Cimitra/GroupWise Implementation

This documentation will guide you all the way through the implementation of a Cimitra Server and the GroupWise Admin Helpdesk Scripts available for free on the GitHub distribution site.

Cimitra has a free version which is limited to 3 users and 3 Cimitra agents. You can implement everything explained in this documentation without any purchase. You must have a SUSE Linux SLES12 SP3 or better server. An OES 2018 SP1 server is SLES12 SP3 compatible.

Outside of download times, the entire process of getting a Cimitra Server implemented is about **5 minutes!** Defining the rest of the Cimtra system will take about 30 minutes.

Console-Based Installation of Docker and Cimitra

The installation portion of this documentation is written in such a manner that **every step is accomplished using console commands**. So all you need to do is determine a SUSE SLES 12 SP3 (or better) server that will act as your Cimitra Server, and open up a console prompt to that server and **copy/paste the commands given in this documentation** into the console session you have with the SLES Server you selected.

Install & Configure Docker Components

1. **Determine** which server will act as your Cimitra Server. Cimtira uses Docker. You will need a server that has 2 GB of memory and an additional 50 GB of disk space.
2. **Install Docker** with the **following command.**

```
yast -i docker
```

3. **Start the Docker daemon** and also **configure** the Docker daemon to **run automatically as a service** even after server reboot events with the following **2 commands**.

```
systemctl start docker.service
```

```
systemctl enable docker.service
```

4. **Install** the **Docker-Compose** utility. The Docker-Compose utility is not distributed with YAST, and must be installed from Docker's repository. Fortunately, doing the installation is very simple! Use the following **3 commands**.

Command # 1

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.25.3/
docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

Command # 2

```
sudo chmod +x /usr/local/bin/docker-compose
```

Command # 3

```
sudo ln -s /usr/local/bin/docker-compose
/usr/bin/docker-compose
```

Install & Configure Cimitra Server Components

5. **Create the Cimitra Server directory** that will contain the Cimitra Server configuration file. We will use the **/var/opt/cimitra/server** directory in this documentation. Use the **following command** which is actually two commands chained together.

```
mkdir -p /var/opt/cimitra/server ; cd
/var/opt/cimitra/server
```

6. **Download the Cimitra Server configuration file** which is a "Docker-Compose YAML file" (**Y**et **A**nother **M**arkup **L**anguage) called: **docker-compose.yml**. We will use the **curl command to** accomplish the **download:**

The Cimitra Server uses an internal Apache server which is using port 80 and 443 by default. If the SUSE Server you are installing the Cimitra Server to, already has an Apache web server, the Cimitra Server and the Apache web server will have a port conflict . . . if you use the defaults. We have provided you **2 different** docker-compose.yml **files to choose from**, **so if you anticipate a port conflict and you are fine using port 81 and 443, you do not need to edit anything. Just choose the right file to download.**

[Everything else explained below is for those who might want to make manual changes to the docker-compose.yml file.]

If you expect that you will have a conflict with the Apache Server already running on SUSE Server, but you do not want use ports 81 and 443 you can simply edit the Cimitra Server configuration file (docker-compose.yml) and edit the two lines associated with the Apache web server ports.

The syntax is:

- SUSE Server Port:Internal Cimitra Server Port

For example:

```
 - 80:80
 - 443:443
```

Only change the SUSE Server Port. Keep the Internal Cimitra Port at it's defaults.

For example:

```
 - 81:80
 - 444:443
```

Download docker-compose.yml file | Port 80 & 443 Version

```
curl https://app.cimitra.com/download/docker-compose.yml -o docker-compose.yml
```

or

Download docker-compose.yml file | Port 81 & 444 Version

```
curl -LJO https://raw.githubusercontent.com/cimitrasoftware/server_config_files/master/docker-compose.yml -o docker-compose.yml
```

7. **Download the Cimitra Server Docker Container**. The image is about 1.3 GB in size. Use the **curl command** to accomplish the **download**.

```
curl https://app.cimitra.com/download/cimitra-free.tgz -o cimitra-free.tgz
```

8. **Import the Cimitra Server** Docker Container with the **docker** command and the **import** directive.

```
docker import ./cimitra-free.tgz cimitra/free
```

9. **Start the Cimitra Server** from within the /var/opt/cimitra/server directory or wherever you downloaded the docker-compose.yml file into with the following command:

```
docker-compose up -d
```

Log Into the Cimitra Server

10. **Log into the Cimitra Server**'s HTTP login page which will be the address of your SUSE Linux Server. So for example, if the SUSE server is at **192.168.1.2** then use the web address

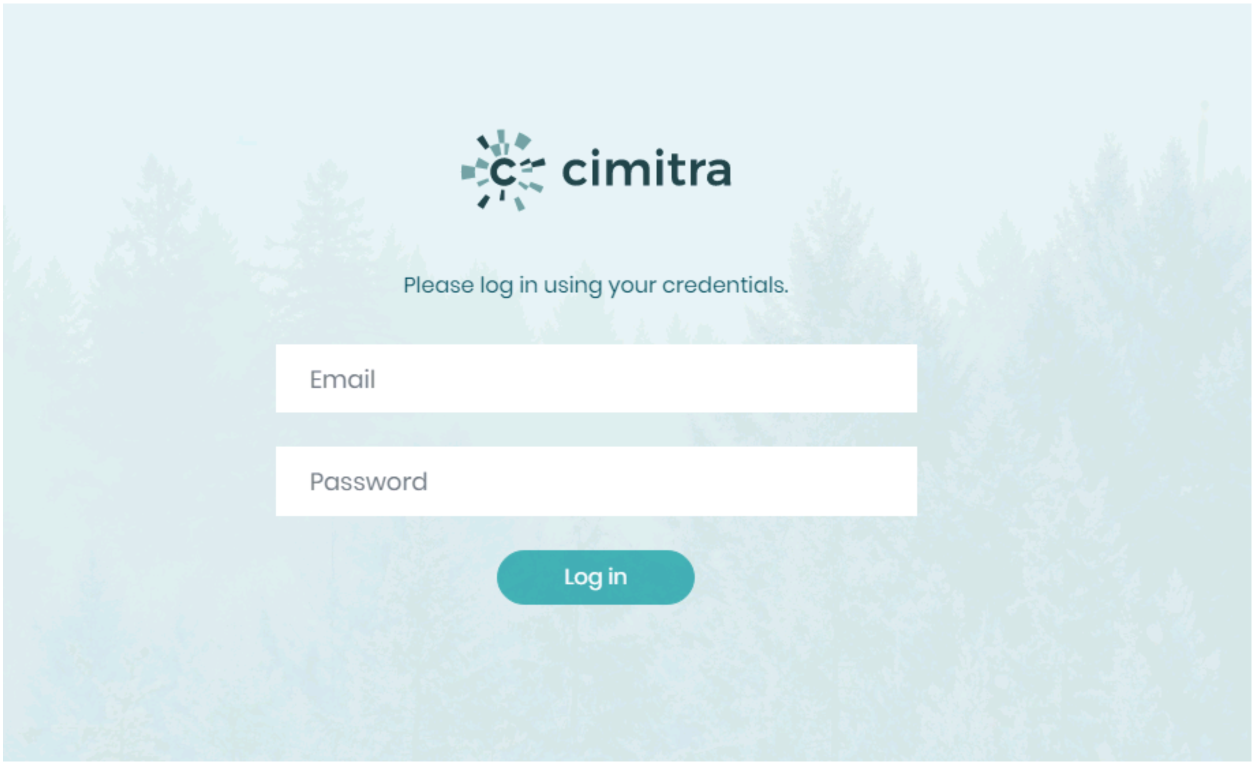
    https://192.168.1.2.

    Or if you used an alternative port such as 444 go to:

    https://192.168.1.2:444.

You should be presented with a screen similar to the one below. The **default credentials** are:

USER: admin@cimitra.com
PASSWORD: **changeme**



In a few moments we will configure things in the Cimitra Server, but first let’s install the GroupWise Admin Helpdesk Scripts by Cimitra.

