Getting ready for the Cloud

One major change, compared to the previous semester, is that everything we do, involving client-server "stuff", must be hosted. We will develop and test locally on localhost as usually, but for all CA's, and several exercises your projects are required to be hosted for real - like in the real world :-)

There are "tons" of Cloud Providers out there, but this semester we have decided for DigitalOcean (https://www.digitalocean.com/) which you should already know from 2. semester. For our purpose DigitalOcean offer what is known as an *iaas* platform (Infrastructure as a Service) which basically means that all we get is an empty fresh (Linux) server. Everything we need (Java, Tomcat, MySQL etc.) must be installed on this server before we can use it. This is complex and cumbersome compared to other strategies like paas (Platform as a Service), but while doing all this you will learn a lot, and once done your server is ready for all your projects. Also with this strategy, we can get whatever server setup we like.

Note: this is an individual exercise (it's YOUR server that must be set up), but do it together with someone else, so you can help each other if something goes wrong.

Getting started

DigitalOcean has some pretty OK descriptions to get you started (which we will use). The major problem for most of us however, will be that we are using a Window OS and many of you are probably not used to work in a terminal (which you will learn).

Windows does not provide an SSH client, which is required for the communication with a Linux server. The following will provide you with the necessary steps to generate the required SSH keys (to login without a password) and show you how you can use GIT Bash as you're SSH client (On Mac OS you have a Unix terminal and a SSH client).

Git Bash as a SSH Client

If not already done, install the git tools from this link: https://git-scm.com/downloads

Select all default values while you go through the installation wizard.

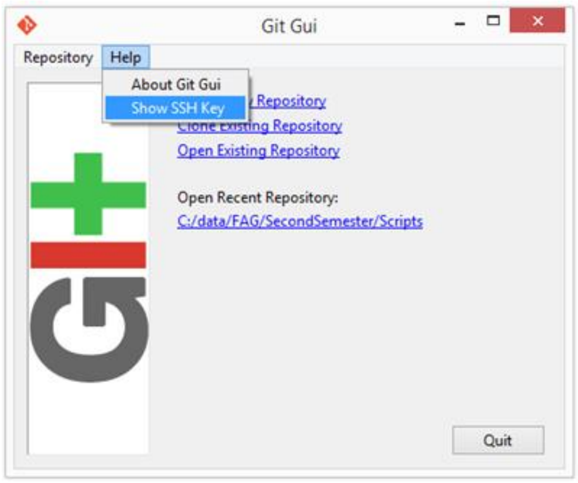
Among the many things installed are:

Git Gui: which we will use to generate the required SSH keys, and access the public key, that we have to place on our server
Git Bash: A "Unix" terminal which we will use as our terminal and SSH client.

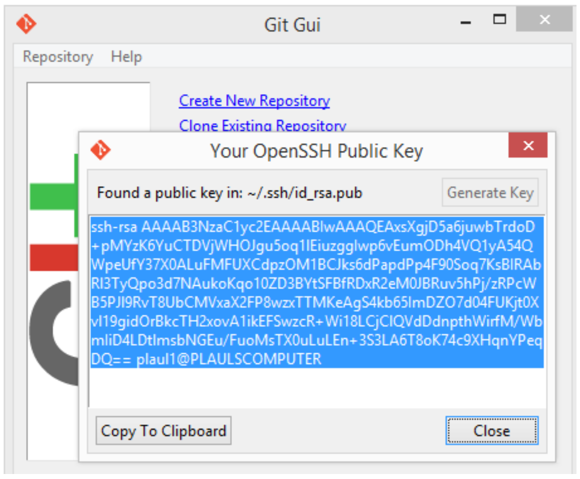
We will come back to SSH in a future lesson. For now, just accept that we are using a strategy which involves a *private* key (as in never ever give it away) and a *public* key (as in, give it to all you want to perform secure communication with).

