Test Plan for FWaaS Fuel Plugin 1.0.0

Revision history

Version	Revision date	Editor	Comment
1.0	06.03.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com)	Created the test-plan.
1.1	07.04.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com)	Modified Developer's specification and Appendix, add Requirements and Limitations
1.2	08.04.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com)	Check links and delete Priority
1.3	17.04.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com)	Modified with new model test_plan_template
1.4	21.05.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com	Add destructive tests
1.5	29.05.2015	Kristina Kuznetsova (kkuznetsova@mirantis.com	Add system tests

FWaaS Plugin

The main goal of this document is to describe the test cases for FWaaS plugin on Mirantis OpenStack project. These tests cases should be used during the acceptance testing for each new release. FWaaS (FireWall-as-a-Service) introduces firewall feature set.

Developer's specification

Blueprint: support-fwaas-in-mos: link

Demo. VPNaas and FWaas Fuel plugins: link

FWaaS Plugin Guide: link

spec review: link

Limitations

FWaaS plugin can be enabled only in environments with Neutron as the networking option.

Test strategy

Type of implemented tests is functional testing. In future all tests for the CLI will be automated. There is plan to add some integration test cases with others plugins (VPNaaS) and some more new functional test cases. We also plan to add tests in Rally. We are going to study what is in Tempest and then we will decided if there is necessary a few changes.

Acceptance criteria

All tests should be passed without any errors and exceptions.

Test environment, infrastructure and tools

Deploy environment (3 nodes with controller role, 1 node with compute role, Neutron networking) with default parameters with installing FWaaS plugin. Other recommendation you can see in the test cases. There are steps, which we need for some test cases:

Steps for create ubuntu image start:

1) In CLI go to the fuel ip: ssh root@<fuel id>
2) Look what controllers we have: fuel node
3) Go to the one of controllers: ssh node-<node id>
4) Download file with ubuntu image: wget https://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-disk1.img
5) Create ubuntu image: glance image-create --name <name_for_the_new image> --file trusty-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --is-public=true --progress

Steps before booting VM:

1) Login to OpenStack Horizon dashboard as admin user
2) Create flavor for the new VM: navigate to Admin->System->Flavors, press “Create Flavor” button. Parametrs: VCPUs = 1, RAM (MB) = 1024, Root Disk (GB) = 5, Ephemeral Disk (GB) = 0, Swap Disk (MB) = 0
3) Navigate to Project->Compute->Access$Security->Key Pairs and click on “Import Key Pair” button. Write name of key and your public key
4) Allow all icpm, tcp and udp rules for sec group

Product compatibility matrix

Requirement	Version/Comment
Fuel	6.1 release
OpenStack compatibility	2014.2 Juno
Operating systems	Ubuntu 14.04 LTS, CentOS 6.5
Plugin version	FWaaS Fuel Plugin 1.0.0

Functional testing

Title	**TC 001: Ability to create firewall rule, rename, edit, share and delete it**
Test Case ID	create_update_delete_firewall_rule
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin and 2 tenants, member-user for the second tenant
Steps	CLI: 1) Go with ssh to the fuel ip: ssh root@<fuel ip> 2) Look what nodes we have: fuel node 3) Go to the controller node: ssh node-<node_id> 4) . openrc

	5) Create firewall rule for the protocol icmp to allow icmp traffic: neutron firewall-rule-create --protocol icmp --action allow --name all_icmp 6) Check that rule have been created: neutron firewall-rule-list 7) Rename and edit this created rule: neutron firewall-rule-update <rule_id> --protocol tcp --action allow --name all_tcp 8) Check that this rule was modified: neutron firewall-rule-show all_tcp 9) Create new firewall rule: neutron firewall-rule-create --protocol icmp --action allow --name all_icmp 10) Share all_tcp for another tenant: neutron firewall-rule-update <rule_id> --shared 11) vim openrc 12) In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type name of the second tenant, member-user and password 13) . openrc 14) Check that only shared rule appeared: neutron firewall-rule-list 15) vim openrc 16) In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type admin's parameters 17) . openrc 18) Delete rule: neutron firewall-rule-delete <rule_id> 19) Check that rule has been deleted: neutron firewall-rule-list
	Dashboard: 1) Login to OpenStack Horizon dashboard 2) Navigate to Project->Network->Firewalls 3) Navigate to tab Firewall Rules 4) Click on Add Rule button to create Firewall rule with name "all icmp". In the field Protocol choose ICMP and click on Add 5) Check that new rule has created 6) Click on Edit rule in the line with new created rule 7) Change name from "all icmp" to "all_tcp" 8) Chose Protocol tcp 9) Click on Save changes 10) Check that name and protocol became new 11) Click on Add Rule button to create Firewall rule with name "all icmp". In the field Protocol choose ICMP and click on Add 12) Check that we can create no only one rule 13) Click on Edit Rule 14) Put a mark against Shared 15) Click on Save Changes 16) Go to another tenant 17) Navigate to Project->Network->Firewalls 18) Navigate to tab Firewall Rules 19) Check that shared created rule appeared here 20) Click on arrow down button in the line with created rule 21) Click on Delete rule

	22) Check that rule disappeared
Expected Result	All steps should be passed, we should have the ability to create, rename, edit, share and delete firewall rule

Title	**TC 002: Ability to create firewall policy, rename, remove rule from the policy, add rule before and after already added rules in the policy, share and delete it**
Test Case ID	create_update_delete_firewall_policy
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin with 2 tenants, member-user for the second tenant
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create firewall rule for the protocol icmp to allow icmp traffic: neutron firewall-rule-create --protocol icmp --action allow --name all_icmp --shared 6. Create firewall rule for the protocol tcp to allow tcp traffic: neutron firewall-rule-create --protocol tcp --action allow --name all_tcp --shared 7. Create firewall rule for the protocol udp to allow udp traffic: neutron firewall-rule-create --protocol udp --action allow --name all_udp --shared 8. Create policy with icmp rule: neutron firewall-policy-create --firewall-rules <all_icmp_rule_id> <name for this policy> 9. Check that policy has been created: neutron firewall-policy-list 10. Rename the policy: neutron firewall-policy-update <policy_id> --name test 11. Check that policy name has been changed: neutron firewall-policy-show <policy_id> 12. Remove firewall rule from policy: neutron firewall-policy-remove-rule <policy_id> <rule_id> 13. Check that policy name has been changed: neutron firewall-policy-show <policy_id> 14. Insert all_icmp rule: neutron firewall-policy-insert-rule <policy_id> <rule_id> 15. Check that policy has been added new rule all_icmp: neutron firewall-policy-show <policy_id>

