PROJEKT

Multitenantné operačné centrum kybernetickej bezpečnosti riešené ako otvorená cloudová služba s prvkami strojového učenia

Previazané s balíkom KPB6 – Publikačné výstupy

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

ÚRAD VLÁDY
SLOVENSKEJ REPUBLIKY

GNS3aaS: Networking lab in OpenStack

Martin Kontsek*, Pavel Segec, Marek Moravcik, Jana Uramova
*Faculty of Management Science and Informatics, University of Zilina, Univerzitna 8215/1, 010 26 Zilina
*e-mail: martin.kontsek@fri.uniza.sk

***Abstract*—This paper explores the potential of GNS3 as a cloud-based solution for academic institutions. It describes how the GNS3aaS has been implemented and its key features. Firstly, a brief explanation of how users can create their own GNS3 server and what data they should provide. Secondly, a detailed description of how the instance is provisioned. A step-by-step tour of the installation and configuration process. Next, the reader will find 2 more features that were tested/implemented because the faculty wanted to know if it was worth implementing. These 2 features are image sharing and authentication. The next part of the paper describes the new version of the GNS3 server, as they changed the architecture of it. Also introduced some useful features like user management, but on the other hand, due to the backend changes, broke the image sharing process that was possible on the older versions.**

***Index Terms*—GNS3 server, cloud services, GNS3 GUI, GNS3aaS**

I. Introduction

The cloud and cloud services are increasingly in demand[1], and it's no different in the academic environment. Cloud technologies can bring applications closer to the user and they're accessible from anywhere and, in most cases, from any device. Many technical universities teach their students how to build their own network, how to configure it or how to find cyber threats. To provide this type of education, universities need network hardware, lots of end devices and, at the same time, space in which to place or teach these skills. Not surprisingly, many schools can't afford such equipment. The solution to this type of problem is GNS3.

Fig. 1. GNS3

It's a tool that lets you virtualize all supported devices[2], so you don't need to buy any dedicated hardware. All you need is an image that you upload to the GNS3 server, create your own topology and run it. The problem is that with current stable versions of GNS3 servers, you have to share your resources with others if we are talking about academic implementations. This is because there will be one, two or, at best, several servers in your university, where you will share all the resources with other students. The solution is to create a cloud service where each student could create their own GNS3 server and use only the resources allocated to them.

II. GNS3aaS

To realize GNS3aaS there are several steps that need to be done. There needs to be a mechanism for the user to pass data to the instance. The user has to specify the version of the GNS3 server (it has to be the same as the version of your GNS3 GUI) and if he wants to enable authentication on it. If he decides to use it, he will also need to pass the username and password for the user. The other thing is the way the instance will be provisioned.

A. User-data

As mentioned above, the user needs to pass certain data to successfully create a GNS3 instance. The data is kept as short as possible to keep it simple for the user. It consists of version, bool value if authentication is introduced and user name and password. Two of these variables, or four in the case of an authentication process, are passed to the installation script present in the custom image. These variables are provided to the installation script by the cloud-init tool as a shell script. The more specific flow is that cloud-init tool will execute script with variables, which will be exported to environment and then it will execute installation script, which will be described in another part of this article. The example of user data could be seen below2

```
#!/bin/bash

GNS3VERSION="2.2.43"
GNS3AUTHENTICATION="True"
GNS3USER="gns3-user"
GNS3PASSWORD="Strong-Password"

export GNS3VERSION
export GNS3USER
export GNS3PASSWORD
export GNS3AUTHENTICATION

sudo -E bash /opt/gns3/script/gns3-install.sh
```

Fig. 2. Example of user-data

B. Provisioning

Deployment or in other words installation of the GNS3 server on an instance is a bit complex. Installation scripts are stored on the custom Ubuntu 22.04 image. It consists of

several parts, such as installation of dependencies, creation of GNS3 system users, bypassing system interaction during installation or creation of system service for GNS3 server.