Generating the SSH keys

Click the start icon and type "Git Gui" and follow the instructions below



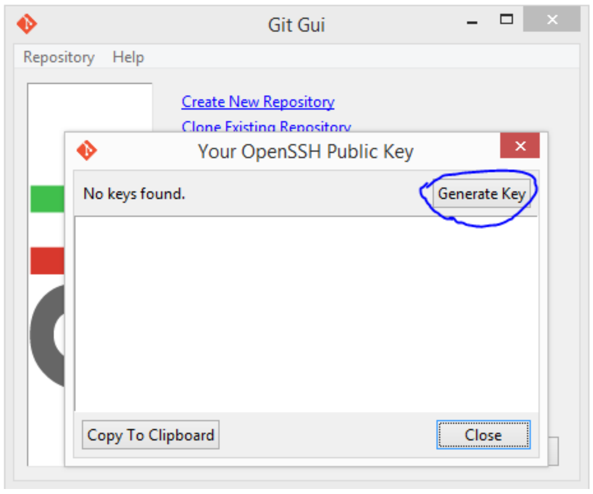
Click Help > Show SSH Key



Found a public key in: ~/.ssh/id_rsa.pub

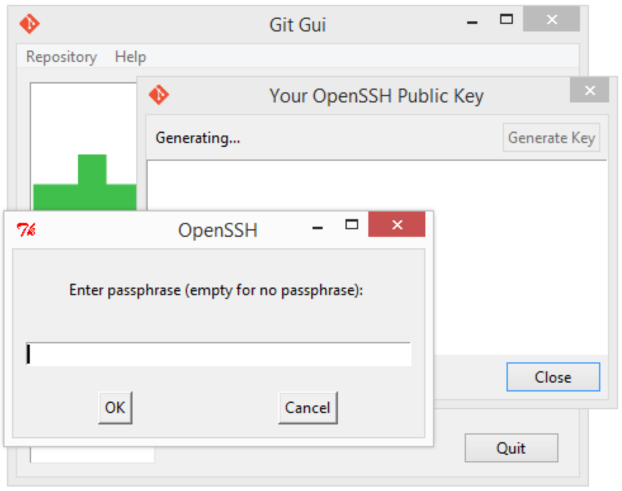
```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAxsXgjD5a6juwbTrdoD
+pMYzK6YuCTDVjWHOJgu5oq1IEiuzgglwp6vEumODh4VQ1yA54Q
WpeUfY37X0ALuFMFUXCdpzOM1BCJks6dPapdPp4F90Soq7KsBIRAb
RI3TyQpo3d7NAukoKqo10ZD3BYtSFBfRDxR2eM0JBRuv5hPj/zRPcW
B5PJI9RvT8UbCMVxaX2FP8wzxTTMKeAgS4kb65ImDZO7d04FUKjt0X
vI19gidOrBkcTH2xovA1ikEFSwzcR+Wi18LCjCIQVdDdnpthWirfM/Wb
mIiD4LDtImsbNGEu/FuoMsTX0uLuLEn+3S3LA6T8oK74c9XHqnYPeq
DQ== plaul1@PLAULSCOMPUTER
```

If you have an existing SSH key, copy the key to the Clipboard and use it when required



No keys found.

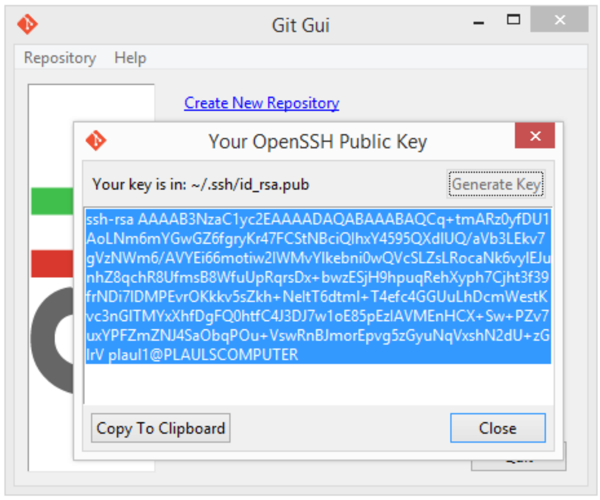
Click generate key



Generating...