	16. Insert all_tcp after all_icmp: neutron firewall-policy-insert-rule <rule_id> <policy_id> --insert-after <rule_id> 17. Insert all_udp before all_icmp: neutron firewall-policy-insert-rule <rule_id> <policy_id> --insert-before <rule_id> 18. Check the order of rules: neutron firewall-policy-show <policy_id> 19. Share the policy: neutron firewall-policy-update <policy_id> --shared 20. vim openrc 21. In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type name of the second tenant, member-user and password 22. . openrc 23. Check that only shared policy appeared: neutron firewall-rule-list 24. vim openrc 25. In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type admin's parameters 26. . openrc 27. Delete the policy: neutron firewall-policy-delete <policy_id>
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project->Network->Firewalls 3. Navigate to tab Firewall Rules 4. Click on Add Rule button to create Firewall rule with name "all_icmp", check shared In the field Protocol choose ICMP and click on Add 5. Click on Add Rule button to create Firewall rule with name "all_tcp", check shared. In the field Protocol choose TCP and click on Add 6. Click on Add Rule button to create Firewall rule with name "all_udp", check shared. In the field Protocol choose UDP and click on Add 7. Navigate to tab Firewall Policies 8. Click on Add Policy button. Type name "1" on the field Name on AddPolicy tab 9. Go to the tab Rules 10. Click on + near all_icmp in the Available Rules 11. Click on Add button 12. Check that new policy has created 13. Click on Edit Policy it the colom Actions in the line with new policy 14. Change name to test_policy instead of 1 and Click on Save Changes 15. Check that name has modified 16. Click on down arrow it the colom Actions in the line with new policy 17. Click on Remove Rule 18. Select all_icmp and click on Save changes 19. Check that all_icmp rule has been deleted from policy 20. Click on down arrow it the colom Actions in the line with new policy 21. Click on Insert rule 22. Select all_icmp and click on Save changes 23. Check that in column Rules appeared all_icmp for the policy

	24. Click on down arrow it the colom Actions in the line with new rule 25. Click on Insert Rule 26. Choose rule all_tcp and in the field after chose all_icmp 27. Click on Save Changes 28. Check, that all_tcp appeared after all_icmp in the column Rules 29. Click on down arrow it the colom Actions in the line with new rule 30. Click on Insert Rule 31. Choose rule all_udp and in the field before chose all_icmp 32. Click on Save Changes 33. Check, that all_udp appeared before all_icmp in the column Rules 34. Click on Edit Policy it the colom Actions in the line with new policy 35. Put a mark against Shared. Click on Save Changes 36. Sign out and go to another tenant login as member-user 37. Navigate to Project->Network->Firewalls 38. Navigate to the tab Firewall Policies 39. Check that shared created policy appeared here 40. Return to the first tenant 41. Navigate to Project->Network->Firewalls 42. Navigate to the tab Firewall Policies 43. Click on arrow down button in the line with created policy 44. Click on Delete policy 45. Check that policy disappeared
Expected Result	All steps should be passed, we should have the ability to create firewall policy, rename, remove rule from the policy, add rule before and after already added rules in the policy, share and delete it

Title	**TC 003: Ability to create firewall, rename, change policy and delete**
Test Case ID	add_update_delete_firewall
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin with 2 tenants, member-user for the second tenant
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create firewall rule for the protocol tcp to allow tcp traffic: neutron firewall-rule-create --protocol tcp --action allow --name all_tcp 6. Create firewall rule for the protocol udp to allow udp traffic: neutron firewall-rule-create --protocol udp --action allow --name all_udp

	7. Create policy with tcp rule: neutron firewall-policy-create --firewall-rules <all_tcp_rule_id> policy_1 8. Create policy with udp rule: neutron firewall-policy-create --firewall-rules <all_udp_rule_id> policy_2 9. Create firewall with policy_1: neutron firewall-create <policy_name> 10. Check that firewall status is active: neutron firewall-show <firewall_id> 11. Rename and change policy: neutron firewall-update <firewall_id> --name 2 --firewall_policy_id <another_policy_id>
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project->Network->Firewalls 3. Navigate to tab Firewall Rules 4. Click on Add Rule button to create Firewall rule with name “all_tcp”. In the field Protocol choose TCP and click on Add 5. Click on Add Rule button to create Firewall rule with name “all_udp”. In the field Protocol choose UDP and click on Add 6. Navigate to tab Firewall Policies 7. Click on Add Policy button. Type name “1” on the field Name on AddPolicy tab 8. Go to the tab Rules 9. Click on + near all_tcp in the Available Rules 10. Click on Add button 11. Click on Add Policy button. Type name “2” on the field Name on AddPolicy tab 12. Go to the tab Rules 13. Click on + near all_udp in the Available Rules 14. Click on Add button 15. Go to the tab Firewalls 16. Click on Create Firewall 17. Type name. Choose created policy 18. Click on Add 19. Update the page and check that status is active 20. Click on Edit Firewall in the with the firewall 21. Type name 2 instead of 1 and choose policy 2 22. Click on Save changes 23. Check that firewall has the policy 2 and name 2 24. Click on arrow down button in the line with created firewall 25. Click on Delete firewall 26. Check that firewall has disappeared
Expected Result	All steps should be passed, we should have the ability to create firewall, rename, change policy and delete

Title	**TC 004: Create icmp firewall to allow ping VM (in admin tenant) from your computer**
Test Case ID	allow_icmp_from_computer_to_tenant
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM: nova floating-ip-create nova floating-ip-associate <VM_port> <floating_ip> 6. In new tab go to this VM1: ssh ubuntu@<floating_ip> 7. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 8. Return to the tab with node controller 9. Create firewall rule for the protocol icmp to allow ping to your computer: neutron firewall-rule-create --protocol icmp --action allow --source_ip_address <your computer ip> 10. Create firewall policy: neutron firewall-policy-create --firewall-rules <firewall rule id> <name for this policy> 11. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 12. Check that firewall status is active: neutron firewall-show <firewall_id> 13. In tab with computer: sudo tcpdump -i any icmp 14. In tab with VM1 try to ping your computer: ping <computer_IP> 15. In tab with computer check that ping is successful 16. In tab with VM1: sudo tcpdump -i any icmp 17. In tab with computer try to ping VM1 18. Check that ping can't be available
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1 in admin tenant 3. Associate Floating IP for this VM1 4. In CLI go to this VM1: ssh ubuntu@<floating_ip>