Install & Configure the GroupWise Admin Helpdesk Scripts by Cimitra

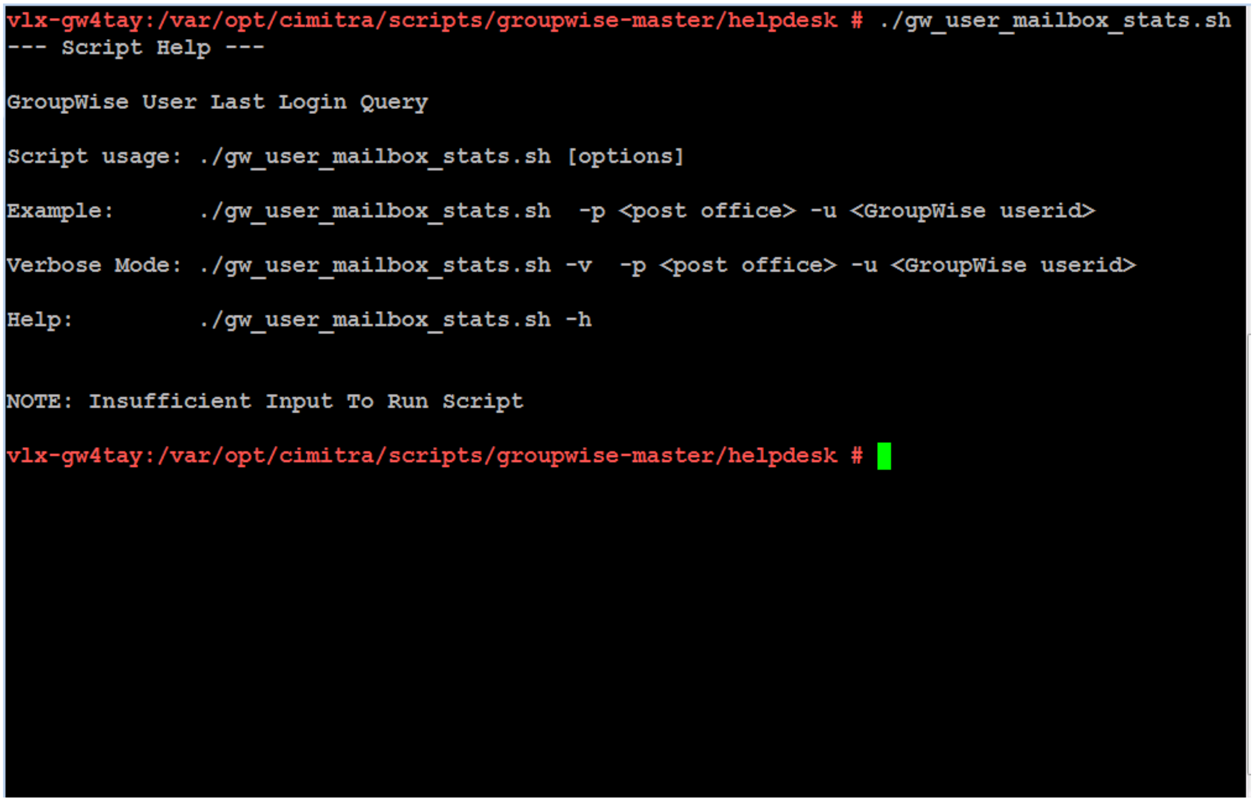
11. **Download and install** the GroupWise Admin Helpdesk Scripts by Cimitra. Use the **curl command** and a few other chained Bash commands as follows:

```
curl -LJO
https://raw.githubusercontent.com/cimitrasoftware/groupwise/
master/install -o ./ ; chmod +x ./install ; ./install
```

Follow the on-screen directions to **set up the configuration file** (`settings_gw.cfg`) that all of the scripts will use. The file is **located at:**

`/var/opt/cimitra/scripts/groupwise-master/helpdesk/settings_gw.cfg`

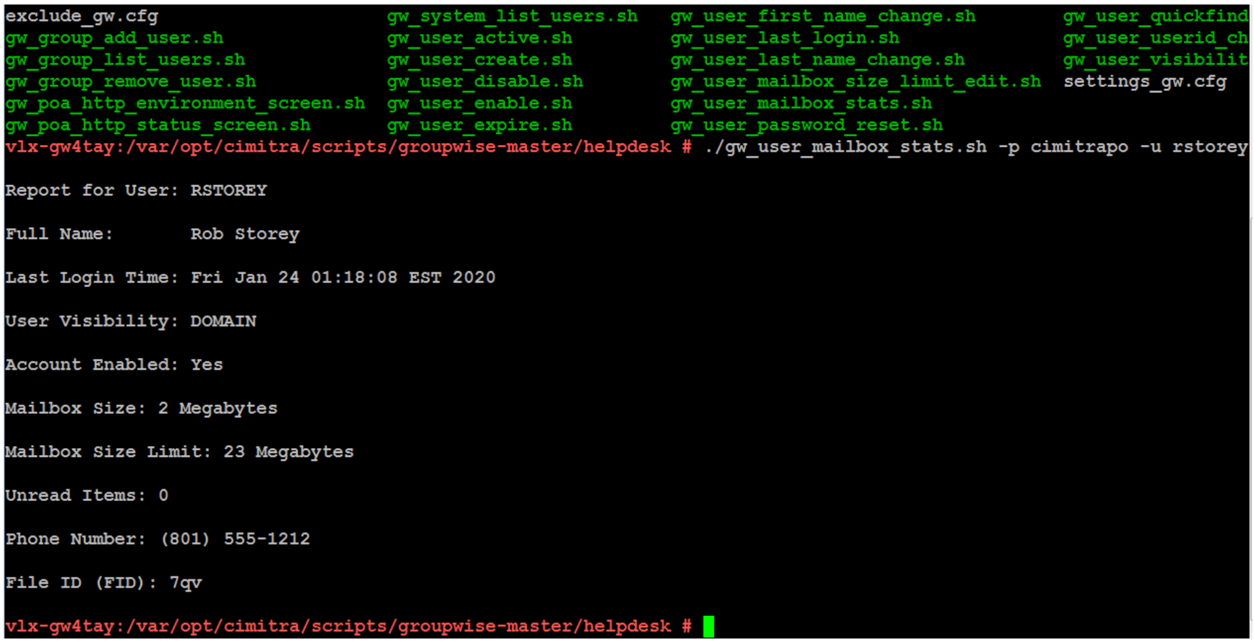
After configuration file is set up, **run one of the scripts** to get the help screen for that script.



```
vlx-gw4tay:/var/opt/cimitra/scripts/groupwise-master/helpdesk # ./gw_user_mailbox_stats.sh
--- Script Help ---

GroupWise User Last Login Query

Script usage: ./gw_user_mailbox_stats.sh [options]

Example:      ./gw_user_mailbox_stats.sh  -p <post office> -u <GroupWise userid>

Verbose Mode: ./gw_user_mailbox_stats.sh -v  -p <post office> -u <GroupWise userid>

Help:         ./gw_user_mailbox_stats.sh -h


NOTE: Insufficient Input To Run Script

vlx-gw4tay:/var/opt/cimitra/scripts/groupwise-master/helpdesk #
```

Then **run the script with the appropriate switches and parameters** to get the output from the script. For example the **User Mailbox Stats** script you would run in this manner:

```
./gw_user_mailbox_stats.sh -p cimitrapo -u rstorey
```



```
exclude_gw.cfg                    gw_system_list_users.sh   gw_user_first_name_change.sh         gw_user_quickfind
gw_group_add_user.sh              gw_user_active.sh         gw_user_last_login.sh                gw_user_userid_ch
gw_group_list_users.sh            gw_user_create.sh         gw_user_last_name_change.sh          gw_user_visibilit
gw_group_remove_user.sh           gw_user_disable.sh        gw_user_mailbox_size_limit_edit.sh   settings_gw.cfg
gw_poa_http_environment_screen.sh gw_user_enable.sh         gw_user_mailbox_stats.sh
gw_poa_http_status_screen.sh      gw_user_expire.sh         gw_user_password_reset.sh
vlx-gw4tay:/var/opt/cimitra/scripts/groupwise-master/helpdesk # ./gw_user_mailbox_stats.sh -p cimitrapo -u rstorey

Report for User: RSTOREY

Full Name:       Rob Storey

Last Login Time: Fri Jan 24 01:18:08 EST 2020

User Visibility: DOMAIN

Account Enabled: Yes

Mailbox Size: 2 Megabytes

Mailbox Size Limit: 23 Megabytes

Unread Items: 0

Phone Number: (801) 555-1212

File ID (FID): 7qv

vlx-gw4tay:/var/opt/cimitra/scripts/groupwise-master/helpdesk #
```

Cimitra System, Agents, Apps and User Creation

Here is a quick conceptuatl introduction to Cimitra. A “Cimitra System” consists of Cimitra **Agents**, Cimitra **Apps**, and Cimitra **Users**.

