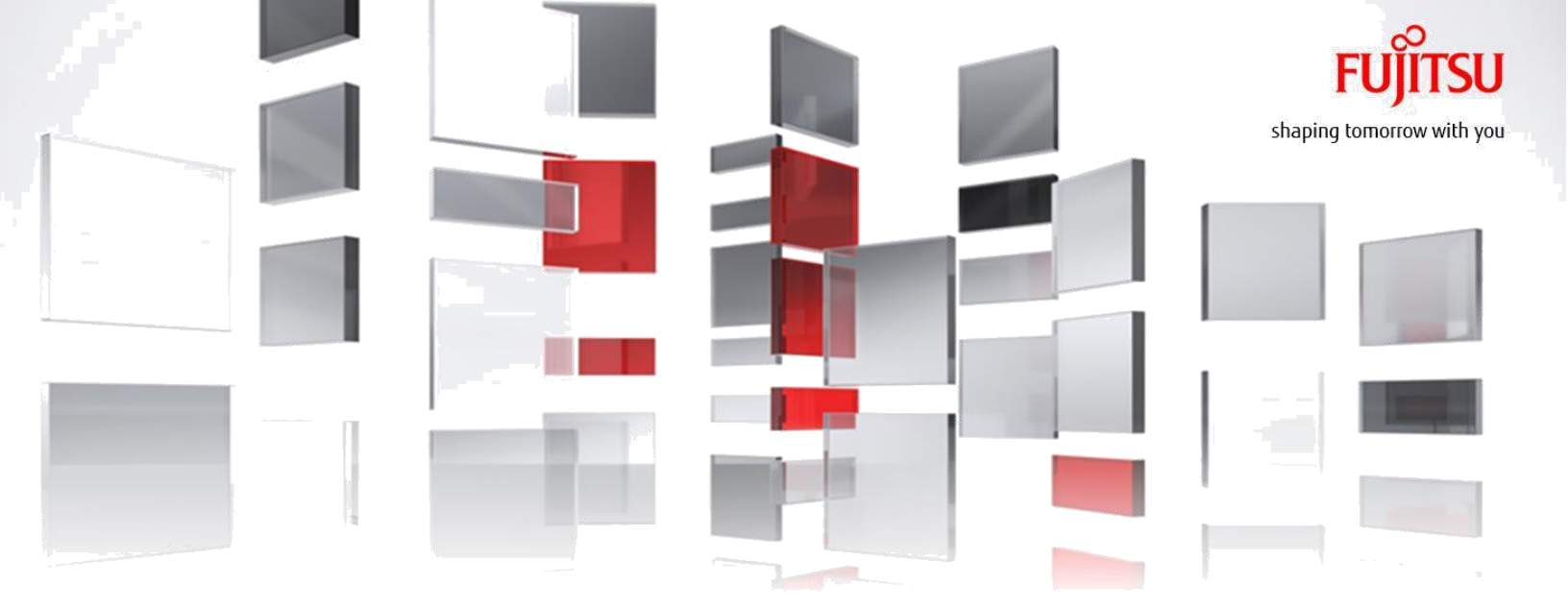


An abstract graphic at the top of the page composed of numerous semi-transparent, 3D rectangular blocks in white, grey, and red, arranged in a complex, overlapping geometric pattern.

FUJITSU Cloud Service K5 IaaS API User Guide

UK Version 1.0
FUJITSU LIMITED

Preface

Purpose of this Manual

This document explains how to use the FUJITSU Cloud Server K5 IaaS REST API by providing sample scripts. It consists of two parts:

- Preparing the environment to use the API which includes shell setup as well as managing projects, users, groups and roles.
- API usage detailing how to create a basic implementation as well as how to delete resources from an implementation.

Audience for this Manual

This manual is intended for those involved in the planning and developing of applications or services using K5 IaaS. This manual is written on the assumption that the reader possesses the following knowledge:

- Basic knowledge of virtualization technology (hypervisors, virtual servers, virtual storage, virtual networks)
- Basic knowledge of OpenStack
- Basic knowledge of your OS
- Basic knowledge of the Internet and Intranet
- Basic security knowledge
- Basic knowledge of system operation, including backups, monitoring, and redundancy

Organisation of Manuals

Refer to the related manuals listed below according to your purposes and methods of use.

Manual Title	Purposes and Methods of Use
IaaS Features Handbook	This document explains the functions provided by this service in detail.
IaaS API User Guide	This document provides instructions on how to use the K5 REST APIs, including how to build an API execution environment and how to use a sample scripts.
IaaS API Reference	This document provides detailed information about how to use REST API.
IaaS Heat Template Specifications	This document explains the format of the Heat Orchestration Template (HOT) used for orchestration.
IaaS Service Portal User Guide	This document explains how to use the functions provided by this service via Service Portal (Web GUI).

Abbreviations Used in This Manual

In this manual, product names are abbreviated as follows.

Official Name	Abbreviation	
FUJITSU Cloud Service K5 IaaS	K5 IaaS	
Microsoft® Windows Server® 2012 SE R2	Windows 2012 R2	Windows
Microsoft® Windows Server® 2008 SE R2	Windows 2008 R2	
Microsoft® Windows Server® 2008 EE R2		
Red Hat® Enterprise Linux® 6.5 (for Intel64)	RHEL6.5	Linux
Community Enterprise Operating System 6.5	CentOS6.5	CentOS
Red Hat Update Infrastructure	RHUI	
Windows Server Update Services	WSUS	

Trademarks

- Microsoft, Windows, Windows Server and other Microsoft product names and model names are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Java is a registered trademark of Oracle Corporation and its subsidiaries or affiliates in the United States and/or other countries.
- Xeon is a trademark of Intel Corporation in the United States and/or other countries.
- Linux® is a registered trademark of Linus Torvalds in the United States and/or other countries.
- Red Hat and Red Hat Enterprise Linux are trademarks of Red Hat, Inc. registered in the United States and/or other countries.
- Ubuntu is a registered trademark of Canonical Ltd.
- OpenStack is a registered trademark of OpenStack, LLC in the United States.
- Other company names and product names mentioned in this manual are trademarks or registered trademarks of their respective companies.

In this manual, the registered trademark symbols (™ or ®) next to system names or product names have been omitted.

Export Administration Regulations

When exporting or giving this document to a third party, be sure to familiarize yourself with the regulations related to export administration valid in your country of residence and the United States, and follow the necessary procedures.

Note....

The content of this manual may change without prior notice.

The reproduction of this manual without permission is prohibited.

Fujitsu do not assume responsibility for any violation of patent rights or any other rights of a third party that may occur due to the use of the data in this manual.

Contents

PREFACE	2
PURPOSE OF THIS MANUAL	2
AUDIENCE FOR THIS MANUAL	2
ORGANISATION OF MANUALS	2
ABBREVIATIONS USED IN THIS MANUAL	3
TRADEMARKS	3
EXPORT ADMINISTRATION REGULATIONS	3
NOTE.....	4
CONTENTS	5
PART I	7
PREPARING THE ENVIRONMENT	7
1. BUILDING THE API OPERATING ENVIRONMENT	8
1.1 ENVIRONMENT.....	9
1.2 CREATING A WORKING DIRECTORY.....	9
1.3 RETRIEVING THE AUTHENTICATION TOKEN.....	12
2. MANAGING PROJECTS.....	14
2.1 CREATING A PROJECT.....	15
3. MANAGING GROUPS.....	17
3.1 CREATING A GROUP.....	18
3.2 ADDING A USER TO A GROUP.....	19
3.3 DELETING A USER FROM A GROUP	20
3.4 DELETING A GROUP	21
4. MANAGING ROLES	22
4.1 LISTING THE EXISTING ROLES.....	23
4.2 ASSIGNING AND REFERENCING A ROLE.....	25
PART II	27
APPENDIX	27
5. APPENDIX 1 - BASIC IMPLEMENTATION - CONFIGURING A VIRTUAL SERVER	28
5.1 BASIC STRUCTURE	29
5.2 CREATING A NETWORK.....	30
5.3 CREATING A SUBNET.....	32
5.4 CREATING A VIRTUAL ROUTER.....	35
5.5 CHANGING THE CONFIGURATION OF A VIRTUAL ROUTER (ATTACHING TO AN EXTERNAL NETWORK)	37

5.6	CHANGING THE CONFIGURATION OF A VIRTUAL ROUTER (ATTACHING TO A SUBNET)	38
5.7	CREATING A SECURITY GROUP.....	40
5.8	CREATING A SECURITY GROUP RULE	42
5.9	EXAMPLES OF SECURITY GROUP RULES.....	44
5.10	CREATING A FIREWALL RULE (IP ADDRESS SPECIFICATION)	46
5.11	CREATING A FIREWALL RULE (PORT NUMBER SPECIFICATION)	48
5.12	CREATING A FIREWALL RULE (ICMP PERMISSION)	51
5.13	CREATING A FIREWALL RULE (DENY RULE)	53
5.14	CREATING A FIREWALL POLICY	55
5.15	CREATING A FIREWALL	58
5.16	CREATING A KEY PAIR.....	60
5.17	CREATING A PORT	62
5.18	PREPARING TO CREATE A VIRTUAL SERVER (LISTING VIRTUAL SERVER IMAGES)	64
5.19	PREPARING TO CREATE A VIRTUAL SERVER (LISTING VIRTUAL SERVER TYPES)	66
5.20	CREATING A VIRTUAL SERVER (CENTOS / PORT SPECIFICATION).....	68
5.21	CREATING A VIRTUAL SERVER (CENTOS / DHCP RETRIEVAL)	70
5.22	CREATING A VIRTUAL SERVER (WINDOWS / PORT SPECIFICATION).....	73
5.23	CREATING A VIRTUAL SERVER (WINDOWS / DHCP RETRIEVAL)	76
5.24	RETRIEVING A VIRTUAL SERVER PORT.....	79
5.25	ALLOCATING A GLOBAL IP ADDRESS AND ASSIGNING IT TO A VIRTUAL SERVER	80
5.26	LOGGING INTO A VIRTUAL SERVER (CENTOS).....	82
5.27	LOGGING INTO A VIRTUAL SERVER (WINDOWS).....	82
6.	APPENDIX 2 - DELETING RESOURCES	84
6.1	DELETING A VIRTUAL SERVER.....	85
6.2	RELEASING A GLOBAL IP ADDRESS	85
6.3	DELETING A PORT.....	86
6.4	DELETING A KEY PAIR	86
6.5	DELETING A FIREWALL.....	87
6.6	DELETING A FIREWALL POLICY	87
6.7	DELETING A FIREWALL RULE	88
6.8	DELETING A SECURITY GROUP RULE.....	88
6.9	DELETING A SECURITY GROUP	89
6.10	DETACHING A VIRTUAL ROUTER FROM A SUBNET	89
6.11	DELETING A VIRTUAL ROUTER	90
6.12	DELETING A SUBNET	90
6.13	DELETING A NETWORK	91

Part I

Preparing the environment

Topics:

- [Building the API operating environment](#)
- [Managing projects](#)
- [Managing groups](#)
- [Managing roles](#)

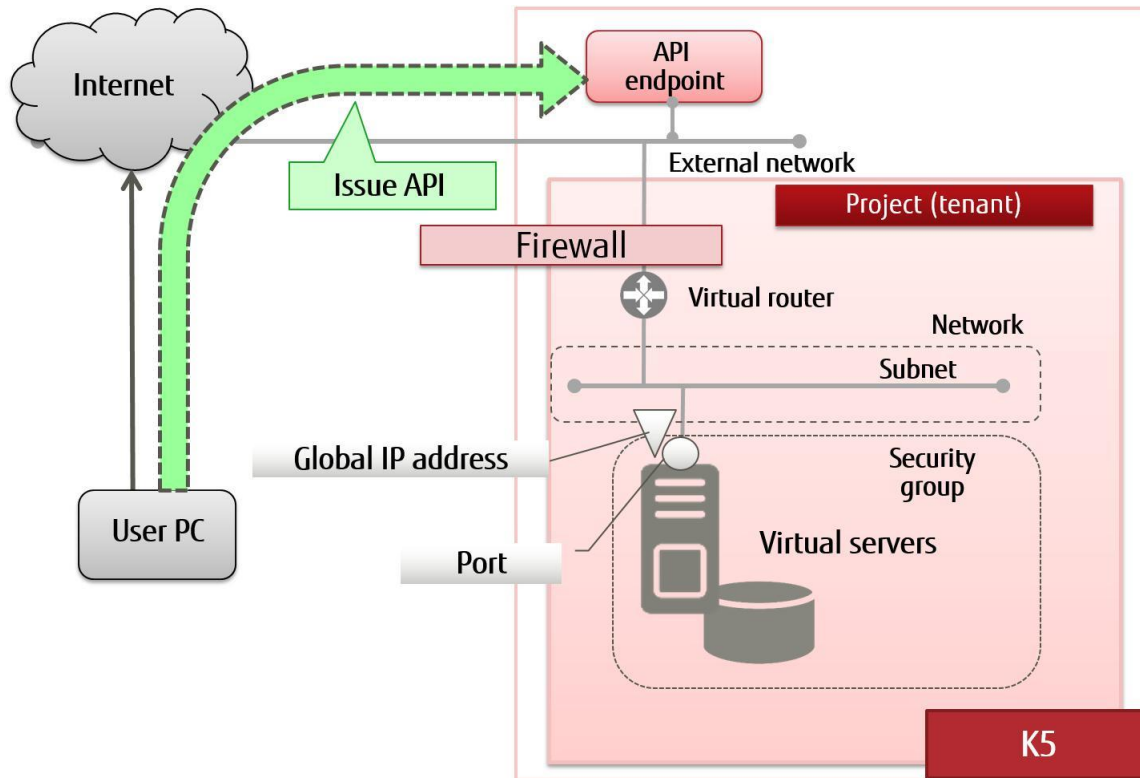
1. Building the API operating environment

Topics:

- [Environment](#)
- [Creating a working directory](#)
- [Retrieving the authentication token](#)

1.1 Environment

This section provides an overview of how to set up an environment to enable the use of the K5 IaaS API. The user (User PC) accesses the API over the internet. This document assumes that the user environment is CentOS.



1.2 Creating a working directory

This section explains how to create a working directory and two scripts: the first to set the required environment variables and the second to retrieve an authentication token. This token will be used to authenticate all future API calls.

1. Create the working directory and then change into it.

```
$ mkdir <anyName>
$ cd <anyName>
```

2. Environment set up script.

Create "init.sh" script substituting the relevant values for those items in <angular brackets>. This sets up environment variables that will be used by other scripts (e.g. get_token.sh) and commands.

init.sh

```
#!/bin/bash
# Account information.
DOMAIN_NAME=<contractNum(Domain)>
DOMAIN_ID=<domainID>
TENANT_ID=<projID>
PROJECT_ID=$TENANT_ID
USER_NAME=<userName>

USER_PW=<password>

# Endpoint shortcut. echo "EP initial setup."
TOKEN=https://identity.cloud.global.fujitsu.com
IDENTITY=$TOKEN
NETWORK=https://networking.uk-1.cloud.global.fujitsu.com
COMPUTE=https://compute.uk-1.cloud.global.fujitsu.com
CEILOMETER=https://telemetry.uk-1.cloud.global.fujitsu.com
TELEMETRY=$CEILOMETER
DB=https://database.uk-1.cloud.global.fujitsu.com
BLOCKSTORAGE=https://blockstorage.uk-1.cloud.global.fujitsu.com
HOST_BLOCKSTORAGEV2=$BLOCKSTORAGE
OBJECTSTORAGE=https://objectstorage.uk-1.cloud.global.fujitsu.com
ORCHESTRATION=https://orchestration.uk-1.cloud.global.fujitsu.com
ELB=https://loadbalancing.uk-1.cloud.global.fujitsu.com
AUTOSCALE=https://autoscale.uk-1.cloud.global.fujitsu.com
IMAGE=https://image.uk-1.cloud.global.fujitsu.com
MAILSERVICE=https://mail.uk-1.cloud.global.fujitsu.com
NETWORK_EX=https://networking-ex.uk-1.cloud.global.fujitsu.com
DNS=https://dns.cloud.global.fujitsu.com

# Initial setup
NAME_FORMAT="TES_$(date +%m%d)_$(who am I | cut -d " " -f1)_"
# Other
alias curl='curl --tlsv1.2' SCRIPT_PATH=`pwd` RES_DIR=response
RES_PATH=$SCRIPT_PATH/$RES_DIR
```

 **Note:** Retrieve the domain ID from the domain ID column under Management > User Management > Group on the Service Portal.

 **Note:** Refer to "Retrieving the authentication token" for details on how to acquire API privileges, and refer to "Creating a project" and "Creating a user" for details on how to enter the project name, project ID and user ID.