	5. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 6. Return to the dashboard 7. Navigate to Project->Network->Firewalls 8. Create Firewall rule to allow icmp traffic from computer: Protocol = ICMP, Source IP Adress = <your computer IP> 9. Create firewall policy with this rule 10. Create firewall with this policy 11. Check that firewall status is ACTIVE 12. Return to the CLI 13. In tab with VM1: sudo tcpdump -i any icmp 14. In tab with computer try to ping the VM: ping <VM_IP> 15. In tab with VM1 check that ping is successful 16. In tab with computer: sudo tcpdump -i any icmp 17. In tab with VM1 try to ping computer 18. Check that ping can't be available
Expected Result	All steps should be passed, we should have the ability to send ICMP traffic from computer to VM1 and we shouldn't have the ability to send ICMP traffic from VM1 to the computer

Title	**TC 005: Create firewall to allow ping only your computer from VM (in admin tenant)**
Test Case ID	allow_icmp_from_tenant_to_computer
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM: nova floating-ip-create nova floating-ip-associate <VM_port> <floating_ip> 6. In new tab go to this VM1: ssh ubuntu@<floating_ip>

	7. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 8. Return to the tab with node controller 9. Create firewall rule for the protocol icmp to allow ping to your computer: 10. neutron firewall-rule-create --protocol icmp --action allow --destination_ip_address <your computer ip> 11. Create firewall policy: neutron firewall-policy-create --firewall-rules <firewall rule id> <name for this policy> 12. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 13. Check that firewall status is active: neutron firewall-show <firewall_id> 14. In tab with computer: sudo tcpdump -i any icmp 15. In tab with VM1 try to ping your computer: ping <computer_IP> 16. In tab with computer check that ping is successful 17. In tab with VM1: sudo tcpdump -i any icmp 18. In tab with computer try to ping VM1 19. Check that ping can't be available
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1 in new tenant 3. Associate Floating IP for this VM1 4. In CLI go to this VM1: ssh ubuntu@<floating_ip> 5. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 6. Return to the dashboard 7. Navigate to Project->Network->Firewalls 8. Create Firewall rule to allow icmp traffic from computer: Protocol = ICMP, Destination IP Address = <you computer IP> 9. Create firewall policy with this rule 10. Create firewall with this policy 11. Check that status of firewall is ACTIVE 12. Return to the CLI 13. In tab with computer: sudo tcpdump -i any icmp 14. In tab with VM1 try to ping your computer: ping <computer_IP> 15. In tab with computer check that ping is successful 16. In tab with VM1: sudo tcpdump -i any icmp 17. In tab with computer try to ping VM1 18. Check that ping can't be available
Expected Result	All steps should be passed, we should have the ability to send ICMP traffic from computer to VM1 and we shouldn't have the ability to send ICMP traffic from computer to the VM1

Title	**TC 006: Create firewall to allow any protocol (icmp, tcp, udp) between your computer and VM (in admin tenant)**
Test Case ID	allow_trafic_between_tenant_and_computer
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM: nova floating-ip-create nova floating-ip-associate <VM_port> <floating_ip> 6. In new tab go to this VM1: ssh ubuntu@<floating_ip> 7. Return to the tab with node controller 8. Create firewall rule for the all protocols to allow traffic from your computer to the tenant: neutron firewall-rule-create --protocol any --action allow --destination_ip_address <your computer ip> 9. Create firewall rule for the all protocols to allow traffic from this tenant: neutron firewall-rule-create --protocol any --action allow --source_ip_address <your computer ip> 10. Create firewall policy with this 2 rules: neutron firewall-policy-create --firewall-rules "<firewall_rule_id_step9> <firewall_rule_id_step10>" <name for this policy> 11. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 12. Check that firewall status is active: neutron firewall-show <firewall_id> 13. In tab with computer: sudo tcpdump -i any icmp 14. In tab with VM1 try to ping your computer: ping <computer_IP> 15. In tab with computer check that ping is successful 16. In tab with VM1: sudo tcpdump -i any icmp 17. In tab with computer try to ping VM1 18. Check that ping is available 19. In tab with VM1 ping 8.8.8.8 and check that ping isn't available 20. In tab with VM1 run iperf -s to check tcp traffic

	21. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP> 22. Verify that TCP traffic allowed 23. In tab with VM1 run iperf -u -s to check udp traffic 24. In tab with computer run (to start udp traffic): iperf -u -n 1m -c <VM1_IP> 25. Verify that UDP traffic allowed 26. In tab with computer run iperf -s to check tcp traffic 27. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id> 28. Verify that TCP traffic allowed 29. In tab with computer run iperf -u -s to check udp traffic 30. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -c <computer_id> 31. Verify that UDP traffic allowed
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1 in new tenant 3. Associate Floating IP for this VM1 4. In CLI go to this VM1: ssh ubuntu@<floating_ip> 5. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 6. Return to the dashboard 7. Navigate to Project->Network->Firewalls 8. Create Firewall rule to allow icmp traffic from computer: Protocol = ANY, Destination IP Address = <you computer IP> 9. Create Firewall rule to allow icmp traffic from computer: Protocol = ANY, Source IP Address = <you computer IP> 10. Create firewall policy with this rules 11. Create firewall with this policy 12. Check that status of firewall is ACTIVE 13. Return to the CLI 14. In tab with computer: sudo tcpdump -i any icmp 15. In tab with VM1 try to ping your computer: ping <computer_IP> 16. In tab with computer check that ping is successful 17. In tab with VM1: sudo tcpdump -i any icmp 18. In tab with computer try to ping VM1 19. Check that ping is available