The fundamental purpose of Cimitra is to create the “buttons and wiring” to commands and scripts that are on servers and computers within an organization. As an example, the **GroupWise Admin Helpdesk Scripts by Cimitra** are scripts that run on a SUSE Linux server in your organization. Without Cimitra here are the information and steps you would need in order to run a script on a Linux server:

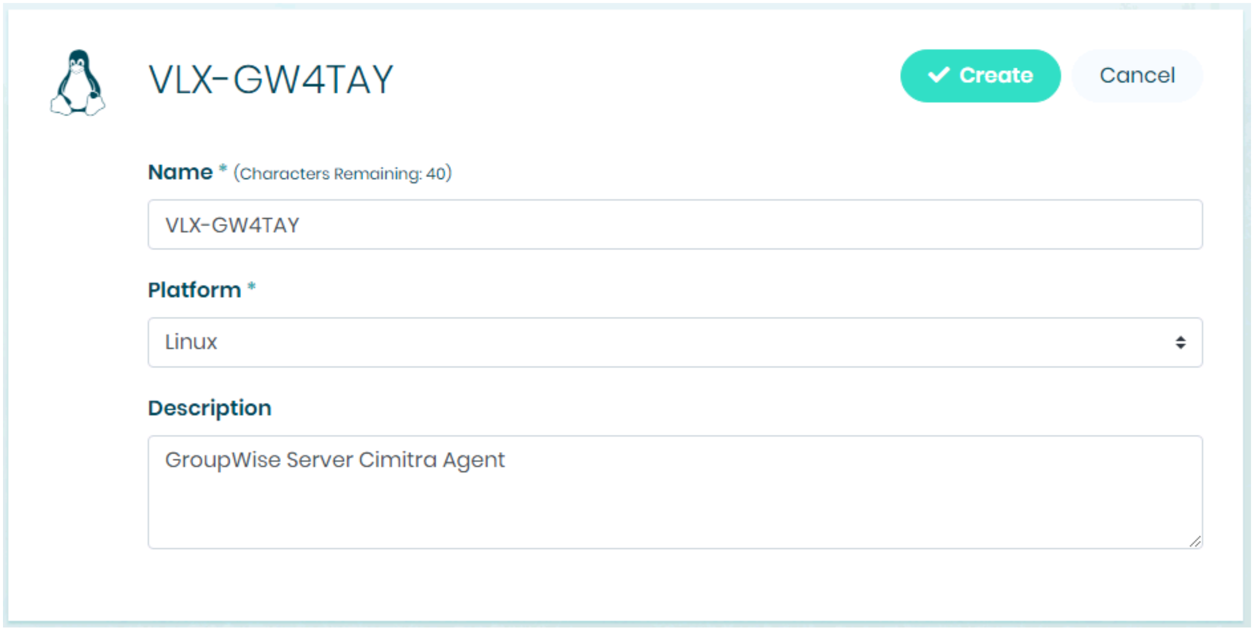
1. What server is the script located on?
2. What is the address of the server?
3. What is the tool to get to the server?
4. What are the credentials to get to the server?
5. Where are the scripts located on the SUSE server?
6. What are the inputs to the script?
7. What are the incorrect inputs to the script?
8. How do I give users access to those scripts without giving them the ability to do other things?

A Cimitra **Agent** is deployed to **any box** where you want **to run a command or script**. Every Cimitra **App** is an object that associates a command or script with a Cimitra Agent. Cimitra **Users** are created so you can safely delegate Cimitra Apps for them to run without all of the privileged access and specific knowledge needed to run a script.

Defining and Deploying a Cimitra Agent

In this step you will deploy a Cimitra Agent to the SUSE Server that has the **GroupWise Admin Helpdesk Scripts by Cimitra** already installed to it.

1. Log in as an Admin User in Cimitra.
    a. IMPORTANT: Make sure to change the Admin password to something **other** than **changeme**.
2. In Cimitra select **Agents | Add Agent**
    a. Giving the Agent a **name** something similar to the SUSE Linux Server that will house the agent is generally easiest.
    b. Set the **platform** to **Linux.**



Name * (Characters Remaining: 40)
VLX-GW4TAY

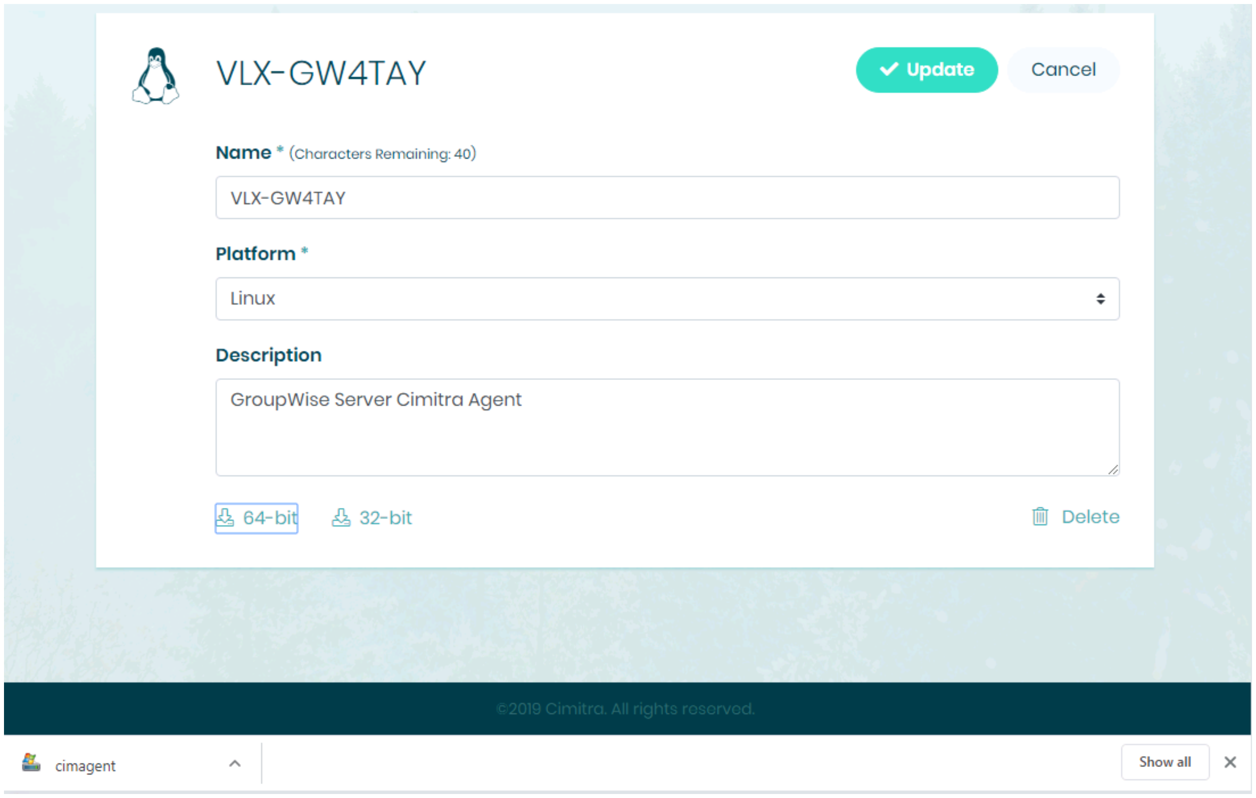
Platform *
Linux

Description
GroupWise Server Cimitra Agent

    c. Select the **Create** button.

3. Now select the **Cimitra Agent** that you just created and then select the **64-Bit Download** button which will download a Cimitra Agent binary program file. The "**cimagent**" file that you download contains within it:

    - The Cimitra Agent code
    - The Cimitra Agent installation code
    - The Cimitra Agent startup script
    - The Cimitra Agent configuration information



VLX-GW4TAY

Name * (Characters Remaining: 40)
VLX-GW4TAY

Platform *
Linux

Description
GroupWise Server Cimitra Agent

©2019 Cimitra. All rights reserved.