 **Note:** This guide uses the command line tool cURL at version 7.45.0.

 **Note:** The API Endpoints in the script above are for the UK Region of K5 IaaS. You can find API Endpoints specific to your projects under Compute > Access & Security > API Access on the Service Portal

3. Run the `get_token.sh` script to retrieve the authentication token.

 **Note:** all commands in this script must be on a single line and not over multiple lines

`get_token.sh`

```
#!/bin/bash

## Script to retrieve token
. ~/<anyName>/init.sh TMPFILE=~/<anyName>/token.txt
echo ""
echo "*****"
echo "**          Retrieve then display token          **"
echo "*****"
echo ""
echo 'Setting content'
echo 'endpoint':$TOKEN
echo ' domain_name':$DOMAIN_NAME echo ' domain_id':$DOMAIN_ID echo '
user_name':$USER_NAME echo ' user_pw':$USER_PW

echo 'project_id':$PROJECT_ID

echo "CURL"

echo 'curl -k -X POST '$TOKEN'/v3/auth/tokens -H "Content-Type:application/
json" -H "Accept:application/json" -d'

echo '{"auth":{"identity":{"methods":["password"],"password":{"user":
{"domain":{"name":"'$DOMAIN_NAME'"},"name": "'$USER_NAME'", "password":
"'$USER_PW'"}}}}, "scope": { "project": { "id": "'$PROJECT_ID'"}}}}' | jq .

echo -n "***** Hit Enter Key *****"

read

curl -k -X POST -si $TOKEN/v3/auth/tokens -H "Content-Type: application/ json" -
H "Accept:application/json" -d '{"auth":{"identity":{"methods":
["password"],"password":{"user":{"domain":{"name":"'$DOMAIN_NAME'"}, "name":
"'$USER_NAME'", "password": "'$USER_PW'"}}}}, "scope": { "project": { "id":
"'$PROJECT_ID'"}}}}' | awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr - d
'\r\n'

OS_AUTH_TOKEN=`cat $TMPFILE | tr -d '\r\n'`

echo "=== Retrieved authentication token starts from here ==="
echo $OS_AUTH_TOKEN
```

```
echo "=== Retrieved authentication token ends here ==="
```

4. Confirmation.

Verify that the scripts have been created in the current directory.

```
$ ls -l
Total 2
-rw-r--r-- 1 username Domain Users 0 MM DD hh:mm get_token.sh
-rw-r--r-- 1 username Domain Users 0 MM DD hh:mm init.sh
```

1.3 Retrieving the authentication token

This section explains the process to retrieve the authentication token required to use the API.

 **Note:** It is assumed that the working directory and two setup scripts have been already been created.

1. Move to the working directory.

```
$ cd <anyName>
```

2. Execute the command.

Token will be overwritten in to the "OS_AUTH_TOKEN" variable (by the get_token.sh script) when the following command is executed.

```
$ . ./get_token.sh
```

Example of a successful response:

```
EP initial setup.

*****
**      Retrieve then display token      **
*****

-Setting content
endpoint:https://identity.cloud.global.fujitsu.com
domain_id:<domainId>
user_name:<userName>
user_id:<userId>
user_pw:<userPassword>

-CURL
```

```
curl -k -X POST https://identity.cloud.global.fujitsu.com/v3/auth/tokens -H
"Content-Type:application/json" -H "Accept:application/json" -d
{
  "auth": {
    "identity": {
      "methods": [ "password"], "password": {
        "user": {
          "domain": {
            "name": "<domainName>"
          },
          "name": "<userName>",
          "password": "<userPassword>"
        }
      }
    }
  }
}

=== Retrieved authentication token starts here ===
<anyString>
=== Retrieved authentication token ends here ===
```

API operations can now be performed in this environment using the retrieved token.

2. Managing projects

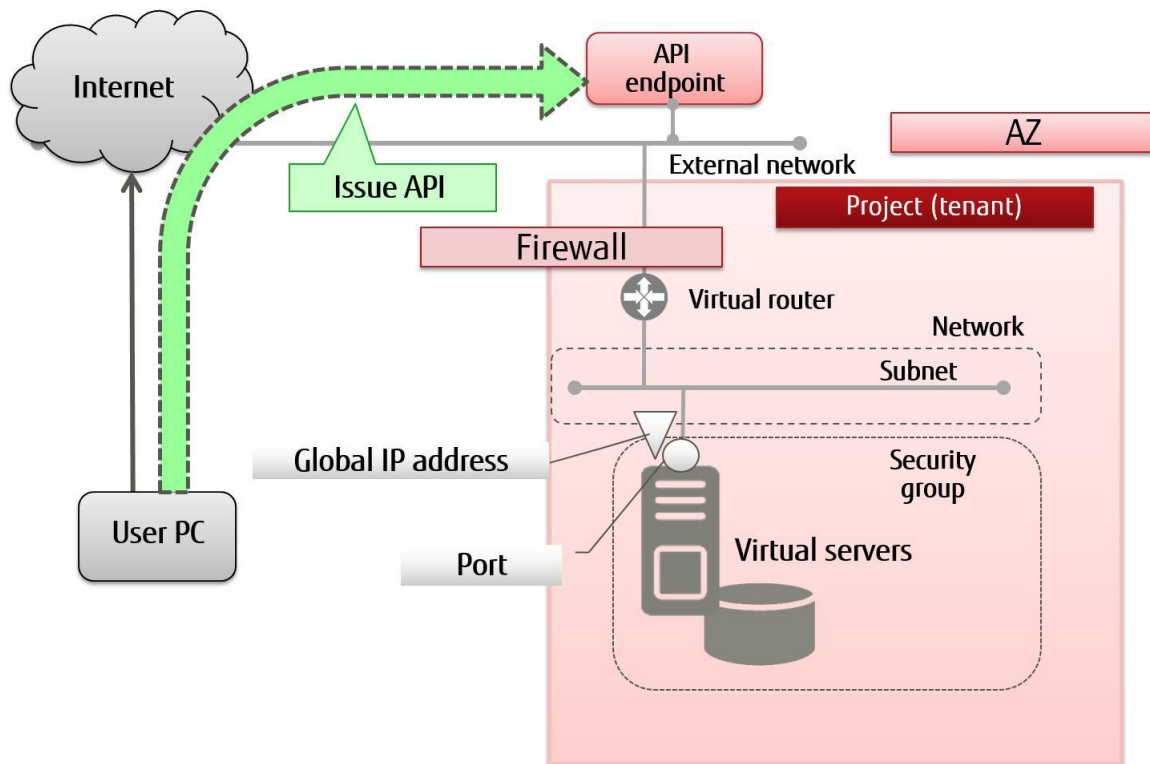
Topics:

- [Creating a project](#)

2.1 Creating a project

This section explains the process to create and verify a project. The virtual resources, groups and users to be used by a domain in K5 are divided and managed into a unit called a project.

It should be noted that a project can also be referred to as a "tenant" in the API.



 **Note:** a created project cannot be deleted.

1. Set environment variables as below:

```
$ TENANT_NAME=<projName>
$ DOMAIN_ID=<domainID>
$ DESCRIPTION=<projDesc>
```

2. Execute the following API call:

```
$ curl -X POST -s $IDENTITY/v3/projects -H "X-Auth-Token:$OS_AUTH_TOKEN" -H
"Content-Type:application/json" -d '{"project": {"name": "'$TENANT_NAME'",
"description": "'$DESCRIPTION'", "domain_id": "'$DOMAIN_ID'", "enabled": true}}' |
jq .
```

Successful response:

```
{
  "project": {
    "description":
      "<projDesc", "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/projects/
<createdProjId>"
      },
    "enabled": true,
    "id": "<createdProjId>",
    "domain_id": "<domainID>",
    "name": "<createdProjectName>"
  }
}
```

3. Execute the following API call to verify that the project has been created:

```
$ curl -X GET -s $IDENTITY/v3/projects?domain_id=$DOMAIN_ID -H "X-Auth-
Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

If the list of projects contains the project name then it has been created successfully.

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/projects",
    "previous": null,
    "next": null
  },
  "projects": [
    ...
    {
      "description":
        "<projDesc>", "links": {
          "self": "http://identity.cloud.global.fujitsu.com/v3/projects/
<createdProjId>"
        },
      "enabled": true,
      "id": "<createdProjId>",
      "domain_id": "<domainID>",
      "name": "<createdProjName>"
    },
    ...
  ]
}
```

3. Managing groups

Topics:

- [Creating a group](#)
- [Adding a user to a group](#)
- [Deleting a user from a group](#)
- [Deleting a group](#)

3.1 Creating a group

This section explains the process to create and verify a group. A group can contain zero, one or more users and is used in role management.

 **Note:** Multiple groups can be created in a domain; groups cannot be nested within each other.

1. Set environment variables as below:

```
$ TMP_GROUP_NAME=<newGroupName>
$ DOMAIN_ID=<domainID>
$ DESCRIPTION=<groupDesc>
```

2. Execute the following API call:

```
$ curl -X POST -s $IDENTITY/v3/groups -H "X-Auth-Token: $OS_AUTH_TOKEN" -H
"Content-Type:application/json" -d '{"group": {"description": "'$DESCRIPTION'",
"domain_id": "'$DOMAIN_ID'", "name": "'$TMP_GROUP_NAME'"}' | jq .
```

Successful response:

```
{
  "group": {
    "domain_id": "<domainNameThatGroupBelongsTo>",
    "description": "<groupDesc>",
    "id": "<newGroupId>",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/groups/ <newGroupId>"
    },
    "name": "<newGroupName>"
  }
}
```

3. Verify created groups by listing all the groups in a domain with the following API call. Only groups within the domain specified will be retrieved.

```
$ curl -X GET -s $IDENTITY/v3/groups?domain_id=$DOMAIN_ID -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

If the response contains the newly created group then the group has been successfully created.

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",
    "previous": null,
    "next": null
  },
}
```

```

"groups": [
  ...
  {
    "domain_id": "<domainId>",
    "description": "<groupDesc>",
    "id": "<groupId>",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/groups/ <groupId>"
    },
    "name": "<groupName>"
  },
  ...
]
}

```

3.2 Adding a user to a group

This section explains the process to add and verify a user to a group.

Users can inherit all the roles of a particular group by adding them to that particular group.

1. Set environment variables as below:

```

$ TMP_USER_ID=<userIdToBeAdded>
$ TMP_GROUP_ID=<groupId>

```

2. Execute the following API call:

```

$ curl -i -X PUT -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users/$TMP_USER_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"

```

Successful response:

```

HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<anyNineDigitNum>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT

```

3. Verify that the user successfully added to the group by executing the following call:

```

$ TMP_GROUP_ID=<groupId>

$ curl -X GET -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .

```

Check the output to verify newly added users are in the correct group:

```
{
  "users": [
    ...
    {
      "domain_id": "<domainId>",
      "name": "<userName>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/users/<userId>"
      },
      "locale": "ja", "enabled": true,
      "id": "<userId>",
      "default_project_id": "<defaultProjId>",
      "description": "<userDesc>"
    },
    ...
  ], "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<groupId>/users",
    "next": null, "previous": null
  }
}
```

3.3 Deleting a user from a group

Users can also be deleted from a group. Follow the steps below to delete a user from a group:

1. Set the environment variables as below:

```
$ TMP_USER_ID=<userIdToBeDeleted>
$ TMP_GROUP_ID=<groupId>
```

2. Execute the following API call:

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users/$TMP_USER_ID -H
"X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

Example response:

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<nineDigitNum>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT
```

Verify users have been removed from the group by using the process above to list the members of a group and then check that the nominated user is no longer listed as a member.

3.4 Deleting a group

This section explains the process to delete and verify the deletion of a group.

1. Set the environment variables as below:

```
$ TMP_GROUP_ID=<groupIdToBeDeleted>
$ DOMAIN_ID=<domainID>
```

2. Execute the following API all:

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TMP_GROUP_ID -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

Successful response:

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<anyNineDigitNum>_204 Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT
```

3. Verify the group has been deleted:

```
$ curl -X GET -s $IDENTITY/v3/groups?domain_id=$DOMAIN_ID -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

Verify the group has been deleted. If the group is not in the list of domain groups it has been successfully deleted:

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",
    "previous": null,
    "next": null
  },
  "groups": [
    ...
    {
      "domain_id": "<domainId>",
      "description": "<groupDesc>",
      "id": "<groupId>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/groups/ <groupId>"
      },
      "name": "<groupName>"
    },
    ...
  ]
}
```

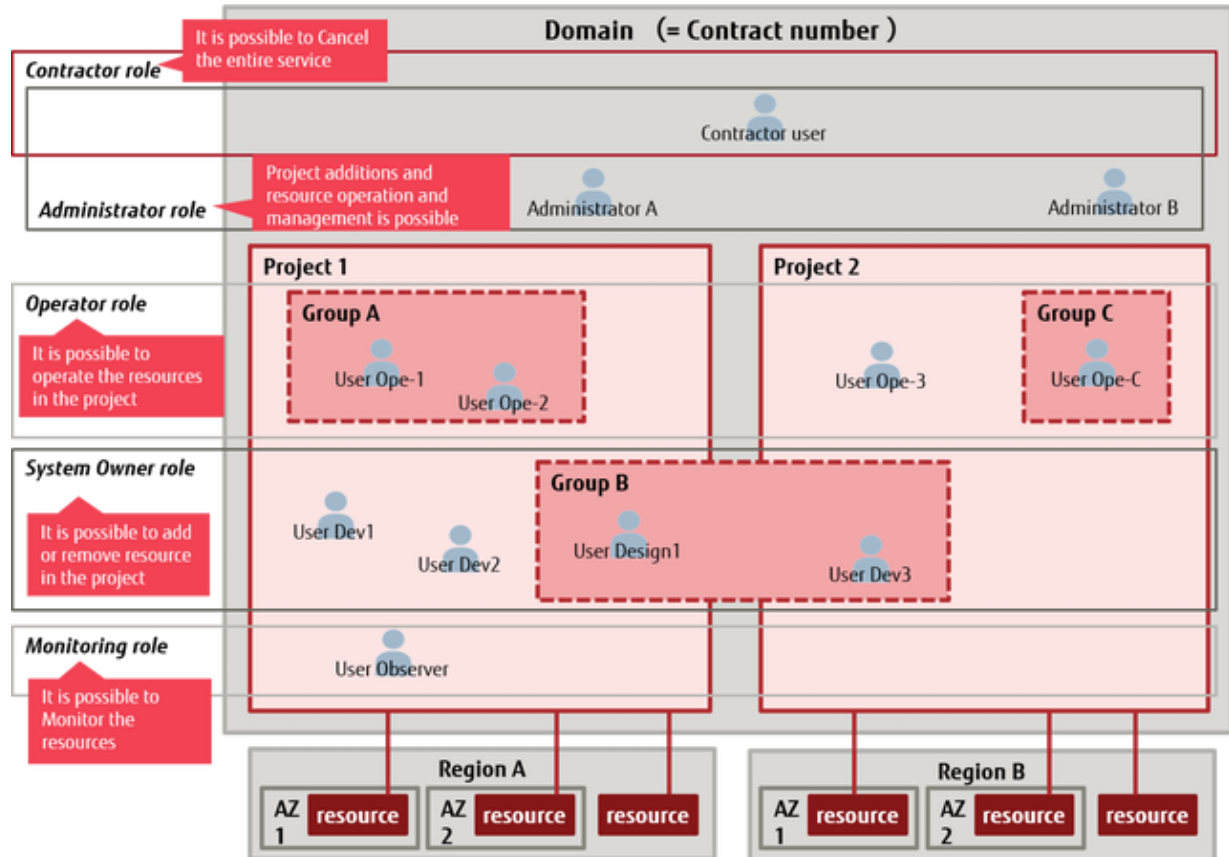
4. Managing roles

Topics:

- [Listing the existing roles](#)
- [Assigning and referencing a role](#)

4.1 Listing the existing roles

This section explains the process to list the pre-set roles that decide the operating privileges of users.



1. List all roles by execute the following API call:

```
$ curl -X GET -s $IDENTITY/v3/roles?domain_id=$DOMAIN_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq '.roles[] | select(.policy_role != true)'
```

Successful response example. Refer to the description in the IaaS Function Manual for pre-defined roles and their privileges.

```
{
  "id": "0739580a550d4a0f9c78f45a9f038c05",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/roles/0739580a550d4a0f9c78f45a9f038c05"
  }
}
```

```

    },
    "name": "cpf_systemowner"
  }
  {
    "id": "20e572b88c544606863548f8abd4ff53", "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/20e572b88c544606863548f8abd4ff53"
    },
    "name": "cpf_operator"
  }
  {
    "id": "3af119c426a742999e7890f6d1f70b36", "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/3af119c426a742999e7890f6d1f70b36"
    },
    "name": "cpf_admin"
  }
  {
    "id": "970ea0105b514c16828626fe4dd50960", "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/970ea0105b514c16828626fe4dd50960"
    },
    "name": "cpf_observer"
  }
  {
    "id": "9fe2ff9ee4384b1894a90878d3e92bab", "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/9fe2ff9ee4384b1894a90878d3e92bab"
    },
    "name": "_member_"
  }
  {
    "id": "df7d043a09d34a7c9e2bad15926ee097", "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/df7d043a09d34a7c9e2bad15926ee097"
    },
    "name": "cpf_org_manager"
  }
}

```

4.2 Assigning and referencing a role

This section explains the process to assign a role to a user or group.
Select a role and assign to users or groups to allow project participation.