	20. In tab with VM1 ping 8.8.8.8 and check that ping isn't available 21. In tab with VM1 run iperf -s to check tcp traffic 22. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP> 23. Verify that TCP traffic allowed 24. In tab with VM1 run iperf -u -s to check udp traffic 25. In tab with computer run (to start udp traffic): iperf -u -n 1m -c <VM1_IP> 26. Verify that UDP traffic allowed 27. In tab with computer run iperf -s to check tcp traffic 28. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id> 29. Verify that TCP traffic allowed 30. In tab with computer run iperf -u -s to check udp traffic 31. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -c <computer_id> 32. Verify that UDP traffic allowed
Expected Result	All steps should be passed, we should have the ability to send ICMP, UDP and TCP traffic between computer and VM1 and we shouldn't have the ability to send ICMP traffic from VM1 to 8.8.8.8

Title	**TC 007: Create firewall to allow icmp traffic (in admin tenant)**
Test Case ID	allow_all_icmp_trafic_for_tenant
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id>

4. . openrc
5. Create new ubuntu VM, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id>
   Associate floating IP for this VM:
   nova floating-ip-create
   nova floating-ip-associate <VM_port> <floating_ip>
6. In new tab go to this VM1: ssh ubuntu@<floating_ip>
7. Return to the tab with node controller
8. Create firewall rule for the icmp protocol to allow all icmp traffic in tenant: neutron firewall-rule-create --protocol icmp --action allow
9. Create firewall policy with this rule: neutron firewall-policy-create --firewall-rules "<firewall_rule_id>" <name for this policy>
10. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id>
11. Check that firewall status is active: neutron firewall-show <firewall_id>
12. In tab with computer: sudo tcpdump -i any icmp
13. In tab with VM1 try to ping your computer: ping <computer_IP>
14. In tab with computer check that ping is successful
15. In tab with VM1: sudo tcpdump -i any icmp
16. In tab with computer try to ping VM1
17. Check that ping is available
18. In tab with VM1 ping 8.8.8.8 and check that ping is available
19. In tab with VM1 run
    iperf -s
    to check tcp traffic
20. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP>
21. Check that TCP traffic lost
22. In tab with VM1 run
    iperf -u -s
    to check udp traffic
23. In tab with computer run (to start udp traffic): iperf -u -i1 -c <VM1_IP>
24. Verify that UDP traffic lost
25. In tab with computer run
    iperf -s
    to check tcp traffic
26. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id>
27. Verify that TCP traffic lost
28. In tab with computer run
    iperf -u -s
    to check udp traffic
29. In tab with VM1 run (to start udp traffic): iperf -u -i1 -c <computer_id>

	30. Verify that UDP traffic lost
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1. Associate Floating IP for this VM1 3. In CLI go to this VM1: ssh ubuntu@<floating_ip> 4. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 5. Return to the dashboard 6. Navigate to Project->Network->Firewalls 7. Create Firewall rule to allow icmp traffic: Protocol = ICMP, action Allow 8. Create firewall policy with this rule 9. Create firewall with this policy 10. Check that status of firewall is ACTIVE 11. Return to the CLI 12. In tab with computer: sudo tcpdump -i any icmp 13. In tab with VM1 try to ping your computer: ping <computer_IP> 14. In tab with computer check that ping is successful 15. In tab with VM1: sudo tcpdump -i any icmp 16. In tab with computer try to ping VM1 17. Check that ping is available 18. In tab with VM1 ping 8.8.8.8 and check that ping is available 19. In tab with VM1 run iperf -s to check tcp traffic 20. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP> 21. Verify that TCP traffic lost 22. In tab with VM1 run iperf -u -s to check udp traffic 23. In tab with computer run (to start udp traffic): iperf -n 1m -c <VM1_IP> 24. Verify that UDP traffic lost 25. In tab with computer run iperf -s to check tcp traffic 26. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id> 27. Verify that TCP traffic lost 28. In tab with computer run iperf -u -s to check udp traffic

	29. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -c <computer_id> 30. Verify that UDP traffic lost
Expected Result	All steps should be passed, we should have the ability to send ICMP traffic between computer and VM1 and we shouldn't have the ability to send ICMP traffic from VM1 to 8.8.8.8

Title	**TC 008: Create icmp firewall for tenant except your computer network**
Test Case ID	allow_all_icmp_trafic_except_from_one_ip
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM1, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM1: nova floating-ip-create nova floating-ip-associate <VM1_port> <floating_ip> 6. In new tab go to VM1: ssh ubuntu@<floating_ip> 7. Return to the tab with node controller 8. Create firewall rule for the icmp protocol to deny icmp from your computer network: neutron firewall-rule-create --protocol icmp --action deny --destination_ip_address <your computer network (172.18.76.0/24)> 9. Create firewall rule for the icmp protocol to allow all icmp traffic in tenant: neutron firewall-rule-create --protocol icmp --action allow 10. Create firewall policy with this rules: neutron firewall-policy-create --firewall-rules "<firewall_rule_id1> <firewall_rule_id2>" <name for this policy> 11. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 12. Check that firewall status is active: neutron firewall-show <firewall_id> 13. In tab with computer: sudo tcpdump -i any icmp 14. In tab with VM1: ping <your computer IP>

	15. Check that ping is impossible 16. In tab with VM1: sudo tcpdump -i any icmp 17. In tab with computer: ping <VM1 floating ip> 18. Check that ping is possible 19. In tab with VM1: ping 8.8.8.8 and check that ping is possible 20. In tab with node controller delete firewall. At first look the firewall id: neutron firewall-list. Then delete it: neutron firewall-delete <firewall-id> (Check, that we can delete firewall) 21. In tab with node controller delete firewall policy. Look what id has policy:: neutron firewall-policy-list. The delete it: neutron firewall-delete <firewall policy id> (Check, that we can delete policy) 22. In tab with node controller delete rules. At first look what id this rule has: neutron firewall-rule-list. Then delete it: neutron firewall-rule-delete <rule id> (Check, that we can delete rule)
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1. Associate Floating IP for this VM1 3. In CLI go to this VM1: ssh ubuntu@<floating_ip> 4. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 5. Return to the dashboard 6. Navigate to Project->Network->Firewalls 7. Create firewall rule for the icmp protocol to deny icmp from your computer: Protocol = ICMP, action Deny, destination ip address <your computer network (172.18.76.0/24)> 8. Create firewall rule for the icmp protocol to allow all icmp traffic in tenant: Protocol = ICMP, action = ALLOW 9. Create firewall policy with this rules (on tab rules choose rule from step 8, then rule step 9 10. Create firewall with this policy 11. Check that firewall status become active 12. In tab with computer: sudo tcpdump -i any icmp 13. In tab with VM1: ping <your computer IP> 14. Check that ping is impossible 15. In tab with VM1: sudo tcpdump -i any icmp 16. In tab with computer: ping <VM1 floating ip> 17. Check that ping is possible 18. In tab with VM1: ping 8.8.8.8 and check that ping is possible 19. Return to UI, choose firewall and delete firewall with “Delete button” (Check, that we can delete firewall) 20. In UI go to the Firewall Policies, choose policy and delete it with “delete policies@ button (Check, that we can delete policy) 21. In UI go to the rules, choose rule from step 8 and delete it with “Delete rules” button (Check, that we can delete rule)

