Extensive Guide

to DaaS application

Produced by: Alireza Mohammadpour
TNT-LAB

alireza.mohammadpour@cnit.it
a.r.mohamadpour@gmail.com

If you come across any problems, contact me at

a.r.mohamadpour@gmail.com

alireza.mohammadpour@cnit.it

23/03/2023

Table of Contents

About This File

This file was created for the benefit of all teachers and students wanting to use this DaaS for exams/lessons etc.

SECTION 1

Brief Introduction to TNT-lab Desktop as a Service

The TNT-lab Desktop as a Service (DaaS) offers a convenient solution for remote desktop access via the secure HTTPS protocol. This service not only provides a reliable Ubuntu desktop environment, but also includes essential software applications and packages required for this course, such as Visual Studio Code, g++, gcc, and other relevant tools.

With TNT-lab DaaS, each user can access a fully functional desktop environment and has all the necessary tools to complete coursework without the need for additional installations or setup. The service also enables user policies to be applied, such as limiting internet access, restricting information sharing, and other requested limitations necessary for the specific courses.

Furthermore, the TNT-lab DaaS can be easily accessed through the Guacamole web application server, providing a seamless and user-friendly experience.

In summary, TNT-lab DaaS provides a comprehensive solution for remote desktop access and offers a convenient, hassle-free approach to accessing course-related applications and software packages.

2

SECTION

Preparing & Installing

The service can run on Ubuntu 18.04 and 20.04 versions of the operating system. While it is recommended to run the Guacamole server on a separate machine from the DaaS service to ensure optimal performance and minimize potential issues, it is worth noting that one can still apply the same requirements on a single machine if needed.

In this guide, the requirements are divided into two different sections for clarity, but the same hardware and software specifications can be met by a single machine. However, separating the services onto two distinct machines can help to prevent potential conflicts and ensure that each service operates smoothly without impacting the performance of the other. It can result in a smoother and more reliable user experience, which is crucial for any service.

Regardless of whether the services are run on one or two machines, it is important to ensure that both machines meet the necessary hardware and software requirements for their respective services. This can help to prevent any potential performance issues or compatibility problems that could arise from running these services on machines that do not meet the necessary specifications.

Overall, while the requirements can be applied to one machine, running the DaaS and Guacamole services on separate machines is highly recommended to ensure optimal performance and minimize potential issues. By following the necessary specifications and ensuring the compatibility of the hardware and software, one can create a smoother and more reliable user experience for their service.

2.1 Install requirements and run DaaS

Deploying the DaaS (Desktop as a Service) service requires a series of steps to be followed carefully. Firstly, the machine where the service will be deployed needs to have ssh installed on it. This can be achieved by using the apk package manager and running the command

```
sudo apt update
sudo apt install openssh-server
```

Once the ssh server has been installed, it is recommended to log in to the machine for the first time to ensure that everything is working as expected.
After this, Ansible needs to be installed on your machine. If you are using Windows, you can install Ansible using WSL. To install Ansible on Ubuntu or Windows WSL, run the command

```
sudo apt install ansible
```

in the terminal. Additionally, sshpass also needs to be installed on your system. This step is necessary because Ansible requires **sshpass** to connect to the DaaS server. To install sshpass, use the command

```
sudo apt install sshpass
```

Next, the DaaS_VNC app needs to be cloned or downloaded from the Github repository. This can be done by navigating to the repository at `https://github.com/AlirezaMohammadpour85/DaaS_VNC.git` and using the "git clone" command to download the source code.

```
git clone https://github.com/AlirezaMohammadpour85/DaaS_VNC.git
```

Once the app has been cloned, the next step is to configure the inventory file for Ansible. The inventory file is a list of servers that Ansible will manage. In this application, you need to modify the inventory file and replace the IP address with the IP address of the Ubuntu machine. The "[ubunto]" section should be the same as the Ubuntu machine username. Moreover, you need also to add the usernames of the users that you want to create (give remote access) in the "[users]" section. These usernames will also define in the yaml configuration file.
Lastly, the Ansible playbook file needs to be executed to deploy the DaaS service. This can be done by running the command:

```
ansible-playbook --become --ask-become-pass --ask-pass playbooks.yaml
```

or you can run the prepared bash file inside the cloned app directory in the **DaaS** folder. It will ask you to enter the ssh password. before running the bash file you need to make it executable by using this command :

```
sudo chmod +x run_daas_playbooks.sh
```

and then execute it **./run_daas_playbooks.sh**.