During the installation process, the user can monitor installation status. Since every logical step during installation is logged, by custom logging function where its format looks like this:

```
**********
LOG MESSAGE
**********
```

All steps which are triggered or executed with their description are described below.

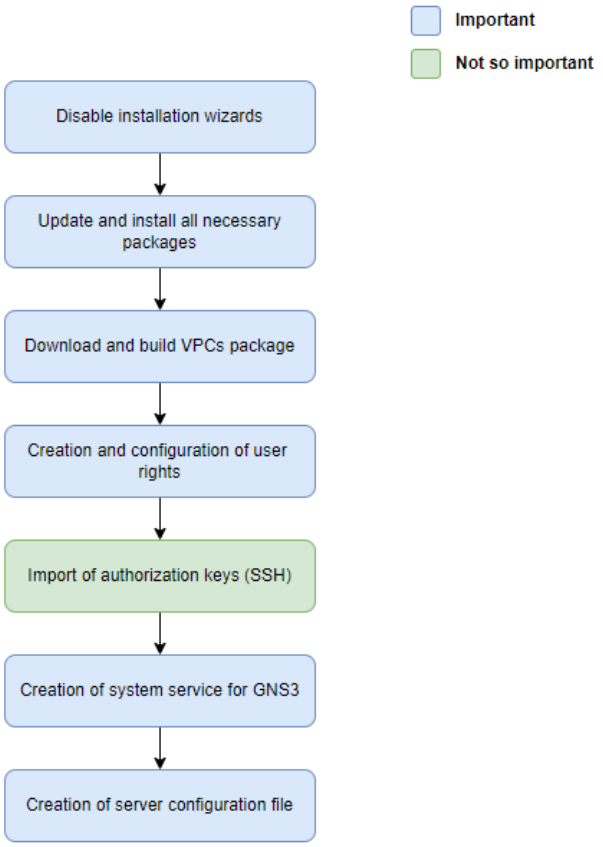

Fig. 3. Installation process of GNS3 server

- The first step is to disable user interaction during package installation, since the script is triggered by user data, there's no way how the user could interact with the installation window and at the same time it's unwanted because it's a cloud service. The user should be able to use the service with no or minimal effort. The commands to disable the installation wizard look like this: `sudo sed -i "1iDEBIAN_FRONTEND=noninteractive\n" /etc/environment`
  `source /etc/environment`

  This was mainly needed because of the Wireshark installation process. Where the user was asked to confirm or deny actions.
- The second step is to install packages or dependencies. There are a lot of dependencies that need to be installed before installing the GNS3 server. The most important ones are python, python-pip, qemu packages, libvirt, Wireshark and many more. After the dependencies are installed, the script can move on to the next part of the installation and that is the installation of the GNS3 server using pip. GNS3 can be installed using the Ubuntu package manager, but the user cannot specify the version of the server. If the user chooses to use pip, he will be able to specify all released versions.
- During the creation of the script, a problem was discovered where one package couldn't be installed during the installation. Even with custom repositories. The package is for simple GNS3 computers. It is a small simple virtual pc that can ping, set IP and so on. The problem was solved by installing VPCs from git[3]. The VPCs are downloaded from the git repository, built on the virtual machine and placed in the GNS3 image folders.
- Another step that must not be forgotten is the configuration of user rights. Specifically, the script set root permissions for the main users used by the GNS3 server, which are ubridge, libvirt, kvm, wireshark, gns3. The last one, gns3 user, is a bit tricky. Many of us would expect the gns3 user to be created automatically during the installation of the GNS3 server, but it isn't and it had to be added to the installation script, otherwise it wouldn't work.
- A step that is not so important for newer versions, but really important for image sharing, is the step where the SSH key is imported into the instance. This allows the instance to connect to the storage system. Which was another Ubuntu instance.
- Another important step is to create system service. By using the system service, the GNS3 server is activated every time the instance is started, it logs all actions in the log directory and it's easy to manage for administrators. Another advantage is that before starting the GNS3 server, we could connect to the shared storage of GNS3 images. The shared storage will be described in the next part of this article.