Expected Result	All steps should be passed, we should have the ability to send ICMP traffic except fot the computer's network

Title	**TC 009: Create icmp firewall for tenant except one VM**
Test Case ID	allow_all_icmp_except_one_vm
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM1, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM1: nova floating-ip-create nova floating-ip-associate <VM1_port> <floating_ip> 6. Create one more new ubuntu VM2 in another private network, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM2: nova floating-ip-create nova floating-ip-associate <VM2_port> <floating_ip> 7. In new tab go to VM1: ssh ubuntu@<floating_ip> 8. In new tab go to VM2: ssh ubuntu@<floating_ip> 9. Return to the tab with node controller 10. Create firewall rule for the icmp protocol to allow all icmp traffic in tenant: neutron firewall-rule-create --protocol icmp --action allow 11. Create firewall rule for the icmp protocol to deny icmp from VM2: neutron firewall-rule-create --protocol icmp --action deny --destination_ip_address <ip in private network> 12. Create firewall rule for the icmp protocol to deny icmp to VM2: neutron firewall-rule-create --protocol icmp --action deny --source_ip_address <ip in private network> 13. Create firewall policy with this rules: neutron firewall-policy-create --firewall-rules "<firewall_rule_id_1> <firewall_rule_id_step_2> <firewall_rule_id_step_3>" <name for this policy>

	14. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 15. Check that firewall status is active: neutron firewall-show <firewall_id> 16. In tab with VM2: sudo tcpdump -i any icmp 17. In tab with VM1: ping <floating_ip_VM2> 18. Check that ping is possible (iptables check first rule with allow and miss others) 19. Return to the tab with node controller 20. Remove rule from policy with allow icmp for all (policy id in list: neutron firewall-policy-list; rule id list: neutron firewall-rule-list: neutron firewall-rule-list): neutron firewall-policy-remove-rule <policy id> <rule id> 21. Add rule to allow all icmp in the end of rules for the policy: neutron firewall-policy-insert-rule --insert-after <rule id, after what should be allow> <policy id> <rule id for adding> 22. In tab with VM2: sudo tcpdump -i any icmp 23. In tab with VM1: ping <floating_ip_VM2> 24. Check that ping is impossible 25. In tab with VM1: ping <computer ip> 26. Check that ping is possible 27. In tab with VM1: sudo tcpdump -i any icmp 28. In tab with VM2: ping <floating_ip_VM1> 29. Check that ping is impossible
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1. Associate Floating IP for this VM1 3. Create one more new ubuntu VM2 in another private network, if it doesn't exist. 4. In CLI go to this VM1: ssh ubuntu@<floating_ip> 5. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 6. Return to the dashboard 7. Navigate to Project->Network->Firewalls 8. Create firewall rule for the icmp protocol to allow all icmp traffic in tenant: Protocol = ICMP, action = ALLOW, destination ip address <your computer network (172.18.76.0/24)> 9. Create firewall rule for the icmp protocol to deny icmp from VM2: Protocol = ICMP, action Deny 10. Create firewall policy with this rules (on tab rules choose rule from step 8, then rule step 9) 11. Create firewall with this policy 12. Check that firewall status become active 13. In tab with VM2: sudo tcpdump -i any icmp

	14. In tab with VM1: ping <floating_ip_VM2> 15. Check that ping is possible (iptables check first rule with allow and miss others) 16. Return to UI. Navigate to Project->Network->Firewalls->Firewall Policies. Find our policy and in column Actions click “Down arrow” button 17. Choose Remove rule and then choose allow icmp rule 18. Again find our policy and in column Actions click “Down arrow” button 19. Choose rule to allow icmp in “Insert rule” field, choose after the ladt deny rule 20. n tab with VM2: sudo tcpdump -i any icmp 21. In tab with VM1: ping <floating_ip_VM2> 22. Check that ping is impossible 23. In tab with VM1: ping <computer ip> 24. Check that ping is possible 25. In tab with VM1: sudo tcpdump -i any icmp 26. In tab with VM2: ping <floating_ip_VM1> 27. Check that ping is impossible
Expected Result	All steps should be passed, we shouldn’t have the ability to send ICMP traffic to VM2 and from VM2 after correct ordered rules

Title	**TC 010: Allow tcp firewall for tenant for one port**
Test Case ID	allow_tcp_for_one_port
Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM1, if it doesn’t exist: nova boot <vm’s_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM1: nova floating-ip-create nova floating-ip-associate <VM1_port> <floating_ip> 6. In new tab go to VM1: ssh ubuntu@<floating_ip> 7. Return to the tab with node controller

	8. Create firewall rule for the tcp protocol to allow tcp traffic for one port in tenant: neutron firewall-rule-create --protocol tcp --action allow --destination_port 2000 9. Create firewall rule for the icmp protocol to allow tcp to the tenant: neutron firewall-rule-create --protocol tcp --action allow --source_port 2000 10. Create firewall policy with this rules: neutron firewall-policy-create --firewall-rules “<firewall_rule_id_step_8> <firewall_rule_id_step_9>” <name for this policy> 11. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 12. Check that firewall status is active: neutron firewall-show <firewall_id> 13. In tab with VM1 run iperf -s -p 2000 to check tcp traffic 14. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2000 -c <VM1_IP> 15. Verify that TCP traffic is allowed 16. In tab with computer run iperf -s -p 2000 to check tcp traffic 17. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2000 -c <computer_id> 18. Verify that TCP traffic is allowed 19. In tab with VM1 run iperf -s -p 2005 to check tcp traffic 20. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2005 -c <VM1_IP> 21. Verify that TCP traffic is lost 22. In tab with computer run iperf -s -p 2005 to check tcp traffic 23. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2005 -c <computer_id> 24. Verify that TCP traffic is lost
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1. Associate Floating IP for this VM1 3. In CLI go to this VM1: ssh ubuntu@<floating_ip> 4. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 5. Return to the dashboard

