Linux Fundamentals - Hands-on

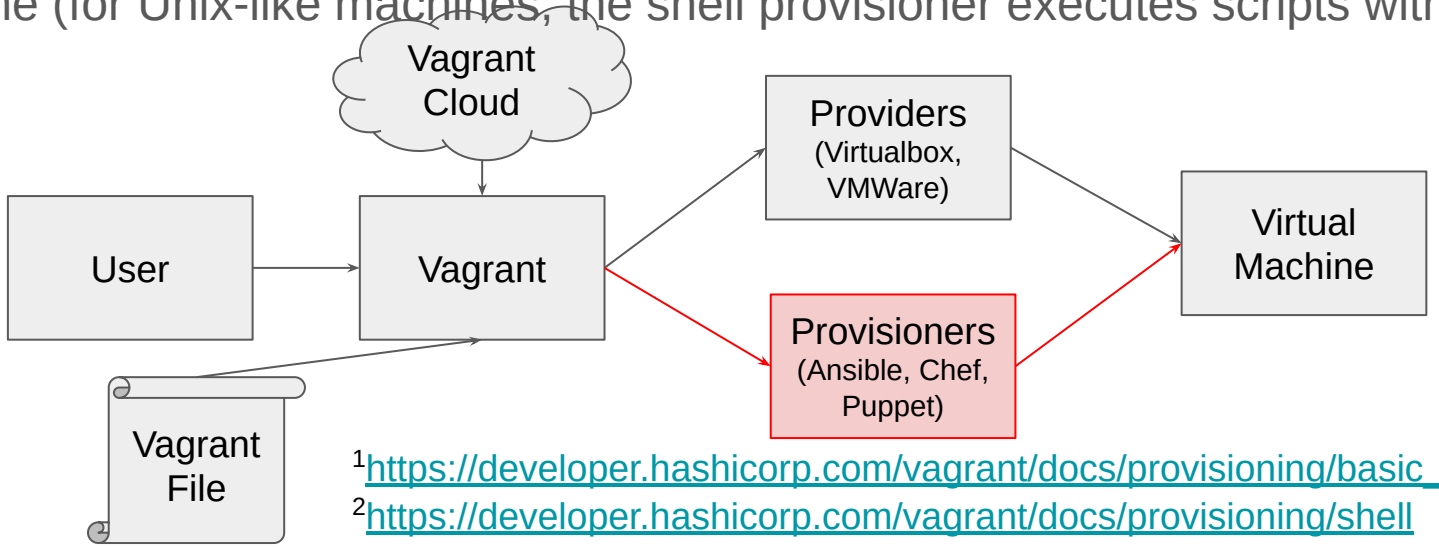
Virtualization and Cloud Computing - ay 2023/2024

Enrico RUSSO <enrico.russo@dibris.unige.it>
Giacomo LONGO <giacomo.longo@dibris.unige.it>

Vagrant: Provisioners and Shell provisioner

Provisioners[1] in Vagrant allow you to automatically install software, alter configurations, and more on the machine as part of the vagrant up process.

The Vagrant Shell provisioner[2] **upload** and **execute** a **script** within the guest machine (for Unix-like machines, the shell provisioner executes scripts with SSH)



[1] https://developer.hashicorp.com/vagrant/docs/provisioning/basic_usage
[2] https://developer.hashicorp.com/vagrant/docs/provisioning/shell

Shell provisioner (example with variables)

```
# Vagrantfile
Vagrant.configure("2") do |config|
 config.vm.box = "generic/ubuntu2204"

 # [...]

 var1 = "This is var1"
 var2 = "This is var2"

 # by default script is privileged

 config.vm.provision :shell, :path =>
'setup.sh', :args => [var1, var2],
privileged: false
end
```

```
# setup.sh

#!/bin/bash

# $1 -> var1
# $2 -> var2

echo "var1 is $1" > /tmp/result.txt
echo "var2 is $2" >> /tmp/result.txt
```

Vagrant: File provisioner

The Vagrant file provisioner allows you to upload a file or directory from the host machine to the guest machine.

```
### configuration parameters ###
SRCFILE = "C:\\users\\enrico\\Documents\\test.txt"

# Vagrantfile
#
Vagrant.configure("2") do |config|
 config.vm.box = "generic/ubuntu2204"

 # [...]

 # copy file specified in SRCFILE to the home directory, renaming it with [destination] filename
 config.vm.provision "key", type: "file", source: KEY_FILE, destination: "/tmp/id.pub"
end
```