1. Set the environment variables as below:

User

```
$ TMP_PROJECT_ID=<projIdThatUserOrGroupIsToParticipateIn>
$ TMP_USER_ID=<userToParticipateInProj>
```

Group

```
$ TMP_GROUP=<groupToParticipateInProj>
$ TMP_ROLE_ID=<roleId>
```

2. Execute the following API call to assign a role:

User

```
$ curl -i -X PUT -s $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/
$TMP_USER_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-
Type:application/json"
```

Group

```
$ curl -i -X PUT -s $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/
$TMP_GROUP_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-
Type:application/json"
```

Example successful response:

```
HTTP/1.1 204 No Content
Vary:X-Auth-Token
Content-Length:0
Date:Www, DD MMM yyyy hh:mm:ss GMT
```

3. Verify the roles of users and groups.

Roles can be verified at 4 different levels:

1. Project - Group
2. Project - User
3. Domain - Group
4. Domain - User

Set the environment variables:

```
$ TMP_PROJECT_ID="<projId>"
$ TMP_DOMAIN_ID="<domainId>"
$ TMP_USER_ID"<userId>"
$ TMP_GROUP_ID="<groupId>"
```

4. Execute the following API calls:

```
$ curl -X GET $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/$TMP_GROUP_ID/ roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

```
$ curl -X GET $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/$TMP_USER_ID/ roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

```
$ curl -X GET $IDENTITY/v3/domains/$DOMAIN_ID/groups/$TMP_GROUP_ID/roles - H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

```
$ curl -X GET $IDENTITY/v3/domains/$DOMAIN_ID/users/$TMP_USER_ID/roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

Part II

Appendix

Topics:

- [Appendix 1 – Basic implementation – Configuring a virtual server](#)
- [Appendix 2 – Deleting Resources](#)

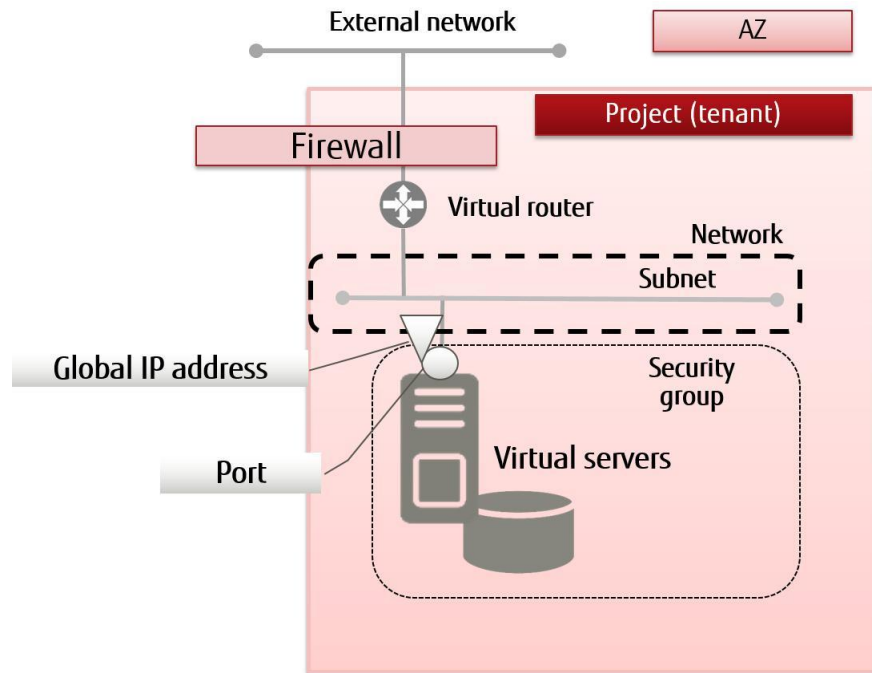
5. Appendix 1 - Basic implementation - Configuring a virtual server

Topics:

- [Basic structure](#)
- [Creating a network](#)
- [Creating a subnet](#)
- [Creating a virtual router](#)
- [Changing the configuration of a virtual router \(attaching to an external network\)](#)
- [Changing the configuration of a virtual router \(attaching to a subnet\)](#)
- [Creating a security group](#)
- [Creating a security group rule](#)
- [Examples of security group rules](#)
- [Creating a firewall rule \(IP address specification\)](#)
- [Creating a firewall rule \(port number specification\)](#)
- [Creating a firewall rule \(ICMP permission\)](#)
- [Creating a firewall rule \(deny rule\)](#)
- [Creating a firewall policy](#)
- [Creating a firewall](#)
- [Creating a key pair](#)
- [Preparing to create a virtual server \(listing virtual server images\)](#)
- [Preparing to create a virtual server \(listing virtual server types\)](#)
- [Creating a port](#)
- [Creating a virtual server \(CentOS / port specification\)](#)
- [Creating a virtual server \(CentOS / DHCP retrieval\)](#)
- [Creating a virtual server \(Windows / port specification\)](#)
- [Creating a virtual server \(Windows / DHCP retrieval\)](#)
- [Retrieving a virtual server port](#)
- [Allocating a global IP address and assigning it to a virtual server](#)
- [Logging into a virtual server \(CentOS\)](#)
- [Logging into a virtual server \(Windows\)](#)

5.1 Basic structure

This section provides an overview of how to create a basic implementation as detailed in the diagram below:



There are two main steps to creating the implementation: network creation/configuration; and virtual server creation. Network creation/configuration must be done before the creation of virtual servers. The sub-steps required for network creation/configuration include:

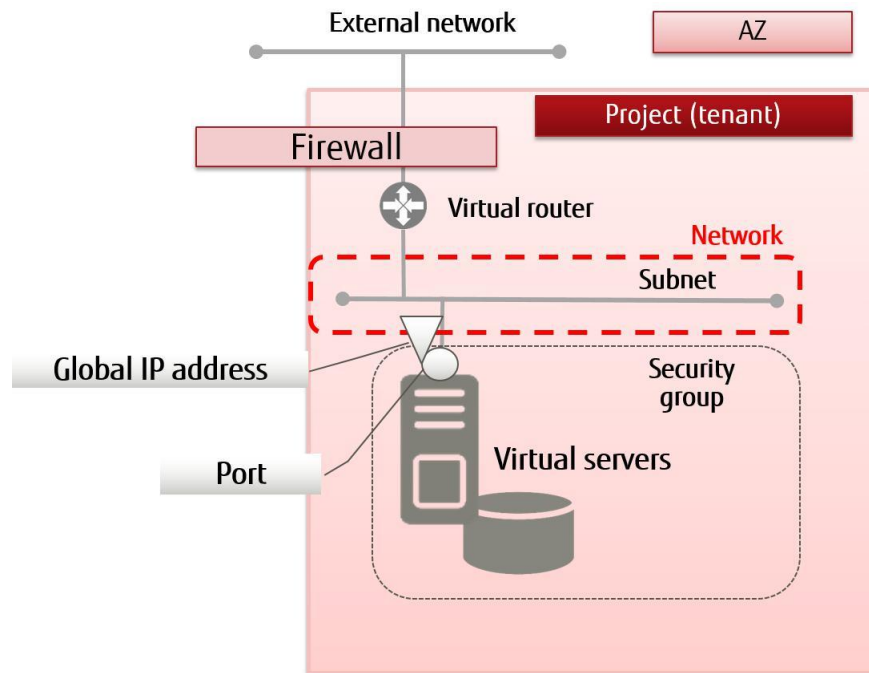
1. Create a network
2. Create a subnet
3. Create a virtual router
4. Attach the virtual router to the external network and subnet
5. Create a security group
6. Create security group rules
7. Create firewall rules
8. Create firewall policies
9. Create a firewall

The sub-steps required for virtual server creation include

1. Create a key pair
2. Create a port
3. Create a virtual server
4. Retrieve the global IP address and assign it to the virtual server

5.2 Creating a network

This section explains the process to create and verify a network within a project. This will enable the deployment of resources such as a virtual server. Multiple networks within a project can be created.



1. Set environment variables as below:

```
$ NW_NAME=<newNetworkName> (any)
$ AZ=<availabilityZoneName>
```

2. Execute the following API call:

```
$ curl -s $NETWORK/v2.0/networks -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" -H
"Content-Type: application/json" -d '{"network":{ "name": "'$NW_NAME'",
"availability_zone": "'$AZ'"}}' | jq .
```

Example successful response:

```
{
  "network": { "status":
    "ACTIVE",
    "subnets": [],
    "name": "<specifiedNetworkName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "shared": false,
```

```

    "id": "<newNetworkId>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

3. Execute the following API call to verify the network was created.

```
$ curl -s $NETWORK/v2.0/networks -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If the list of networks contains the new network name, then the network has been successfully created.

 **Note:** The network names "inf_az1_ext_net01" and "inf_az2_ext_net01" are external networks that exist by default.

```

{
  "networks": [
    {
      "status": "ACTIVE", "subnets": [

        "5079f324-5db0-44ee-92ac-3a6b7977b23f",
        "a56b6058-0479-43a1-8b27-01c1c05e96a2",
        "c1da3ee7-51c3-4801-bb97-aa03a4383ef0",
        "e96e55b8-84bb-4777-a782-a5d6e8340039",
        "f5e9ec37-88ec-494b-ac55-dae101a54cc1"
      ],
      "name": "inf_az2_ext-net01",
      "router:external": true,
      "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
      "admin_state_up": true,
      "shared": true,
      "id": "375c49fa-a706-4676-b55b-2d3554e5db6a",
      "availability_zone": "uk-1b"
    },
    ...

    {
      "status": "ACTIVE", "subnets": [],
      "name": "<networkName>",
      "router:external": false,
      "tenant_id": "<projId>",
      "admin_state_up": true,
      "shared": false,
      "id": "<networkId>",
      "availability_zone": "<availabilityZone>"
    },
    {
      "status": "ACTIVE", "subnets": [

```

```

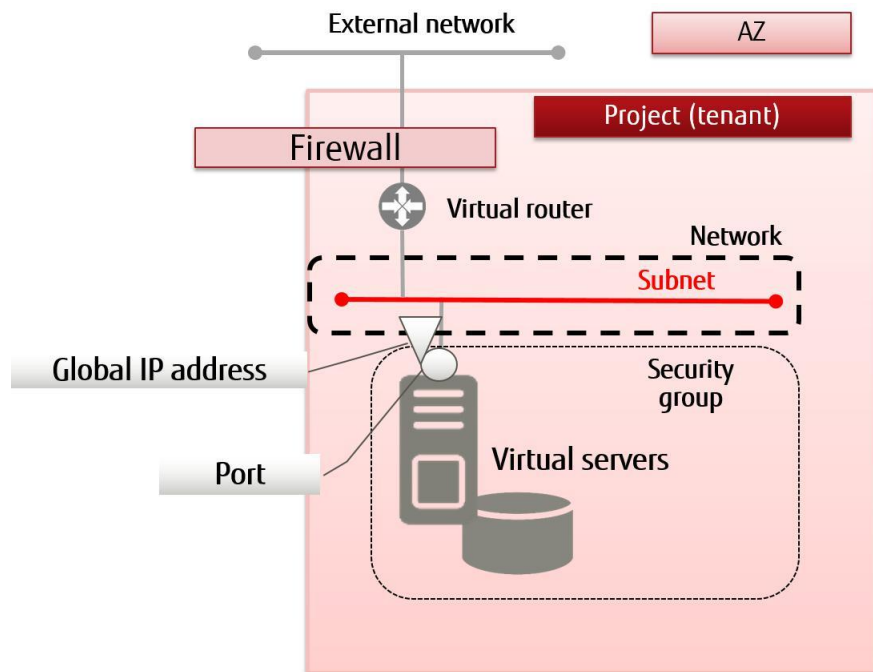
    "37ca5225-6ca6-4ee6-a49d-d479400a632b",
    "3c2419a7-7745-452c-bee5-664db03129bb",
    "7c7feecd-082c-415b-865a-82d4e5de97e5",
    "d96cdd6c-6e1e-4331-9c8e-42b52588b767",
    "ff4a9bd6-37be-4c03-9b87-1f693f807b48"
  ],
  "name": "inf_az1_ext-net01",
  "router:external": true,
  "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
  "admin_state_up": true,
  "shared": true,
  "id": "af4198a9-b392-493d-80ec-a7c6e5a1c22a",
  "availability_zone": "uk-1a"
}
]
}

```

In order to deploy resources such as virtual servers, a subnet will have to be created within the new network.

5.3 Creating a subnet

A subnet is required for managing private IP addresses, for resources to be connected to a network and for automatic setting of IP addresses by DHCP. This section explains the process to create and verify a subnet.



1. Set the environment variables as below:

 **Note:** In this example a DNS server is available on the internet. Verify the settings of the security group function or firewall service to allow communication with a DNS server (protocol: TCP/UDP, port number: 53).

```
$ SUBNET=<newSubnetName> (any)
$ NETWORK_ID=<networkId>
$ CIDR=<networkAddr> (any, using format XXX.XXX.XXX.0/24)
$ GATEWAY=<defaultGatewayIpAddr> (any, using format XXX.XXX.XXX.1)
$ DNS="\133.162.XXX.XXX\"\\, \"133.162.XXX.XXX\" (obtain from the administrator as
this will differ depending on the environment)
```

2. Execute the following API call:

```
$ curl -k -s $NETWORK/v2.0/subnets -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" -H
"Content-Type: application/json" -d '{"subnet": {"name": "'$SUBNET'",
"network_id": "'$NETWORK_ID'", "cidr": "'$CIDR'", "dns_nameservers": ['$DNS'],
"ip_version": 4, "gateway_ip": "'$GATEWAY_IP'", "availability_zone": "'$AZ'"}'
| jq .
```

Example successful response:

```
{
  "subnet": {
    "name": "<newSubnetName>",
    "enable_dhcp": true,
    "network_id": "<specifiedNetworkId>",
    "tenant_id": "<projId>",
    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "start": "XXX.XXX.XXX.2", "end": "XXX.XXX.XXX.254"
      }
    ],
    "host_routes": [], "ip_version": 4,
    "gateway_ip": "<specifiedDefaultGatewayIpAddr>",
    "cidr": "<specifiedNetworkAddr>",
    "id": "<newSubnetId>",
    "availability_zone": "<sameAvailabilityZoneAsSpecifiedNetwork>"
  }
}
```

Availability zones AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

3. Execute the following API call to verify the subnet was created.

```
$ curl -s $NETWORK/v2.0/subnets -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

An example of a successful response is shown below. If the new subnet is contained within the list of all subnets then it has been created successfully.

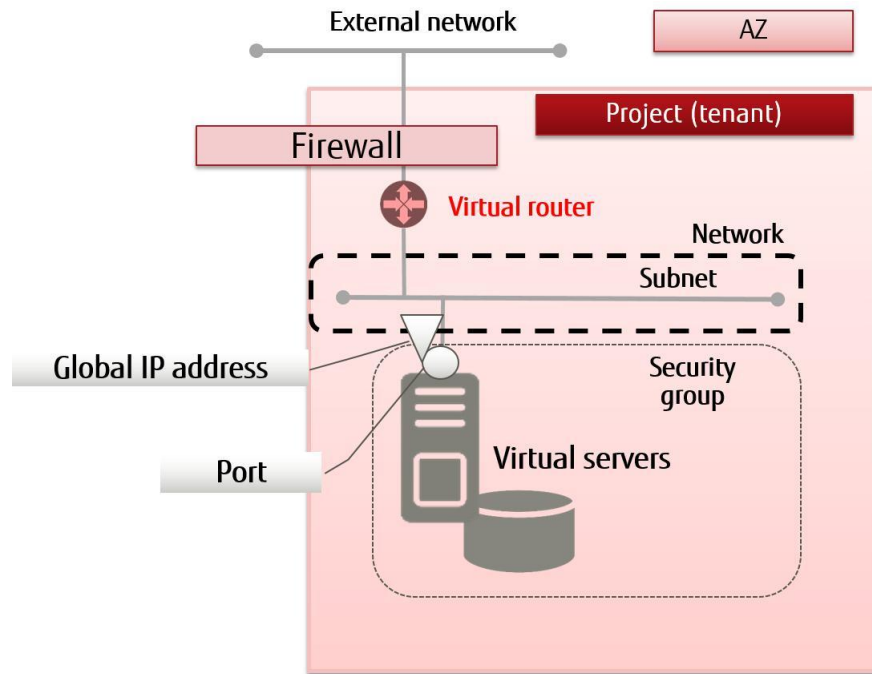
Subnet names that begin with "inf_az1_fip-pool", "inf_az2_fip-pool", "inf_az1_ext-subnet", and "inf_az1_ext-subnet", are associated with external networks that exist by default.

```
{
  "subnets": [
    ...