	6. Navigate to Project->Network->Firewalls 7. Create firewall rule for the tcp protocol to allow tcp traffic in tenant for one port: Protocol = TCP, action = ALLOW, destination port 2000 8. Create firewall rule for the tcp protocol to allow tcp traffic in tenant for one port: Protocol = TCP, action = ALLOW, source port 2000 9. Create firewall policy with this rules 10. Create firewall with this policy 11. Check that firewall status become active 12. In tab with VM1 run iperf -s -p 2000 to check tcp traffic 13. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2000 -c <VM1_IP> 14. Verify that TCP traffic is allowed 15. In tab with computer run iperf -s -p 2000 to check tcp traffic 16. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2000 -c <computer_id> 17. Verify that TCP traffic is allowed 18. In tab with VM1 run iperf -s -p 2005 to check tcp traffic 19. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2005 -c <VM1_IP> 20. Verify that TCP traffic is lost 21. In tab with computer run iperf -s -p 2005 to check tcp traffic 22. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2005 -c <computer_id> 23. Verify that TCP traffic is lost
Expected Result	All steps should be passed, we should have the ability to send tcp traffic only for one port, for other it should be impossible

Title	**TC 011: Allow udp firewall for tenant for all ports except one**
Test Case ID	allow_all_udp_except_one_port

Prerequisites	1 pre deployed ubuntu cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Go with ssh to the fuel ip: ssh root@<fuel ip> 2. Look what nodes we have: fuel node 3. Go to the controller node: ssh node-<node_id> 4. . openrc 5. Create new ubuntu VM1, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM1: nova floating-ip-create nova floating-ip-associate <VM1_port> <floating_ip> 6. In new tab go to VM1: ssh ubuntu@<floating_ip> 7. Return to the tab with node controller 8. Create firewall rule for the udp protocol to allow udp traffic for all ports in tenant: neutron firewall-rule-create --protocol udp --action allow 9. Create firewall rule for the udp protocol to deny udp traffic for one port in tenant: neutron firewall-rule-create --protocol udp --action deny --destination_port 2000 10. Create firewall rule for the udp protocol to allow udp to the tenant: neutron firewall-rule-create --protocol udp --action deny --source_port 2000 11. Create firewall policy with this rules: neutron firewall-policy-create --firewall-rules "<firewall_rule_id_2> <firewall_rule_id_3> <firewall_rule_id_1>" <name for this policy> 12. Create firewall for this policy (if there are another firewall delete it before creation new): neutron firewall-create <policy_id> 13. Check that firewall status is active: neutron firewall-show <firewall_id> 14. In tab with VM1 run iperf -u -s -p 2005 to check udp traffic 15. In tab with computer run (to start udp traffic): iperf -u -n 1m -p 2005 -c <VM1_IP> 16. Verify that UDP traffic is allowed 17. In tab with computer run iperf -u -s -p 2005 to check udp traffic 18. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -p 2005 -c <computer_id> 19. Verify that UDP traffic is allowed 20. In tab with VM1 run iperf -u -s -p 2000 to check udp traffic

	21. In tab with computer run (to start udp traffic): iperf -u -n 1m -p 2000 -c <VM1_IP> 22. Verify that UDP traffic is lost 23. In tab with computer run iperf -u -s -p 2000 to check udp traffic 24. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -p 2000 -c <computer_id> 25. Verify that UDP traffic is lost
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1. Associate Floating IP for this VM1 3. In CLI go to this VM1: ssh ubuntu@<floating_ip> 4. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 5. Return to the dashboard 6. Navigate to Project->Network->Firewalls 7. Create firewall rule for the udp protocol to allow udp traffic for all ports: Protocol = UDP, action = ALLOW 8. Create firewall rule for the udp protocol to deny udp traffic for one port: Protocol = UDP, action = DENY, destination port 2000 9. Create firewall rule for the udp protocol to deny udp traffic in tenant for one port: Protocol = UDP, action = DENY, source port 2000 10. Create firewall policy with this rules (first 8 and 9 rules, then 7) 11. Create firewall with this policy 12. Check that firewall status become active 13. In tab with VM1 run iperf -u -s -p 1000 to check udp traffic 14. In tab with computer run (to start udp traffic): iperf -u -n 1m -p 1000 -c <VM1_IP> 15. Verify that UDP traffic is allowed 16. In tab with computer run iperf -u -s -p 1000 to check udp traffic 17. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -p 1000 -c <computer_id> 18. Verify that UDP traffic is allowed 19. In tab with VM1 run iperf -u -s -p 2000 to check udp traffic 20. In tab with computer run (to start udp traffic): iperf -u -n 1m -p 2000 -c <VM1_IP> 21. Verify that UDP traffic is lost

	22. In tab with computer run iperf -u -s -p 2000 to check udp traffic 23. In tab with VM1 run (to start udp traffic): iperf -u -n 1m -p 2000 -c <computer_id> 24. Verify that UDP traffic is lost
Expected Result	All steps should be passed, we should have the ability to send tcp traffic for all ports, for port 2000 it should be impossible

Title	**TC 012: Create firewall policy in non admin tenant with shared from admin tenant policy**
Test Case ID	share_firewall_policy
Prerequisites	1 pre deployed CEntOS cloud with firewall plugin. Executed steps for create ubuntu image start and steps before booting VM
Steps	CLI: 1. Add all rules for icmp, tcp and udp to sec groups as for admin tenant 2. Go with ssh to the fuel ip: ssh root@<fuel ip> 3. Look what nodes we have: fuel node 4. Go to the controller node: ssh node-<node_id> 5. vim openrc 6. In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type nonadmin's parameters 7. . openrc 8. Create new ubuntu VM1, if it doesn't exist: nova boot <vm's_name> --flavor <name of flavor> --image <name of image> --nic net-id=<net_id> Associate floating IP for this VM1: nova floating-ip-create nova floating-ip-associate <VM1_port> <floating_ip> 9. In new tab go to VM1: ssh ubuntu@<floating_ip> 10. Return to the tab with node controller 11. vim openrc 12. In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type admin's parameters 13. . openrc 14. Create firewall rule for the icmp protocol to allow tcp traffic for all tenant: neutron firewall-rule-create --protocol tcp --action allow --shared True