Vagrant: running provisioners

- Provisioners are run in three cases: the initial `vagrant up`, `vagrant provision`, and `vagrant reload --provision`
- A `--no-provision` flag can be passed to up and reload if you do not want to run provisioners
- The `--provision-with` flag can be used if you only want to run a specific provisioner
- Multiple provisioners will be run in the order they are defined

Exercise 1

Configure Vagrant to automatically enable SSH access using your personal key
generate ssh key on the host machine (or use your personal one)

- generate ssh key on the host machine (or use your personal one)
- use a file provisioner to copy the **public key** on /tmp/id.pub
- use a script provisioner (setup_pubkey.sh) to append the above key on the authorized_keys of the user vagrant
    - use a configuration variable KEY_FILE to specify the file containing the public key
    - check if the key already exists in the authorized_keys before adding it
- test whether it works with an ssh client and vscode

Exercise 2: install a LAMP Stack[1]

- A “LAMP” stack is a group of open source software that is typically installed together in order to enable a server to host dynamic websites and web apps written in PHP
- This term is an acronym which represents
  - the Linux operating system with the Apache web server
  - the site data is stored in a MySQL database
  - dynamic content is processed by PHP
- it requires the following packages: `apache2`, `mysql-server`, `php`, `libapache2-mod-php`, `php-mysql` (installation must be **NON interactive**)
- test by browsing `http://[ip_address]`

[1] https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-22-04

LAMP

- configuration file of Apache are in `/etc/apache2`
- the default directory for the web pages is `/var/www/html`
- mysql can be managed as an administrator (root) through the user mysql: `sudo mysql`
- a query can be executed from the command line with sudo `mysql -e "[query]"` (e.g., `mysql -e "select 1"`)
- test the PHP engine by adding a page `info.php`

```
<?php

phpinfo();
```

Exercise 3: create a virtual host [1]

Virtual hosts (VHs) allow an Apache server to host more than one domain from a single server.

- add a local domain `test.local` pointing to the GUEST IP address in the `hosts` file of the HOST
- create a provision script that [1]
  - uses a `DOMAIN` variable for specifying the domain name handled by the VH
  - creates a directory `/var/www-[nameofthedomain]`
  - changes the owner in `vagrant.vagrant`

* Windows: `C:\Windows\System32\drivers\etc\hosts`

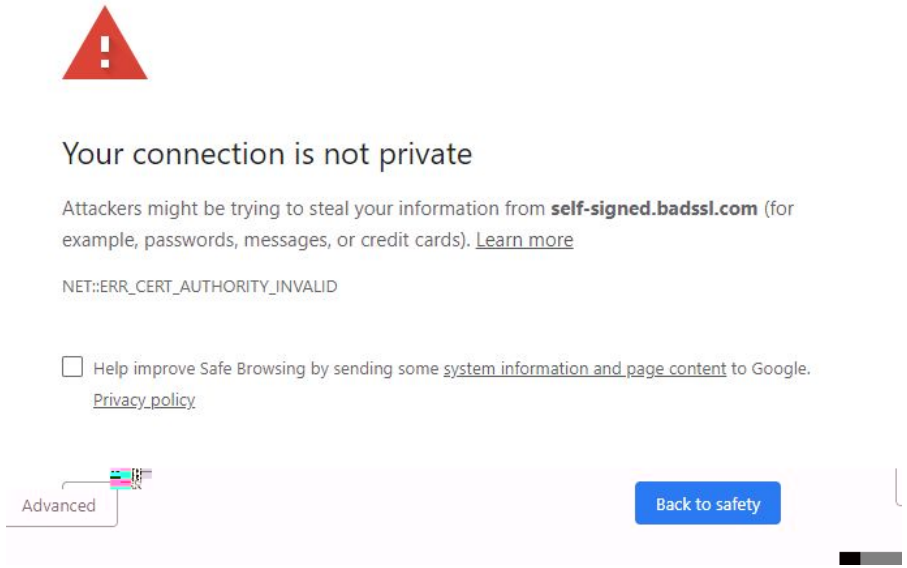
Exercise 3: create a virtual host [2]

- create a provision script that [2]
  - copy the local file `vh.conf` in `/etc/apache2/sites-available/DOMAIN.conf`
  - update the `DOMAIN.conf` by substituting the token `DOMAIN` with the domain name for the VH
    - `sed 's/DOMAIN/[domain name]/' vh.conf` is a possible solution
  - enable the vh: `a2ensite DOMAIN`
  - reload Apache: `apache2ctl restart`
- test by creating a index page in the web root and opening `http://DOMAIN`