    {
      "name": "<subnetName>",
      "enable_dhcp": true,
      "network_id": "<networkId>",
      "tenant_id": "<projId>",
      "dns_nameservers": [
        "133.162.XXX.XXX",
        "133.162.XXX.XXX"
      ],
      "allocation_pools": [
        {
          "start": "XXX.XXX.XXX.2", "end": "XXX.XXX.XXX.254"
        }
      ],
      "host_routes": [],
      "ip_version": 4,
      "gateway_ip": "<defaultGatewayIpAddr>",
      "cidr": "<networkAddr>",
      "id": "<subnetId>",
      "availability_zone": "<sameAvailabilityZoneAsNetwork>"
    },
    ...
  ]
}
```

5.4 Creating a virtual router

This section explains the process to create and verify a virtual router which has to be connected to an external and an internal network.



1. Set the environment variables as below:

```
$ ROUTER_NAME=<newRouterName> (any)
$ AZ=<availabilityZoneName>
```

2. Execute the API call.

```
$ curl -k -s $NETWORK/v2.0/routers -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" -H
"Content-Type: application/json" -d '{"router": {"name": "'$ROUTER_NAME'",
"tenant_id": "'$TENANT_ID'", "availability_zone": "'$AZ'"}}' | jq .
```

Example successful response:

```
{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": null,
    "name": "<newVirtualRouterName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
```

```

    "id": "<newVirtualRouterId>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

3. Execute the following API call to verify the virtual router was created:

```
$ curl -s $NETWORK/v2.0/routers -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If the list of all routers contains the name of the new virtual router then it has been created successfully.

```

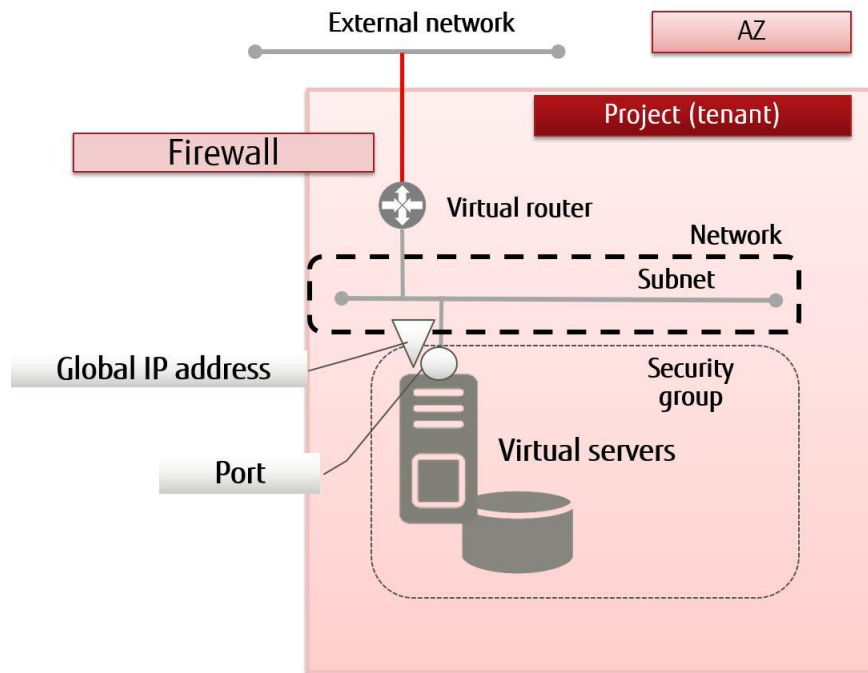
{
  "routers": [
    ...
    {
      "status": "ACTIVE",
      "external_gateway_info": null,
      "name": "<newVirtualRouterName>",
      "admin_state_up": true,
      "tenant_id": "<projId>",
      "id": "<newVirtualRouterId>",
      "availability_zone": "<specifiedAvailabilityZone>"
    },
    ...
  ]
}

```

Refer to Changing the configuration of a virtual router sections below for the procedure to attach a virtual router to a subnet or external network.

5.5 Changing the configuration of a virtual router (attaching to an external network)

This section explains the process to connect a virtual router to an external network.



1. Set environment variables as below:

```
$ ROUTER_ID=<targetVirtualRouterId>
$ EXT_NET_ID=<externalNetworkIdToBeConnected>
```

2. Execute the following API call:

```
$ curl -k -s $NETWORK/v2.0/routers/$ROUTER_ID -X PUT -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"router":
{"external_gateway_info": { "network_id": "'$EXT_NET_ID'"}}}' | jq .
```

Example successful response:

```
{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": {
      "network_id": "<specifiedExternalNetworkId>",
      "enable_snat": true
    },
    "name": "<specifiedVirtualRouterName>",
```

```

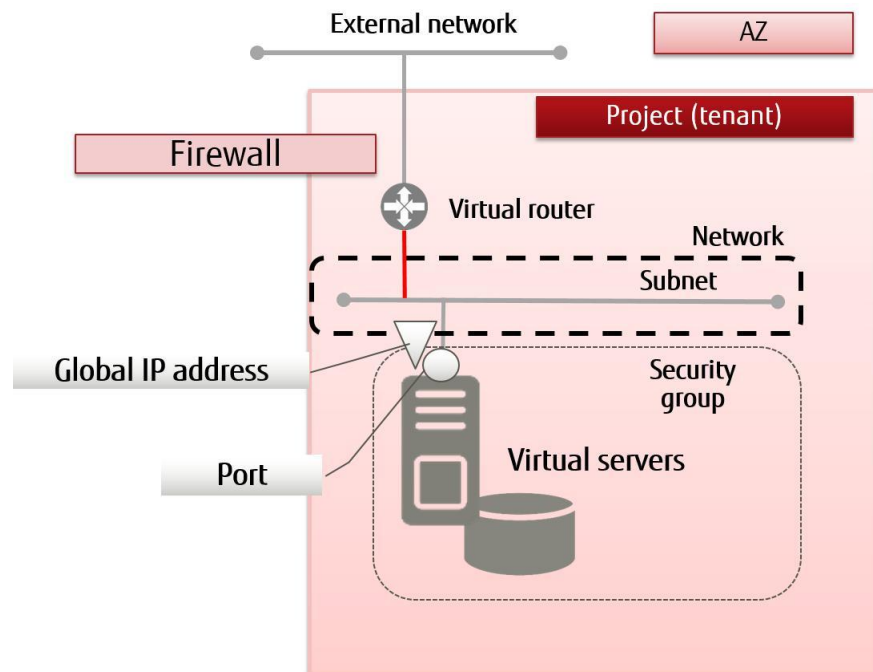
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "routes": [],
    "id": "<specifiedVirtualRouterId>",
    "availability_zone": "<availabilityZone>"
  }
}

```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

5.6 Changing the configuration of a virtual router (attaching to a subnet)

This section explains the process to connect a virtual router with an existing subnet of K5 setting information of existing virtual router.



1. Set the environment variables as below:

```

$ ROUTER_ID=<targetVirtualRouterId>
$ SUBNET_ID=<subnetIdToBeConnected>

```

2. Execute the following API call:

```
$ curl -k -s $NETWORK/v2.0/routers/$ROUTER_ID/add_router_interface -X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"subnet_id": "'$SUBNET_ID'" }' | jq .
```

Example successful response:

```
{
  "subnet_id": "<specifiedSubnetId>",
  "tenant_id": "<projId>",
  "port_id": "<portId>",
  "id": "<virtualRouterId>",
  "availability_zone": "<availabilityZone>"
}
```

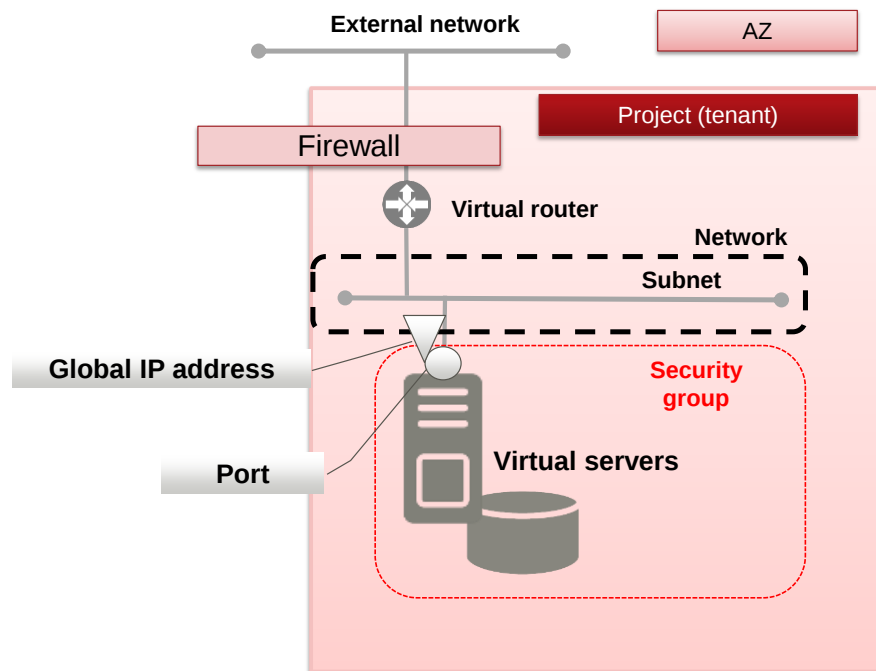
Availability zones such as AZ1 is presented by "uk-1a", and AZ2 is presented by "uk-1b".

5.7 Creating a security group

Security groups allow you to define and configure groups of rules that when applied perform packet filtering on virtual server ports.

A security group may have multiple rules. For a packet to pass through a requested port it must match one of the rules in the security group, otherwise it will be blocked. (whitelist method, OR condition)

Note: A security group cannot be set on a port of either a virtual router or DHCP server.



If no security group is specified for a virtual server then the default security group will be applied to the ports. The default security group will block all communication on a port. However, a new security group can be created and rules applied to allow the desired communication on a virtual server's ports. To create a security group:

1. Set the environment variable as below:

Security groups are not associated with availability zones, therefore, it is not necessary to specify one.

```
$ SG_NAME=<newSecGroupName> (any)
```

2. Execute the following API call:

```
$ curl -k -s $NETWORK/v2.0/security-groups -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"security_group":
{"name": "'$SG_NAME'"}' | jq .
```

Example successful response:

```
{
  "security_group": {
    "tenant_id": "<projId>",
    "description": "",
    "name": "<newSecGroupName>",
    "security_group_rules": [
      {
        "remote_group_id": null,
        "direction": "egress",
        "remote_ip_prefix": null,
        "protocol": null,
        "ethertype": "IPv6",
        "port_range_max": null,
        "security_group_id": "<newSecGroupId>",
        "port_range_min": null,
        "tenant_id": "<projId>",
        "id": "<defaultSecGroupRuleId>"
      },
      {
        "remote_group_id": null,
        "direction": "egress",
        "remote_ip_prefix": null,
        "protocol": null,
        "ethertype": "IPv4",
        "port_range_max": null,
        "security_group_id": "<newSecGroupId>",
        "port_range_min": null,
        "tenant_id": "<projId>",
        "id": "<defaultSecGroupRuleId>"
      }
    ],
    "id": "<newSecGroupId>"
  }
}
```

3. Execute following API call to verify the security group was created.

```
$ curl -s $NETWORK/v2.0/security-groups -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN"
| jq '.security_groups[] | .name, .description, .id'
```

If the list of security groups contains the new security group name then it the new security group has been created successfully.

```
"default"
"default"
"<defaultSecGroupIdOfProj>"
...
```

```
"<newSecGroupName>"

""
"<newSecGroupId>"

...
```

5.8 Creating a security group rule

This section explains the process to setup and verify a security group rule in a security group.

 **Tip:** Best practice is to use the security group ID for communication among virtual servers in the same security group.

1. Set the environment variables as below:

```
$ DIRECTION=<communicDirection> (specify ingress or egress)
$ PROTOCOL=<communicProtocol> (specify tcp, udp, icmp or 0-65535)
$ MIN_PORT_NUM=<minPortNum> (specify 0-65535)
$ MAX_PORT_NUM=<maxPortNum> (specify 0-65535)
$ REMOTE_GROUP_ID=<secGroupIdToRestrict> or
$ REMOTE_IP=<ipAddrToRestrict> (specify using a format such as XXX.XXX.XXX.0/24)
$ SG_ID=<secGroupIdToAddRuleTo>
```

2. Either execute the following API call to set packet filtering rules for a remote group of servers identified by the environment variable \$REMOTE_GROUP_ID

```
$ curl -k -s $NETWORK/v2.0/security-group-rules -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d
'{"security_group_rule":{"direction": "'$DIRECTION',"port_range_min":
'$MIN_PORT_NUM',"port_range_max": '$MAX_PORT_NUM',"protocol":
"'$PROTOCOL',"remote_group_id": "'$REMOTE_GROUP_ID',"security_group_id":
"'$SG_ID'"} }' | jq .
```

Or execute this API call to set rules for a specific remote server identified by \$REMOTE_IP

```
$ curl -k -s $NETWORK/v2.0/security-group-rules -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d
'{"security_group_rule":{"direction": "'$DIRECTION',"port_range_min":
'$MIN_PORT_NUM',"port_range_max": '$MAX_PORT_NUM',"protocol":
"'$PROTOCOL',"remote_ip_prefix": "'$REMOTE_IP'", "security_group_id":
"'$SG_ID'"} }' | jq .
```

Example successful response:

```
{
```

```

"security_group_rule": {
  "remote_group_id": "<secGroupIdToRestrict>",
  "direction": "<communicDirection>",
  "remote_ip_prefix": "<ipAddrToRestrict>",
  "protocol": "<protocol>",
  "tenant_id": "<projId>",
  "port_range_max": <maxPortNum>,
  "security_group_id": "<secGroupIdWithNewRule>",
  "port_range_min": <minPortNum>,
  "ethertype": "IPv4",
  "id": "<secGroupRuleId>",
  "availability_zone": null
}
}

```

3. Execute the following API call to verify the security group rule was set.

```

$ curl -s $NETWORK/v2.0/security-group-rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq .

```

If the list of rules contains the name of the new rule then it the security group rule has been created successfully.

```

{
  "security_group_rules": [

    ...

    {
      "remote_group_id": "<secGroupIdToRestrict>",
      "direction": "<commDirection>",
      "remote_ip_prefix": "<ipAddrToRestrict>",
      "protocol": "<protocol>",
      "tenant_id": "<projId>",
      "port_range_max": <maxPortNum>,
      "security_group_id": "<secGroupIdWithNewRule>",
      "port_range_min": <minPortNum>,
      "ethertype": "IPv4",
      "id": "<secGroupRuleId>",
    }

    ...

  ]
}

```

5.9 Examples of security group rules

This section provides examples of common security group rules.

Examples for the following settings:

- ingress: PING (ICMP: 0-255)
- ingress: SSH (TCP: 22)
- ingress: HTTP (TCP: 80)
- ingress: HTTPS (TCP: 443)
- ingress: NTP (UDP: 123)
- ingress: KMS (TCP: 1688)
- ingress: RDP (TCP: 3389)

The default values will be used for ingress: PING (ICMP: 0-255)

```
$ DIRECTION=ingress
$ PROTOCOL=icmp
$ MIN_PORT_NUM=0
$ MAX_PORT_NUM=255
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

ingress: SSH (TCP: 22)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=22
$ MAX_PORT_NUM=22
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

ingress: HTTP (TCP: 80)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=80
$ MAX_PORT_NUM=80
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

ingress: HTTPS (TCP: 443)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=443
$ MAX_PORT_NUM=443
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
```

```
$ SG_ID=<secGroupIdToBeSet>
```

ingress: NTP (UDP: 123)

```
$ DIRECTION=ingress
$ PROTOCOL=udp
$ MIN_PORT_NUM=123
$ MAX_PORT_NUM=123
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

ingress: KMS (TCP: 1688)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=1688
$ MAX_PORT_NUM=1688
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

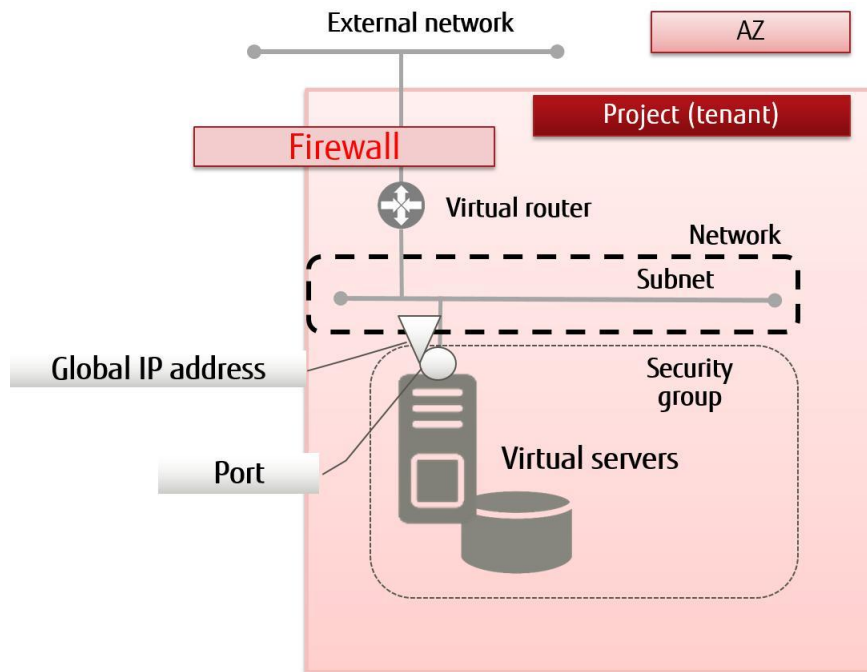
ingress: KMS (TCP: 3389)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=3389
$ MAX_PORT_NUM=3389
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24) or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>
```

Refer to the "IaaS Function Manual" to learn more about security rules.

5.10 Creating a firewall rule (IP address specification)

This section explains the process to create and verify firewall rules (IP address specification) in order to use firewall services for setting the packet filters in a virtual router.



Tip: The firewall service settings consist of the following elements and have filtering information which is configured in numerical order. The specified filtering is performed by linking a firewall with virtual router.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

Configure the following settings to create firewall rules.

- Remote IP address – takes the form "XXX.XXX.XXX.0/24" (address is environment specific, alternatively use 0.0.0.0/0 to specify all IP addresses)
- Private IP address – takes the form "XXX.XXX.XXX.0/24" (address is environment specific, i.e. the IP that was specified when creating the subnet)

Common rule examples:

- Allow: Remote IP address: TCP
- Allow: Remote IP address: UDP
- Allow: Remote IP address: ICMP
- Allow: Private IP address: TCP
- Allow: Private IP address: UDP
- Allow: Private IP address: ICMP

This guide has detailed the most commonly used rules for creating a firewall; other rules exist. Refer to the IaaS Function Manual to learn more about firewall settings and setup methods.

1. Set the environment variables as below:

```
$ FWR_NAME=<firewallRuleName> (any)
$ AC=<accessControl> (specify allow or deny)
$ SOURCE_IP=<ipAddrToRestrict> (specify using a format such as XXX.XXX.XXX.0/24)
$ PROTOCOL=<protocol> (specify tcp, udp, icmp, any)
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"firewall_rule":
{ "name": "'$FWR_NAME'", "action": "'$AC'", "source_ip_address": "'$SOURCE_IP'",
"protocol": "'$PROTOCOL'", "availability_zone": "'$AVAILABILITY_ZONE'" } }' | jq .
```

Example successful response:

```
{
  "firewall_rule": { "protocol": "<protocol>",
    "description": "", "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
    "source_ip_address": "<specifiedIpAddr>",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

3. Execute the following API call to verify the firewall rules were created:

```
$ curl -s $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'
```

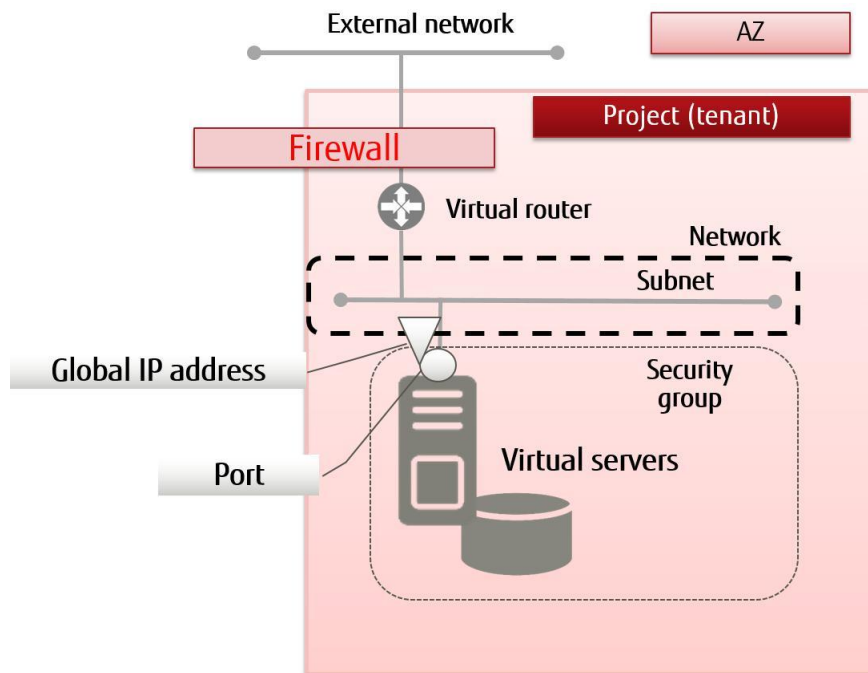
If the list contains the specified firewall rules then the firewall rules have been created successfully.

```
...
"<newFirewallRuleName>"
"<newFirewallRuleId>"
...
```

Proceed to create a firewall policy with all the required firewall rules after creating the individual rules for the firewall.

5.11 Creating a firewall rule (port number specification)

This section explains the process to create and verify a firewall rule for a specific port number in order to use the firewall service for setting packet filters in virtual routers.



Tip: The firewall service settings consist of the following elements and have filtering information which is configured in numerical order. The specified filtering is performed by linking a firewall with virtual router.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

Common rule examples.

- Allow: SSH (TCP: 22)
- Allow: HTTP (TCP: 80)
- Allow: HTTPS (TCP: 443)
- Allow: NTP (UDP: 123)
- Allow: KMS (TCP: 1688)
- Allow: RDP (TCP: 3389)

This guide has detailed the most commonly used rules for creating a firewall; other rules exist. Refer to the IaaS Function Manual to learn more about firewall settings and setup methods.

1. Set the environment variables:

```
$ FWR_NAME=<firewallRuleName> (any)
$ AC=<accessControl> (specify allow or deny)
$ DESTINATION_PORT=<destinationPortNumOrRange> (for example, "0 - 255", 22, etc.)
$ PROTOCOL=<protocol> (specify tcp, udp, icmp, any)
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"firewall_rule": { "name": "'$FWR_NAME'", "action": "'$AC'", "destination_port": "'$DESTINATION_PORT'", "protocol": "'$PROTOCOL'", "availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

Example successful response:

```
{
  "firewall_rule": {
    "protocol": "<protocol>",
    "description": "",
    "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
```

```

    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": "<specifiedPortNum>",
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk-1b".

3. Execute the following API call to verify the firewall rules were created:

```

$ curl -s $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'

```

If the list of rules contains the specified firewall rule names then the firewall rules have been created successfully.

```

...

"<newFirewallRuleName>"
"<newFirewallRuleId>"

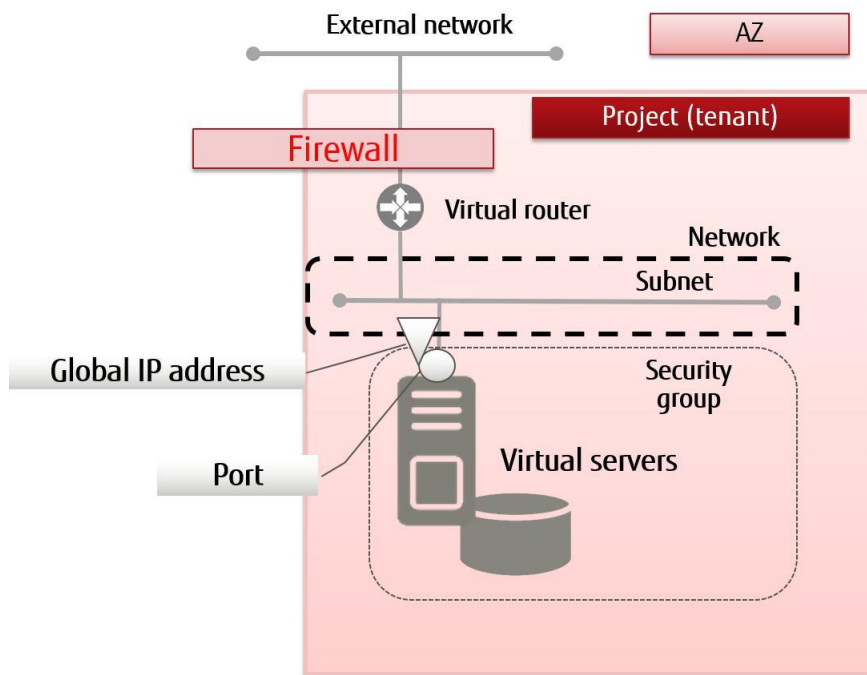
...

```

Proceed to create a firewall policy containing all the required firewall rules.

5.12 Creating a firewall rule (ICMP permission)

This section explains the process to create and verify a firewall rule for ICMP permission in order to use the firewall service for setting packet filters in virtual routers.



Tip: The firewall service settings consist of the following elements and have filtering information which is configured in numerical order. The specified filtering is performed by linking a firewall with virtual router.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

The API call to create the following rule is detailed below

- Allow: PING (ICMP: 0 to 255)

This guide has detailed the most commonly used rules for creating a firewall; other rules exist. Refer to the IaaS Function Manual to learn more about firewall settings and setup methods.

1. Set the environment variables:

```
$ FWR_NAME=<firewallRuleName> (any)
$ AC=<accessControl> (specify allow or deny)
$ PROTOCOL=icmp
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"firewall_rule":
{ "name": "'$FWR_NAME'", "action": "'$AC'", "protocol": "'$PROTOCOL'",
"availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

Example successful response:

```
{
  "firewall_rule": {
    "protocol": "icmp",
    "description": "",
    "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk-1a" and AZ2 is presented by "uk -1b".

3. Execute the following API call to verify the firewall rules were created:

```
$ curl -s $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'
```

If the list of rules contains the specified firewall rule names then the firewall rules have been created successfully.

```
...

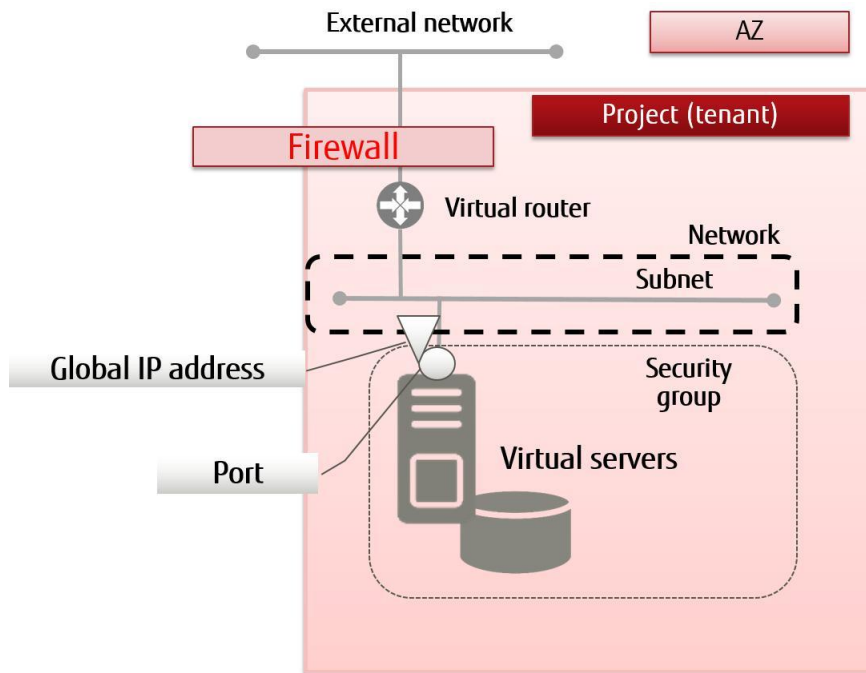
"<newFirewallRuleName>"
"<newFirewallRuleId>"
```

...

Proceed to create a firewall policy with all the required firewall rules after creating the rules for firewall.

5.13 Creating a firewall rule (deny rule)

This section explains the process to create and verify a firewall deny rule.



Tip: The firewall service settings consist of the following elements and have filtering information which is configured in numerical order. The specified filtering is performed by linking a firewall with virtual router.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

The API call to create the following rule is detailed below

- Close all the ports that have not been set

This guide has detailed the most commonly used rules for creating a firewall; other rules exist. Refer to the IaaS Function Manual to learn more about firewall settings and setup methods.

1. Set the environment variables as below:

```
$ FWR_NAME=<firewallRuleName> (any)
$ AC=deny
$ PROTOCOL=any
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"firewall_rule":
{ "name": "'$FWR_NAME'", "action": "'$AC'", "protocol": "'$PROTOCOL'",
"availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

Example successful response:

```
{
  "firewall_rule": {
    "protocol": null,
    "description": "",
    "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "deny",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk -1a" and AZ2 is presented by "uk -1b".

3. Execute the following API call to verify the firewall rules were created:

```
$ curl -s $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'
```

If the list of rules contains the specified firewall rule names then the firewall rules have been created successfully.

```
...

"<newFirewallRuleName>"
```

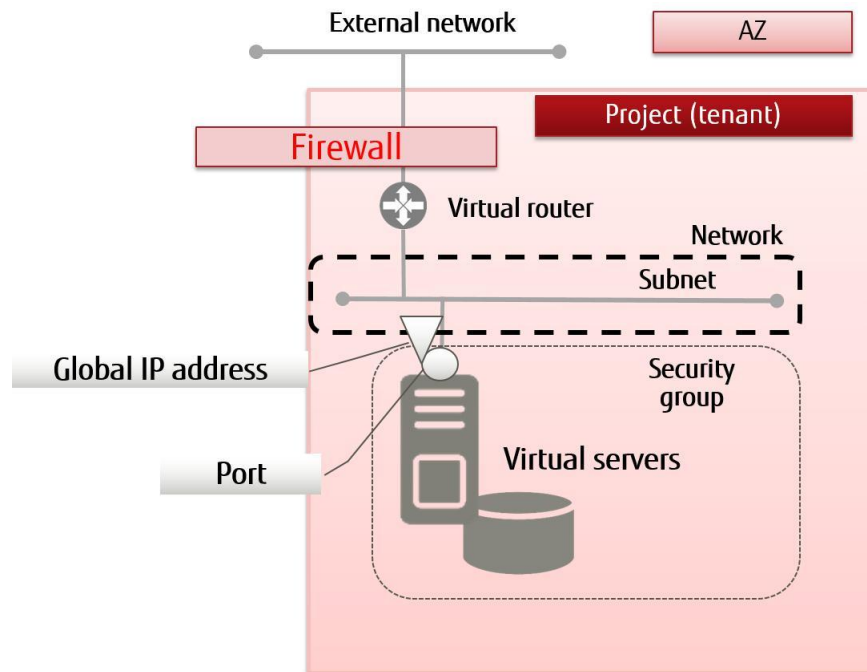
```
"<newFirewallRuleId>"
```

...

Proceed to create firewall a policy with all the required firewall rules.

5.14 Creating a firewall policy

This section explains the process to create and verify a firewall policy to group a number of firewall rules.



There is priority order set for the rules in accordance with their sequential number in the list. These rules are verified in an order from top to bottom and controls the permission of communication.

Tip: one rule of "DENY ALL" is automatically added to the end of the policy. This ensures that all the traffic without a defined permission rule will be intercepted by default. (White list method), however, "DENY ALL" rule which is automatically added is an implicit rule and does not appear in the policy.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

1. Execute the following API call to retrieve the list of firewall rules:

```
$ curl -s $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'
```

List of all firewall rules. Select the rules which are required for the policy.

```
...

"<firewallRuleName>"
"<firewallRuleId>"

...
```

2. Set the environment variables as below:

```
$ NAME=<firewallPolicyName> (any)
$ FWR1=<firewallRuleId1>
$ FWR2=<firewallRuleId2>
$ FWR3=<firewallRuleId3>
```

Note: Specify as many firewall rule IDs as required for the firewall policy.

```
$ AVAILABILITY_ZONE=<availabilityZoneSetInFirewallRules>
```

3. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewall_policies -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d
'{"firewall_policy":{"firewall_rules": [ "'$FWR1'", "'$FWR2'", "'$FWR3',
<remainingSpecifiedRuleIdEnvVars>"], "name": "'$NAME'", "availability_zone":
"'$AVAILABILITY_ZONE'"} }' | jq .
```

Example successful response. The sequential order in which firewall rule IDs are displayed is their priority order.

```
{
  "firewall_policy": {
    "name": "<newFirewallPolicyName>",
    "firewall_rules": [
      "<specifiedFirewallRuleId>(1)",
      "<specifiedFirewallRuleId>(2)",
      ...
      "<specifiedFirewallRuleId>(n)"
    ],
    "tenant_id": "<projId>",
    "audited": false,
    "shared": false,
    "id": "<newFirewallPolicyId>",
```

```

    "description": "",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

Availability zones such as AZ1 is presented by "uk -1a" and AZ2 is presented by "uk -1b".

4. Execute the following API call to verify the firewall policy was created:

```

$ curl -s $NETWORK/v2.0/fw/firewall_policies -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq .

