University of Antwerp
Faculty of Applied Engineering

Lab 1

Distributed Systems

Raul Cuervo Bello, Andreas Gavrielides

University of Antwerp – imec, IDLab – Faculty of Applied Engineering, Belgium

imec
embracing a better life

Teaching Assistant

- **Raul Cuervo Bello**
  - Email
    - raul.cuervo@uantwerpen.be
- **Andreas Gavrielides**
  - E-mail
    - Andreas.Gavrielides@uantwerpen.be

- **GitHub**
  - Username: (add the following usernames always to your projects)
    - Raulitocuervo
    - agavri

Overview of lab sessions

- **Lab 1:**
  - Introductory lab & TCP/UDP
  - Setup Remote nodes
- **Lab 2:**
  - REST protocol
  - Remote nodes
  - 2 sessions
- **Lab 3: Project**
  - Naming server
  - Remote nodes
  - 3 sessions
- **Lab 4: Project**
  - Discovery service
  - Remote nodes
  - 3 sessions
- **Lab 5: Project**
  - Replication
  - Remote nodes
  - 4 sessions
- **Lab 6: Project**
  - Agents
  - Remote nodes
  - 1 session
- **Lab 7: Project**
  - GUI
  - Remote nodes
  - 1 session
- **Finishing project**
  - Remote nodes
  - 4 sessions

Schedule

Month	Monday					Wednesday					Thursday					Friday			
	Date	AM/PM	Type	Content	Comment	Date	AM/PM	Type	Content	Comment	Date	AM/PM	Type	Content	Comment	Date	AM/PM	Type	Content
February	10/02/25	8:30	Lecture	Intro/Characteriza							13/02/25	10:45	Lecture	System Models					
						26/02/25	16:00	Lecture	Interprocess		27/02/25	10:45	Lecture	RMI / Rest					
March						5/3/25	16:00	Laboratory	Intro / Interprocess communications		6/3/2025	10:45	Laboratory	Project / REST					
	10/3/25	16:00	Lecture	Indirect communication							13/3/2025	10:45	Lecture	Naming					
	17/3/25	16:00	Lecture			19/3/25	16:00	Laboratory	Naming		20/3/25	10:45	Laboratory	Naming					
	24/3/25	16:00	Laboratory	Discovery		26/3/25	16:00	Laboratory	Discovery		27/3/24	10:45	Lecture						
March/April	31/3/25	16:00	Lecture	Distributed Computing paradigms		2/4/25	16:00	Laboratory	Replication		3/4/25	10:45	Lecture			4/4/25	8:30	Laboratory	Replication
April																			
											24/4/25	10:45	Laboratory	Replication					
	28/4/25	16:00	Laboratory	Replication		30/4/25	16:00	Lecture	Seminars Group 1 and 2										
May	5/5/25	16:00	Lecture	Seminars Group 3 and 4		7/5/25	16:00	Lecture	Seminars Group 5		8/5/25	10:45	Laboratory	Agents		9/5/25	8:30	Laboratory	Agents
						14/5/25	16:00	Laboratory	GUI		15/5/25	10:45	Laboratory	Finishing Project		16/5/25	8:30	Laboratory	Finishing Project
						21/5/25	16:00	Laboratory	Finishing Project		22/5/25	10:45	Laboratory	Finishing Project					
June [EXAM]	28/05/2025																		
Sept [2deEx]	To be confirm																		

Deadlines for Lab Reports

Lab report	Deadline
Lab 1 – Introduction and TCP/UDP	19/03/2025
Lab 2 – REST protocol	20/03/2025
Lab 3 – Naming Server	02/04/2025
Lab 4 – Discovery	24/04/2025
Lab 5 – Replication	15/05/2025
Lab 6 – Agents	23/05/2025
Lab 7 – GUI	23/05/2025
Final Report	23/05/2025
Peer evaluation	26/05/2025
Final Exam	28/05/2025

General rules

- **60% of the overall grade**
  - Project portfolio
  - Presentation (defense + video)
- **You should join lab session in time**
  - No delays of more than 5 minutes, is not tolerable
  - Absence –> e-mail Raul Cuervo Bello and Andreas Gavrielides
- **Presence is mandatory**
- **Lab reports are mandatory**
- **Split the workload with your teammates!**
- **Polls and surveys are not grated**