The **playbooks.yaml** file contains the instructions for deploying the DaaS application which includes copying the source code to the DaaS server, installing the necessary dependencies, and starting the application. After the playbook has finished running, the DaaS service should be accessible by Guacamole. The remote desktop application also can reach the defined users via VNC.

It is worth noting that, before running the previous command, you must customize the **playbooks.yaml** file. In Section 2.1.1, a complete instruction for customizing this file is proposed.

2.1.1 Modifying playbooks.yaml file

As previously mentioned, this file contains all the necessary steps to run the DaaS service. In order to tailor the output to your specific needs, there are certain lines in the file that need to be modified. Firstly, navigate to line 9 in the "users" section and remove the default usernames and replace them with your own usernames. This can be done by editing the list in this section. Refer to Figure 1 for an example. Once you have added your desired usernames, these users will be created on the machine with a default password of "ubuntu".

```
9   - users: # Replace with your usernames
10    - 5167230-104779
11    - 5216510-104779
12    - 5180846-104779
13    - 5206225-104779
14    - 4373074-104779
15    - 4484448-104779
16    - 5193530-104779
17    - 4648295-104779
18    - 5052444-104779
19    - prof-admin
```

Figure 1: Create usernames

Next, proceed to line 111 and replace/add/remove the usernames with the ones you defined in the previous section. This step is important in order to ensure that the correct users are granted access to the service. Figure 2 provides an example of how this section should look once it has been updated.

```
107 - name: Edit rc.local file to start program after boot for policies
108   lineinfile:
109     path: /etc/rc.local
110     line: '{{ item }}'
111   loop: # Replace with your usernames
112     - '#!/bin/bash'
113     - chmod -R 777 /home/ubuntu/.config/Code /home/ubuntu/.vscode/extensions
114     - '# apply policies for all users'
115     - /home/ubuntu/.http_drop.sh 5167230-104779
116     - /home/ubuntu/.http_drop.sh 5216510-104779
117     - /home/ubuntu/.http_drop.sh 5180846-104779
118     - /home/ubuntu/.http_drop.sh 5206225-104779
119     - /home/ubuntu/.http_drop.sh 4373074-104779
120     - /home/ubuntu/.http_drop.sh 4484448-104779
121     - /home/ubuntu/.http_drop.sh 5193530-104779
122     - /home/ubuntu/.http_drop.sh 4648295-104779
123     - /home/ubuntu/.http_drop.sh 5052444-104779
124     - exit 0
```

Figure 2: Create startup services for each user

Moving on, navigate to line 176 in the "password_remote" section and replace the default list items with dedicated passwords for each user. This will ensure that each user has a unique password for accessing the service remotely.

In line 189 in the "users_remote" section, you can define the remote usernames for each user. We highly recommend using the same usernames as the ones defined in the previous sections for

consistency. Figure 3 provides an example of a completed section where both remote usernames and passwords have been defined.

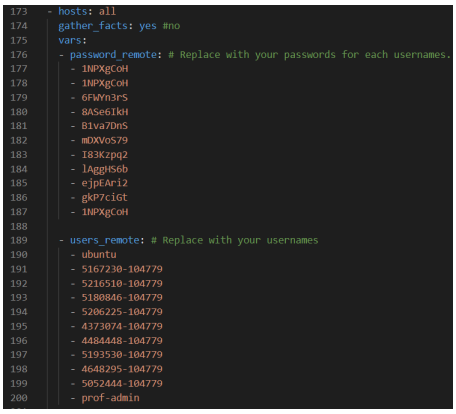

Figure 3: Create usernames and passwords for remote users

To further customize the DaaS service, navigate to the "Edit .run_vncserver.sh part 2" play and locate the "blockfile" section. Starting from line 297, you will need to replace, define, add, or remove the if conditions with the usernames that were defined in the previous sections. This is crucial in order to ensure that the correct users have access to the service. Additionally, you will need to assign a number to each "display" in increasing order, starting from the "ubuntu" display number which should be left untouched. Refer to Figure 4 for a pre-completed example of this play.
Moving on, in line 349, replace the list of usernames with the ones that were defined in the previous sections. This step will ensure that the correct users are granted access to the service.