```

If the list of firewall policies and rules contains the new policy then the firewall policy has been created successfully.

```

{
  "firewall_policies": [

    ...

    {
      "name": "<newFirewallPolicyName>",
      "firewall_rules": [
        "<specifiedFirewallRuleId1>",
        "<specifiedFirewallRuleId2>",
        ...
        "<specifiedFirewallRuleIdN>"
      ],
      "tenant_id": "<projId>",
      "audited": false,
      "shared": false,
      "id": "<newFirewallPolicyId>",
      "description": "",
      "availability_zone": "<specifiedAvailabilityZone>" },

    ...

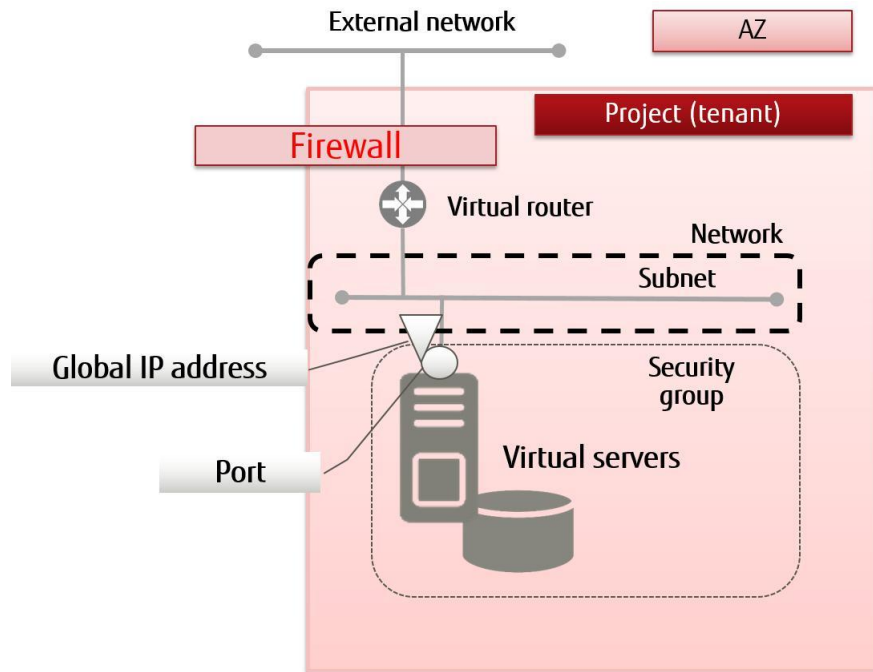
  ]
}

```

Proceed to Creating a firewall which will illustrate linking a virtual router with a firewall policy.

5.15 Creating a firewall

This section explains the process to create and verify a firewall in virtual router.



Tip: The firewall service settings consist of the following elements and have filtering information which is configured in numerical order. The specified filtering is performed by linking a firewall with virtual router.

By default all traffic is intercepted. Only the traffic defined by permissive rules (AC=allow) can pass through the firewall. There are three steps that must be followed in order to configure a firewall on a virtual router:

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall and link it with the virtual router

1. Set the environment variables as below:

```
$ FW_NAME=<newFirewallName> (any)
$ FWP_ID=<firewallPolicyIdToBeSet>
$ ROUTER_ID=<virtualRouterIdToBeApplied>
$ AVAILABILITY_ZONE=<sameAvailabilityZoneAsSetInTheFirewallPolicy>
```

2. Execute the following API call:

```
$ curl -X POST -k -s $NETWORK/v2.0/fw/firewalls -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"firewall":{"name":
```

```
"'$FW_NAME',"firewall_policy_id": "'$FWP_ID'", "router_id": "'$ROUTER_ID'",
"availability_zone": "'$AVAILABILITY_ZONE'"} }' | jq .
```

Example successful response:

```
{
  "firewall": {
    "status": "PENDING_CREATE",
    "router_id": "<specifiedVirtualRouterId>",
    "name": "<newFirewallName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "firewall_policy_id": "<specifiedFirewallPolicyId>",
    "id": "<newFirewallId>",
    "description": "",
    "availability_zone": "<availabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk -1a" and AZ2 is presented by "uk -1b".

3. Execute the following API call to verify the firewall was created:

```
$ curl -s $NETWORK/v2.0/fw/firewalls -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" |
jq .
```

If the list of firewalls contains the specified firewall policy name then the firewall has been created successfully.

```
{
  "firewalls": [

    ...

    {
      "status": "PENDING_CREATE",
      "router_id": "<specifiedVirtualRouterId>",
      "name": "<newFirewallName>",
      "admin_state_up": true,
      "tenant_id": "<projId>",
      "firewall_policy_id": "<specifiedFirewallPolicyId>",
      "id": "<newFirewallId>",
      "description": "",
      "availability_zone": "<availabilityZone>" },

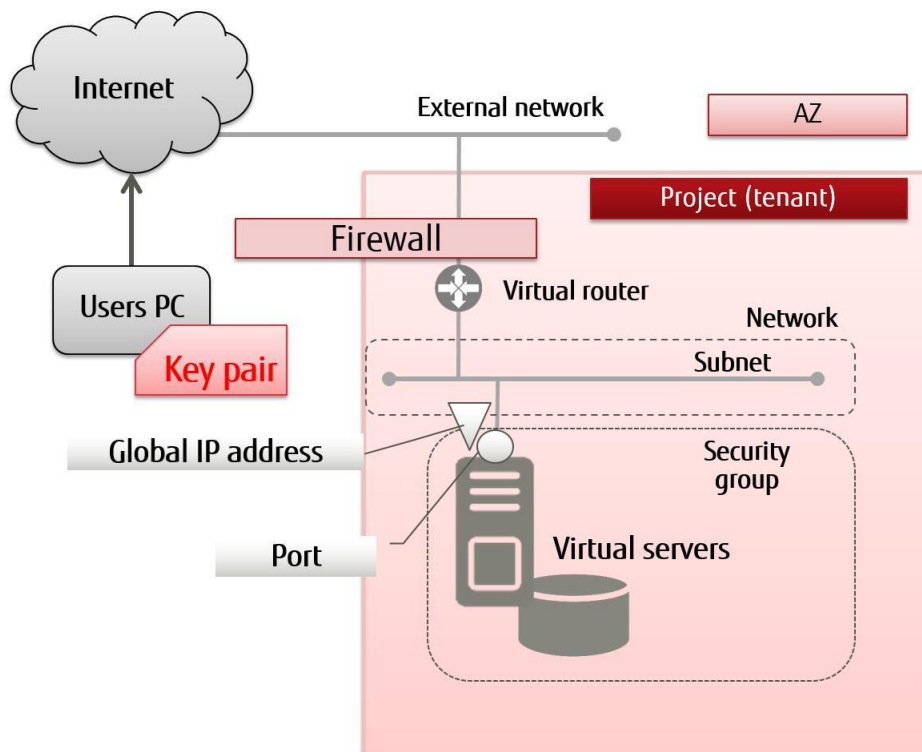
    ...

  ]
}
```

Warning: If a firewall policy or any rules are set which are not owned by the project then the firewall will not work properly. In this case, the user will not be able to delete the firewall and it will be necessary for the K5 administrator to perform such deletions. Create the firewall and its policies/rules so that they have the same project ID.

5.16 Creating a key pair

This section explains the process to create and verify a key pair to be used while logging in to a virtual server using SSH.



A key file (*.pem) which is used to authenticate SSH can be retrieved by registering a key pair. One can login to the virtual server by using this key pair.

Important: The key file should be managed with care. The .pem file contains the private key. The private key can be used to decrypt all encrypted traffic.

1. Set the environment variable as below:

```
$ KEYPAIR=<keyPairName> (any)
```

2. Execute the following API call:

```
$ curl -k -s $COMPUTE/v2/$PROJECT_ID/os-keypairs -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json" -d '{"keypair": {"name":
"$KEYPAIR"}}' | jq -r .keypair.private_key > $KEYPAIR.pem
```

3. Execute following commands to enable the key file was created:

```
$ chmod 600 $KEYPAIR.pem
```

4. Execute following commands to verify the content of key file:

```
cat $KEYPAIR.pem
```

Example successful response:

```
-----BEGIN RSA PRIVATE KEY-----

~~omitted~~

-----END RSA PRIVATE KEY-----
```

5. Execute following API to verify the Key Pair was created:

```
$ curl -X GET -s $COMPUTE/v2/$PROJECT_ID/os-keypairs -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

If the list of key pairs contains the specified key pair name then that key pair has been created successfully.

```
{
  "keypairs": [

    ...

    {
      "keypair": {
        "public_key": "ssh-rsa
        ~~content omitted~~

        == Generated by Nova\n", "name": "<keyPairName>",
        "fingerprint": "64:96:18:f9:f2:96:e9:7d:f2:b3:dd:ee:bc:45:eb:a0"
      }
    },

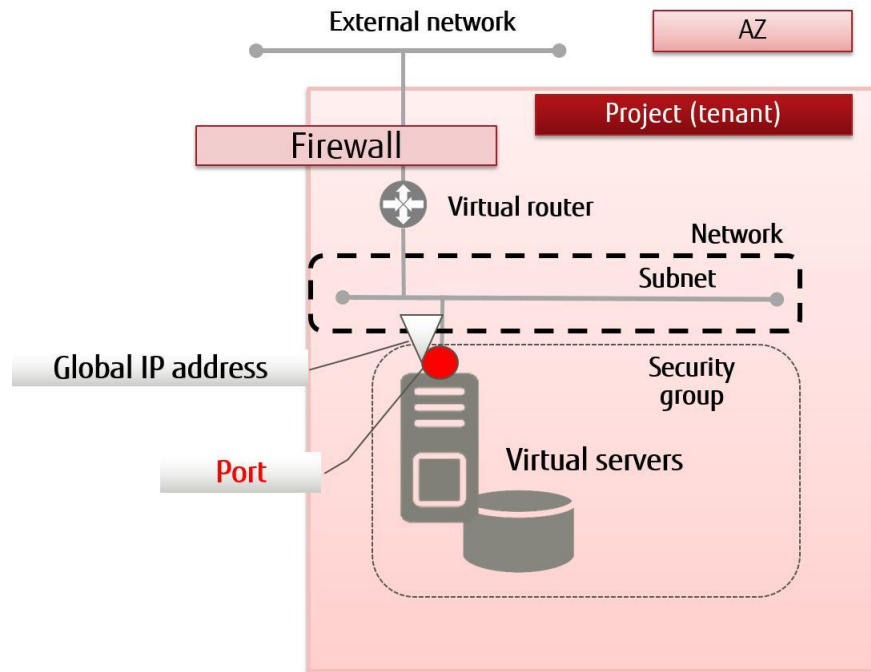
    ...

  ]
}
```

 **Tip:** It is also possible to import externally created key pairs.

5.17 Creating a port

This section explains the process to create and verify the port (network interface) to be linked with an IP address for connecting a resource such as virtual server over a network



If the subnet itself is specified while creating a resource as below, then the system will automatically create a port and assign an IP address to it.

- Virtual server

 **Note:** When an IP address for a virtual server is assigned manually instead of being assigned automatically by DHCP then create a port associated with that IP address in advance and assign it to the virtual server.

 **Tip:** Multiple ports can be added to a virtual server.

- Virtual router

 **Note:** A port is assigned automatically only when it is created for a default gateway (x.x.x.1). When a virtual router has to be connected to network which already has a virtual router which is connected with an IP address of x.x.x.1 then it is necessary to configure the port manually.

1. Set the environment variables as below:

```
$ PORT_NAME=<portName>(any)
$ NETWORK_ID=<networkId>
$ SUBNET_ID=<subnetId>
$ FIXED_IP_ADDRESS=<ipAddrToBeSpecified> (specify using format "XXX.XXX.XXX.XXX"
within a range specified during creation of subnet)
```

2. Execute the following API call:

```
$ curl -s $NETWORK/v2.0/ports -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" -H
"Content-Type: application/json" -d '{"port":{"network_id": "'$NETWORK_ID'",
"name": "'$PORT_NAME'", "availability_zone":"'$AZ'", "fixed_ips": [{"subnet_id":
"'$SUBNET_ID'", "ip_address": "'$FIXED_IP_ADDRESS'"}] }}' | jq .
```

Example successful response:

```
{
  "port": {
    "status": "<status>",
    "name": "<portName>",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "<networkId>",
    "tenant_id": "<projId>",
    "binding:vnic_type": "normal",
    "device_owner": "",
    "mac_address": "<macAddr>",
    "fixed_ips": [
      {
        "subnet_id": "<subnetId>",
        "ip_address": "<ipAddrToBeSpecified>XXX.XXX.XXX.XXX"
      }
    ],
    "id": "<portId>",
    "security_groups": [
      null
    ],
    "device_id": "",
    "availability_zone": "<availabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk -1a" and AZ2 is presented by "uk -1b".

3. Execute the following API call to verify the port was created:

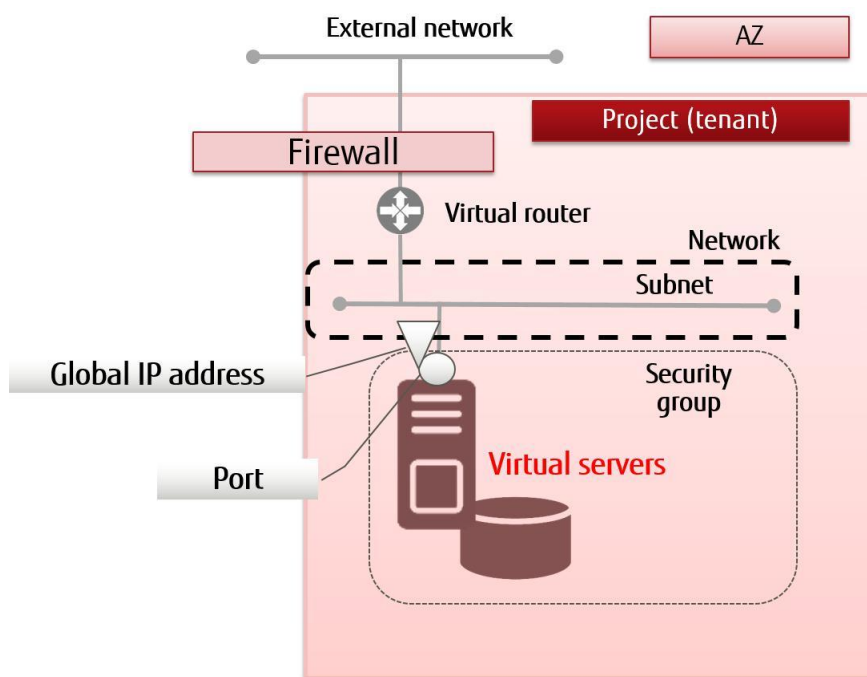
```
$ curl -s $NETWORK/v2.0/ports -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq
'.ports[] | .name, .status, .id, .fixed_ips[]'
```

If the list of ports contains the specified port names then the ports have been created successfully.

```
...  
"  
"<status>  
"<portId>  
{  
  "subnet_id": "<subnetId>",&br/>  "ip_address": "<privateIpAddr>"  
}  
...
```

5.18 Preparing to create a virtual server (listing virtual server images)

This section explains the process to obtain a list of available OS images.



Select the OS for a virtual server from the list given below.

Table 1: OS environments to be provided (as of December 1, 2015)

OS type	OS to be provided
Windows	• Windows Server 2008 SE R2 SP1 64-bit (English) • Windows Server 2012 SE R2 64-bit (English)
Linux	• CentOS 6.5 64-bit (English) • Red Hat Enterprise Linux 6.5 64-bit (English) • Ubuntu Server 14.04 LTS (English)



Note: Customers can upload their own Images to the platform providing they are one of the support Operating Systems for K5 IaaS.

1. Execute the following API call to display the list:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/images/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.images[] | .name, .id, .status'
```

The response is structured in the format shown below:

```
"<imgName>"
"<imgId>"
"<imgStatus>"
```

Example of a successful response:

```
"Ubuntu Server 14.04 LTS (English) 01" "ffa17298-537d-40b2-a848-0a4d22b49df5"
"ACTIVE"
"FJK5-TemplateBuilder-V02" "14117885-4104-45a1-9bcd-6dc572e9ee5f" "ACTIVE"
"Windows Server 2012 SE 64bit (Japanese) 01" "30718484-1002-43bf-9cf5-
f1777c6ed4cb" "ACTIVE"
"Windows Server 2012 R2 SE 64bit (Japanese) 03" "5ab16551-c229-4611-834b-
a16e074c187e"
"ACTIVE"
"Red Hat Enterprise Linux 6.5 64bit (English) 02" "db9766f0-c95c-4f1c-bb29-
304a90405e3e"
"ACTIVE"
"Windows Server 2008 R2 EE 64bit SP1 (Japanese) 02" "dcbd4261-e5d1-4d49-9e4f-
7434a14dbf4e"
"ACTIVE"
"CentOS 6.5 64bit (English) 03" "839c1db6-738c-4e2b-9a1d-c14977564203" "ACTIVE"
"Windows Server 2008 R2 SE 64bit SP1 (Japanese) 02" "0e2bd896-0ede-4e00-bf71-
248ef92c2202"
"ACTIVE"
```

Specify the image ID while creating a virtual server.

5.19 Preparing to create a virtual server (listing virtual server types)

This section explains the process to obtain the list of virtual server types (flavours) which are available.