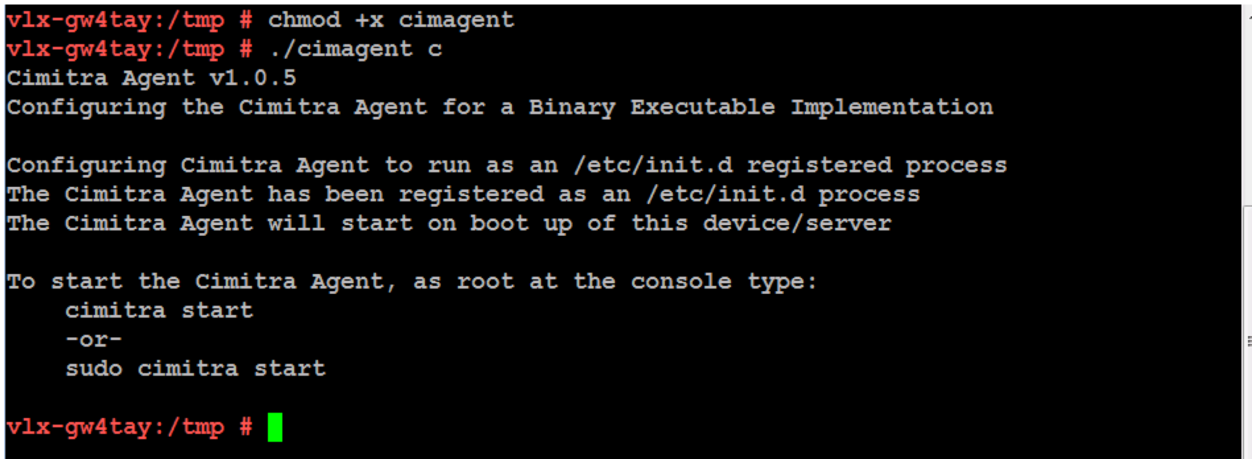
cimagent

4. Now you need to get the `cimagent` file that you just downloaded, over to the SUSE server where the **GroupWise Admin Helpdesk Scripts by Cimitra** are installed. You can put the `cimagent` file in any directory, as it will not install to that directory, it just needs to be run from someplace to kick of the installation routine.

5. Make the `cimagent` file **executable** and then **run** the `cimagent` file with the "**c**" parameter to **configure** the Cimitra Agent to install.

```
chmod -x ./cimagent
```

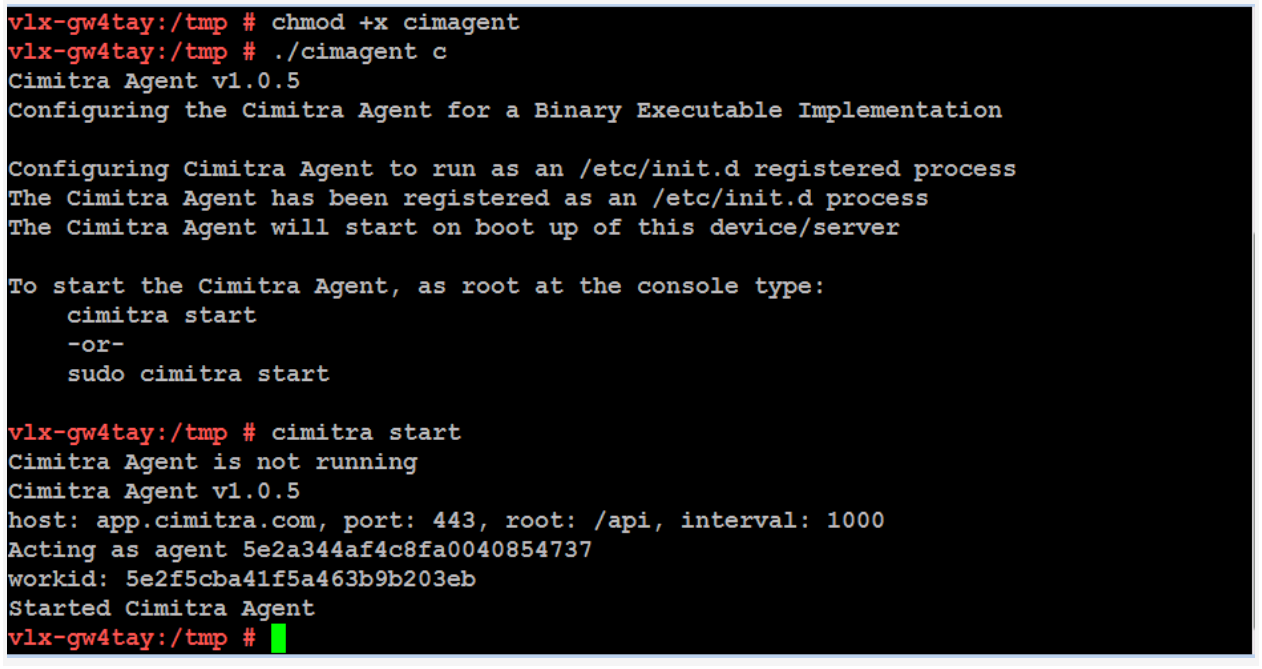
```
./cimagent c
```



```
vlx-gw4tay:/tmp # chmod +x cimagent
vlx-gw4tay:/tmp # ./cimagent c
Cimitra Agent v1.0.5
Configuring the Cimitra Agent for a Binary Executable Implementation

Configuring Cimitra Agent to run as an /etc/init.d registered process
The Cimitra Agent has been registered as an /etc/init.d process
The Cimitra Agent will start on boot up of this device/server

To start the Cimitra Agent, as root at the console type:
    cimitra start
    -or-
    sudo cimitra start

vlx-gw4tay:/tmp #
```

6. Start the Cimitra Agent with the **cimitra start** command which behind the curtain, is calling a Bash script to start the Cimitra Agent that you just installed.

```
cimitra start
```



```
vlx-gw4tay:/tmp # chmod +x cimagent
vlx-gw4tay:/tmp # ./cimagent c
Cimitra Agent v1.0.5
Configuring the Cimitra Agent for a Binary Executable Implementation

Configuring Cimitra Agent to run as an /etc/init.d registered process
The Cimitra Agent has been registered as an /etc/init.d process
The Cimitra Agent will start on boot up of this device/server

To start the Cimitra Agent, as root at the console type:
    cimitra start
    -or-
    sudo cimitra start

vlx-gw4tay:/tmp # cimitra start
Cimitra Agent is not running
Cimitra Agent v1.0.5
host: app.cimitra.com, port: 443, root: /api, interval: 1000
Acting as agent 5e2a344af4c8fa0040854737
workid: 5e2f5cba41f5a463b9b203eb
Started Cimitra Agent
vlx-gw4tay:/tmp #
```

The Cimitra Agent is now defined and deployed. The Cimitra Agent runs as a background service and so once you close out your session the Cimitra Agent will continue to run.

Defining a Cimitra App

Cimitra Apps are pointers to scripts. When you are defining a Cimitra App, you are trying to point the Cimitra Agent to the Cimitra App and giving it all of the information it needs to run the Cimitra App.

Generally it is best to put a Cimitra App into a "**Folder**" in Cimitra. Folders are helpful for organizing similar Cimitra **Apps**, and Cimitra **Folder**s are also necessary for **delegating/sharing Apps** with other Cimitra **Users**.