Git

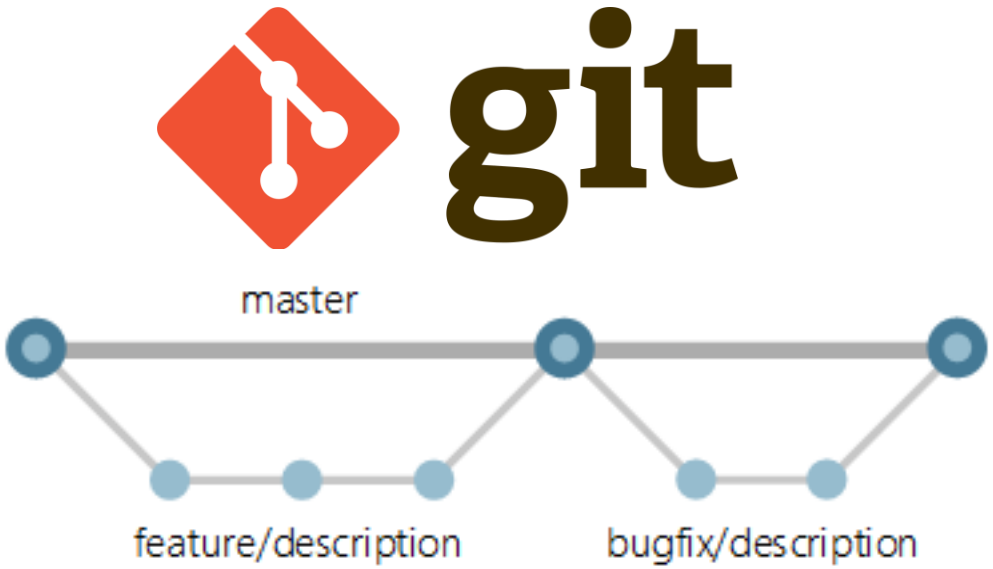
Why do we need Git?

- Open-source version control system
- Code hosting platform for collaboration
- Work on the same code
- Conflict management
- Keeps track of code version (and changes made by specified users)
- Snapshot of code taken as *commit*
- Control of contributions by *push* command
- Contribution integration
- Remote repository on GitHub, public or private

Step 1. Create (on GitHub) a repository

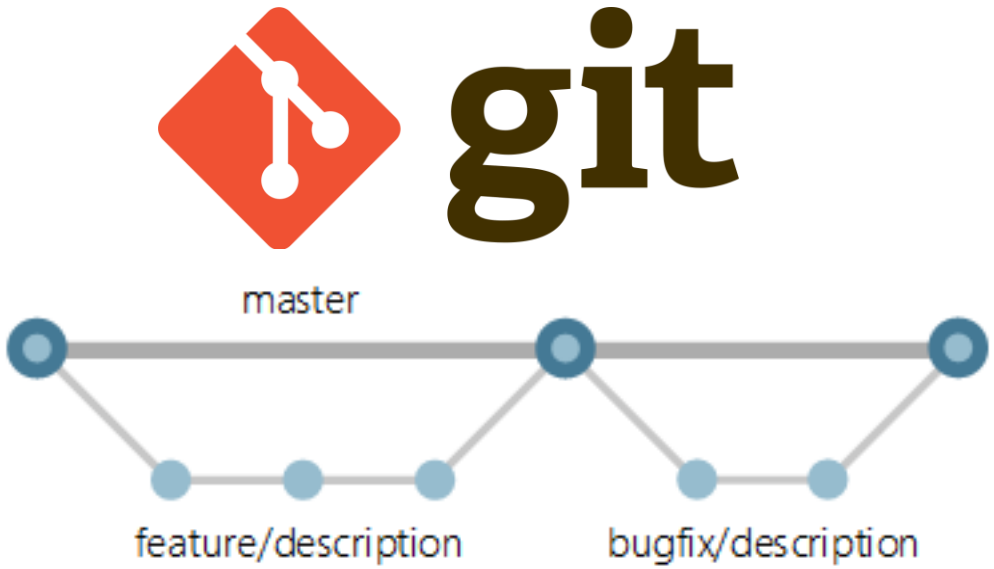
- Repository usually used for one project
- Contain files, images, videos, datasets, etc i.e., anything that project requires to properly run
- Special: README, usually a .md file that provides the summary of project and some basis instructions on ow to make use of it
- Repository can be created via Command Line Interface (CLI) or using GitHub GUI