  The configuration file could be divided into 3 parts. The first part is the specification of the steps to be executed before the GNS3 server. In this case it is communication with shared storage of images and configuration files, for that the service needs to execute two commands. First one will copy configuration file and second one will mount images on the instance/GNS3 server. The second part is the path to the executable or startup file of the GNS3 server and the last part is the user who will run the startup

file. In this case it is root, because it must have the highest privileges.

- The last step is to create the GNS3 server configuration file, which contains all the main attributes, such as whether authentication is enabled, user name and password. Most importantly, it also contains the paths to the image, template, appliance and projects folders.

C. Image sharing

The next requested feature is image sharing. Since images for GNS3 are not small, and it would be pointless to copy images to every single instance of the GNS3 server. Somehow a way had to be found to share images across all instances. It's possible to do this using the scp and sshfs packages. The first package is used to download the image configuration file to the instance. The file contains the specification of memory, CPU, RAM, etc. of each individual image. The next package is used to mount the image folder into the instance. The first package could also be used for this, but there is no point in transferring or copying instances into the instance every time, if you can just mount the folder and load only the images you need.

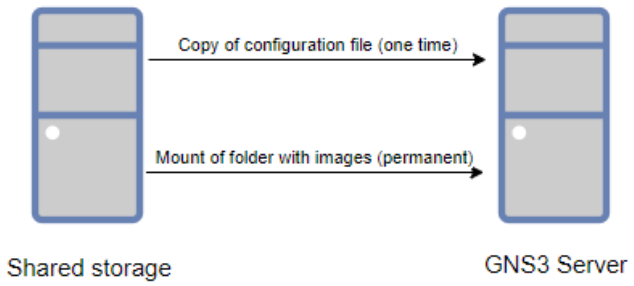

Fig. 4. Connection with shared storage

By using such packages we could achieve safe and efficient image sharing, but with introduction of new version of GNS3 server, which will be described in other chapters of the article, there is no way how the image sharing could work this way, since they store all configuration files in database and finding a way how to collect data from such files using only built-in function would be quite a difficult task.

D. Authentication in GNS3

Another aspect of GNS3 is its accessibility. If you host a GNS3 server, it's accessible to everyone on your network. The only way to make it a bit more secure is to enable bult-in authentication, which only consists of one GNS3 user. So if you use it as your own GNS3 server, which is only for you or a small group of colleagues, it is basically all you need. You set a username and password, share it with your colleagues and that's it. The username and password are set during the instance initialization via cloud-init data, which needs to be provided.

If authentication is enabled, the user will be asked to enter valid credentials. If the credentials are incorrect, they won't be able to access the dashboard, create a project or anything else. Another problem with the current type of authentication is that if users, which are using the GNS3 server forget their password. There is no way for them to reset it, for that you need to be a bit more technical type of person and have access to the CLI of GNS3 server and change it from there. On the other hand, in the latest versions of GNS3 (beta and pre-release [3.x.x]), you don't suffer from this kind of problem, because there is always someone who has administrator role and is so reliable that he doesn't forget his password and if he does. He should have the technical background to be able to reset it via the CLI, and this person is responsible for resetting passwords for users who have forgotten them.

III. On-boarding of new GNS3 version

As mentioned above, the GNS3 development team has introduced a new version of the GNS3 server that drastically changes the way the server is implemented[4]. On the other hand, it should improve performance, user management, scalability and network handling. The question now is whether it can be used in a university or school environment. The current problem with the current GSN3 server is security or authentication. Any user who has access to the GNS3 server also has access to other users' projects, and that's uncomfortable. Now is the time to find out if the new version will change this.