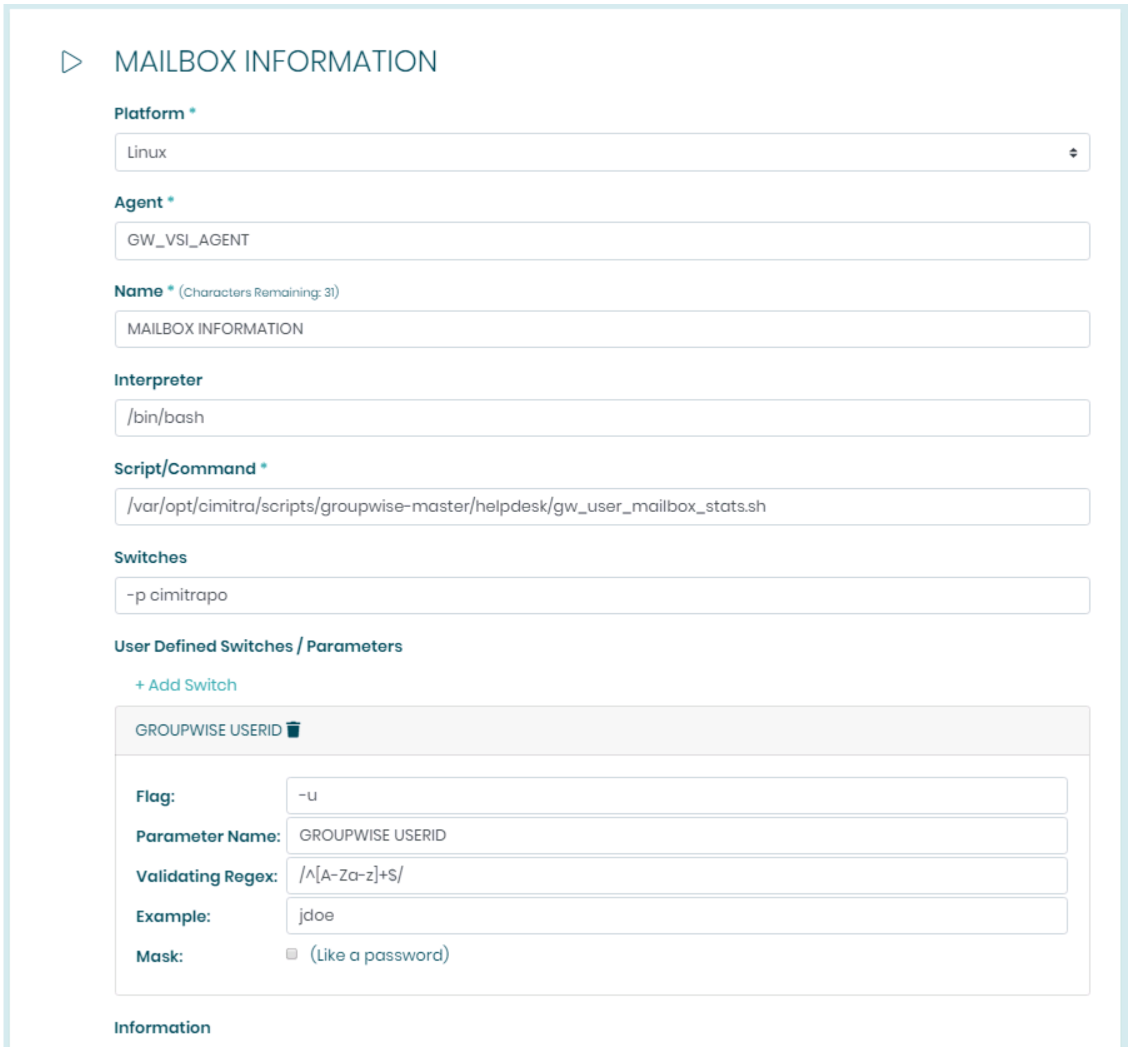
1. In the Cimitra web console, select **Create | Folder** to create a folder called **GROUPWISE**

2. Get in the GROUPWISE Folder and select **Create | App**

3. For the **App** we will add a very simple App first which is the **Mailbox Information** Cimitra **App** which will be a pointer to one of the **GroupWise Admin Helpdesk Scripts by Cimitra** called: **`gw_user_mailbox_stats.sh`**

   First off, it is best to run the script to get familiar with what inputs that it needs. The **`gw_user_mailbox_stats.sh`** script needs two inputs, the **post office** for the user, and the user's **userid**. These inputs should be proceeded with their command-line parameters: **`-p <user's post office>`** and **`-u <userid>`**.

NOTE: Here is how the App will look after we have filled in all the appropriate fields.



MAILBOX INFORMATION

Platform *
Linux

Agent *
GW_VSI_AGENT

Name * (Characters Remaining: 31)
MAILBOX INFORMATION

Interpreter
/bin/bash

Script/Command *
/var/opt/cimitra/scripts/groupwise-master/helpdesk/gw_user_mailbox_stats.sh

Switches
-p cimitrapo

User Defined Switches / Parameters

+ Add Switch

GROUPWISE USERID

Flag:	-u
Parameter Name:	GROUPWISE USERID
Validating Regex:	/^[A-Za-z]+$/
Example:	jdoe
Mask:	☐ (Like a password)

Information

4. Choose the **Platform | Linux**

5. Now choose the previously defined **Agent | <The Agent You Defined Earlier>**

6. Give it the **Name | Mailbox Information**

7. These are Bash scripts, so **Interpreter | /bin/bash**

8. Point to the **Script/Command |**

   /var/opt/cimitra/scripts/groupwise-master/helpdesk/gw_user_mailbox_stats.sh

9. **Switches | -p <post office >** so in our system it is: **-p cimitrapo**

10. Select **Add Switch** and fill it out as follows:

    a. Flag: **-u**
    b. Parameter Name: **GroupWise Userid**
    c. Validating Regex: **/^[A-Za-z]+$/**

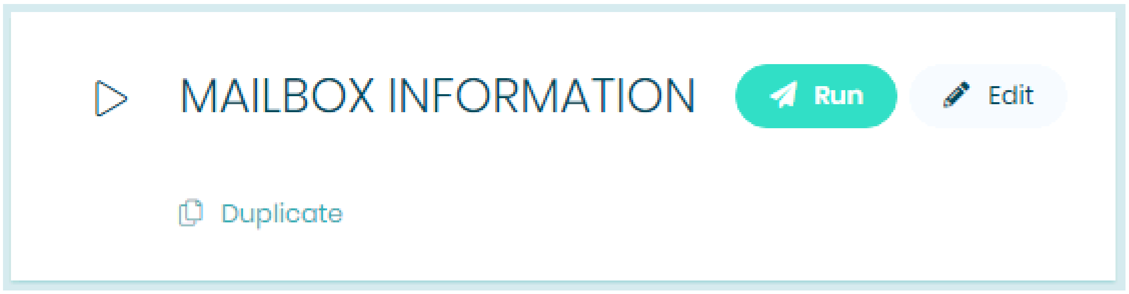
NOTE: Validating Regex is a very important feature. Make sure that when you give users the ability to input information, you should always employ the Validating Regex feature. If you do not, then a clever user could hack their way around your script and cause damage.