Enter passphrase (empty for no passphrase):

Enter a passphrase or leave blank for non-production use



Your key is in: ~/.ssh/id_rsa.pub

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCq+tmARz0yfDU1
AoLNm6mYGwGZ6fgryKr47FCStNBciQIhxY4595QXdIUQ/aVb3LEkv7
gVzNWm6/AVYEi66motiw2IWMvYIkebni0wQVcSLZsLRocaNk6vyIEJu
nhZ8qchR8UfmsB8WfuUpRqrsDx+bwzESjH9hpuqRehXyph7Cjht3f39
frNDi7IDMPEvrOKkkv5sZkh+NeltT6dtml+T4efc4GGUuLhDcmWestK
vc3nGITMYxXhfDgFQ0htfC4J3DJ7w1oE85pEzIAVMEnHCX+Sw+PZv7
uxYPFZmZNJ4SaObqPOu+VswRnBJmorEpvg5zGyuNqVxshN2dU+zG
lrV plaul1@PLAULSCOMPUTER
```

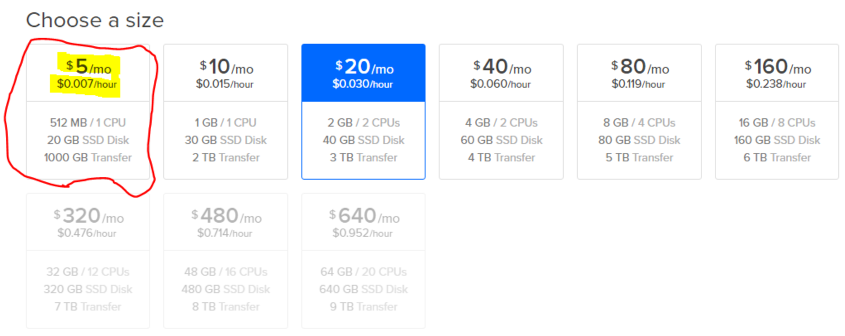
Copy the key to the and use when required

Alternatively you can use git bash to set up the keys as explained here: https://help.github.com/articles/generating-ssh-keys/#platform-all

Sign up on DigitalOcean

Go to https://www.digitalocean.com/ and create an account. This is not free, so it requires a credit card. This however, should not be a problem. We don't expect you to buy any books (other educations on Cphbusiness require students to buy books for way beyond kr. 1000 pr. semester) but do expect you to spend money on things like this (and outsourcing later in the semester).

Just make sure to pick the $5 image size for your image when you get to that part (unless you are willing to pay for more resources) and also observe that it is $5 + an hourly fee which depends on the traffic and the up time, on your site.

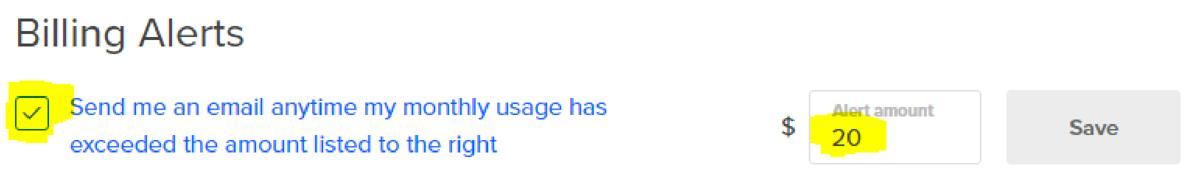


Choose a size

$5/mo $0.007/hour 512 MB / 1 CPU 20 GB SSD Disk 1000 GB Transfer	$10/mo $0.015/hour 1 GB / 1 CPU 30 GB SSD Disk 2 TB Transfer	$20/mo $0.030/hour 2 GB / 2 CPUs 40 GB SSD Disk 3 TB Transfer	$40/mo $0.060/hour 4 GB / 2 CPUs 60 GB SSD Disk 4 TB Transfer	$80/mo $0.119/hour 8 GB / 4 CPUs 80 GB SSD Disk 5 TB Transfer	$160/mo $0.238/hour 16 GB / 8 CPUs 160 GB SSD Disk 6 TB Transfer
$320/mo $0.476/hour 32 GB / 12 CPUs 320 GB SSD Disk 7 TB Transfer	$480/mo $0.714/hour 48 GB / 16 CPUs 480 GB SSD Disk 8 TB Transfer	$640/mo $0.952/hour 64 GB / 20 CPUs 640 GB SSD Disk 9 TB Transfer			

Make sure to check your bank account at least once per month, and destroy your Droplet if no longer used. Any charges related to this is your problem (but should never be a problem if you use the hint below)

How to prevent surprises: As the very first thing, you should go to *Settings* → *Billing* → *Billing Alerts* and set an amount and check the email-check box. This should ensure you get an email if the amount exceeds what you expect.

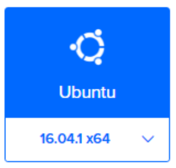


Billing Alerts

✓ Send me an email anytime my monthly usage has exceeded the amount listed to the right

$ Alert amount 20

Create your Droplet


Ubuntu

16.04.1 x64

1) Once you have signed up, create an Ubuntu Droplet on DigitalOcean.

You can create more than one server, but they will each cost you a minimum of $5. Since you can install (almost) whatever you like on your Droplet (server) it will probably make more sense to upgrade your existing server if required.

2) Select Frankfurt (or at least a location in Europe) as *datacentre region*

3) Copy your public key, as described above, into the clipboard. Select "New SSH Key" and paste your key into the TextArea. Provide the key with a name (myLaptop, myHomeComputer or similar). You can add more than one key, if this is a droplet for a team, or your want to access it from more than one computer.

Initial Server Setup

The following is a rewrite of this tutorial:

https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04

The rewrite is basically made to make things a bit simpler and to introduce Git bash as our terminal. Click the start menu and type git bash. This is a "unix" terminal as on a Linux box or Mac OS, and is the terminal we will use as our client (if you are a Windows user) to access our Ubuntu Server

Introduction

When you first create a new Ubuntu 16.04 server, there are a few configuration steps that you should take early on as part of the basic setup. This will increase the security and usability of your server and will give you a solid foundation for subsequent actions.

Step One — Root Login

To log into your server, you will need to know your server's public IP address. You do not need a password since we installed an SSH key for authentication for the "root" user's account. If you have not already logged into your server, you may want to follow the first tutorial in this series, How to Connect to Your Droplet with SSH, which covers this process in detail.

If you are not already connected to your server, go ahead and log in as the root user using the following command (substitute the red part with your server's public IP address):