A. Deployment of new versions

From a deployment point of view, there is nothing new. GNS3 server could be installed by its scripts or even the custom image for cloud usage works with new versions as well. All the user needs to do is follow the instructions above and the user can deploy their own GNS3 server. Scale it as desired, upload your own images and play with it.

B. User management

The most significant change for users or administrators is the ability to manage users. They introduced a role system, administrator can also create his own role, change rules, assign projects, images and so on. The standard policy is RBAC[5, 4], which stands for Role-based Access Control.

Administrator can create users, the small inconvenience is that GNS3 can't connect to AD, so the automatic user creation won't work. Another part is group creation, users can be assigned to groups of people where they share projects and access policies. Another important part of user management is role creation. There are 7 built-in roles that basically cover all user needs, but there is also the ability to create your own role with specific rules and last but not least is the resource pool creator. It's used to specify the environment, groups, etc. The administrator can assign specific projects to users

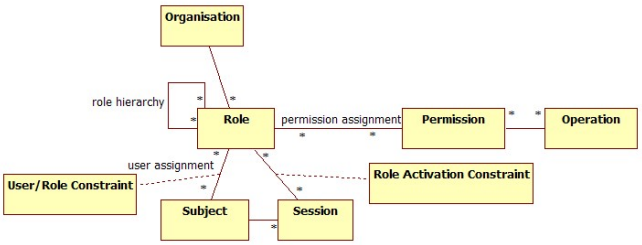

Fig. 5. Connection with shared storage

or groups via resource pools. Finally, there is a setting or configuration called ACL, where administrators can allow or disallow specific endpoints, roles and much more.

C. Back-end changes

The most significant changes have been made to the backend. Firstly, the REST API framework has been changed and now uses it FastAPI[6]. This framework provides universal and stable REST API communication and up-to-date Swagger documentation for users and developers who want to use its API calls.

Another important change is the new way of storing configuration files. As mentioned above, configuration was stored in multiple config files. One config file was for images, one for templates and so on. Each type of image was stored in the same config file. Now they started to separate things and store them not in configuration files but in database tables. This also improved their administration, stability and performance. The database system used is sqlite3. The various config files are now separated into 25 tables. This consists of tables for specific image types (qemu, iou, dynamips,..) and maps to specific projects. All the tables are listed below.

```
sqlite> .tables
acl                          qemu_templates
alembic_version              resource_pool_map
cloud_templates              resource_pools
computes                     resources
docker_templates             roles
dynamips_templates           templates
ethernet_hub_templates       user_group_map
ethernet_switch_templates    user_groups
image_template_map           users
images                       virtualbox_templates
iou_templates                vmware_templates
privilege_role_map           vpcs_templates
privileges
```

Fig. 6. GNS3 server database tables

Please note that it is possible that the current tables look different, as this version is still a work in progress, so there is a possibility that they will add, remove or change some tables.

D. The process of image storage in new versions of GNS3

As mentioned before, the way how it works on the server side in the backend is completely reworked. The interesting part is how the images are stored, since they have introduced a database system in their backend system. The storage system could be divided into two parts, the first one is the authorization and the way how to have correct privileges to upload and store the image. It's not that important, so the focus will be on the second part, and that is the analysis of the image, creating database records and so on.

The first step is when the user uploads the image to the server. The image is stored in a temporary location. The server will then load all the necessary attributes, such as the destination folder where it will be saved, the path to the current folder, and it will also check if the image doesn't currently exist on the GNS3 server. If it's not, it will cancel all operations, nevertheless it will check it, only by it's name, so if user will upload same image with different file name, it will continue with next step.