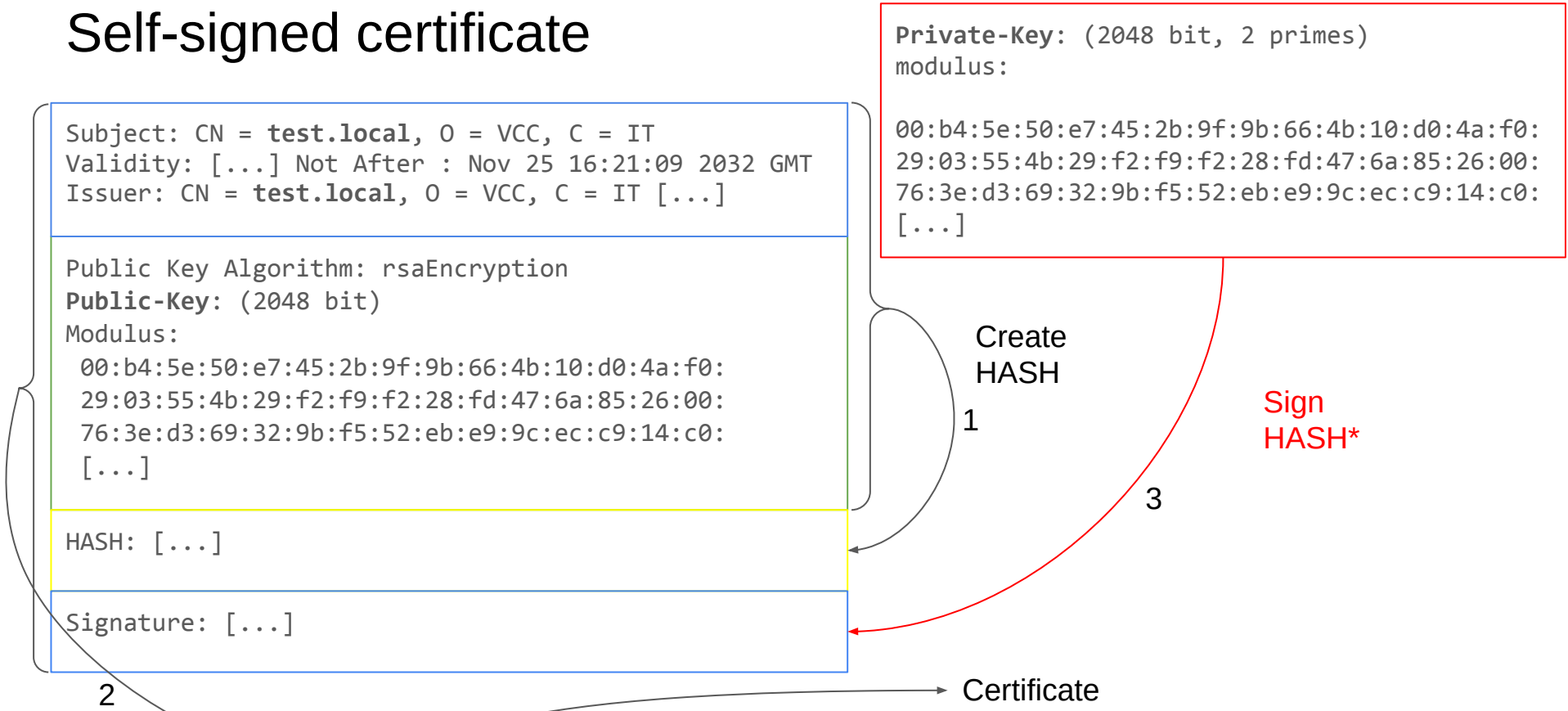
```
# vh.conf

<VirtualHost *:80>
    ServerName DOMAIN
    ServerAlias www.DOMAIN

    ServerAdmin webmaster@DOMAIN
    DocumentRoot /var/www-DOMAIN

    <Directory /var/www-DOMAIN>
        AllowOverride All
        Options -Indexes
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/DOMAIN-error.log
    CustomLog ${APACHE_LOG_DIR}/DOMAIN-access.log
combined
</VirtualHost>
```

Exercise 4: HTTPS virtual host

- We need to configure Apache to secure the browsing of web pages by supporting TLS
- It requires a digital certificate
- We generate one locally, i.e., a self-signed certificate
- Self-signed certificates are **not trusted** by browsers



Self-signed certificate



*for a **trusted** certificate we require an Issuer ID and the signature made with the private key of a **trusted** Certification Authority (trusted CAs' certificates are **preloaded** in the browser, e.g., chrome://settings/security > Certificates Managed by Chrome)

