id | 50 |
| router:external | False |
| shared | False |
| status | ACTIVE |
| subnets | 3bff6cb0-8422-4596-857f-a02d68a051b1 |
| | dedcd9e0-d4ee-4306-a6bf-194104baa9b4 |
| tenant_id | c17a85ac5283496da1152ce68b79e606 |
| vlan_transparent | True |
+-----+-----+
```

For more information on setting up provider networks, see *Titanium Server System Administration: Configuring Provider Networks* and *Configuring Provider Networks Using the CLI*. For more information on setting up tenant networks, see [Creating Tenant Networks](#) on page 34.

Table 1 VLAN Transparent Compatibility

	802.1p	802.1q	802.1ad	QinQ
<b>Flat Provider Network</b>	Yes	Yes	Yes	Yes
<b>VLAN Provider Network</b>	Yes	Yes	Yes	Yes
<b>VXLAN Provider Network</b>	Yes	Yes	Yes <sup>1</sup>	Yes
<b>VLAN Tagged Subnets</b>	Yes	Yes	No	No

## Guest VLANs

Use guest VLANs to segregate IP traffic from a single virtual Ethernet interface on a virtual machine into dedicated VLANs. Together with the capability to define multiple IP subnets on a single tenant network, guest VLANs facilitate the transitioning of existing guest applications to run on the Titanium Server virtualized network environment.

Guest VLANs are useful when guest applications rely on the capability to configure a single virtual Ethernet interface with multiple, probably overlapping, IP addresses. From the point of view of the guest, this is done by defining VLAN Ethernet interfaces, and associating one or more IP addresses to them. If implementing overlapping IP addresses, typically in support of VPN applications, the guest must use different VLAN IDs to separate traffic from different VPNs.

For example, on a Linux guest, the virtual interfaces eth0.10:1, eth0.10:2, and eth0.20 refer to the same eth0 physical interface with two virtual interfaces on VLAN ID 10, and a single virtual interface on VLAN ID 20. A common scenario in a VLAN application is to allocate distinct IP addresses from the same IP subnet to virtual interfaces on the same VLAN. In this example, eth0.10:1 and eth0.10:2 could be assigned distinct IP addresses from the subnet 192.168.1.0/24, and eth0.20 an address from the subnet 192.168.2.0/24. In the case of a VPN application, overlapping IP addresses are allowed to exist on eth0.20 and either eth0.10:1 or eth0.10:2.

Titanium Server supports these deployment scenarios with the help of guest VLANs which enable the transport of IP subnets traffic over VLAN-tagged Ethernet frames. To support the example above, a tenant user would define the following two IP subnets, both on the same tenant network, using guest VLAN IDs as follows:

- Subnet 192.168.1.0/24 with guest VLAN ID set to 10
- Subnet 192.168.2.0/24 with guest VLAN ID set to 20

The subnet-to-VLAN\_ID mapping can be one-to-one, as in this example, or many-to-one. This means that tenant users can use a single VLAN ID of their choice to encapsulate traffic from one or more IP subnets.

Alternately, tenant networks can be configured to use VLAN Transparent mode, in which VLAN tagged guest packets are encapsulated within a provider network segment without removing or modifying the guest VLAN tag. For more information, see *Titanium Server Tenant User's Guide: [VLAN Transparent](#)* on page 31.

<sup>1</sup> VLAN priority attributes not propagated to IP header differentiated services field.

## Guest VLAN Implementation

Guest VLANs are implemented using available segmentation ranges from suitable provider networks, just as it is done when new tenant networks are created. Therefore all network design considerations regarding the configuration of L2 switches and the Neutron allocation strategy, described in [Tenant Networks](#) on page 29, must be taken into consideration.

Additionally, note that the AVS will silently discard incoming VLAN-tagged VM traffic with unknown VLAN IDs, that is, with VLAN IDs not defined as guest VLANs on the particular tenant network.

## Creating Tenant Networks

You can use the CLI or Web interface to set up tenant networks and their associated IP subnets.

### Procedure

1. List the tenant networks currently defined on the system.

Select **Admin > System > Networks** to open the Networks page.

# Networks

Filter

Q

+ Create Network

<input type="checkbox"/>	Project	Network Name	Subnets Associated	DHCP Agents	Shared	Status	Admin State	Actions
No items to display.								
Displaying 0 items								

2. Create the tenant network.

From the **Networks** tab, click **Create Network** and fill in the form as illustrated below:

## Create Network

Name

tenant1-mgmt-net

Project \*

tenant1

Provider Network Type \* ?

vlan

Physical Network ?

provider-net-a

Segmentation ID ?

Admin State \*

UP

QoS Policy

No Policy

☐ Shared

☐ External Network

☐ VLAN Transparent ?

Description:

Create a new network for any project as you need.  
Provider specified network can be created. You can specify a physical network type (like Flat, VLAN, GRE, and VXLAN) and its segmentation\_id or physical network name for a new virtual network.  
In addition, you can create an external network or a shared network by checking the corresponding checkbox.

Cancel

Create Network

Click **Create Network** to commit the changes.

3. Create any required subnets.
  - a) Select **Networks** in the System Panel section of the **Admin** tab to open the Networks page.
  - b) Select the name of the tenant network you just created.
  - c) Click **Create Subnet**.
  - d) Complete the forms on the Subnet and Subnet Details tabs, then click **Create**.

## Setting Up Tenant Networks Using the CLI

You can use the CLI to set up tenant networks and their associated IP subnets.

This exercise sets up two tenant networks, **external-net**, **internal-net**, **tenant1-mgmt-net**, and **tenant2-mgmt-net**.

### Procedure

1. Create and configure the tenant network **external-net**.

- a) Create the tenant network.

```
~(keystone_admin)$ neutron net-create --tenant-id ${admin_tenant_UUID} \
--provider:physical_network=provider-net-a \
--provider:network_type=vlan \
--provider:segmentation_id=10 --router:external external-net
```

- b) Define the IP subnet.

```
~(keystone_admin)$ neutron subnet-create --tenant-id ${admin_tenant_UUID} \
--name external-subnet --gateway 192.168.2.1 --disable-dhcp external-net
192.168.2.0/24
```

- c) Set up the UUID variable for the new tenant network.

```
~(keystone_admin)$ export external_net_UUID=$(neutron net-list \
| grep external-net | awk '{print $2}')
```

## 2. Create and configure the tenant network **internal-net**.

- a) Create the tenant network.

```
~(keystone_admin)$ neutron net-create --tenant-id ${admin_tenant_UUID} \
--provider:physical_network=provider-net-b \
--provider:network_type=vlan \
--shared internal-net
```

- b) Define the IP subnet.

```
~(keystone_admin)$ neutron subnet-create --tenant-id ${admin_tenant_UUID} \
--name internal-subnet \
--no-gateway --wrs-net:vlan_id 5 internal-net 10.1.1.0/24
```



---

**NOTE:** The `--wrs-net:vlan_id 5` option defines a guest VLAN. Packets sent to this subnet by any virtual machine must be tagged with VLAN ID 5.

---

- c) Set up the UUID variable for the new tenant network.

```
~(keystone_admin)$ export internal_net_UUID=$(neutron net-list \
| grep internal-net | awk '{print $2}')
```

## 3. Create the tenant network **tenant1-mgmt-net**.

- a) Create the tenant network.

```
~(keystone_admin)$ neutron net-create --tenant-id ${tenant1_UUID} \
--provider:physical_network=provider-net-a \
--provider:network_type=vlan \
tenant1-mgmt-net
```

- b) Set up the UUID variable for the new tenant network.

```
~(keystone_admin)$ export tenant1_mgmt_net_UUID=$(neutron net-list \
| grep tenant1-mgmt-net | awk '{print $2}')
```

## 4. Create the tenant network **tenant2-mgmt-net**.

- a) Create the tenant network.

```
~(keystone_admin)$ neutron net-create --tenant-id ${tenant2_UUID} \
--provider:physical_network=provider-net-a \
```



```
--provider:network_type=vlan \  
tenant2-mgmt-net
```

b) Set up the UUID variable for the new tenant network.

```
~(keystone_admin)$ export tenant2_mgmt_net_UUID=`neutron net-list \  
| grep tenant2-mgmt-net | awk '{print $2}'`
```

The external and internal tenant networks are set up and ready to be used. The tenant management networks are available, but their IP subnets have yet to be created.

The tenant networks are assigned segmentation IDs automatically. You can list the assigned IDs for a provider network as follows:

```
~(keystone_admin)$ neutron net-list-on-providernet provider-net-a
```

id	name	vlan_id	providernet_type	segmentation_id
3d5dac09-a407- ...	external-net	0	vlan	10
93fa41f5-1f2e- ...	tenant2-mgmt-net	0	vlan	664
f699313d-a069- ...	tenant1-mgmt-net	0	vlan	623



# 5

## *Volume and Image Management*

<a href="#">Volume and Image Management Overview</a>	<a href="#">39</a>
<a href="#">Creating Cached RAW Boot Images</a>	<a href="#">39</a>
<a href="#">Creating Cinder Volumes for Boot Images</a>	<a href="#">40</a>
<a href="#">Cinder Volume Backups</a>	<a href="#">41</a>

### **Volume and Image Management Overview**

Volume Management is primarily the same as upstream OpenStack.

Generally, you should refer to upstream OpenStack documentation for information about volume management. The topics in this section highlight Titanium Server extensions.

### **Creating Cached RAW Boot Images**

You can use cached RAW boot image files for faster volume creation on Ceph-backed systems.

On systems using Ceph storage, boot image files must be converted to RAW format before they can be used to create volumes. You can accelerate volume creation in Titanium Server (and therefore instance launch time) by caching the RAW images as they are created. The cached images are maintained in the Glance image storage space and used for volume creation, eliminating conversion time.

By default, caching is performed in the background to ensure low processing overhead. If multiple requests are made at the same time, they are executed serially. You can use the `--wait` option to override background processing, so that the command does not return until the operation is complete and the raw image is ready for use. If you include a time in seconds (for

example, **--wait 10**), the command returns after the time has elapsed and reports the current status of the operation.

This feature is available from the CLI only. To create a cache, include the **--cached-raw** option when creating an image, as shown in the following example:

```
~(keystone_admin)$ glance image-create --name "cirros" --visibility public \
--disk-format=qcow2 --container-format=bare \
--file /home/wrsroot/images/cirros.img --cache-raw
```

Property	Value
Property 'cache_raw'	True
checksum	9ffe77b56668bf3fad95c64df4a8ac4f
container_format	bare
created_at	2016-01-07T22:06:22.769563
deleted	False
deleted_at	None
disk_format	qcow2
id	a5c6ef41-fe73-4d25-abbc-64380646e705
visibility	public
min_disk	0
min_ram	0
name	cirros
owner	f34e518e265a402aa53ce7c29475cab1
protected	False
size	636485632
status	active
updated_at	2016-01-07T22:06:39.225343
virtual_size	None

The cache size and status are included in the images list for the CLI.

```
~(keystone_admin)$ glance -v image-list
```

ID	Name	Status	Cache Size	Raw Cache
a5c...	cirros	active	5636485632	Cached

In the web administration interface, this list is shown on the Images page accessible from the **System** menu.




---

**NOTE:** Until image caching is complete, an image created using **--cached-raw** behaves as any regular uncached image.

---

## Creating Cinder Volumes for Boot Images

You can create Cinder volumes to store VM boot images.

The use of Cinder volumes is recommended for fast instance booting.




---

**NOTE:** You can accelerate volume creation by using a cached boot image. For more information, see [Creating Cached RAW Boot Images](#) on page 39.

---

### Procedure

1. Log in as the appropriate tenant.
2. On the **Project** menu, open the **Compute** section, and then click **Volumes**.
3. On the Volumes page, click **Create Volume**
4. Complete the Create Volume form.

## Cinder Volume Backups

Cinder volume backups are executed selectively, one volume at a time. They can include virtual disk images and any other Cinder volumes available in the system.

Consider the following recommendations when executing Cinder volume backups.

Aim at backing up an idle system

In an idle system, there are no running virtual machines. All VMs are either shut down or terminated. This is the ideal state for ensuring that the current content of all volumes is captured for replication at the next restore operation. Volumes reported as *available* are not associated with any VMs, while volumes reported as *in-use* are associated only with shut-down VMs, and are not being actively modified.

In a non-idle system, at least one VM is running. Volumes reported as *in-use* may be subject to write operations. In this case, you can create snapshots and back them up, but there is a risk of changes after the snapshot that are not captured.

Instructions to back up Cinder volumes in *available* and *in-use* states are provided in the following sections.

Back up one volume at a time

Backing up Cinder volumes can be a time- and space- consuming process, as determined by the size of the volumes themselves. Having two or more backup processes running simultaneously is likely to result in a severe performance hit to the controller and the running instances. This not only slows the normal system operations, but also makes the backup operations themselves take much longer.

Clean up as soon as possible

Backup files for Cinder volumes can fill up the **/opt/backups** partition on the controller very quickly. This is likely to cause further backup operations to fail in the middle of the execution. Therefore, as soon as a Cinder volume is backed up, and its backup files transferred and securely stored elsewhere:

- remove any snapshots you may have created during the backup process
- remove the backup files themselves

## Backing Up Available Cinder Volumes

You can back up Cinder volumes in the *available* state.

A Cinder volume in the *available* state is the ideal candidate for a backup operation in that there are no running instances using it.

### Procedure

1. Log in as Keystone user **admin** to the active controller.

Log in as user **wrsroot** to the active controller and source the script **/etc/nova/openrc** to obtain administrative privileges. For more information, see *Titanium Server System Administration: Linux User Accounts*.

2. Identify the volume you want to back up.

- a) List Cinder volumes from all tenants.

```
~(keystone_admin)$ cinder list --all-tenants
```

ID	Status	Display Name	Size	Volume Type	Bootable
593111d8-...	in-use	volume-1	1	None	true
53ad007a-...	available	volume-2	10	None	false

```

+-----+
| Attached to |
+-----+
| b998107b-... |
|               |
+-----+

```

Cinder volumes are listed also on the Volumes page of the web administration interface.

- b) Identify the volume to back up.

In the list above, **volume-1** is a disk image associated with a running instance. The volume **volume-2**, with UUID **53ad007a-...**, is available and can be safely backed up.

3. Export the target volume to the controller's file system.

```
~(keystone_admin)$ cinder export 53ad007a-841a-4fd3-a22b-813d4a6a8a85
```

Property	Value
display_description	
id	53ad007a-841a-4fd3-a22b-813d4a6a8a85
size	1
status	exporting
updated_at	2014-09-18T21:36:15.247881
volume_type	None

Exporting the volume takes some time, longer times for larger volumes. While it is taking place, the status of the volume is set to *exporting*, which you can verify by issuing the command **cinder show UUID**, or **cinder list --all-tenants** again. This new state is automatically reported on the Web administration interface.

You must wait for the export operation to complete. Use the command **cinder show 53ad007a-...** and look for the **backup\_status** field to determine whether the export operation was successful. The state of the volume should be set to *available* again.

When the export operation completes, the backup file is ready, as illustrated in the following example:

```
~(keystone_admin)$ ls -lh /opt/backups
total 306M
drwx----- 2 root root 16K Sep 10 17:42 lost+found
drwxr-xr-x 2 root root 4.0K Sep 18 21:42 temp
-rw-r--r-- 1 root root 305M Sep 18 21:42 volume-53ad007a-841a-4fd3-
a22b-813d4a6a8a85-20140918-213918.tgz
```

4. Transfer the backup file to an external storage resource.

The following example uses the **scp** command to transfer the backup file to the directory **/backups/titanium** on a server named **backups-server.wrs.com**:

```
~(keystone_admin)$ scp /opt/backups/volume-53ad007a-841a-4fd3-
a22b-813d4a6a8a85-20140918-213918.tgz admin@backups-server.wrs.com:/backups/titanium
```

5. Delete the backup file.

You may want to free up disk space on the controller to accommodate additional Cinder volume backups.

The volume backup is complete. Repeat this procedure with all other Cinder volumes in the *available* state in the system.

## Backing Up In-Use Cinder Volumes

You can back up Cinder volumes in the *in-use* state.

A Cinder volume in the *in-use* state is associated with an existing instance, either as the instance's disk image or as an attached storage resource. The instance can be either running or shut down. If the instance is running, you must take a snapshot of the volume first, and then export it to the controller's file system. An effective and reliable strategy is to take snapshots of all *in-use* volumes, regardless of whether the associated instances are running. If you prefer, you can use the **nova list** command with the option **--all-tenants** to identify which instances are running, and then take snapshots selectively.

```
~[keystone_admin]$ nova list --all-tenants
```

ID	Name	Tenant ID	Status	Task State	Power State	Networks
1f6...	t1-vm1	5e38b9...	ACTIVE	-	Running	tenan...
b92...	t1-vm2	5e38b9...	ACTIVE	-	Running	tenan...

### Procedure

1. Log in as Keystone user **admin** to the active controller.

Log in as user **wrsroot** to the active controller and source the script **/etc/nova/openrc** to obtain administrative privileges as described in *Linux User Accounts*.

2. Identify the volume you want to back up.

- a) List Cinder volumes from all tenants.

```
~(keystone_admin)$ cinder list --all-tenants
```

ID	Status	Display Name	...	Bootable	Attached to
53ad007a-...	in-use	volume-1	...	false	c820df55-...
578da734-...	in-use		...	true	c820df55-...
593111d8-...	available	volume-2	...	true	

Cinder volumes are also listed on the Volumes page of the Web administration interface.

- b) Identify the volume to back up.

In the list above, two volumes are listed in the *in-use* state:

- A volume named **volume-1** with ID **53ad007a-...**, attached to the running image **c820df55-....**. The bootable flag is *false*, which indicates that this is not a disk image.
- An unnamed volume with ID **578da734-...**, attached to the running image **c820df55-....**. The bootable flag is *true*, which indicates that this is the instance's disk image. Unnamed volumes are dynamically created when the option **Instance Boot Source** in the Launch Instance window of the Web administration interface is set to *Boot from image (creates a new volume)* or *Boot from volume snapshot (creates a new volume)*.

This example uses the unnamed disk image to illustrate the backup procedure, but you must back up all *in-use* volumes appropriately.

3. Create a volume snapshot.

```
~(keystone_admin)$ cinder snapshot-create --force True --display-name test-vm-snapshot \
578da734-a416-47e7-97de-137aa10b90f8
```

Property	Value
created_at	2014-09-19T14:18:47.269583
display_description	None
display_name	test-vm-snapshot
id	53b5237c-5cd0-4939-ad6b-cf612e449216
metadata	{}
size	1
status	creating
volume_id	578da734-a416-47e7-97de-137aa10b90f8

The status of the new snapshot is *creating*, which means that it is in process. You must wait until it becomes *available* before proceeding.

You can verify the status of all Cinder snapshots using the following command:

```
~(keystone_admin)$ cinder snapshot-list --all-tenants
```

ID	Volume ID	Status	Display Name	Size
53b5237c-...	578da734-...	available	test-vm-snapshot	1

In this example, the state of the snapshot **test-vm-snapshot** is *available* already.

4. Export the snapshot to the controller's file system.