Step 2. Create a branch



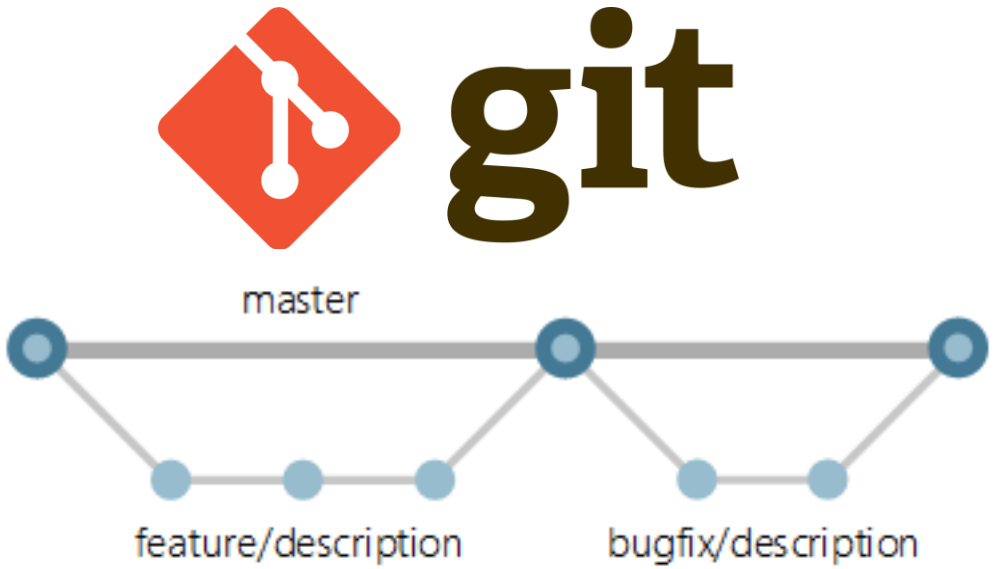
- Working on different versions of repository at the same time
- Default branch *master*
- Other branches used for experimentation (e.g., bugfix, new features, etc.), before committing them to master branch

Step 3. Make and commit changees



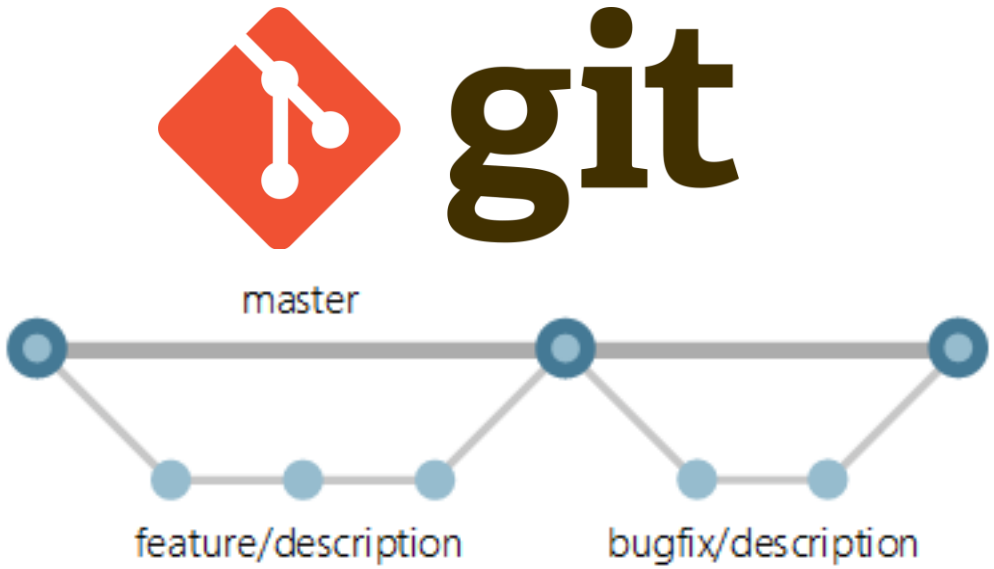
- **Snapshot of current code that is going to be saved is called commit**
- **Each commit has a *commit message***
  - *In your lab report -> how to write a proper commit message? What are the rules? How are they constructed? …*
- **Messages help other developers to understand the changes that were made**