	15. Create firewall policy with this rule: neutron firewall-policy-create --firewall-rules “<firewall_rule_id>” --shared True <name for this policy> 16. vim openrc 17. In fields OS_TENANT_NAME, OS_USERNAME, OS_PASSWORD type nonadmin’s parameters 18. . openrc 19. Create firewall for shared policy: neutron firewall-create <policy_id> 20. Check that firewall status is active: neutron firewall-show <firewall_id> 21. In tab with computer: sudo tcpdump -i any icmp 22. In tab with VM1 try to ping your computer: ping <computer_IP> 23. In tab with computer check that ping isn’t successful 24. In tab with VM1 run iperf -s to check tcp traffic 25. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP> 26. Verify that TCP traffic is allowed 27. In tab with computer run iperf -s to check tcp traffic 28. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id> 29. Verify that TCP traffic is allowed
	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Create new ubuntu instance VM1 in non admin tenant (add all rules for icmp, tcp and udp to sec groups as for admin tenant). Associate Floating IP for this VM1 3. In CLI go to this VM1: ssh ubuntu@<floating_ip> 4. Send ICMP(using ping util: ping) traffic from VM1 to your computer and verify that all packages are encrypted (using tcpdump on your computer: sudo tcpdump -i any -v icmp). 5. Return to the dashboard 6. Navigate to Project->Network->Firewalls 7. Create firewall rule for the tcp protocol to allow tcp traffic for all ports: Protocol = TCP, action = ALLOW, shared = True 8. Create shared firewall policy with this rule (mark share) 9. Sign out and login to another tenant 10. Navigate to Project -> Network - > Firewalls 11. Navigate to tab Firewalls 12. Click on Create firewall 13. Type name 1, choose shared policy and click on Add 14. Upload page and check that status is active 15. In tab with computer: sudo tcpdump -i any icmp

	16. In tab with VM1 try to ping your computer: ping <computer_IP> 17. In tab with computer check that ping isn't successful 18. In tab with VM1 run iperf -s to check tcp traffic 19. In tab with computer run (to start tcp traffic): iperf -P 10 -c <VM1_IP> 20. Verify that TCP traffic is allowed 21. In tab with computer run iperf -s to check tcp traffic 22. In tab with VM1 run (to start tcp traffic): iperf -P 10 -c <computer_id> 23. Verify that TCP traffic is allowed
Expected Result	All steps should be passed, we should have the ability to send tcp traffic for all ports, for port 2000 it should be impossible

Title	**TC 013: Negative: Create firewall with empty policy**
Test Case ID	negative_create_firewall_with_empty_policy
Prerequisites	1 pre deployed cloud with firewall plugin
Steps	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project -> Network -> Firewalls 3. Choose tab Firewalls 4. Click on Create Firewall button 5. Type name and click on Add button 6. Check that firewall can't be added and there is error message "This field is required" under the Policy field
Expected Result	All steps should be passed, a firewall should not created with empty policy

Title	**TC 014: Negative: Create 2 firewalls**
Test Case ID	negative_create_2_firewalls
Prerequisites	1 pre deployed cloud with firewall plugin

Steps	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project->Network->Firewalls 3. Navigate to tab Firewall Rules 4. Click on Add Rule button to create Firewall rule with name “all_tcp”. In the field Protocol choose TCP and click on Add 5. Click on Add Rule button to create Firewall rule with name “all_udp”. In the field Protocol choose UDP and click on Add 6. Navigate to tab Firewall Policies 7. Click on Add Policy button. Type name “1” on the field Name on AddPolicy tab 8. Go to the tab Rules 9. Click on + near all_tcp in the Available Rules 10. Click on Add button 11. Click on Add Policy button. Type name “2” on the field Name on AddPolicy tab 12. Go to the tab Rules 13. Click on + near all_udp in the Available Rules 14. Click on Add button 15. Go to the tab Firewalls 16. Click on Create Firewall 17. Type name. Choose created policy 18. Click on Add 19. Update the page and check that status is active 20. Click on Create Firewall 21. Type name 2 and choose policy 2 22. Click on Add 23. Check that we can’t create one more firewall wirr error message
Expected Result	All steps should be passed, the second firewall should not created

Title	**TC 015: Delete policy which is used in the firewall**
Test Case ID	delete_used_policy
Prerequisites	1 pre deployed cloud with firewall plugin
Steps	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project->Network->Firewalls 3. Navigate to tab Firewall Rules 4. Click on Add Rule button to create Firewall rule with name “all_tcp”. In the field Protocol choose TCP and click on Add

	5. Navigate to tab Firewall Policies 6. Click on Add Policy button. Type name “1” on the field Name on AddPolicy tab 7. Go to the tab Rules 8. Click on + near all_tcp in the Available Rules 9. Click on Add button 10. Go to the tab Firewalls 11. Click on Create Firewall 12. Type name. Choose created policy 13. Click on Add 14. Update the page and check that status is active 15. Navigate to tab Firewall policies 16. Click on down arrow near policy 1 17. Click on delete 18. Check that the policy can’t be deleted if it used in the firewall
Expected Result	All steps should be passed, the policy shouldn’t be deleted if it used in the firewall

Title	**TC 016: Delete rule which is used in the policy**
Test Case ID	delete_used_policy
Prerequisites	1 pre deployed cloud with firewall plugin
Steps	Dashboard: 1. Login to OpenStack Horizon dashboard 2. Navigate to Project->Network->Firewalls 3. Navigate to tab Firewall Rules 4. Click on Add Rule button to create Firewall rule with name “all_tcp”. In the field Protocol choose TCP and click on Add 5. Navigate to tab Firewall Policies 6. Click on Add Policy button. Type name “1” on the field Name on AddPolicy tab 7. Go to the tab Rules 8. Click on + near all_tcp in the Available Rules 9. Click on Add button 10. Navigate to tab Firewall rules 11. Click on down arrow near rule all_icmp 12. Click on delete 13. Check that the rule can’t be deleted if it used in the policy
Expected Result	All steps should be passed, the rule shouldn’t be deleted if it used in the policy