```
~(keystone_admin)$ cinder snapshot-export test-vm-snapshot
```

Property	Value
----------	-------



display_description	None
id	53b5237c-5cd0-4939-ad6b-cf612e449216
status	exporting
updated_at	2014-09-19T14:18:47.393220
volume_size	1
volume_type	None

The status of the snapshot is *exporting*, which means that it is not yet ready on the controller's file system. As before, you can use the **cinder snapshot-show** *UUID* command to verify the current status, and wait until the snapshot's state is *available* again.

When the export operation completes, the backup file is ready, as illustrated in the following example:

```
~(keystone_admin)$ ls -lh /opt/backups
total 258M
drwx----- 2 root root 16K Sep 10 17:42 lost+found
drwxr-xr-x 2 root root 4.0K Sep 19 14:30 temp
-rw-r--r-- 1 root root 258M Sep 19 14:30 volume-578da734-
a416-47e7-97de-137aa10b90f8-20140919-142732.tgz
```

Note that the UUID used in the file name is the volume's and not the snapshot's.

##### 5. Transfer the backup file to an external storage resource.

The following example uses the **scp** command to transfer the backup file to a server named **backups-server.wrs.com** in the directory **/backups/titanium**:

```
~(keystone_admin)$ scp /opt/backups/volume-578da734-
a416-47e7-97de-137aa10b90f8-20140919-142732.tgz \
admin@backups-server.wrs.com:/backups/titanium
```

##### 6. Delete the snapshot.

```
~(keystone_admin)$ cinder snapshot-delete test-vm-snapshot
```

You can confirm that the snapshot is deleted using **cinder snapshot-list --all-tenants**.

If a standard deletion fails, you can force deletion using the following command:

```
~(keystone_admin)$ cinder snapshot-force-delete snapshot-id [snapshot-id]
```

Once the snapshot is exported, you should remove it from the Cinder repositories to free up disk space.

##### 7. Delete the backup file.

You may want to free up disk space on the controller to accommodate additional Cinder volume backups.

The volume backup is complete. Repeat this procedure with all other Cinder volumes in the *in-use* state in the system.



# 6

## *Virtual Router Management*

Virtual Routers	47
Virtual Router Administration	52
Adding Virtual Routers	53
Adding a Gateway to a Virtual Router	54
Configuring SNAT on a Virtual Router	56
Configuring Port-based DNAT on a Virtual Router	57
Adding and Maintaining Routes for a VXLAN Network	60
Address Filtering on Virtual Interfaces	61

### Virtual Routers

Virtual routers provide internal and external network connectivity for tenants.

The user associated with a tenant can add a designated number of virtual routers (Neutron routers) to the tenant. The maximum number is specified in the quotas for the tenant.

The virtual router automatically provides a connection to system services required by the tenant, such as the metadata service that supplies instances with user data. You can configure the virtual routers with interfaces to tenant networks to provide internal and external connectivity.

A virtual router can be implemented as a centralized router, or a distributed virtual router (DVR).

Only the **admin** user can specify a distributed router. For other tenants, this choice is not available, and a centralized router is implemented by default. The **admin** user can change a centralized router to a distributed router on behalf of other tenants (see *Titanium Server Tenant User's Guide*: [Virtual Router Administration](#) on page 52).

### Centralized

A centralized virtual router is instantiated on a single compute node. All traffic using the router must pass through the compute node.

### Distributed

A distributed virtual router is instantiated in a distributed manner across multiple hosts. Distributed virtual routers provide more efficient routing than standard virtual routers for east-west (tenant-to-tenant) or floating IP address traffic. Local traffic is routed within the local host, while external L3 traffic is routed directly between the local host and the gateway router.

To implement the distributed model, a centralized-portion of the router is still deployed on one host. The centralized portion manages north-south (external network) traffic and source network address translation (SNAT) traffic, as well as agents deployed on other hosts. The agents offload the centralized router for east-west (tenant-to-tenant) routing and floating IP network address translation.

In some cases, the use of virtual routers on the tenant networks can result in multiple default routes for a virtual machine (VM). If this happens, you can establish alternative VM access to the metadata server; see [#unique\\_23/unique\\_23.Connect 42.accessing\\_metadata\\_server](#) on page 49.

You can enable SNAT on a virtual router. For more information, see *Titanium Server Tenant User's Guide*: [Configuring SNAT on a Virtual Router](#) on page 56.

You can also enable DNAT on a virtual router. For more information, see *Titanium Server Tenant User's Guide*: [Configuring Port-based DNAT on a Virtual Router](#) on page 57.

To add a virtual router, see *Titanium Server Tenant User's Guide*: [Adding Virtual Routers](#) on page 53. To create interfaces, see *Titanium Server Cloud Administration: Adding Virtual Router Interfaces*.

## Managed and Unmanaged Subnets

Use the **System Managed Subnet** and **Enable DHCP** subnet attributes to determine how IP addresses are allocated and offered on an IP subnet.

With the proper configuration in place, DHCP services can be provided by the built-in Neutron DHCP server, by a standalone server available from an external network infrastructure, or by both.

When creating a new IP subnet for a tenant network you can specify the following attributes:

### System Managed Subnet

When this attribute is enabled, the subnet is *system managed*. The Neutron service automatically allocates an IP address from the address allocation pools defined for the subnet to any new virtual machine (VM) instance with a virtual Ethernet interface attached to the tenant network. Once allocated, the pair (MAC address, IP address) is registered in the Neutron database as part of the overall registration process for the new virtual machine.

When the system managed subnet attribute is disabled, the subnet is *unmanaged*. No automatic allocation of IP addresses takes place, and the Neutron DHCP service for the subnet is disabled. Allocation of IP addresses for new virtual machines must be done at boot time using the CLI or the API interfaces.

## Enable DHCP

When this attribute is enabled, a virtual DHCP server becomes available when the subnet is created. It uses the (MAC address, IP address) pairs registered in the Neutron database to offer IP addresses in response to DHCP discovery requests broadcast on the subnet. DHCP discovery requests from unknown MAC addresses are ignored.

The Neutron DHCP server can only be enabled on system managed subnets. DHCP services for unmanaged subnets, if required, must be provisioned by external, non-Neutron, DHCP servers.

When the DHCP attribute is disabled, all DHCP and DNS services, and all static routes, if any, must be provisioned externally.

## Allocation Pools

This is a list attribute where each element in the list specifies an IP address range, or address pool, in the subnet address space that can be used for dynamic offering of IP addresses. By default there is a single allocation pool comprised of the entire subnet's IP address space, with the exception of the default gateway's IP address.

An external, non-Neutron, DHCP server can be attached to a system managed subnet to support specific deployment needs as required. For example, it can be configured to offer IP addresses on ranges outside the Neutron allocation pools to service physical devices attached to the tenant network, such as testing equipment and servers.

Allocation pools can only be specified on system managed subnets.

The method used to access the server depends on your deployment scenario.

- If the VM is attached to a tenant network with a virtual router then the metadata server is reachable using the virtual router as the route gateway.
- If the VM instance is attached to multiple tenant networks, each with access to a virtual router, then any of the virtual routers provides access to the metadata server. However, installing multiple default routes on the VM might impact the VM's ability to route packets back to external networks.
- If the VM is attached to a tenant network that does not have a virtual router then the route to the metadata server can be retrieved from the Neutron DHCP service, provided that this service is enabled. If DHCP is enabled on a tenant network, then the VM can use DHCP option 121, Classless Static Route, to retrieve the metadata server static route information, which uses the DHCP server's address as the gateway.

Note that when using DHCP option 121, the answer from the DHCP server will also include other applicable static routes. They include:

- any static routes configured when the IP subnet was created
- default route to the subnet gateway IP address, if one is configured on the subnet
- on-link routes for all other subnets on the same network and same guest VLAN

The DHCP service only responds to option 121 if there are no virtual routers on the network.

The following requirements must be satisfied in order for a guest application to access the metadata service:

- There is a route table entry to route traffic destined to the 169.254.169.254 address via a Neutron router, or via a suitable static route to the 169.254.169.254 address.
- The metadata server knows about the virtual machine instance associated with the MAC and IP addresses of the virtual Ethernet interface issuing the metadata service request. This is

necessary for the metadata server to be able to validate the request, and to identify the virtual machine's specific data to be returned.

On system managed subnets, the Neutron service has all address information associated with the virtual machines in its database.

On unmanaged subnets, you must tell the Neutron service the IP address of the network interface issuing the metadata service requests.

### DiffServ-Based Priority Queuing and Scheduling

Differentiated Services, or DiffServ, is a packet-based traffic management mechanism that allows end devices to specify an expected per-hop behavior (PHB) from upstream switches and routers when handling their traffic during overload conditions.

DiffServ marking and architecture are IEEE specifications, IEEE RFC 2474 and RFC 2475.

Support for DiffServ is implemented on the AVS for all input and output ports, on all attached tenant networks. On each port, it uses eight queues associated with the CS0 to CS7 DiffServ class selectors, which are processed by a round-robin scheduler with weights of 1, 1, 2, 2, 4, 8, 16, and 32. Overflow traffic is tail-drop discarded on each queue. Guest applications in the Titanium Server cluster can set a Differentiated Services Code Point (DSCP) value in the Differentiated Service (DS) field of the IP header to mark the desired upstream PHB.

On ingress, a packet being processed to be sent to the VM is directed to the appropriate DiffServ output queue. On overflow, that is, if the virtual machine cannot keep up with the incoming traffic rate, lower priority packets are discarded first.

On egress, a packet sent from the virtual machine (VM) to the AVS is first enqueued into the appropriate DiffServ input queue according to the specified DSCP value, the CS0 queue being the default. On overload, that is, if the AVS cannot keep up with the incoming traffic, lower priority packets are discarded first; in this case, you should consider re-engineering the AVS, possibly assigning it more processing cores. Once serviced by the DiffServ class scheduler, the packet is processed according to the QoS policy in place for the tenant network, as described in *Titanium Server Tenant User's Guide*: [Quality of Service Policies](#) on page 50.

### Quality of Service Policies

Quality of Service (QoS) policies specify relative packet processing priorities applied by the AVS switch on each compute node to incoming tenant network's traffic during overload conditions.

The QoS polices play no role under normal traffic loads, when no input traffic queues in the AVS are close to their overflow limits.

QoS policies are created by the cluster administrator, and selected by the tenant users to apply on a per-tenant network basis. To create a new QoS policy, log in as administrator, select **Admin > System > Networks**, and then select the **QoS Policies** tab.

Networks

Qos Policies

Filter

Q

Create QoS Policy

Project	Name	Policy	Actions
No items to display.			

Click the button **Create QoS Policy** to display the Create QoS Policy window.

**Create QoS Policy** ✕

**Name \***

**Description**

**Scheduler Weight \***

**Project**

**Description:**  
You can create a qos policy to be assigned to one or more tenant networks.

Cancel

Create QoS Policy

The following fields are available:

#### Name

The name of the new QoS policy. This is the name that the tenant user sees as available for use.

This field is mandatory.

#### Description

A free form text for your reference.

#### Scheduler Weight

The relative weight the AVS traffic scheduler uses on overload conditions, as compared with the scheduler weight of all other QoS policies.

The scheduler weight is a positive integer number associated with each QoS policy. On overload, the AVS schedules traffic processing for each tenant network according to its assigned QoS policy. By default, with no specific QoS policies defined, traffic from all tenant networks is processed in round-robin mode, one tenant network after another. Effectively, the default behavior is similar to assigning a scheduler weight of 1 to all tenant networks.

When a specific QoS policy with a scheduler weight greater than 1 is applied to a tenant network, its traffic is scheduled with a relative higher frequency. For example, if the scheduler weight for tenant network A is 10, and the one for tenant network B is 100, then on overload, tenant network B will see its queued traffic processed 10 times as often as tenant network A.

The handling of the scheduler weight is implemented as a per-packet token bucket for each tenant network. This means that each unit in the scheduler weight counts as one packet to be processed in sequence. In the previous example, the AVS processes 10 consecutive packets from tenant network A, followed by 100 packets from tenant network B. This implementation ensures a fair-share behavior that prevents any tenant network from running into total bandwidth starvation, even if its scheduler weight is relatively low.

The range of values for the scheduler weight is arbitrary. You must however ensure that the values assigned to the different policies make sense for the intended applications.

This field is mandatory.

## Project

A drop-down menu displaying the currently defined tenants (projects). The new QoS policy is available to only the selected tenant. If no tenant is selected, it is available to all tenants in the cluster.

Tenant users can select available QoS policies when the tenant networks are created from the Create Network window. They can also apply a QoS policy to an already existing tenant network from the Edit Network window.

## Virtual Router Administration

Users with administrative privileges can review and edit router information for all tenants.

The Routers page accessed from **Admin > System > Routers** provides central control of all virtual routers. From this page, you can edit router names, change router types, change the status of routers (**Up** or **Down**), and delete routers.

Only users with administrative privileges can convert a centralized router to a distributed router.

An administrative user can delete routers on any tenant. A non-administrative user can delete routers on the tenant associated with the user.

As an alternative to the web administration interface, you can use the CLI. The following examples assume you have become the **admin** user.

- To list routers:

```
~(keystone_admin)$ neutron router-list
```

- To show details for a router:

```
~(keystone_admin)$ neutron router-show router_id
```

where *router\_id* is the name or UUID of the router.

- To update a router:

```
~(keystone_admin)$ neutron router-update router_id --distributed=True
```

where *router\_id* is the name or UUID of the router. This example updates a router to make it a distributed virtual router.

- To delete a router:

```
~(keystone_admin)$ neutron router-delete router_id
```



## Adding Virtual Routers

You can add virtual routers to a tenant using the web administration interface or the CLI. To add a router to a tenant, you must be logged in as the user associated with the tenant.



---

**NOTE:** Only the **admin** tenant can add a distributed virtual router.

---

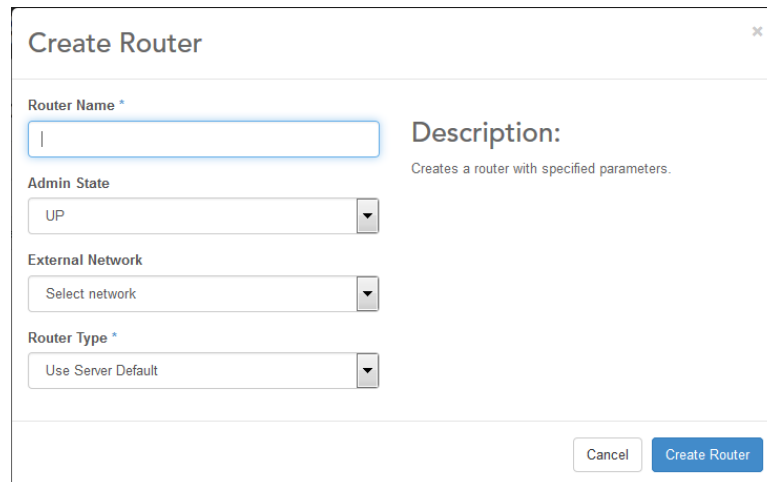
### Procedure

1. Log in as the user associated with the tenant.

The following instructions assume that the **admin** tenant is logged in.

2. Select **Project > Network > Routers**.
3. On the Routers page, click **Create Router**.

The Create Router dialog box appears.



The 'Create Router' dialog box is shown. It has a title bar with a close button. Inside, there are four input fields: 'Router Name' (required, with an asterisk), 'Admin State' (set to 'UP'), 'External Network' (set to 'Select network'), and 'Router Type' (required, with an asterisk, set to 'Use Server Default'). To the right of these fields is a 'Description:' section with the text 'Creates a router with specified parameters.' At the bottom right are two buttons: 'Cancel' and 'Create Router'.

4. Provide a **Router Name**.
5. Optional: To add a connection to a tenant network configured for external access, select from the **External Network** drop-down menu.

This adds a gateway for the router. Alternatively, you can add a gateway after the router is created. For more information, see *Adding Virtual Router Interfaces*.

6. Select the **Router Type**.



---

**NOTE:** The **Router Type** field is shown only for the **admin** user. For other tenants, a centralized router is added. The **admin** user can change this setting on behalf of another tenant after the router has been added (see [Virtual Router Administration](#) on page 52).

---

#### Use Server Default

Use the default setting specified on the controller (for a standard Titanium Server installation, centralized).

#### Centralized

Add a centralized virtual router.

#### Distributed

Add a distributed virtual router (DVR).

For more information about these choices, see [Virtual Routers](#) on page 47.

7. To save your settings and close the dialog box, click **Create Router**.

### Postrequisites

To attach router interfaces to tenant networks, see *Adding Virtual Router Interfaces*.

## Adding Virtual Routers Using the CLI

As an alternative to the web administration interface, you can use CLI commands to add a virtual router. First, become the appropriate keystone user.

You must be logged in as the **admin** user.

To create a router from the CLI, use the **neutron router-create** command. For example:

```
~(keystone_admin)$ neutron router-create router_name --distributed=True
```

Where *router\_name* is a name assigned to the router.

This example creates a distributed virtual router. If the **--distributed** option is omitted, the default setting (centralized) is used.

## Adding a Gateway to a Virtual Router

You can configure a gateway for a virtual router as a tenant or an administrator. Users with administrator privileges have access to additional controls.

### Procedure

1. Log in as the user associated with the tenant.
2. Open the Routers page.
  - To open the page with tenant privileges, select **Project > Network > Routers**.
  - To open the page with administrator privileges, select **Admin > System > Routers**.
3. In the list of existing routers, click the name of the router to be configured.

The Router Details page appears.

**Router Details** 9/1/2015, 3:12:50 PM

Set Gateway

Overview Interfaces Port Forwarding

Name	tenant1-router
ID	c42238b0-9038-4265-8d36-c6224c48187c
Project ID	1180117155a243f691abc2efae9a5dbd
Status	ACTIVE
Admin State	UP
External Gateway	None

4. On the **Interfaces** tab, click **Set Gateway**.



**NOTE:** If a gateway has already been added, a **Clear Gateway** button is present instead.

The Set Gateway dialog box appears.

Set Gateway

External Network \*

Select network

Router Name \*

tenant1-router

Router ID \*

c42238b0-9038-4265-8d36-c6224c48187c

IP Address (optional) ⓘ

☒ Enable SNAT

Description:

You can connect a specified external network to the router. The external network is regarded as a default route of the router and the router acts as a gateway for external connectivity.

Cancel

Set Gateway

### External Network

Select a tenant network configured to provide external access.

### Router Name

This field is shown for information only.

### Router ID

This field is shown for information only.

### IP Address (optional)

This field is shown only for users with administrator privileges. You can optionally specify an IP address for the gateway interface. By default, the server allocates the next available external IP address.

### Enable SNAT

This control is shown only for users with administrator privileges. For more about the SNAT option, see [Configuring SNAT on a Virtual Router](#) on page 56.

To save the gateway gettings and close the dialog box, click **Set Gateway**.

## Configuring SNAT on a Virtual Router

You can enable or disable source network address translation (SNAT) for each individual external network connection on a virtual router. By default, SNAT is enabled for all external networks.

Both SNAT and port-based DNAT (destination network address translation) use tables to translate between IP addresses and ports used on an internal network, and an IP address and multiple ports used on an external network. However, SNAT provides translation for internally initiated UDP and TCP traffic flows only. An instance connected to an internal tenant network can initiate connections to servers on an external, usually public, network, but external servers cannot initiate connections to external addresses mapped to internal addresses. For externally initiated connections, the use of floating IP addresses or port-based DNAT is required. For more about DNAT, see [Configuring Port-based DNAT on a Virtual Router](#) on page 57.

System administrators can enable or disable SNAT using the following command:

```
~(keystone_admin)$ neutron router-update routename --external_gateway_info type=dict \
network_id=externalnetwork,enable_snat=boolean
```

where:

**routename**

is the name or UUID of the virtual router

**externalnetwork**

is the name or UUID of the external network

**boolean**

is **true** (to enable SNAT) or **false** (to disable SNAT)



---

**CAUTION:** For SNMP implemented on guests, the following limitations apply when SNAT is enabled:

- An SNMP agent on an internal network cannot receive requests from SNMP managers on external networks.
- An SNMP manager on an internal network cannot receive traps from SNMP agents on external networks, even if communication is initiated by the manager. This is because SNMP traps are sent to UDP port 162, not to the source port.

Only agent responses to requests from a manager on the internal network are processed appropriately.

To overcome these limitations, you can use floating IP addresses or make internal addresses public. For more information, see *Titanium Server Cloud Administration: [Connecting Virtual Machines to External Networks](#)* on page 18.

---

## Configuring Port-based DNAT on a Virtual Router

You can use port-based destination network address translation (DNAT), also known as port forwarding, to support externally-initiated connections to multiple VMs using a single external IP address.

As with SNAT (source network address translation), port-based DNAT uses a network address translation table to translate between IP addresses and ports used on an internal network, and the gateway IP address of a router attached to an external network. With port-based DNAT, internally initiated connections are not required to establish or maintain table entries.

As an alternative to the web administration interface, you can use the CLI. For more information, see [Configuring Port-based DNAT Using the CLI](#) on page 59.

### Prerequisites

Before you can use port-based DNAT, you must enable SNAT to activate support for network address translation tables. For more information, see [Configuring SNAT on a Virtual Router](#) on page 56.

Before you can add an entry, a VM IP address is required. IP addresses are assigned to instances when they are launched, and preserved until they are deleted.

### Procedure

1. On the **Admin** menu of the web administration interface, in the **System** section, select **Routers**.
2. On the Routers page, click the **Name** of the router to be configured for port-based DNAT.

The screenshot shows the 'Routers' page in a web administration interface. At the top, there is a navigation bar with 'admin' and 'Logged in as: admin'. Below this, the page title 'Routers' is displayed along with the date and time 'Tue Nov 10 05:42:45 2015'. A search filter is present on the right. Below the filter, there is a table with the following columns: Name, Status, Distributed, External Network, Admin State, Project, Host, and Actions. The table contains one entry: 'tenant1-router' with status 'Active', 'No' for distributed, 'external-net' for external network, 'UP' for admin state, 'tenant1' for project, and 'compute-0' for host. The Actions column for this entry has a 'Clear Gateway' button. At the bottom of the table, it says 'Displaying 1 item'.

<input type="checkbox"/>	Name	Status	Distributed	External Network	Admin State	Project	Host	Actions
<input type="checkbox"/>	tenant1-router	Active	No	external-net	UP	tenant1	compute-0	Clear Gateway

3. On the Router Details page, select the **Port Forwarding** tab.

admin

Router Details Mon Oct 26 10:13:26 2015

Logged in as: admin Settings Help Sign Out

Clear Gateway

Overview Interfaces Port Forwarding

Port Forwarding Rules

+ Add Rule

	Port	Private Address	Private Port	Public Port	Protocol	Description	Actions
No items to display.							
Displaying 0 items							

- To add a rule, click **Add Rule**, and then complete the Add Port Forwarding Rule dialog box.

Add Port Forwarding Rule

IP Address \*

192.168.102.3 : instance-1

Private Port \*

1234

Public Port \*

5678

Protocol \*

TCP

Description

Router Name \*

tenant1-router

Router ID \*

722f8091-157b-48dd-8372-0e1315d5177d

Description:

You can create a port forwarding rule from a public port to a private address and port.

Cancel

Add Rule

#### IP Address

The IP address of the VM. You can select existing addresses from the drop-down list.

#### Private Port

The port to use on the VM IP address.

#### Public Port

The port on the external gateway interface of the router.

#### Protocol

The applicable protocol for the rule. Select from the drop-down list.

#### Description

An optional text description of the rule.

#### Router Name

The name of the router (shown for information only).

#### Router ID

The UUID of the router (shown for information only).

### 4. Click **Add Rule** to apply the settings.

The new rule is displayed.

<div>Overview Interfaces <b>Port Forwarding</b></div>							
<div>Port Forwarding Rules <span>+ Add Rule</span> <span>✖ Delete Rule</span></div>							
<input type="checkbox"/>	Port	Private Address	Private Port	Public Port	Protocol	Description	Actions
<input type="checkbox"/>	(76f0b669)	192.168.102.3	1234	5678	tcp		<div>Update Rule</div> <div></div>
Displaying 1 item							

## Configuring Port-based DNAT Using the CLI

If you prefer, you can use the command-line interface to configure port-based DNAT for external access to VMs.

For more about using port-based DNAT, including instructions for using the web administration interface, see [Configuring Port-based DNAT on a Virtual Router](#) on page 57.

To add a network translation table entry, use a command of the following form:

```
~(keystone_admin)$ neutron portforwarding-create router --inside-addr addr \
--inside-port inside_port --outside-port outside_port --protocol protocol_type \
[--description desc]
```

where

router

is the name of the virtual router

addr

is the IP address of the VM interface connected to the virtual router

inside\_port

is a port on the VM IP address

outside\_port

is the port on the external gateway interface of the router

protocol\_type

is the protocol (**tcp**, **udp**, **udp-lite**, **scrp**, or **dccp**)

desc

is an optional text description of the entry

To obtain the VM IP address, use the **nova show** command and provide the instance UUID.

For example:

```
~(keystone_admin)$ nova show 915ea4da-1dd8-44ad-8a5b-375b9af6237f | grep network
| internal-net network          | 10.1.1.2      |
| tenant1-mgmt-net network      | 192.168.102.3 |
| tenant1-net network           | 172.31.1.2    |
...
~(keystone_admin)$ neutron portforwarding-create tenant1-router \
--inside-addr 192.168.102.3 --inside-port 1234 \
--outside-port 5678 --protocol tcp --description port_forwarding_rule
Created a new portforwarding:
+-----+-----+
| Field      | Value                                     |
+-----+-----+
| description | port_forwarding_rule                   |
| id          | c5cbf9a8-b041-4ce4-bb55-5527722eebf8   |
| inside_addr | 192.168.102.3                           |
| inside_port | 1234                                     |
| outside_port | 5678                                    |
| port_id     | 76f0b669-f493-4e5a-8302-4baeadab90c6   |
| protocol    | tcp                                     |
| router_id   | 722f8091-157b-488d-8372-0e1315d5177d   |
| tenant_id   | b75b40702c054c78b8cf22860e2ded51      |
+-----+-----+
```

To modify an entry, use a command of the following form:

```
~(keystone_admin)$ neutron portforwarding-update router [--inside-addr addr] \
[--inside-port inside_port] [--outside-port outside_port] \
[--protocol protocol_type] [--description desc]
```

To view entries for a virtual router, use a command of the following form:

```
~(keystone_admin)$ neutron portforwarding-show router_uuid
```

To view entries for all routers, use the following command:

```
~(keystone_admin)$ neutron portforwarding-list
```

### Prerequisites

Before you can use port-based DNAT, you must enable SNAT to activate support for network address translation tables. For more information, see [Configuring SNAT on a Virtual Router](#) on page 56.

## Adding and Maintaining Routes for a VXLAN Network

You can add or delete routing table entries for hosts on a VXLAN network using the CLI.

### Prerequisites

The compute node must be locked.



To add routes, use the following command.

```
~(keystone_admin)$ system host-route-add node ifname network prefix gateway metric
```

where

**node**

is the name or UUID of the compute node

**ifname**

is the name of the interface

**network**

is an IPv4 or IPv6 network address

**prefix**

is the netmask length for the network address

**gateway**

is the default gateway

**metric**

is the cost of the route (the number of hops)

To delete routes, use the following command.

```
~(keystone_admin)$ system host-route-delete uuid ifname network prefix gateway metric
```

where **uuid** is the UUID of the route to be deleted.

To list existing routes, including their UUIDs, use the following command.