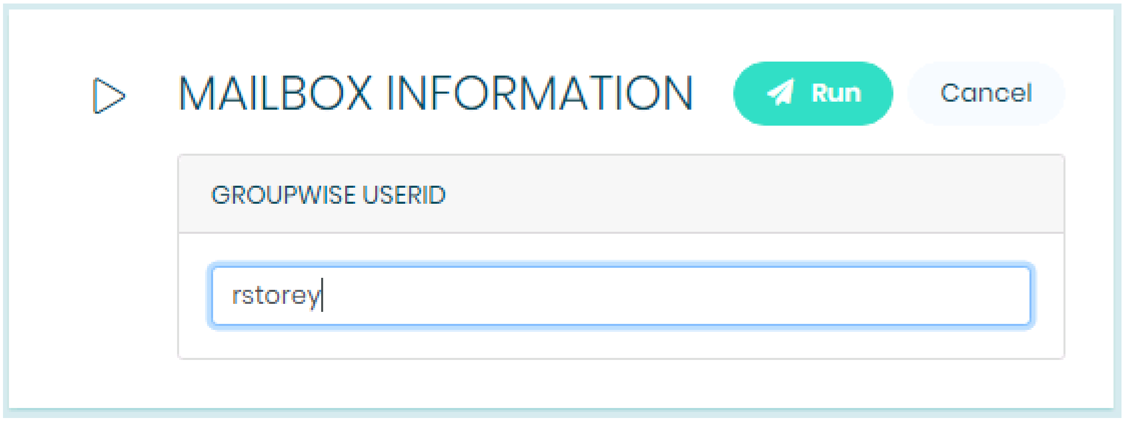
    d. Example: **jdoe**

11. Anything that you put in the **Information** field acts an **in-line help** when the Cimitra **App is run**.

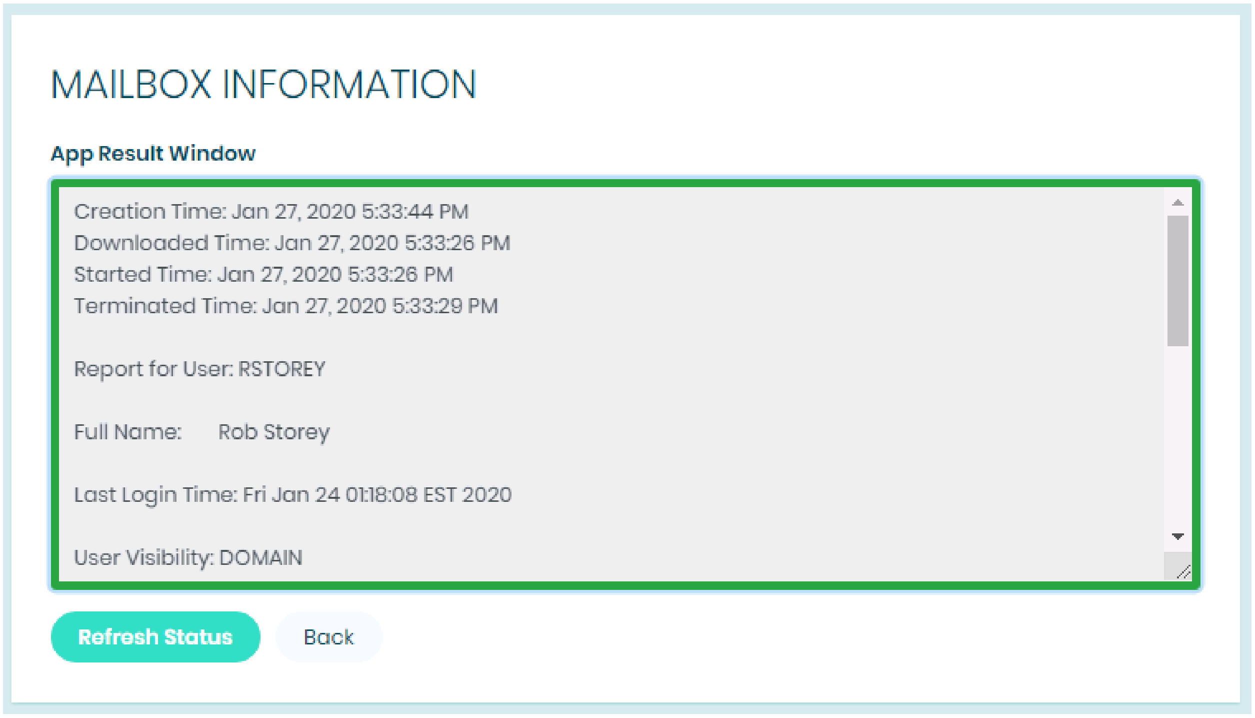
12. Select **Update** to save your newly defined Cimitra App.

13. Now you should **Run** the Cimitra App that you created to make sure everything works just fine.





rstorey



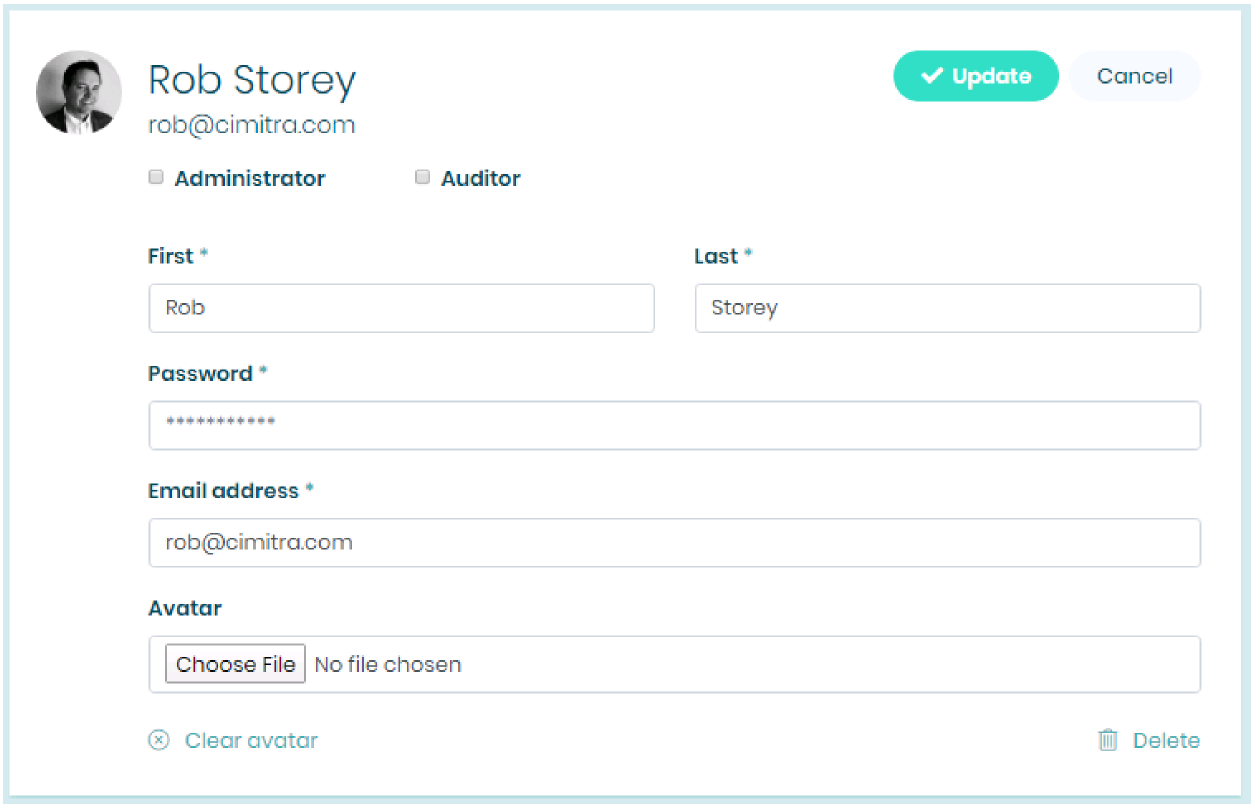
```
Creation Time: Jan 27, 2020 5:33:44 PM
Downloaded Time: Jan 27, 2020 5:33:26 PM
Started Time: Jan 27, 2020 5:33:26 PM
Terminated Time: Jan 27, 2020 5:33:29 PM

Report for User: RSTOREY

Full Name:     Rob Storey

Last Login Time: Fri Jan 24 01:18:08 EST 2020

User Visibility: DOMAIN
```

Safely Delegating Cimitra Apps to Users

In order to delegate Cimitra Apps to users, you first need to define a user or users, and then share the GROUPWISE folder with that user. Here are the steps to doing so in the Cimitra web console.

1. Select **Users**
2. Select **Add User**

3. Define all of the fields for the user. This user will not have Admin or Auditing rights. You can even upload a picture of the user, or some other kind of avatar.



Rob Storey
rob@cimitra.com

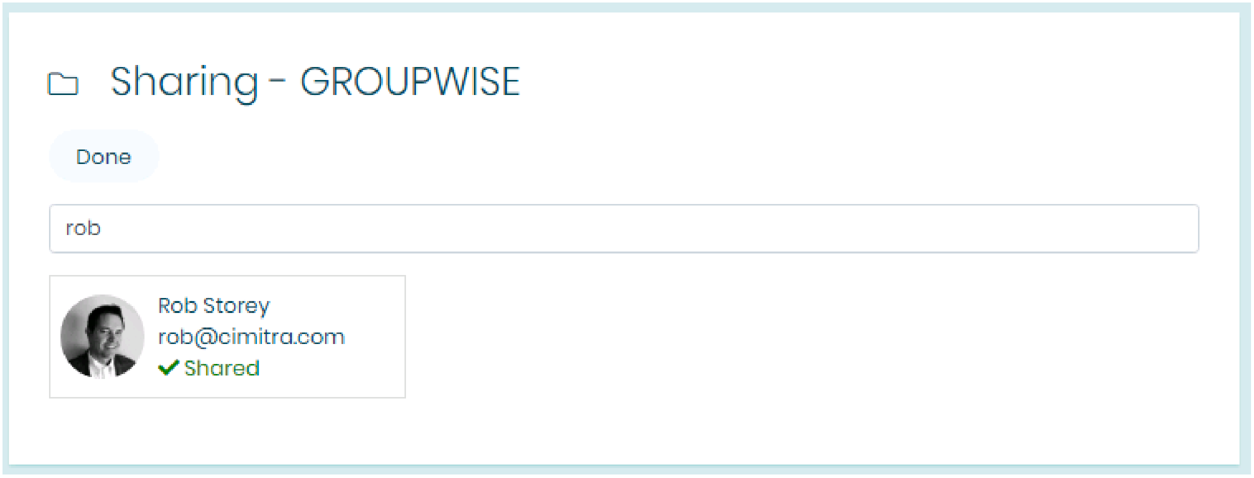
☐ Administrator ☐ Auditor

First *	Rob
Last *	Storey
Password *	***********
Email address *	rob@cimitra.com
Avatar	Choose File No file chosen

Now that a user is created, you can **Share** the App with the user by doing the following.