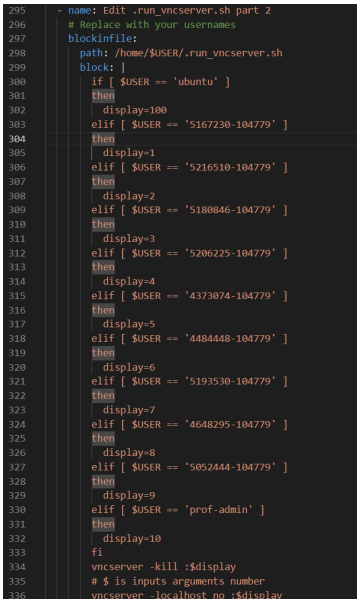

Figure 4: Define display number for VNC

"Lastly, in line 354 (refer to Figure 5), you will need to complete the "display_number" list. This information should be taken from the previous "blockfile" section and should be equal to the number of users. Starting from one, you will need to increase the number by one until you reach

the number of users. This step is important in order to ensure that each user is assigned a unique display number.

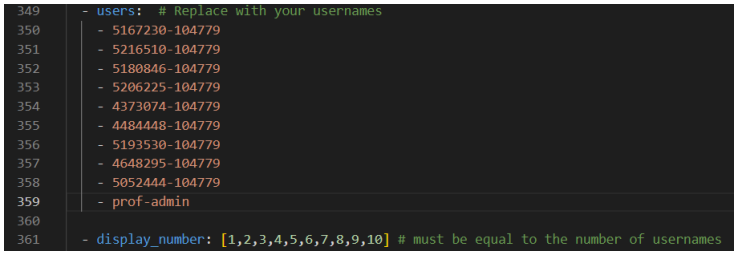

Figure 5: Set to each username a display number

By following these steps, you will have successfully customized the DaaS service to your specific needs.

2.2 Install requirements and run Guacamole

Guacamole is a powerful web-based remote desktop gateway that allows you to connect to remote servers, virtual machines, and other devices using just a web browser. This makes it incredibly convenient and accessible, as you don't need to install any software on your local machine to use it.

To get started with Guacamole, you'll first need to install it on a server. We recommend installing Guacamole on a separate server, but it's also possible to install it on the same server as your DaaS (Desktop-as-a-Service) solution, as long as you have enough resources to handle both. If you're using Ubuntu, you'll need to make sure that the openssh-server package is installed on the machine, as Guacamole uses SSH to connect to remote devices.

Once you have your server set up, you can move on to installing Guacamole itself. Start by navigating to the **guacamole** folder on your server, and opening the inventory file. This file contains information about the server and the devices you want to connect to, and you'll need to update it with the IP address of the machine you want to install Guacamole on. Make sure to save the changes once you're done.

There are two ways to install Guacamole: you can either use an Ansible playbook, or a bash file provided inside the guacamole folder. To install it using the Ansible playbook, simply run the following command:

$$ansible-playbook \quad --become \quad --ask-become-pass \quad --ask-pass \quad guacamole\_playbook.yaml$$

or you can use the bash file provided inside the **guacamoel** folder. To use this script, make sure that it is executable by running the following command:

$$sudo \quad chmod \quad +x \quad run\_guacamole\_playbook.sh$$

Then, execute the script by running the following command: **./run_guacamole_playbook.sh** in the terminal when you are inside the *guacamole* folder.

SECTION 3

Using service

After successfully installing the guacamole web service, you can access it by entering the IP address of the machine where Guacamole is installed, followed by the port number 8080. Upon accessing the web app, you will be prompted to log in with the default username and password, which are both **guacadmin**. It is crucial to create an admin user and remove the default user from the server to ensure security.

Additionally, it is essential to create a separate user for each student defined in the **playbooks.yaml** file and a corresponding connection for each user inside the guacamole web application. To learn more about creating users and connections in Guacamole, please refer to the official documentation at `https://guacamole.apache.org/doc/gug/using-guacamole.html`.

In our proposed application, the connections should be defined based on the information specified in the **playbooks.yaml** file. The port number for each connection should resemble the display number in the Guacamole web app. To define the port number when creating a connection, you need to add the display number to 5900 and insert the result in the port section. For instance, if a user has a display number of 1, which belongs to the user *5167230-104779* in Figure 4, the port number for that user will be 5901. To obtain the remote password for each user, refer to Figure 3, where you define the usernames and passwords for remote access. Therefore, you can configure each connection based on the user's display number and remote password, as demonstrated in Figure 6 for the user *5167230-104779*.

To ensure a secure and reliable connection to the guacamole web service, it is essential to carefully follow the steps outlined above for creating users and connections. Once the connections are established, each student can access their remote desktop by logging in using their credentials.