```
~(keystone_admin)$ system host-route-list compute-0
```

## Address Filtering on Virtual Interfaces

The AVS on compute nodes can be configured to filter out packets based on the source MAC address.

MAC addresses for virtual network interfaces on virtual machines are dynamically allocated by the system. For most scenarios, the assigned MAC addresses are expected to be used on all outgoing packets from the virtual machine instances. However, there are scenarios where the source MAC address is not expected to match the original assignment, such as when a L2 switch is implemented internally on the virtual machine.

By default, the AVS on compute nodes accepts any source MAC address on the attached virtual network interfaces. However, it can be configured to filter out all incoming packets with non-system-generated source MAC address, if required. When evaluating the use of the filtering capability, you must consider the following:

- Source MAC address filtering can be enabled and disabled by the administrator user only, not by tenants.
- Filtering is enabled on a per-tenant basis only. Higher granularity, such as per-instance filtering, is not supported.
- When enabled, source MAC address filtering applies to all new virtual interfaces created by the Neutron service. Address filtering is not active on virtual interfaces created before filtering is enabled.

You can use the following CLI command to enable source MAC address filtering:

```
~(keystone_admin)$ neutron setting-update --tenant-id=<TENANTID> \  
--mac-filtering={True|False}
```

# Stack Management

Managing Stacks 63

## Managing Stacks

You can create and manage collections of resources, or *stacks*, using Heat, the OpenStack orchestration service. Titanium Server extensions are included for enhanced scope and reliability.

With Heat, you can define a stack configuration in a *template* file, and then apply the template to create or modify the stack resources and connections. The Heat orchestration layer includes life-cycle management features to simplify the addition, modification, and deletion of stacks.

## Stacks

A collection of resources created and managed using the Heat orchestration service is called a *stack*.

Stack resources can be flavors, images, instances, tenant networks, volumes, security groups, users, floating IP addresses, or any other entities defined and configured in Heat templates. Most types of resources offered by OpenStack can be defined in Heat templates and included in stacks.

Using the CLI, you can obtain information on existing stacks in several ways:

- **heat stack-list** shows the names of stacks
- **heat stack-show** *name* shows details about the named stack
- **heat stack-create** *name* creates the named stack with given parameters
- **heat stack-modify** *name* modifies the named stack according to given parameters
- **heat stack-delete** *name* deletes the named stack

Stacks can include *static* and *autoscaling* resources.

#### Static resource

A resource that is defined when the stack is launched, and does not change unless a **heat stack-modify** command is issued.

#### Autoscaling resource

A resource that can be created, modified, or removed in response to performance metrics collected by Ceilometer. These can include metrics generated by the Titanium Server or OpenStack platform, and metrics generated by VM instances using CloudWatch.

## Templates

A template is a formal set of instructions for defining and configuring a stack.

Templates specify the resources to include in a stack, and relationships between them.

A template contains several sections, including:

#### Description

You can use this section to provide comments about the stack, such as its purpose or any special considerations when using it.

#### Parameters

This section defines the parameters that you can pass to the stack template. You can pass parameters using the command line, a web form, or an environment file.

You can include a default value for a parameter, and optionally pass a different value when the template is deployed. If no default value is included, you must provide a value when the template is deployed. Parameters without default values are called *mandatory parameters*.

#### Resources

This section defines the resources to be created. They can be specified in any order.

There are two official formats for templates: CloudFormation (CFN) and Heat Orchestration Template (HOT).

#### CFN

This format is associated with Amazon Web Services (AWS) technology. A CFN template is indicated by the first line:

```
HeatTemplateFormatVersion: '2012-12-12'
```

#### HOT

This format is associated specifically with the OpenStack project. A HOT template is indicated by the first line:

```
heat_template_version: 2013-05-23
```

In HOT, some of the built-in functions are different, and lowercase is used for attribute names.

Both formats support Amazon Web Services (AWS) resources and OpenStack resources, in any combination.

Titanium Server extensions to Heat functionality are supported only on OpenStack resources, which are the resources primarily used with HOT format.

Titanium Server includes sample stack templates in HOT format for demonstration purposes (see [Sample Templates for Titanium Server](#) on page 71). Example CFN-formatted Heat templates can be found at <https://github.com/openstack/heat-templates/tree/master/cfn>. The templates are expressed using YAML, which is the recommended notation for HOT. Heat templates may also be expressed using JSON, which is the usual notation for CFN.

For more information on the content and syntax of Heat templates, see the online [OpenStack Template Guide](#).

## Parameters

You can specify parameters for a stack, such as authentication, storage, networking, and so on, when you invoke a template.

You can supply parameter values by:

- including them in the template as defaults
- specifying each one using a form in the web administration interface, or using the **-parameters** option on the **heat create-stack** command line
- saving them in an environment file, which you can specify using the **-e** option on the **heat create-stack** command line

This enables you to customize the behavior of a common stack template when the stack is created or modified.

You can use environment files or command-line entries to supplement parameters in the template, or to override them if they already exist. Environment-file values override existing values in the template. Command-line values override existing values in the environment file or the template.

The syntax for specifying parameters is as follows:

- on the command line, using the **-P** option. Multiple **-P** arguments are supported.

```
~(keystone_admin)$ heat stack-create -P key1=val1 -P key2=val2
```

- in an environment file:

```
parameters:
  key:value
  key:value
```

## Titanium Server Extensions to Heat

Several extensions to Heat are included with Titanium Server.

The Titanium Server Extensions are supported only for OpenStack resources, which are the primary resources used with HOT format.

### Multiple NIC Support

When launching a VM instance initially or in an autoscaling stack, you can specify multiple network interfaces. The open-source version of Heat allows only a single network interface to be specified at launch.

The syntax for the extension is as follows:

```
...
vm:
  type: OS::Nova::Server
  properties:
    ...
    networks :
      - { network: 'tenant1-mgmt-net', vif-model: virtio}
      - { network: { get_param: port0 }, vif-model: avp}
      - { network: { get_param: port1 }, vif-model: avp}
    ...
```

As the example shows, this extension also adds the ability to specify a different **vif-model** for each interface. In addition, a new **avp** option is introduced, supporting the use of optimized AVP device drivers.

The valid **vif-model** values are as follows:

- **avp** (Accelerated Virtual Port)
- **e1000** (Intel e1000 Emulation)
- **ne2k\_pci** (NE2000 Emulation)
- **pcnet** (AMD PCnet/PCI Emulation)
- **rtl8139** (Realtek 8139 Emulation)
- **virtio** (VirtIO Network)
- **pci-passthrough** (PCI Passthrough Device)
- **pci-sriov** (SR-IOV device)

### Simplified VM Instance Naming

Titanium Server introduces minor changes to the OpenStack VM naming convention to make Heat-generated names more user-friendly.

- For a static resource, launched VM instances are named using the **name** attribute of the **OS::Nova::Server** structure (without including the **<StackTemplateName>**).

```
Serving_Gateway:
  Type: OS::Nova::Server
  Properties:
    name : 'Serving_Gateway'
    image : 'cirros'
    flavor: 'm1.tiny'

~(keystone_admin)$ heat stack-create -f <file> EPC
~(keystone_admin)$ nova list
```

ID	Name	Status	...
581b3495-3cf1-4410-9587-5cf04fccfed2	Serving_Gateway	ACTIVE	...

### Enhanced Support for Server Groups

You can specify a maximum group size and a best-effort attribute for Server Groups.

The syntax for adding a Server Group Resource is as follows:

```
resources:
  OS_Nova_ServerGroup:
```

```
type: OS::Nova::ServerGroup
properties:
  policies: ['affinity']
  best_effort: true
  group_size: 2
```

### Relaxed Requirements for Passing User Data

The property **UserData** is a Wind River template extension that you can use to pass user data to an instance even if the instance does not have **cloud-init** installed. For more information, see [Customizing Images with User Data In a Heat Template](#) on page 70.

### Improved User Access to Stacks

Stacks can be created, modified, or deleted by admin or non-admin users.

### Greater Control over Resource Allocations

When creating a network resource using **OS::Neutron::Net**, you can use a **depends\_on** attribute to ensure that the requirements of other resources are given priority before the resource is created. The attribute takes another resource as an argument. In the following example, it is used to specify that the resource **external\_network** must be created before **internal\_network** is created.

```
internal_network:
  type: OS::Neutron::Net
  properties:
    name: { get_param: INTERNALNET }
    depends_on: { get_resource: external_network }
    shared: false
    tenant_id: {get_param: TENANT_ID}
```

### Support for Local Files in Glance Image

Support is added for specifying a local file or web URL as the location of a Glance image.

```
...
wrl5_Glance_Image:
  type: OS::Glance::Image
  properties:
    name: {get_param: IMAGE_NAME }
    visibility: public
    container_format : bare
    disk_format : qcow2
    location: file:///home/wrsroot/images/cust-guest.img
...
```

### Support for name Property in Nova Flavor

A **name** property is added to the Flavor resource to support user-friendly names for Heat-generated flavors.

### Additional Heat Resources

In addition to the standard OpenStack resources available for Heat templates, you can use the following resources:

**OS::WR::ScalingPolicy**

Defines a Resource UP/Down autoscaling property.

- WR::Neutron::ProviderNet

Defines a provider network.
- WR::Neutron::ProviderNetRange

Defines a segmentation range for a provider network.
- WR::Neutron::QoSPolicy

Defines a packet scheduling weight that can be referenced by a tenant network (OS::Neutron::Net).
- WR::Neutron::PortForwarding

Defines a destination NAT mapping between an external IP address (a router address) and external port, and an internal (VM) IP address and port.

Launching a Stack

You can launch stacks using the web administration interface, or you can use the command-line interface.

These instructions assume you are using the web administration interface. For CLI assistance, use the following command.