```
ssh root@SERVER_IP_ADDRESS
```

Complete the login process by accepting the warning about host authenticity, if it appears.

About Root

The root user is the administrative user in a Linux environment that has very broad privileges. Because of the heightened privileges of the root account, you are actually *discouraged* from using it on a regular basis. This is because part of the power inherent with the root account is the ability to make very destructive changes, even by accident.

The next step is to set up an alternative user account with a reduced scope of influence for day-to-day work. We'll teach you how to gain increased privileges during the times when you need them.

Step Two — Create a New User

Once you are logged in as root, we're prepared to add the new user account that we will use to log in from now on.

This example creates a new user called "sammy", but you should replace it with a username that you like:

```
adduser sammy
```

You will be asked a few questions, starting with the account password.

Enter a strong password and, optionally, fill in any of the additional information if you would like. This is not required and you can just hit ENTER in any field you wish to skip.

Note: When following a tutorial like this it's very easy to just do things without thinking. Make sure that you remember the password you added in the step just above.

Step Three — Root Privileges

Now, we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks.

To avoid having to log out of our normal user and log back in as the root account, we can set up what is known as "superuser" or root privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word sudo before each command.

To add these privileges to our new user, we need to add the new user to the "sudo" group. By default, on Ubuntu 16.04, users who belong to the "sudo" group are allowed to use the sudo command.

As root, run this command to add your new user to the *sudo* group (substitute the highlighted word with your new user):

```
  usermod -aG sudo sammy
```

Now your user can run commands with superuser privileges! For more information about how this works, check out this sudoers tutorial.

If you want to increase the security of your server, follow the rest of the steps in this tutorial.

Step Four — Add Public Key Authentication for the New User (Recommended)

The next step in securing your server is to set up public key authentication for your new user. Setting this up will increase the security of your server by requiring a private SSH key to log in also for this user.

We did add our public key (at the digitalocean web site) when we created the droplet, but that was for the root user. Now we have to add the key for our new user.

Install the key

It's the same key (your public key) we will use, so copy the key into the clipboard as explained earlier.

To enable the use of SSH key to authenticate as the new remote user, you must add the public key to a special file in the user's home directory.

On the server, as the **root** user, enter the following command to temporarily switch to the new user (substitute your own user name):

```
  su - sammy
```

Now you will be in your new user's home directory.

Create a new directory called .ssh and restrict its permissions with the following commands:

```
mkdir ~/.ssh

chmod 700 ~/.ssh
```

Now open a file in .ssh called authorized_keys with a text editor. We will use nano to edit the file:

```
nano ~/.ssh/authorized_keys
```

Now insert your public key (which should be in your clipboard) by pasting it into the editor.

Hit CTRL-x to exit the file, then y to save the changes that you made, and then ENTER to confirm the file name.

Note: As explained earlier you can install more than one key, if you want to access the server from more than one computer

Now restrict the permissions of the *authorized_keys* file with this command:

```
chmod 600 ~/.ssh/authorized_keys
```

Type this command **once** to return to the root user:

```
exit
```

Now your public key is installed, and you can use SSH keys to log in as your user.

To read more about how key authentication works, read this tutorial: How To Configure SSH Key-Based Authentication on a Linux Server.

Next, we'll show you how to increase your server's security by disabling password authentication.

Step Five — Disable Password Authentication (Recommended)

Now that your new user can use SSH keys to log in, you can increase your server's security by disabling password-only authentication. Doing so will restrict SSH access to your server to public key authentication only. That is, the only way to log in to your server (aside from the console) is to possess the private key that pairs with the public key that was installed.

Note: Only disable password authentication if you installed a public key to your user as recommended in the previous section, step four. Otherwise, you will lock yourself out of your server!

To disable password authentication on your server, follow these steps.

As **root** or **your new sudo user**, open the SSH daemon configuration:

```
sudo nano /etc/ssh/sshd_config
```

Find the line that specifies *PasswordAuthentication*, uncomment it by deleting the preceding #, then change its value to "no". It should look like this after you have made the change:

```
    PasswordAuthentication no
```

Here are two other settings that are important for key-only authentication and are set by default. If you haven't modified this file before, you *do not* need to change these settings:

sshd_config — Important defaults

```
   PubkeyAuthentication yes
   ChallengeResponseAuthentication no
```

When you are finished making your changes, save and close the file using the method we went over earlier (CTRL-X, then Y, then ENTER).

Type this to reload the SSH daemon:

```
sudo systemctl reload sshd
```

Password authentication is now disabled. Your server is now only accessible with SSH key authentication.

Step Six — Test Log In

Now, before you log out of the server, you should test your new configuration. Do not disconnect until you confirm that you can successfully log in via SSH.

In a new terminal on your **local machine**, log in to your server using the new account that we created. To do so, use this command (substitute your username and server IP address):

```
ssh sammy@SERVER_IP_ADDRESS
```

If you added public key authentication to your user, as described in steps four and five, your private key will be used as authentication. Otherwise, you will be prompted for your user's password.
Note about key authentication: If you created your key pair with a passphrase, you will be prompted to enter the passphrase for your key. Otherwise, if your key pair is passphrase-less, you should be logged in to your server without a password.

Once authentication is provided to the server, you will be logged in as your new user.
Remember, if you need to run a command with root privileges, type "sudo" before it like this:

```
sudo command_to_run
```

At this point, you have a solid foundation for your server. You can install any of the software you need on your server now.
In our case we need java, because we need it to run both or chat server (eventually) and a java based web server (Tomcat)

Install Java on your server

We need Java on our server, for the Chat server, for some of the exercises (like this one) and for Tomcat.

While logged in with your new user account, type (to, update your apt-get package index):

```
sudo apt-get update
```

Then install the Java Development Kit package with apt-get: `sudo apt-get install default-jdk`

Verify that Java is installed by typing: `java -version`

Install the Java TCP-echo server on our Ubuntu server

On the Server:

Create a folder */opt* on your server using the command: `mkdir opt`

Create a subfolder using the command: `mkdir opt/javaservers`

On your laptop:

Open a new Git Bash instance, and navigate to the folder that holds your Echo Server jar file

Now we will upload the file via SSH (replace XXX with your username):

```
scp Echo.jar username@yourIP:/home/XXX/opt/javaservers
```

On the Server:

Go back to the server terminal window and verify that the file has been uploaded to the *myjavaservers* folder.

Start the server by typing: `java -jar echo.jar <ip> <port>`

This assumes that you have designed your server to take the bind ip-address and port number from the command line. The value for <ip> could be your public IP, but does not have to be. Type `ifconfig` in the server terminal, to get the right value, and then select a value for the port.

With my IP and port 9000 as example, it should look like:

```
lam@ubuntu64:~/opt/myjava$ java -jar Echo.jar 139.59.143.106 9000
Sever started. Listening on: 9000, bound to: 139.59.143.106
```

Finally, use telnet on your laptop to verify that you can access the server.