Figure 7 depicts the login page of the remote desktop, which has limited internet access and has

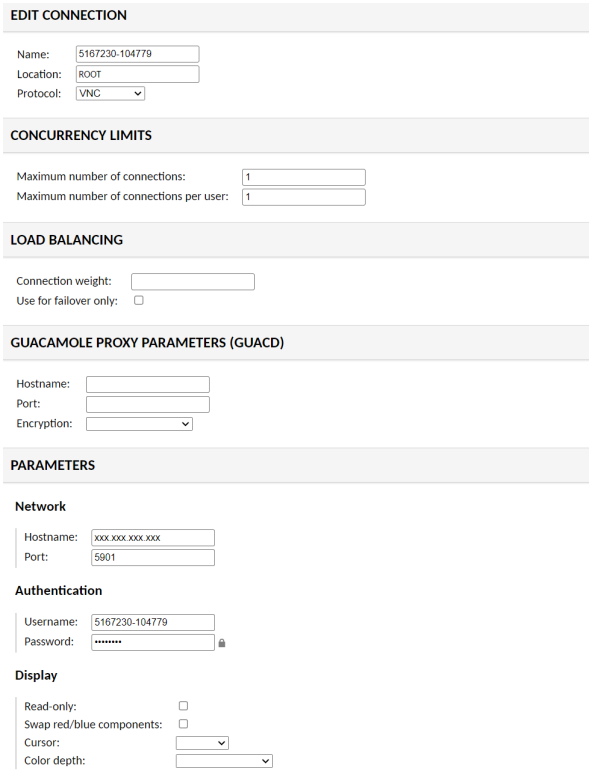

Figure 6: A sample for creating a connection inside Guacamole web application.

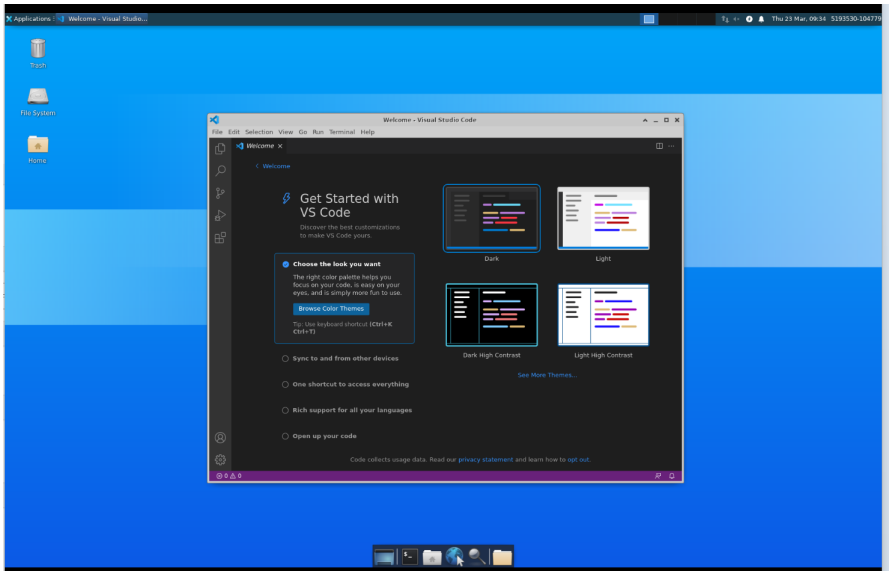

Figure 7: A screenshot of logged-in user in the application.

VS Code pre-installed and ready to use. To log out of the Guacamole service, each student should press "Ctrl+Alt+Shift" and then click on the "Logout" button. However, it is crucial to note that logging out from the Ubuntu server may cause login problems in the future, requiring service restart or other debugging methods. Therefore, students should only log out of the Guacamole service, not the Ubuntu server.

4

SECTION

Debugging

While using the Guacamole web service, users may encounter a few problems that can be overcome using the following methods.
If a user logs out of the Ubuntu desktop, they may not be able to log in again to the service. In such cases, the admin should log in to the server via SSH and restart the service related to that specific user using the following command:

$$systemctl \quad restart \quad rc-vnc<username>.service$$

Alternatively, the admin can run the hidden bash **.run_vncserve** file to restart the service.
If one needs to stop the service for a specific user, it can be done using the following command:

$$systemctl \quad stop \quad rc-vnc<username>.service$$

It is crucial to note that since both Guacamole and Daas are implemented as services, restarting the servers can solve most of the problems without losing any information. Therefore, the admin can restart the services instead of trying to troubleshoot individual user problems. By following these methods, the students can continue their remote desktop sessions without any interruptions.