```
~(keystone_admin)$ heat help stack-create
```

Prerequisites

Ensure that you have a template for the stack.

Depending on the template, you may need to collect some parameters in advance. For details, see the documentation for the specific template, or refer to the Parameters section of the template.

Procedure

1. Open the list of existing stacks.

From the **Project** menu of the web administration interface, select **Orchestration > Stacks**.

The Stacks list appears.

Stacks

Launch Stack

Stack Name	Created	Updated	Status	Actions
No items to display.				
Displaying 0 items				
2. Launch a new stack and specify a template.

Click **Launch Stack**, and then complete the Select Template dialog box.



**Select Template**

Template Source \*

URL  
URL  
File  
Direct Input

**Description:**  
Use one of the available template source options to specify the template to be used in creating this stack.

Cancel Next

The **Template Source** drop-down list offers several ways to specify a template.

Options	Description
URL	Provides a text field for URL access to a template file.
File	Provides a <b>Browse</b> button for access to locally stored template files.
Direct Input	Provides a text window for direct entry of template content.

3. Complete the Launch Stack form to provide any parameter values required by the template.



**NOTE:** The form is created dynamically based on the Parameters section of the template, and varies depending on the template. The form shown here is an example only.

### Launch Stack

Stack Name \*

Creation Timeout (minutes) \*

60

Rollback On Failure

☐

Password for user "admin" \*

InstanceName1

guest-1

InstanceName2

guest-2

KeyName \*

PublicNetId \*

ImageId \*

InstanceType \*

Description:

Create a new stack with the provided values.

Cancel

Launch

## Customizing Images with User Data In a Heat Template

You can provide bootstrap configuration for an instance at launch by including user data in a Heat template.

You can include user data by defining a **UserData** property for an instance. This sends configuration instructions to the instance at launch. For an example of user data, see the **CombinationAutoScaling.yaml** template included with Titanium Server.

Normally, this requires **cloud-init** to be installed in the guest, because the user data is converted to MIME format and then interpreted by **cloud-init**. For instances that do not include **cloud-init**, you can bypass the MIME conversion of user data by invoking a Wind River extension property and assigning the value **'RAW'**. This allows the VM to retrieve the specified user data in unaltered format through a simple REST API call to the metadata server. The name of the property follows the convention for the associated resource type. For an AWS resource (for example, **AWS::EC2::Instance**), use **UserDataType**. For an OpenStack resource (for example, **OS::Nova::Server**), use **user\_data\_format**. The following code fragment shows the property used with an OpenStack resource.

```
...
Srv:
  type: OS::Nova::Server
```

```
properties:
  name: l3-forwarding
  ...
  user_data_format: 'RAW'
  user_data:
    Fn::Base64:
      Fn::Replace:
        - 'OS::stack_name': {Ref: 'OS::stack_name'}
        - |
          #cloud-config