4. Go back to the **GROUPWISE folder** that you created **within Cimitra**, and click the **pencil icon** to the right of the folder name to **edit** the folder.

5. Select the **Sharing** option, **Add Share**, and choose the Cimitra **User** you created to share this Cimitra **Folder** with. A green checkmark should appear next to their name to indicate that the Cimitra Folder is shared with them.


Sharing - GROUPWISE

Done

rob

Rob Storey
rob@cimitra.com
✓ Shared

6. Now you should log in as the Cimitra User that you shared the Cimitra Folder with so that you see the end-user/non-admin user experience.

Defining Additional Cimitra Apps

Now that you know the basics, you should define all of the Cimitra Apps that you would like to delegate to others.

A lot of the fields related to a Cimitra App are easy to define. The most difficult aspect of Cimitra is understanding the "Validating Regex" string you should use to assure that the user is only able to input valid characters that you feel comfortable passing into a script. It cannot be emphasized enough how **IMPORTANT** **Validating Regex** is. If you were to keep the Validating Regex field blank, you could be opening yourself up to hacking that would make your system vulnerable.

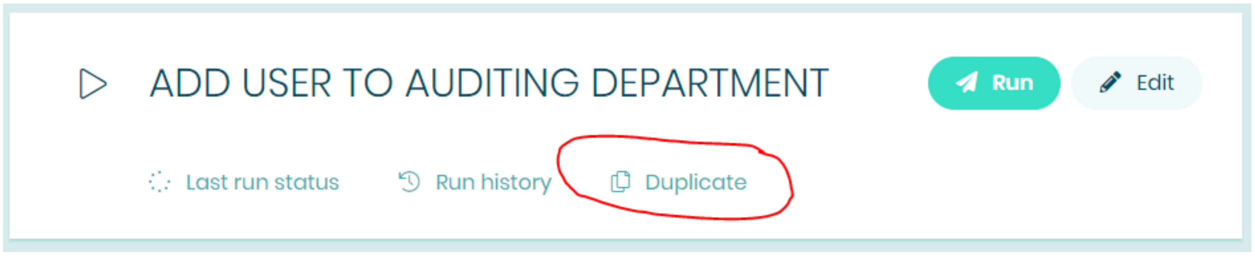
Google is your friend when defining regex strings. Here are some examples of regex strings you might use:

Description	**Regex String**
Letters only, uppercase and lowercase	**/^[A-Za-z]+$/**
Numbers and dashes only	**/^[0-9-]+$/**
Numbers only	**/^[0-9]+$/**
Numbers and forward slashes only	**/^[0-9\/]+$/**
Letters, dashes and spaces only	**/^[a-zA-Z-'\s]+$/**

Numbers, letters, underscores, plus symbol, dashes, equals symbols	**/^[0-9A-Za-z_+-=]+$/**

Using the **Duplicate** Option to Create More Cimitra Apps

The easiest way to create a new Cimitra App is that when you have created one Cimitra App, just **duplicate it**, change the name of the App, and the script name and parameters, etc. as needed. This can speed up the process of creating Cimtra Apps for several scripts and commands that are on the same server.



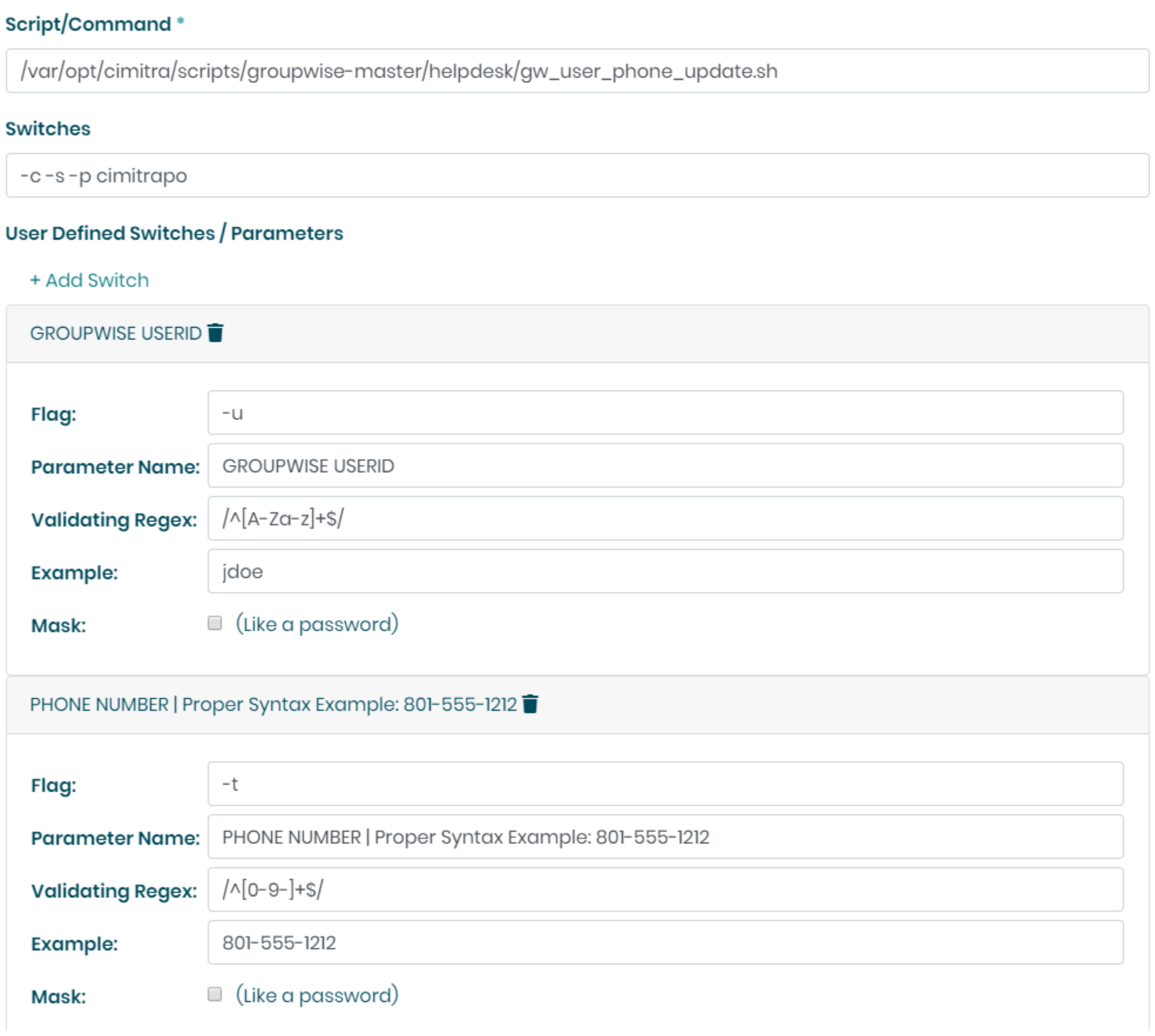
Defining Multiple Input Fields

Cimitra allows you to define as many input fields as you would like. For the user's sake you want to give them the least amount of fields to enter data into. So make sure to put as many parameters on the **Switches** field as possible so that you are only prompting for the things the end-user is actually needing to give input on.