Second step is image analysis, by analysis is meant the calculation of the checksum. It's calculated using the MD5 algorithm, which provides fast and reliable results. Also during the calculation there is a function that decides what type of image it is. This means if it is an IOS, IOU or Qemu image, also based on this result templates are created. On the picture below you can see the algorithm that decides the type of image.

```
def check_valid_image_header(data: bytes, allow_raw_image: bool = False) -> str:

    if data[:7] == b'\x7fELF\x01\x02\x01':
        # for IOS images: file must start with the ELF magic number, be 32-bit, big endian and have an ELF version of 1
        return "ios"
    elif data[:7] == b'\x7fELF\x01\x01\x01' or data[:7] == b'\x7fELF\x02\x01\x01':
        # for IOU images: file must start with the ELF magic number, be 32-bit or 64-bit, little endian and
        # have an ELF version of 1 (normal IOS images are big endian!)
        return "iou"
    elif data[:4] == b'QFI\xfb' or data[:4] == b'KDMV':
        # for Qemu images: file must be QCOW2 or VMDK
        return "qemu"
    else:
        if allow_raw_image is True:
            return "qemu"
        raise InvalidImageError("Could not detect image type, please make sure it is a valid image")
```

Fig. 7. Image header algorithm in GNS3 server

The last steps are to verify that the server has the right to store the image on the system, to verify that there is no image with the same checksum already stored on the system, because if there is, it would mean that the same image is already stored there, and finally to update the image table on the database system.

IV. Conclusion

Implementing GNS3aaS could be a huge step forward for many faculties and even schools. If the faculty has enough resources to make a part of it available to students, they could use it to build their own GNS3 server and maybe work on their own research without the interaction of someone else or even using their resources. It's all possible thanks to a custom image of Ubuntu 22.04, when used to create an instance and provided with the correct data, the GNS3 server will be automatically installed and configured. The user can specify

the version he wants. The other advantage is, as already mentioned, that the user can specify the version, because there is a complication that the GNS3 server and the GUI must have the same version, otherwise the connection won't be established. The ability to share images was also tested. So, if each cloud user is allowed to create their own GNS3 server, they don't need to upload their images, but the GNS3 server will connect to the shared storage. This also saves resources. So the GNS3aaS is fully implemented and ready to use, even without installing other applications, because the GNS3 server is installed with its web application, so the user can access the GUI on the instance IP address and port 3080.

A comparison of new versions of GNS3 with older versions showed the future capabilities of GNS3 as an educational tool. Mainly because of the improved user management system. With the new versions, there could be only one GNS3 server (suitable for smaller faculties), where administrators could specify different sets of permissions on how they could use the GNS3 server. This would allow administrators to avoid many of today's problems, such as deleting images, deleting other users' projects, etc. However, there is one small drawback, and that is that there is no way to automate user creation or even connect it to AD. So manually creating users will be time consuming. On the other hand, there is still the option to deploy a new version of GNS3 with a custom image for everyone that needs it. Image sharing won't be possible with the current implementation, but the user should still take care of his instance. Anyway, the new version of GNS3 Server is a huge step forward in the way it works and there's a big chance that it will be used by many organizations.

ACKNOWLEDGMENT

Not so sure if that needs to be here

REFERENCES

[1] Dhanush Kumar. "Describe the benefits of using cloud services". In: *https://medium.com/@danushidk507/describe-the-benefits-of-using-cloud-services-6652af50682d* (2024).

[2] Tim Canter. "What is GNS3- and Why do You Need it?" In: *https://allconsuming.net/what-is-gns3-and-why-do-you-need-it/* ().

[3] AYOUB LASER. "VPCS executable version must be ¿=0.6.1 but not 0.8". In: *https://gns3.com/vpcs-executable-version-must-be-greater-than-0-6-1-but-not-0-8* (2020).

[4] Jeremy Grossmann. "What's new in GNS3 version 3.0". In: *https://gns3.com/community/blog/whats-new-in-gns3-version-3-0* (2023).

[5] Alex Olivier. "RBAC vs ABAC — Which is better for your application?" In: *https://www.cerbos.dev/blog/rbac-vs-abac* (2024).

[6] Jeremy Grossmann. "GNS3 3.0.0 Beta 1 Released!" In: *https://gns3.com/community/blog/gns3-3-0-0-beta-1-released* (2023).