      runcmd:
        - ifconfig eth1 1.1.1.1/24 up
        - ifconfig eth2 2.2.2.1/24 up
        - echo 1 > /proc/sys/net/ipv4/ip_forward
        - sed -i -e 's#.*\ (net.ipv4.ip_forward)\.*#\1 = 1#g' /etc/sysctl.conf
        - ip route add 192.50.0.0/16 via 1.1.1.10
        - ip route add 192.51.0.0/16 via 2.2.2.10
  ...
```




---

**NOTE:** For a VM to access user data, it must have a DHCP-enabled interface on a tenant network that has a Neutron router. Typically, this is the interface to the VM's OAM network or an internal network. The Neutron router provides a route to the metadata server, which provides instances with access to user data.

---

## Resource Scaling (Autoscaling)

You can use Heat to reassign stack resources automatically to meet changing conditions.

You can define and monitor performance thresholds for metrics such as CPU activity, and then add or remove resources when the thresholds are crossed. This allows you to make efficient use of the hardware in the cluster, by allocating resources only when they are needed, and assigning them where they are most required.

Titanium Server supports two types of scaling:

### In/Out

This type of scaling (also known as *horizontal scaling*) adds or removes instances as needed.

### Up/Down

This type of scaling (also known as *vertical scaling*) increases or decreases resources (for example, vCPUs) for individual instances as needed. For more about up/down scaling, see *Titanium Server Tenant User's Guide: [Scaling Virtual Machine Resources](#) on page 19.*

Performance metrics can be collected and reported by the Titanium Server platform, or by the guests using guest metrics.

## Sample Templates for Titanium Server

You can evaluate selected features of Heat using sample templates. The samples also demonstrate some Titanium Server extensions.

Sample Heat templates are included with the Titanium Server SDK. A brief description is provided for each sample. For more information about the SDK, refer to the *Titanium Server Guest Software Development Kit* documentation.

The samples are also copied to the Titanium Server controllers at system installation. They can be found in the `/etc/heat/templates/hot` directory.

Creating a Static Resource

The **BootFromCinder.yaml** template illustrates the use of templates to create static resources.

This HOT template creates a volume and a VM using this volume. The VM name is specified as a parameter in the template, illustrating the use of Titanium Server extensions to simplify VM instance naming. It is used by the **OS::Nova::Server** resources, which define the construction of the VMs.

The VM has two attached networks.

Procedure

- 1. Collect the required parameter values for this template.
  - a) On the **Admin > System Panel > Flavors** page, locate a **Flavor Name** (a resource profile).
  - b) On the **Project > Compute > Access & Security** page, locate a **Keypair Name** to provide secure access to the stack.
  - c) On the **Admin > System Panel > Images** page, locate an **Image Name** to specify the VM image used by the stack.
  - d) On the **Admin > System Panel > Networks** page, click the appropriate network **Name** to open the Network Overview page, and then obtain the network **ID**.
  - e) On the same page, locate and record an internal network to provide connectivity for the stack.
- 2. Open the list of existing stacks.

From the **Project** menu of the web administration interface, select **Orchestration > Stacks**.  
The Stacks list appears.

Stacks

Launch Stack

Stack Name	Created	Updated	Status	Actions
No items to display.				
Displaying 0 items				

- 3. Launch a new stack and specify a template.

Click **Launch Stack**, and then complete the Select Template dialog box.

Select Template

Template Source \*

URL

URLFileDirect Input

Description:

Use one of the available template source options to specify the template to be used in creating this stack.

Cancel

Next

72

The **Template Source** drop-down list offers several ways to specify a template.

Options	Description
<b>URL</b>	Provides a text field for URL access to a template file.
<b>File</b>	Provides a <b>Browse</b> button for access to locally stored template files.
<b>Direct Input</b>	Provides a text window for direct entry of template content.

4. Select the template, then click **Browse** and select **BootFromCinder.yaml** from the local disk.
5. Use the Launch Stack form to provide the required parameter values for the template.

### Launch Stack

**Stack Name \***

**Creation Timeout (minutes) \***

**Rollback On Failure**

☐

**Password for user "admin" \***

**InstanceName1**

**InstanceName2**

**KeyName \***

**PublicNetId \***

**ImageId \***

**InstanceType \***

**Description:**  
Create a new stack with the provided values.

## Creating an In/Out Autoscaling Resource

The **CombinationAutoscaling.yaml** template illustrates the use of a template to create an in/out autoscaling resource.

This HOT template creates a load balancer VM (**OS::Nova::Server**) and a scalable pool of server VMs (**OS::Heat::AutoScalingGroup** of **OS::Nova::Server**). The associated **OS::Ceilometer::Alarm**

and OS::Heat::ScalingPolicy resources are based on a passed-in platform link utilization meter. Instances are added or removed based on this information.

Use an **admin** account to run this template.

Procedure

1. Collect the required parameter values for this template.
  - a) On the **Admin > System Panel > Flavors** page, locate a **Flavor Name** (a resource profile).
  - b) On the **Project > Compute > Access & Security** page, locate a **Keypair Name** to provide secure access to the stack.
  - c) On the **Admin > System Panel > Images** page, locate an **Image Name** to specify the VM image used by the stack.
  - d) On the **Admin > System Panel > Networks** page, click the **Name** of each required network in turn, in order to open the Network Overview page and obtain the network **ID**. For this template, a public network ID and an internal network ID are required.

2. Open the list of existing stacks.

From the **Project** menu of the web administration interface, select **Orchestration > Stacks**.

The Stacks list appears.

Stacks

Launch Stack

Stack Name	Created	Updated	Status	Actions
No items to display.				
Displaying 0 items				

3. Launch a new stack and specify a template.

Click **Launch Stack**, and then complete the Select Template dialog box.

Select Template

Template Source \*

URL

URL

File

Direct Input

Description:

Use one of the available template source options to specify the template to be used in creating this stack.

Cancel

Next

The **Template Source** drop-down list offers several ways to specify a template.

Options	Description
URL	Provides a text field for URL access to a template file.
File	Provides a <b>Browse</b> button for access to locally stored template files.
Direct Input	Provides a text window for direct entry of template content.

4. Select **File**, then click **Browse** and select **CombinationAutoscaling.yaml** from the local disk.

5. Use the Launch Stack form to provide the required parameter values for the template.

Launch Stack

Stack Name \*

Creation Timeout (minutes) \*

60

Rollback On Failure

☐

Password for user "admin" \*

MetricHighWaterMark

60

MinClusterSize

1

ViifModel

virtio

MaxClusterSize

3

CustomMeterName

guest\_cpu\_avg

Description:

Create a new stack with the provided values.

ImageId \*

KeyName \*

MetricLowWaterMark

5

PublicNetId \*

InstanceType \*

InternalNetId \*

CustomMeterUnit

%

Cancel

Launch



**NOTE:** The parameters are presented in a single column. This image has been edited to present them in two columns.

6. Optional: Adjust existing values to meet your requirements.

Options	Description
KeyName	The <b>KeyPair</b> name from the Access & Security page.
ViFModel	The virtual interface model to use. For valid choices, see <i>Multiple NIC Support</i> in <a href="#">Titanium Server Extensions to Heat</a> on page 65.

Options	Description
<b>InstanceType</b>	The <b>Flavor Name</b> from the Flavors page.
<b>ImageId</b>	The <b>Image Name</b> from the Images page.
<b>PublicNetId</b>	The <b>ID</b> from the Network Overview page.
<b>InternalNetId</b>	The <b>ID</b> from the Network Overview page.
<b>MinClusterSize</b>	The minimum number of VMs to launch. The stack cannot autoscale below this value.
<b>MaxClusterSize</b>	The maximum number of VMs that can be launched.
<b>CustomMeterName</b>	The name of a custom Ceilometer metric to store.
<b>CustomMeterUnit</b>	The unit of measurement to display for the custom metric; for example, 'Percent'.
<b>MetricHighWaterMark</b>	The value that triggers autoscaling to add resources.
<b>MetricLowWaterMark</b>	The value that triggers autoscaling to remove resources.

## Creating an Up/Down Autoscaling Resource

The **VMAutoscaling.yaml** template illustrates the use of a template to create an up/down autoscaling resource.

This HOT template creates a VM instance with vCPU autoscaling. The number of online vCPUS for the instance is scaled up or down depending on vCPU utilization. The scaling range (minimum and maximum number of vCPUs) is determined by the launch flavor.

For up/down scaling, the CPU measurement uses a **vcpu\_util** meter, which takes into account the current number of online vCPUs when measuring usage. This is in contrast to the **cpu\_util** meter, which always uses the maximum number of vCPUs. Using **cpu\_util** for up/down scaling would produce an inaccurate usage/availability ratio if any vCPUs were offline, since it assumes all VCPUs are available.

The VM image selected for launch must be capable of scaling. For more information about VM scaling, see [Scaling Virtual Machine Resources](#) on page 19.

### Procedure

1. Collect the required parameters.
  - a) On the **Admin > System Panel > Flavors** page, locate a **Flavor Name** (a resource profile).
  - b) On the **Project > Compute > Access & Security** page, locate a **Keypair Name** to provide secure access to the stack.
  - c) On the **Admin > System Panel > Images** page, locate an **Image Name** to specify the VM image used by the stack.



- d) On the **Admin > System Panel > Networks** page, obtain the **Name** of the private network.
- e) On the **Admin > Identity Panel > Users** page, locate the **User Name** for the tenant where the VM will be launched.

2. Open the list of existing stacks.

From the **Project** menu of the web administration interface, select **Orchestration > Stacks**.

The Stacks list appears.

Stacks

Launch Stack

	Stack Name	Created	Updated	Status	Actions
No items to display.					
Displaying 0 items					

3. Launch a new stack and specify a template.

Click **Launch Stack**, and then complete the Select Template dialog box.

### Select Template

Template Source \*

URL

URL

File

Direct Input

Description:

Use one of the available template source options to specify the template to be used in creating this stack.

Cancel

Next

The **Template Source** drop-down list offers several ways to specify a template.

Options	Description
<b>URL</b>	Provides a text field for URL access to a template file.
<b>File</b>	Provides a <b>Browse</b> button for access to locally stored template files.
<b>Direct Input</b>	Provides a text window for direct entry of template content.

4. Select **File**, then click **Browse** and select **VMAutoScaling.yaml** from the local disk.
5. Adjust the parameters in the Launch Stack form to meet your requirements.

Options	Description
<b>KeyName</b>	The <b>KeyPair</b> name from the Access & Security page.
<b>TenantUserID</b>	The tenant user ID associated with the instance launch. Scaling of the instance resources can only be performed by the tenant associated with the launch.
<b>FlavorName</b>	The <b>Flavor Name</b> from the Flavors page.

Options	Description
<b>ImageName</b>	The <b>Image Name</b> from the Images page.
<b>PrivateNetName</b>	The <b>Name</b> from the Network Overview page.
<b>InstanceName</b>	The name of the instance to create.
<b>CustomMeterName</b>	The name of a custom Ceilometer metric to store.
<b>CustomMeterUnit</b>	The unit of measurement to display for the custom metric; for example, 'Percent'.
<b>MetricHighWaterMark</b>	The value that triggers autoscaling to add resources.
<b>MetricLowWaterMark</b>	The value that triggers autoscaling to remove resources.

## Supported Heat Resource Types

The Titanium Server implementation of Heat has been tested with most commonly-used resource types.

The following resource types have been tested for use with Titanium Server.

- AWS::AutoScaling::AutoScalingGroup
- AWS::AutoScaling::LaunchConfiguration
- AWS::AutoScaling::ScalingPolicy
- AWS::CloudFormation::Stack
- AWS::EC2::Instance
- AWS::EC2::InternetGateway
- AWS::EC2::NetworkInterface
- AWS::IAM::AccessKey
- AWS::IAM::User
- OS::Ceilometer::Alarm
- OS::Cinder::Volume
- OS::Cinder::VolumeAttachment
- OS::Glance::Image
- OS::Heat::AccessPolicy
- OS::Heat::AutoScalingGroup
- OS::Heat::InstanceGroup
- OS::Heat::ScalingPolicy
- OS::Heat::Stack
- OS::Neutron::FloatingIP

- OS::Neutron::FloatingIPAssociation
- OS::Neutron::Net
- OS::Neutron::NetworkGateway
- OS::Neutron::Port
- OS::Neutron::Router
- OS::Neutron::RouterGateway
- OS::Neutron::RouterInterface
- OS::Neutron::SecurityGroup
- OS::Neutron::Subnet
- OS::Nova::Flavor
- OS::Nova::FloatingIP
- OS::Nova::FloatingIPAssociation
- OS::Nova::KeyPair
- OS::Nova::Server
- OS::Nova::ServerGroup
- OS::WR::ScalingPolicy
- WR::Neutron::PortForwarding
- WR::Neutron::ProviderNet
- WR::Neutron::ProviderNetRange
- WR::Neutron::QoSPolicy

## Further Reading

Additional resources are available for understanding and working with OpenStack Heat Orchestration and templates.

The following online resources are suggested.

- [Create and manage stacks](#) (from the *OpenStack End User Guide*)
- [Heat](#) (from the OpenStack wiki)
- [Heat commands](#) (from the *OpenStack End User Guide*)
- [Heat Orchestration Template \(HOT\) Guide](#)
- [An Introduction to OpenStack Heat](#) (slide show from Mirantis)
- [AWS::CloudFormation::Init](#) (from Amazon's *AWS Documentation*; discusses metadata options for an EC2 instance)



# Swift Object Storage Management

Swift Object Storage	81
VM Configuration for Access to Swift Object Storage	82
Swift Upload File Size Considerations	82

## Swift Object Storage

Systems with dedicated storage hosts can provide object storage using OpenStack Swift. VMs and system users can use this to store and exchange Ceph-backed files.

Swift object storage uses *Swift containers*. These are similar to directories in a file system, except that they cannot be nested. However, each Swift container can contain areas called *folders*, which you can use to organize content. Hierarchies of folders are supported.

VMs and OpenStack users (including OpenStack services) can create Swift containers for public or private use, and then access them for file uploads and downloads.

In Titanium Server, a storage pool implemented on Ceph-backed storage hosts is used to hold Swift containers and objects. The pool is created when the Swift service is started. For VMs, this offers a place to store files that persist independently and can be exchanged with other VMs or with the Titanium Server platform.

You can add Swift support from the command line at any time after installation. For more information, see *Titanium Server System Administration:Configuring Swift Object Storage*.

System administrators and tenant users can manage Swift containers and their contents from the CLI or the Web administration interface. VMs can use a Swift client or the Swift REST API to manage and access Swift containers and objects. For more information, consult the public OpenStack documentation.

## VM Configuration for Access to Swift Object Storage

For access to Swift storage, you must attach the VM to an external network with a route to the controller OAM network. The VM uses a Swift client or Swift REST API to manage Swift containers and objects, sending commands to the Titanium Server Swift endpoint public URL. The Swift API service on the Titanium Server controller has access to the Swift containers and objects in the Ceph storage pool over the infrastructure network.

## Swift Upload File Size Considerations

For uploads using the CLI, the file size is limited by the PUT maximum, but you can use an option to split the file into segments. For uploads using the web administration interface, additional limitations apply.

The default maximum PUT size is 50GB. For larger files, you can use the `-S` option on the **swift upload** command to upload the file in segments:

```
~(keystone_admin)$swift upload -S segment_size filepath
```

where *segment\_size* is the maximum segment size in bytes.

For upload using the web administration interface, the split option is not available, and an additional limitation is imposed by the scratch space. The maximum upload size is the available scratch space (7.81 GB or less) or the maximum PUT size, whichever is less.

You can view the available scratch space as follows:

```
$ cd /scratch/lighttpd
$ df -k | grep scratch
Filesystem                                1K-blocks    Used Available Use% Mounted on
/dev/mapper/cgts--vg-scratch--lv          7932336     36024   7470328    1% /scratch
```