Destructive testing

Title	**TC 017: Destroy primary controller**
Test Case ID	destructive_destroy_primary_controller
Prerequisites	1 pre deployed cloud with firewall plugin
Steps	1. Create firewall as in case 10 2. Find primary controller with command on the controllers: hiera role 3. Destroy this controller from lab (choose variant which your lab supports) a) virsh destroy <node> b) VBoxManage controlvm fuel-slave-2 poweroff 4. Wait 30 sec 5. In tab with VM1 run iperf -s -p 2000 to check tcp traffic 6. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2000 -c <VM1_IP> 7. Verify that TCP traffic is allowed 8. In tab with computer run iperf -s -p 2000 to check tcp traffic 9. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2000 -c <computer_id> 10. Verify that TCP traffic is allowed 11. In tab with VM1 run iperf -s -p 1000 to check tcp traffic 12. In tab with computer run (to start tcp traffic): iperf -P 10 -p 1000 -c <VM1_IP> 13. Verify that TCP traffic is lost 14. In tab with computer run iperf -s -p 1000 to check tcp traffic 15. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 1000 -c <computer_id> 16. Verify that TCP traffic is lost
Expected Result	All steps should be passed, firewall should work after destroy primary controller

Title	**TC 018: Reset primary controller**
Test Case ID	destructive_reset_primary_controller
Prerequisites	1 pre deployed cloud with firewall plugin
Steps	1. Create firewall as in case 10 2. Find primary controller with command on the controllers: hiera role 3. Reset this controller from lab (choose variant which your lab supports) a) virsh reset <node> b) VBoxManage controlvm fuel-slave-2 reset 4. Wait 30 sec 5. In tab with VM1 run iperf -s -p 2000 to check tcp traffic 6. In tab with computer run (to start tcp traffic): iperf -P 10 -p 2000 -c <VM1_IP> 7. Verify that TCP traffic is allowed 8. In tab with computer run iperf -s -p 2000 to check tcp traffic 9. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 2000 -c <computer_id> 10. Verify that TCP traffic is allowed 11. In tab with VM1 run iperf -s -p 1000 to check tcp traffic 12. In tab with computer run (to start tcp traffic): iperf -P 10 -p 1000 -c <VM1_IP> 13. Verify that TCP traffic is lost 14. In tab with computer run iperf -s -p 1000 to check tcp traffic 15. In tab with VM1 run (to start tcp traffic): iperf -P 10 -p 1000 -c <computer_id> 16. Verify that TCP traffic is lost
Expected Result	All steps should be passed, firewall should work after reset primary controller

System testing

Title	**TC 019: Install plugin and deploy environment**
Test Case ID	install_plugin_deploy_env
Prerequisites	1 non deployed cloud
Steps	1. Upload plugin to the master node 2. Install plugin: fuel plugins --install fwaas-plugin-<x.x.x>.rpm 3. Ensure that plugin is installed successfully using cli: fuel plugins 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 1 node with Compute role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc) 10. Run OSTF
Expected Result	Plugin is installed successfully, cluster is created, network verification and OSTF are passed, and all plugin services is enabled and worked as expected.

Title	**TC 020: Modifying env with enabled plugin (removing/adding controller nodes)**
Test Case ID	modify_env_with_plugin_remove_add_controller
Prerequisites	1 non deployed cloud
Steps	1. Upload plugin to the master node 2. Install plugin: fuel plugins --install fwaas-plugin-<x.x.x>.rpm 3. Ensure that plugin is installed successfully using cli: fuel plugins 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 1 node with Compute role 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc)

	10. Run OSTF 11. Remove 1 nodes with Controller role /*remove node, where plugin's services available, to ensure that according to ha mode all plugins resources will be replaced and available on another live node and continue to work as expected*/ 12. Re-deploy cluster 13. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc) 14. Run OSTF 15. Add 1 new node with Controller role 16. Re-deploy cluster 17. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc) 18. Run OSTF
Expected Result	Plugin is installed successfully, cluster is created, network verification and OSTF are passed, and all plugin services is enabled after migration in ha mode and worked as expected after modifying of environment.

Title	**TC 021: Modifying env with enabled plugin (removing/adding compute node)**
Test Case ID	modify_env_with_plugin_remove_add_compute
Prerequisites	1 non deployed cloud
Steps	1. Upload plugin to the master node 2. Install plugin: fuel plugins --install fwaas-plugin-<x.x.x>.rpm 3. Ensure that plugin is installed successfully using cli: fuel plugins 4. Create environment with enabled plugin in fuel ui 5. Add 3 nodes with Controller role and 2 nodes with compute roles 6. Apply network settings 7. Run network verification 8. Deploy the cluster 9. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc)

	10. Run OSTF 11. Remove 1 compute node 12. Re-deploy cluster 13. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc) 14. Run OSTF 15. Add 1 compute node 16. Re-deploy cluster 17. Check plugin health using cli on controller: neutron firewall-list (after deployment the list should be empty, run this command after . openrc) 18. Run OSTF
Expected Result	Plugin is installed successfully, cluster is created, network verification and OSTF are passed, and all plugin services is enabled and worked as expected after modifying of environment.

Title	**TC 022: Uninstall of plugin**
Test Case ID	positive_uninstall_plugin
Prerequisites	1 non deployed cloud
Steps	1. Install plugin: fuel plugins --install fwaas-plugin-<x.x.x>.rpm 2. Check that it was successfully installed: fuel plugins 3. Remove plugin: fuel plugins --remove fwaas-plugin==<version> 4. Check that it was successfully removed: fuel plugins
Expected Result	Plugin was installed and then removed successfully

Title	**TC 023: Negative: Uninstall of plugin with deployed env**
Test Case ID	negative_uninstall_plugin
Prerequisites	1 non deployed cloud
Steps	1. Install plugin: fuel plugins --install fwaas-plugin-<x.x.x>.rpm 2. Deploy env with this plugin 3. Run OSTF

	4. Try to delete plugin and ensure that present in cli alert: "400 Client Error: Bad Request (Can't delete plugin which is enabled for some environment.)" 5. Remove env 6. Remove plugin: fuel plugins --remove fwaas-plugin==<version> 7. Check that it was successfully removed: fuel plugins
Expected Result	Plugin was installed successfully. Alert is present when we trying to delete plugin which is attached to enabled environment. When environment was removed, plugin is removed successfully too.