Self-signed certificate: openssl

server.crt

```
Subject: CN = test.local, O = VCC, C = IT
Validity: [...] Not After : Nov 25 16:21:09 2032 GMT
Issuer: CN = test.local, O = VCC, C = IT [...]

Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit)
Modulus:
 00:b4:5e:50:e7:45:2b:9f:9b:66:4b:10:d0:4a:f0:
 29:03:55:4b:29:f2:f9:f2:28:fd:47:6a:85:26:00:
 76:3e:d3:69:32:9b:f5:52:eb:e9:9c:ec:c9:14:c0:
 [...]

HASH: [...]

Signature: [...]
```

server.key

```
Private-Key: (2048 bit, 2 primes)
modulus:

00:b4:5e:50:e7:45:2b:9f:9b:66:4b:10:d0:4a:f0:
29:03:55:4b:29:f2:f9:f2:28:fd:47:6a:85:26:00:
76:3e:d3:69:32:9b:f5:52:eb:e9:9c:ec:c9:14:c0:
[...]
```

```
openssl req -subj '/CN=test.local/O=VCC/C=IT' -new -newkey rsa:2048 -sha256 -days 3650
-nodes -x509 -keyout server.key -out server.crt
```

Exercise 4: HTTPS virtual host

- generate a self-signed certificate:
  - private key is stored in `/etc/ssl/private/DOMAIN.key`, owned by `root.ssl-cert` and `u+rw`, `g+r` only (`640`)
  - public key is stored in `/etc/ssl/certs/DOMAIN.pem`, owned by `root.root` and `u+rw`, `go+r` only (`644`)
- create a provision script that [2]
  - copy the local file `vh-ssl.conf` in `/etc/apache2/sites-available/DOMAIN-ssl.conf`
  - update the `DOMAIN.conf` by substituting the token `DOMAIN` with the domain name for the VH
  - enable Apache SSL module: `a2enmod ssl`
  - enable the vh: `a2ensite DOMAIN-ssl`
  - reload Apache: `apache2ctl restart`
- test by opening `https://DOMAIN`

```
# vh-ssl.conf

<VirtualHost *:443>
    ServerName DOMAIN
    ServerAlias www.DOMAIN

    ServerAdmin webmaster@DOMAIN
    DocumentRoot /var/www-DOMAIN

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/DOMAIN.pem
    SSLCertificateKeyFile /etc/ssl/private/DOMAIN.key

    <Directory /var/www-DOMAIN>
        AllowOverride All
        Options -Indexes
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/DOMAIN-ssl-error.log
    CustomLog ${APACHE_LOG_DIR}/DOMAIN-ssl-access.log combined
</VirtualHost>
```

Exercise 5: WordPress[1]

WordPress is a popular, open-source **content management system** (CMS) that allows users to create, customize, and manage content on their website. It requires a LAMP stack and

- a dedicated MySQL database
- a list of additional PHP extensions
- a list of enabled Apache modules
- a configuration file (`wp-config.php`)

[1] https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-on-ubuntu-22-04-with-a-lamp-stack

Exercise 5a: create database and user for WordPress

Use the following queries (check if the DB already exists before executing them). `DBNAME`, `DBUSER`, `DBPASS` are configuration parameters.

- `CREATE DATABASE [DBNAME] DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;`
- `CREATE USER '[DBUSER]'@'%' IDENTIFIED WITH mysql_native_password BY '[DBPASS]';`
- `GRANT ALL ON [DBNAME].* TO '[DBUSER]'@'%';`
- `FLUSH PRIVILEGES;`
- test if the connection works before continuing the execution of the script

Exercise 5b: PHP/Apache and add WordPress sources

- add PHP modules (using packages): `php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip`
- enable Apache modules: `rewrite`
- (check if `AllowOverride All` is enabled in the VH); reload Apache
- download latest version of WordPress in /tmp: `curl -O` https://wordpress.org/latest.tar.gz
- extract the archive in the VH directory (strip the first directory) and delete the archive
- change the ownerships of the extracted directories and files
  - `chown -R www-data:www-data /var/www-DOMAIN`
  - `find /var/www/www-DOMAIN/ -type d -exec chmod 750 {} \;`
  - `find /var/www/www-DOMAIN/ -type f -exec chmod 640 {} \;`

Exercise 5c: generate a config file for WordPress

- copy the sample file (`wp-config-sample.php`) in `wp-config.php`
- change the values `database_name_here`, `username_here`, and `password_here` with the corresponding values `DBNAME`, `DBUSER`, and `DBPASS`
  - `sed -i "s/database_name_here/$DBNAME/" wp-config.php` is a possible solution
- generate a random phrase: `openssl rand -base64 32`
- change the values `put your unique phrase here` with the random phrase
  - `sed -i "s:put your unique phrase here:[random phrase]:" wp-config.php` is a possible solution
- completing the installation through the web interface: https://test.local