Table 2: Types of virtual server (flavours) available

Type	Virtual CPU count	Memory (GB)
S-1	1	4
S-2	2	8
S-4	4	16
S-8	8	32
S-16	16	64
M-1	1	8
M-2	2	16
M-4	4	32
M-8	8	64
M-16	16	128
XM-4	4	128

1. Execute following API call to display the list:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/flavors/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.flavors[] | {name: .name, id: .id}'
```

A successful response will be structured as shown below:

```
{
  "name": "<flavorName>",
  "id": "<flavorId>"
}
```

Example of a successful response:

```
{
  "name": "S-1",
  "id": "1101"
}
{
  "name": "S-2",
  "id": "1102"
}
{
  "name": "S-4",
```

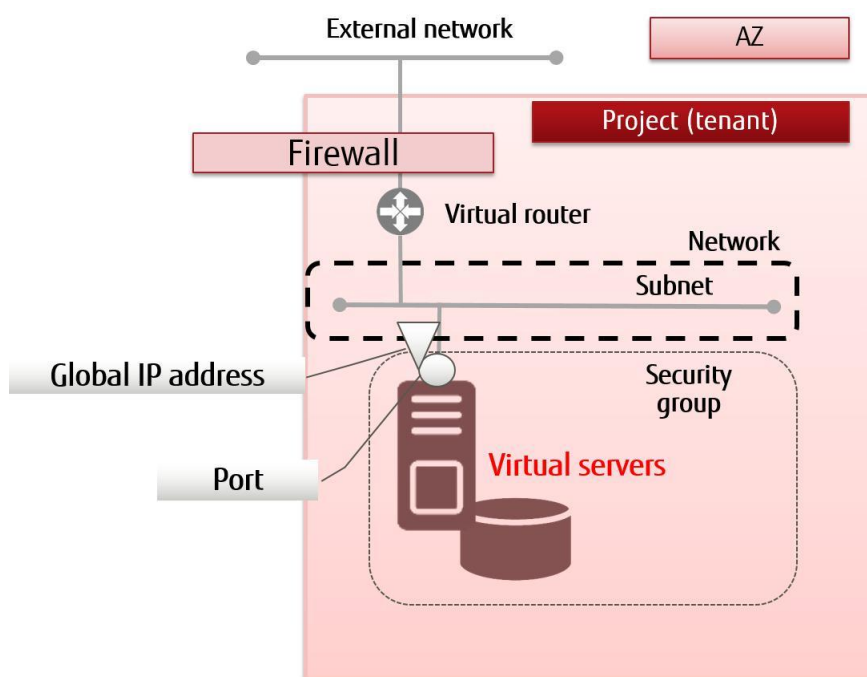
```
"id": "1103"
}
{
  "name": "S-8",
  "id": "1104"
}
{
  "name": "S-16",
  "id": "1105"
}
{
  "name": "M-1",
  "id": "1201"
}
{
  "name": "M-2",
  "id": "1202"
}
{
  "name": "M-4",
  "id": "1203"
}
{
  "name": "M-8",
  "id": "1204"
}
{
  "name": "M-16",
  "id": "1205"
}
```

Specify the flavour ID while creating a virtual server.

5.20 Creating a virtual server (CentOS / port specification)

This section explains the process to create and verify a (CentOS) virtual server by specifying the port (private IP address) using DHCP.

For an alternative to this method see "Creating a virtual server (CentOS / DHCP retrieval)".



1. Set the environment variables as below:

```
$ VM_NAME=<virtualServerName> (any)
$ FLAVOUR_ID=<specOfVirtualServer> (listed flavor ID)
$ VOL_SIZE=30
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
$ IMAGE_REF_ID=<imgIdOfCentos>
$ SOURCE=<blockStorageType>
$ DESTINATION=<connectionDestinationVol>
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of
the virtual server, or 0 otherwise)
$ KEYNAME=<keyPairName>
$ INSTANCE_MAX=<maxNumOfServers> $ INSTANCE_MIN=<minNumOfServers>
$ PORT_ID=<portIdToBeSpecified>
```

2. Execute the following API call:

```
curl -k -s $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"server": {"name":
"$VM_NAME", "imageRef": "", "flavorRef": "$FLAVOUR_ID",
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "$IMAGE_REF_ID",
"volume_size": "$VOL_SIZE", "device_name": "$DEVICE_NAME", "source_type":
"$SOURCE", "destination_type": "$DESTINATION", "delete_on_termination":
"$ISDELETE"} ] , "key_name": "$KEYNAME", "max_count": "$INSTANCE_MAX",
"min_count": "$INSTANCE_MIN", "networks": [{"port": "$PORT_ID"}] }}' | jq .
```

Example successful response:

```
{
  "server": { "security_groups": [
    {
      "name": "default"
    }
  ],
  "OS-DCF:diskConfig": "MANUAL",
  "id": "<newVirtualServerId>",
  "links": [
    {
      "href": "http://10.3.0.201/v2/<projId>/servers/ <newVirtualServerId>",
      "rel": "self"
    },
    {
      "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
      "rel": "bookmark"
    }
  ]
}
```

3. Execute the following API call to verify the virtual server was created:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.servers[] | {status: .status, network_name:.addresses |
keys, key_name: .key_name, flavor_id: .flavor | .id, vm_id:.id,
security_group: .security_groups[] | .name, name: .name, }'
```

If the list of virtual servers contains the specified virtual server name then the virtual server has been created successfully.

```
...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ]
}
```

```

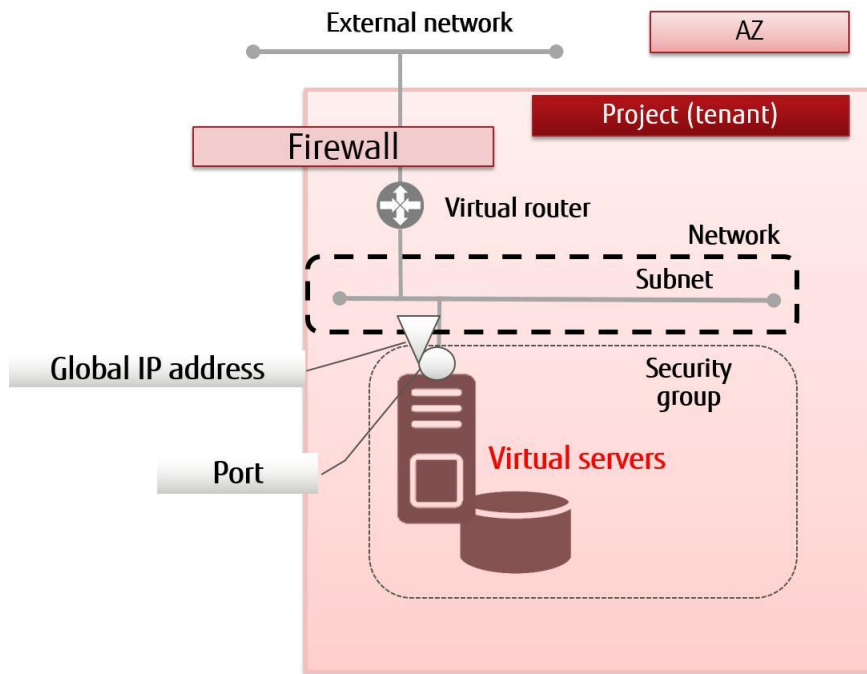
],
"key_name": "<keyPairName>",
"flavor_id": "<flavorId>",
"vm_id": "<virtualServerId>",
"security_group": "<secGroupName>",
"name": "<virtualServerName>"
}
...

```

5.21 Creating a virtual server (CentOS / DHCP retrieval)

This section explains the process to create and verify a (CentOS) virtual server using DHCP to retrieve the port (private IP address).

For an alternative to this method see "Creating a virtual server (CentOS / port specification)".



1. Set the environment variables as below:

```

$ VM_NAME=<virtualServerName> (any)
$ FLAVOUR_ID=<specOfVirtualServer> (listed flavor ID) $ VOL_SIZE=30
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
$ SOURCE=<blockStorageType>
$ IMAGE_REF_ID=839c1db6-738c-4e2b-9a1d-c14977564203<imgIdOfCentos>
$ DESTINATION=<connectionDestinationVol> (specify volume)

```

```
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of
the virtual server, or 0 otherwise)
$ KEYNAME=<keyPairName>
$ INSTANCE_MAX=<maxNumOfServers>
$ INSTANCE_MIN=<minNumOfServers>
$ NETWORK_ID=<networkIdThatYouWantToConnectTo>
$ SG_NAME=<secGroupNameToBeSpecified>
```

2. Execute the following API call:

```
curl -k -s $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"server": {"name":
"$VM_NAME", "availability_zone": "$AZ", "imageRef": "", "flavorRef":
"$FLAVOR_ID", "block_device_mapping_v2": [ {"boot_index": "0",
"uuid": "$IMAGE_REF_ID", "volume_size": "$VOL_SIZE", "device_name":
"$DEVICE_NAME", "source_type": "$SOURCE", "destination_type":
"$DESTINATION", "delete_on_termination": "$ISDELETE"} ] , "key_name":
"$KEYNAME", "max_count": "$INSTANCE_MAX", "min_count": "$INSTANCE_MIN",
"networks": [{"uuid": "$NETWORK_ID"}], "security_groups": [{"name":
"$SG_NAME"}]} | jq .
```

Example successful response:

```
{
  "server": { "security_groups": [
    {
      "name": "<specifiedSecGroupName>"
    }
  ],
  "OS-DCF:diskConfig": "MANUAL",
  "id": "<newVirtualServerId>",
  "links": [
    {
      "href": "http://10.3.0.201/v2/<projId>/servers/ <newVirtualServerId>",
      "rel": "self"
    },
    {
      "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
      "rel": "bookmark"
    }
  ]
}
```

3. Execute the following API call to verify the virtual server was created:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.servers[] | {status: .status, network_name:.addresses |
keys, key_name: .key_name, flavor_id: .flavor | .id, vm_id:.id,
security_group: .security_groups[] | .name, name: .name, }'
```

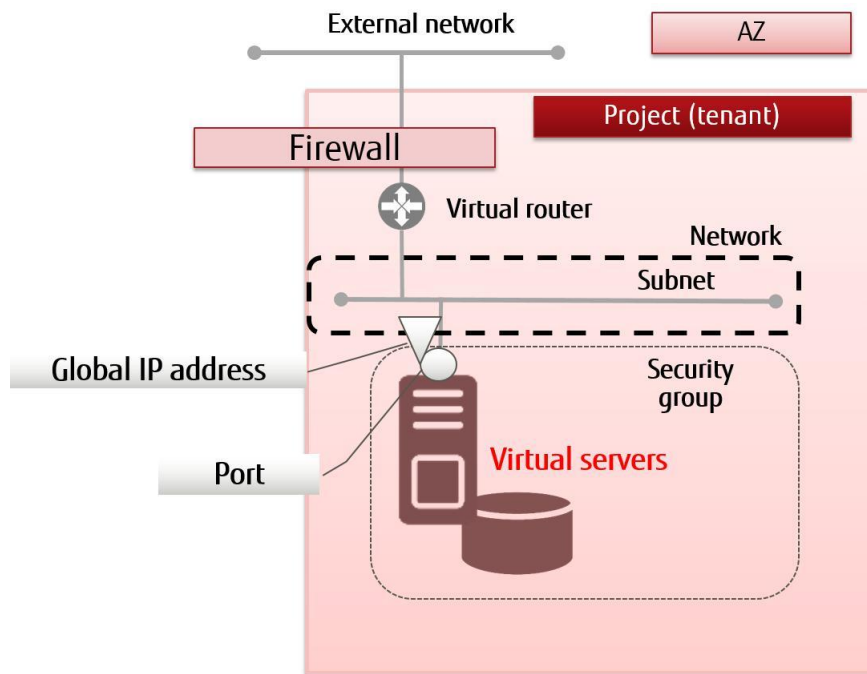
If the list of virtual server names contains the new virtual server then the virtual server has been created successfully

```
...  
  
{  
  "status": "<statusOfVirtualServer>",  
  "network_name": [  
    "<connectedNetworkName>"  
  ],  
  "key_name": "<keyPairName>",  
  "flavor_id": "<flavorId>",  
  "vm_id": "<virtualServerId>",  
  "security_group": "<specifiedSecGroupName>",  
  "name": "<virtualServerName>"  
}  
  
...
```

5.22 Creating a virtual server (Windows / port specification)

This section explains the process to create and verify a (Windows) virtual server by specifying the port (private IP address) using DHCP.

For an alternative to this method see "Creating a virtual server (Windows / DHCP retrieval)".



Create a virtual server and obtain its password. A password is required for logging into the virtual server's remote desktop:

- Create a virtual server specifying a key pair for encrypting and decrypting a password
- Retrieve the encrypted password
- Decrypt the password using the same key pair

⚠ DANGER: Other users may be able to access the password when using a Windows virtual server which is deployed with `admin_pass` specified in the metadata therefore it is recommended to change the password after deployment.

1. Create a virtual server, specify the key pair for encrypting and decrypting the password. Set the environment variables as below:

```
$ VM_NAME=<virtualServerName> (any) $ IMAGE_REF_ID=<windowsimgId>
$ FLAVOUR_ID=<specOfVirtualServer> (listed flavor ID) $ VOL_SIZE=80
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
$ SOURCE=<blockStorageType>
$ DESTINATION=<connectionDestinationVol> (specify volume)
```

```
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of
the virtual server, or 0 otherwise)
$ KEYNAME=<keyPairName>
$ INSTANCE_MAX=<maxNumOfServers>
$ INSTANCE_MIN=<minNumOfServers>
$ PORT_ID=<portIdToBeSpecified>
```

2. Execute the following API call:

```
curl -s $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"server": {"name":
"$VM_NAME", "imageRef": "", "flavorRef": "$FLAVOUR_ID",
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "$IMAGE_REF_ID",
"volume_size": "$VOL_SIZE", "source_type": "$SOURCE", "destination_type":
"$DESTINATION", "delete_on_termination": "$ISDELETE"} ] , "max_count":
"$INSTANCE_MAX", "min_count": "$INSTANCE_MIN",
"key_name": "$KEYNAME", "networks": [{"port": "$PORT_ID"}]} }' | jq .
```

Example successful response:

```
{
  "server": {
    "security_groups": [
      {
        "name": "<specifiedSecGroupName>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<newVirtualServerId>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<projId>/servers/ <newVirtualServerId>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. Execute the following API call to verify the virtual server was created:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.servers[] | {status: .status, network_name:.addresses |
keys, key_name: .key_name, flavor_id: .flavor | .id, vm_id:.id,
security_group: .security_groups[] | .name, name: .name, }'
```

If the list contains the specified virtual server name then the virtual server has been created successfully.

```
...

{
  "status": "<statusOfVirtualServer>",
  "network_name": [ "<connectedNetworkName>"
],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<specifiedSecGroupName>",
  "name": "<virtualServerName>"
}

...
```

4. Retrieve the encrypted password.

Password can be retrieved after an interval of 10 minutes once the virtual server has been created. Set the environment variables as below:

```
$ SERVER_ID=<virtualServerId>
$ PROJECT_ID=<projId>
```

5. Execute the following API call:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-server-password -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

Example successful response:

```
{
  "password": "~~~~ <encryptedPassword> ~~~~=="
}
```

6. Retrieve the password by decrypting the response with the key pair. Set the encrypted password in an environment variable as below:

```
$ PASSWORD=<retrievedPassword>
```

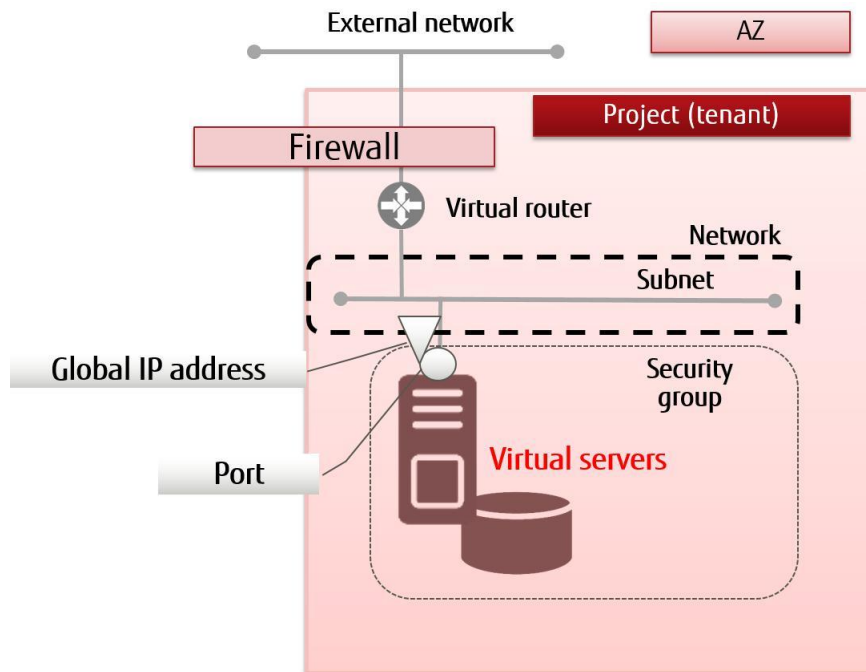
7. Execute the following command:

```
$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt -inkey
<pathToSpecifiedKeyFile>
```

5.23 Creating a virtual server (Windows / DHCP retrieval)

This section explains the process to create and verify a (Windows) virtual server using DHCP to retrieve the port (private IP address).

For an alternative to this method see "Creating a virtual server (Windows / port specification)".



Create a virtual server and obtain its password. A password is required for logging into the virtual server's remote desktop:

- Create a virtual server specifying a key pair for encrypting and decrypting a password
- Retrieve the encrypted password
- Decrypt the password using the same key pair

⚠ DANGER: Other users may be able to access the password when using a Windows virtual server which is deployed with `admin_pass` specified in the metadata therefore it is recommended to change the password after deployment.

1. Create a virtual server by setting a key pair for encrypting and decrypting the password. Set the environment variables as below:

```
$ VM_NAME=<virtualServerName> (any)
$ IMAGE_REF_ID=<windowsImgId>
$ FLAVOUR_ID=<specOfVirtualServer> (listed flavor ID)
$ VOL_SIZE=80
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
$ SOURCE=<blockStorageType>
$ DESTINATION=<connectionDestinationVol> (specify volume)
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of
the virtual server, or 0 otherwise)
$ KEYNAME=<keyPairName>
$ INSTANCE_MAX=<maxNumOfServers>
$ INSTANCE_MIN=<minNumOfServers>
$ NETWORK_ID=<networkIdThatYouWantToConnectTo>
$ SG_NAME=<secGroupNameToBeSpecified>
```

2. Execute the following API call:

```
curl -s $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"server": {"name":
"$VM_NAME", "imageRef": "", "flavorRef": "$FLAVOUR_ID",
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "$IMAGE_REF_ID",
"volume_size": "$VOL_SIZE", "source_type": "$SOURCE", "destination_type":
"$DESTINATION", "delete_on_termination": $ISDELETE} ] , "max_count":
$INSTANCE_MAX, "min_count": $INSTANCE_MIN, "key_name": "$KEYNAME",
"networks": [{"uuid": "$NETWORK_ID"}], "security_groups": [{"name":
"$SG_NAME"}]}}' | jq .
```

Example successful response:

```
{
  "server": { "security_groups": [
    {
      "name": "<specifiedSecGroupName>"
    }
  ],
  "OS-DCF:diskConfig": "MANUAL",
  "id": "<newVirtualServerId>", "links": [
    {
      "href": "http://10.3.0.201/v2/<projId>/servers/ <newVirtualServerId>",
      "rel": "self"
    },
    {
      "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
      "rel": "bookmark"
    }
  ]
}
```

```
}
```

3. Execute the following API call to verify virtual server was created:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" | jq '.servers[] | {status: .status, network_name:.addresses |
keys, key_name: .key_name, flavor_id: .flavor | .id, vm_id:.id,
security_group: .security_groups[] | .name, name: .name, }'
```

If the list contains the specified virtual server name then the virtual server has been created successfully.

```
...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<specifiedSecGroupName>",
  "name": "<virtualServerName>"
}
...
```

4. Retrieve the encrypted password.

Password can be retrieved after an interval of 10 minutes once virtual server is created. Set the environment variables as below:

```
$ SERVER_ID=<virtualServerId>
$ PROJECT_ID=<projId>
```

5. Execute the following API call:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-server-password -X GET -
H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

Example successful response:

```
{
  "password": "~~~~ <encryptedPassword> ~~~~=="
}
```

6. Retrieve the password by decrypting with the key pair. Set the environment variable as below:

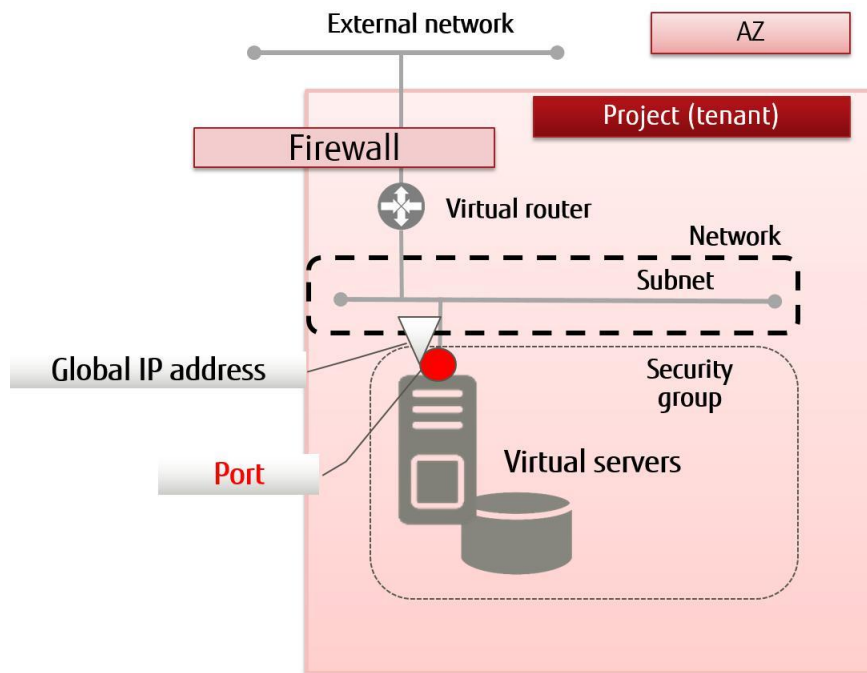
```
$ PASSWORD=<retrievedPassword>
```

7. Execute the following command:

```
$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt -inkey
<pathToSpecifiedKeyFile>
```

5.24 Retrieving a virtual server port

This section explains the process to retrieve the port ID of a virtual server which is required for assigning global IP addresses to the virtual server.



1. Set the environment variable as below to specify the virtual server whose port has to be retrieved:

```
$ SERVER_ID=<virtualServerIdToRetrievePortFrom>
```

2. Execute the following API call to retrieve the port:

```
$ curl -k -s $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface -X GET -H
"X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

Example successful response:

```
{
  "interfaceAttachments": [
    {
      "port_state": "<status>",

```

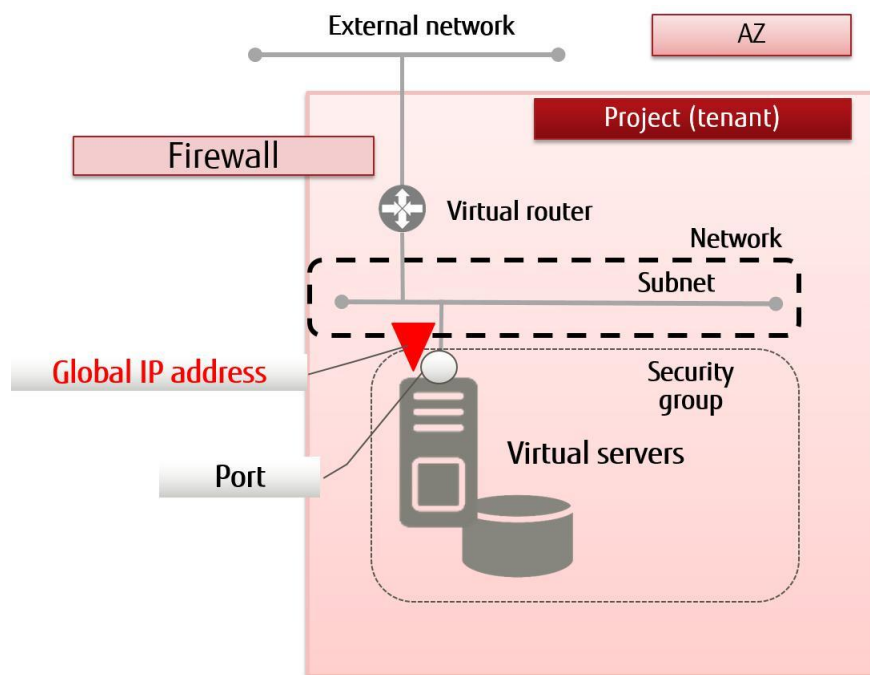
```

    "fixed_ips": [
    {
        "subnet_id": "<virtualServerId>",
        "ip_address": "<privateIpAddr>"
    }
    ],
    "port_id": "<portId>",
    "net_id": "<networkIdThatVirtualServerIsConnectedTo>",
    "mac_addr": "<macAddr>"
  }
]
}

```

5.25 Allocating a global IP address and assigning it to a virtual server

This section explains how to allocate a global IP address for accessing virtual resources over the internet and assign it to a virtual server.



1. Set the environment variables as below:

```

$ NETWORK_ID=<networkId> (specify the external network ID of the availability
zone where the virtual server is located)
$ VM_PORT_ID=<virtualServerId>

```

2. Execute the following API call:

```
$ curl -k -s $NETWORK/v2.0/floatingips -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN"
-H "Content-Type:application/json" -d '{"floatingip":
{"floating_network_id":"' $NETWORK_ID "'", "port_id":"' $VM_PORT_ID "'",
"availability_zone": "' $AZ'"}' | jq .
```

Example successful response:

```
{
  "floatingip": {
    "router_id": "<virtualRouterIdOfDefaultGateway>",
    "status": "DOWN",
    "tenant_id": "<projId>",
    "floating_network_id": "<externalNetworkAddr>",
    "fixed_ip_address": "<privateIpAddr>",
    "floating_ip_address": "<globalIpAddr>",
    "port_id": "<specifiedPortId>",
    "id": "<globalIpAddrId>",
    "availability_zone": "<availabilityZone>"
  }
}
```

Availability zones such as AZ1 is presented by "uk -1a" and AZ2 is presented by "uk -1b".

3. Execute the following API call to verify the global IP address was allocated:

```
$ curl -s $NETWORK/v2.0/floatingips.json -X GET -H "X-Auth-Token:
$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

If the list of IP addresses contains the newly assigned global IP address then the address has been allocated successfully.

```
{
  "floatingips": [

    ...

    {
      "router_id": "<virtualRouterIdOfDefaultGateway>",
      "status": "<status>",
      "tenant_id": "<projId>",
      "floating_network_id": "<externalNetworkAddr>",
      "fixed_ip_address": "<privateIpAddr>",
      "floating_ip_address": "<globalIpAddr>",
      "port_id": "<portId>",
      "id": "<globalIpAddrId>",
      "availability_zone": "<availabilityZone>"
    }
  ]
}
```

```
...  
]  
}
```

5.26 Logging into a virtual server (CentOS)

This section explains the process to login to a virtual server (CentOS).

Confirm the following points before logging to a virtual server:

- The status of the global IP address assigned to the virtual server targeted for login must be "ACTIVE"
- The security group rules must be set correctly for the operating environment
- The firewall rules and policies must be set correctly for the operating environment
- A key file (.pem) must exist in the working directory

1. Execute the following command:

```
$ ssh -i <k5user>@<globalIpAddrAssignedToTargetVirtualServer>  
  
The authenticity of host '<globalIpAddrAssignedToTargetVirtualServer>'  
can't be established.  
  
RSA key fingerprint is: <fingerprint>.  
Are you sure you want to continue connecting (yes/no)? <> Warning:  
Permanently added  
'<globalIpAddrAssignedToTargetVirtualServer>' (RSA) to the list of known  
hosts.
```

2. Switch to the root user.

A special command is required in CentOS provided by K5 for retrieving the root privileges after login.

```
$ sudo su -
```

5.27 Logging into a virtual server (Windows)

This section explains the process to login to a virtual server (Windows).

Before logging in, check the following:

- The status of the global IP address assigned to the virtual server targeted for login must be "ACTIVE"
- The security group rules must be set correctly for the operating environment
- The firewall rules and policies must be set correctly for the operating environment

Run the remote desktop from the client PC and login to connect.

 **Warning:** After login to the virtual server, it becomes necessary to configure a firewall in the virtual server for security reasons.

Use the server only after verifying whether the firewall has been configured or not.

6. Appendix 2 - Deleting resources

Topics:

- [Deleting a virtual server](#)
- [Releasing a global IP address](#)
- [Deleting a port](#)
- [Deleting a key pair](#)
- [Deleting a firewall](#)
- [Deleting a firewall policy](#)
- [Deleting a firewall rule](#)
- [Deleting a security group rule](#)
- [Deleting a security group](#)
- [Detaching a virtual router from a subnet](#)
- [Deleting a virtual router](#)
- [Deleting a subnet](#)
- [Deleting a network](#)

6.1 Deleting a virtual server

This section details the steps to delete a virtual server.

1. Set the environment variable as below:

```
$ TMP_VM_ID=<virtualServerIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $COMPUTE/v2/$PROJECT_ID/servers/$TMP_VM_ID -X DELETE -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime> Keep-Alive: timeout=5, max=100
```

6.2 Releasing a global IP address

This section details the steps to release a global IP address. The global IP address is not released as part of a virtual server's deletion, hence this separate process.

1. Set the environment variables:

```
$ TMP_FLOATINGIP_ID=<globalIpAddrIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/floatingips/$TMP_FLOATINGIP_ID -X DELETE -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

```
HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runTime> Keep-Alive: timeout=5, max=100
```

6.3 Deleting a port

This section details the steps to delete a port. A port is not deleted when the virtual server it is attached to is deleted, hence this separate process.

1. Set the environment variable as below:

```
$ TMP_PORT_ID=<portIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/ports/$TMP_PORT_ID -X DELETE -H "X-Auth-Token:
$OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime> Keep-Alive: timeout=5, max=100
```

6.4 Deleting a key pair

This section details the steps to delete a key pair.

1. Set the environment variable as below:

```
$ TMP_KEYPAIR_NAME=<keyPairIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -X DELETE -s -i $COMPUTE/v2/$PROJECT_ID/os-keypairs/$TMP_KEYPAIR_NAME -H
"X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime> Keep-Alive: timeout=5, max=100
Content-Type: text/html; charset=UTF-8 Content-Length: 0
```

6.5 Deleting a firewall

This section details the steps to delete a firewall.

1. Set the environment variable as below:

```
$ TMP_FW_ID=<firewallIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/fw/firewalls/$TMP_FW_ID -X DELETE -H "X-Auth-Token:
$OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

6.6 Deleting a firewall policy

This section details the steps to delete a firewall policy.

1. Set the environment variable as below:

```
$ TMP_FW_ID=<firewallPolicyIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/fw/firewall_policies/$TMP_FWP_ID -X DELETE -H "X-
Auth-Token: $OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

6.7 Deleting a firewall rule

This section details the steps to delete a firewall rule.

1. Set the environment variable below as follows:

```
$ TMP_FW_ID=<firewallRuleIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/fw/firewall_rules/$TMP_FWR_ID -X DELETE -H "X-Auth-Token: $OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache

Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

6.8 Deleting a security group rule

This section details the steps to delete a security group rule.

 **Tip:** The deletion of a security group will cause the deletion of any rules it contains.

1. Set the environment variable as below:

```
$ TMP_SGR_ID=<secGroupRuleIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/security-group-rules/$TMP_SGR_ID -X DELETE -H "X-Auth-Token: $OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache

Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-Control: no-cache X-Request-Id: <xReqId>
X-Runtime:<runtime>
```

6.9 Deleting a security group

This section details the steps to delete a security group.

1. Set the environment variable as below:

```
$ TMP_SGR_ID=<secGroupIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/security-groups/$TMP_SG_ID -X DELETE -H "X-Auth-Token: $OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

6.10 Detaching a virtual router from a subnet

This section details the steps to detach a subnet from virtual router.

1. Set the environment variables as below:

```
$ TMP_ROUTER_ID=<virtualRouterIdToBeDetached>
$ TMP_SUBNET_ID=<subnetIdAttachedTo>
```

2. Execute the following API call:

```
$ curl -s $NETWORK/v2.0/routers/$TMP_ROUTER_ID/remove_router_interface -X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" -d '{"subnet_id": "'$TMP_SUBNET_ID'" }' | jq .
```

Example of a successful response:

```
{
  "subnet_id": "<subnetIdTargetedForDetachment>",
  "tenant_id": "<projId>",
  "port_id": "<connectedPortId>",
  "id": "<virtualRouterIdTargetedForDetachment>",
  "availability_zone": "<availabilityZone>"
}
```

6.11 Deleting a virtual router

This section details the steps to delete a virtual router.

1. Set the environment variable as below:

```
$ TMP_ROUTER_ID=<virtualRouterIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/routers/$TMP_ROUTER_ID -X DELETE -H "X-Auth-Token:
$OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-id Cache-Control:
no-cache X-Request-Id: xReqId
X-Runtime: <runtime>
```

6.12 Deleting a subnet

This section details the steps to delete a subnet.

1. Set the environment variable as below:

```
$ TMP_SUBNET_ID=<subnetIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/subnets/$TMP_SUBNET_ID -X DELETE -H "X-Auth-Token:
$OS_AUTH_TOKEN"

HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

6.13 Deleting a network

This section details the steps to delete a network.

1. Set the environment variable as below:

```
$ TMP_NETWORK_ID=<networkIdTargetedForDeletion>
```

2. Execute the following API call:

```
$ curl -i -s $NETWORK/v2.0/networks/$TMP_NETWORK_ID -X DELETE -H "X-Auth-Token:$OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT x-compute-request-id: req-<reqId> Cache-
Control: no-cache X-Request-Id: <xReqId>
X-Runtime:<runtime>
```