Here is an example. There is a script amongst the **GroupWise Admin Helpdesk Scripts by Cimitra** called:

`gw_user_phone_update.sh`

This User Phone Update script has some command line parameters that are optional. That you may want to use. For example, if you use the **-c** switch the phone number that is entered is **converted** from the entry syntax of 801-555-1212 to (801)555-1212 when the script saves the phone number into GroupWise. If you also use the **-s** switch a space is added between the right parenthesis and the rest of the phone number, like this: (801) 555-1212. So the **-c** and the **-s** switches should also be added to the **Switches** field. See the screenshot below to see how this is done.


Script/Command *

/var/opt/cimitra/scripts/groupwise-master/helpdesk/gw_user_phone_update.sh

Switches

-c -s -p cimitrapo

User Defined Switches / Parameters

+ Add Switch

GROUPWISE USERID

Flag:	-u
Parameter Name:	GROUPWISE USERID
Validating Regex:	/^[A-Za-z]+$/
Example:	jdoe
Mask:	☐ (Like a password)

PHONE NUMBER | Proper Syntax Example: 801-555-1212

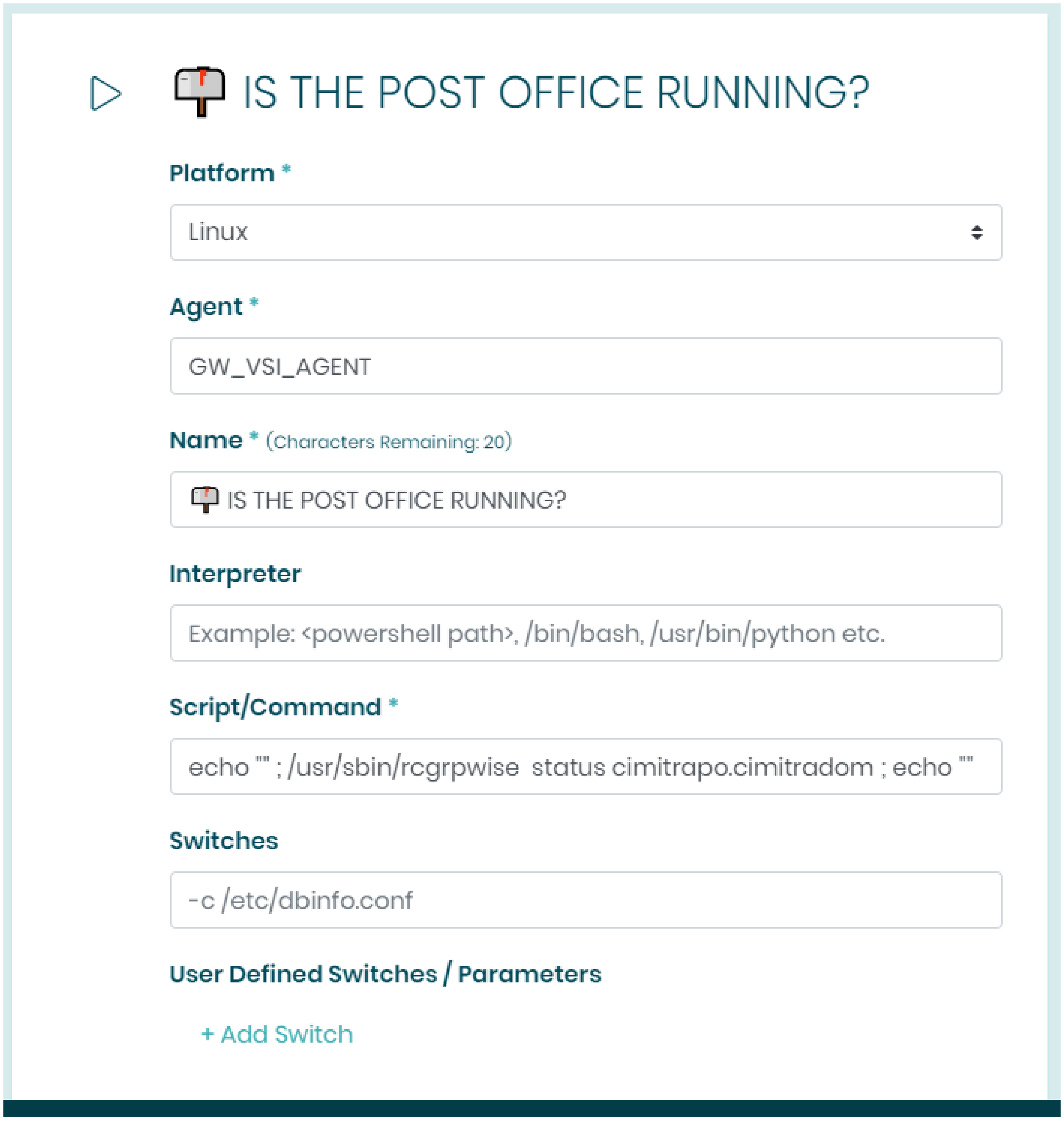
Flag:	-t
Parameter Name:	PHONE NUMBER | Proper Syntax Example: 801-555-1212
Validating Regex:	/^[0-9-]+$/
Example:	801-555-1212
Mask:	☐ (Like a password)

Using Commands That Ship With GroupWise

GroupWise ships with a few commands and scripts. For example, the **grpwise command**. This command is actually mapped to the command: **/usr/sbin/rcgrpwise**

In to invoke the **grpwise** command, you must have a Cimitra Agent deployed to that server. **Or** you can implement a Cimitra Agent on one server and then get access to run the grpwise commands using SSH keys. For more information on how to do that **[CLICK HERE]** This is particularly helpful in scenarios in which you might have portions of your GroupWise system running on SLES 11 boxes, because the Cimitra Agent is only compatible with SUSE SLES 12 and higher boxes. FYI, the Cimitra Agent for Linux is also compatible with many other flavors of Linux.

To make a Cimitra App to use the **grpwise** command is easy. Here is a screen shot, and the fields are explained below:



📫 IS THE POST OFFICE RUNNING?

Platform *

Linux

Agent *

GW_VSI_AGENT

Name * (Characters Remaining: 20)

📫 IS THE POST OFFICE RUNNING?

Interpreter

Example: <powershell path>, /bin/bash, /usr/bin/python etc.

Script/Command *

```
echo "" ; /usr/sbin/rcgrpwise  status cimitrapo.cimitradom ; echo ""
```

Switches

-c /etc/dbinfo.conf

User Defined Switches / Parameters

+ Add Switch

Name Field

The name field supports text and any emoticons you can get off of the Internet. So we just copied and pasted an emoticon that we found on the Internet: 📫

Just be aware that different browsers and operating systems interpret emoticons differently. So what you enter and what a user sees might be different with emoticons.

Interpreter Field

The grpwise command is registered in such a manner that is similar to an operating system command. So there is no interpreter needed. Keep the **Interpreter field blank**.

Script/Command Field

We are using a method of executing a command called chaining So the Script/Command field in this example is actually 3 commands. They are:

```
echo ""
```

```
/usr/sbin/grpwise status cimitrapo.cimitradom
```

```
echo ""
```

And we are chaining the commands using the semicolon character ; in the following manner:

```
echo "" ; /usr/sbin/rcgrpwise status cimitrapo.cimitradom ;
echo ""
```

The purpose behind the `echo ""` commands is to give a little bit more space to the output so that it looks a little more pleasing to the eye when the Cimitra App is run.

Defining Cimitra Link Objects

Cimitra has a very simple and handy object. It’s called a “Link”. A Cimitra Link object is just an HTTP bookmark object. Cimitra Link objects are very helpful when you want to share things such as Wiki articles, documentation, URLs, or HTTP consoles with other users. Cimitra Link objects have to be shared via a Cimitra Folder just like Cimitra Apps are shared. And only Administrators can share Cimitra Folders, so non-Admins cannot share Cimitra Folders with other people. But non-Admins can create their own Cimitra Folders and their own Cimitra Links for their own benefit.

Conclusion

Setting up the first few Cimitra Apps can make you feel like you are in some strange wilderness. But after you get the hang of it, you will be making CImitra Apps in a matter of a few minutes. You will find all kinds of commands and scripts that you will connect to Cimtra Apps for both your benefit as an administrator, and to allow you to delegate your routine and even mundane tasks to a larger community of IT problem solvers.