Step 4. Open a pull request



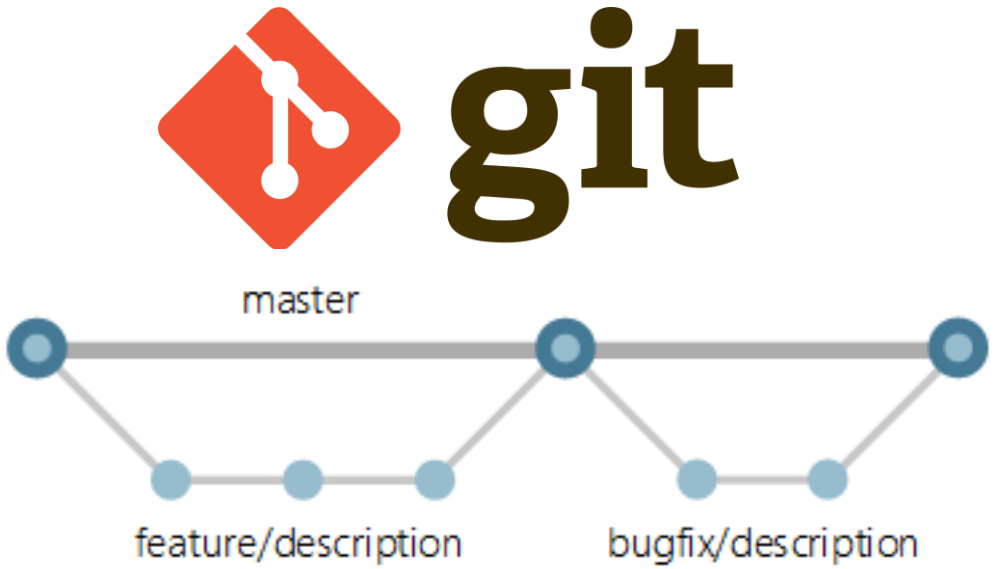
- Work on additional branch is ready to be reviewed
- Pull request allows contributors to see the difference between branches
- It opens the discussion before merging feature branch in to master

Step 5. Merge Pull request



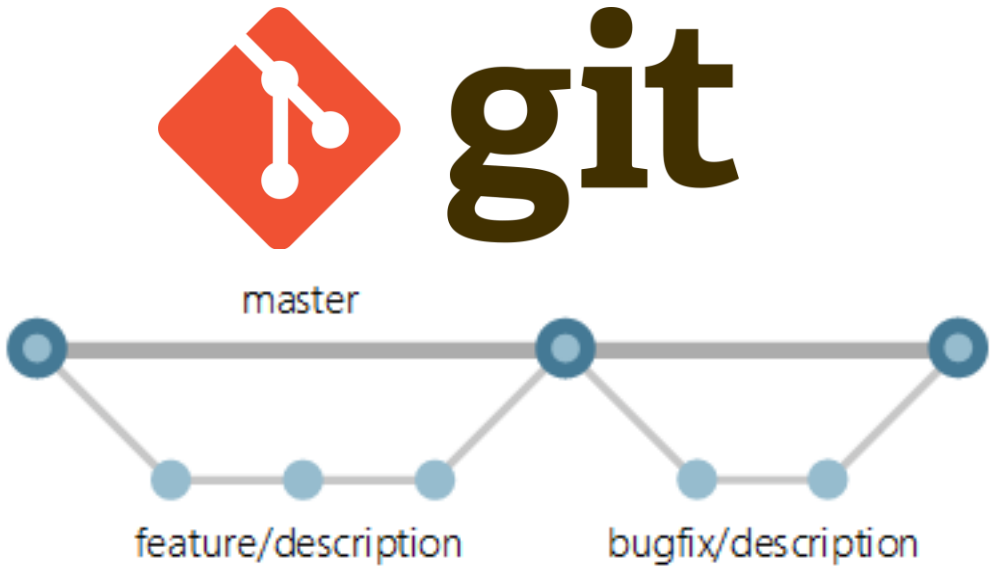
- Brings changes together
- Once the feature branch is merged to master, it can be deleted

Basic commands (1/3)



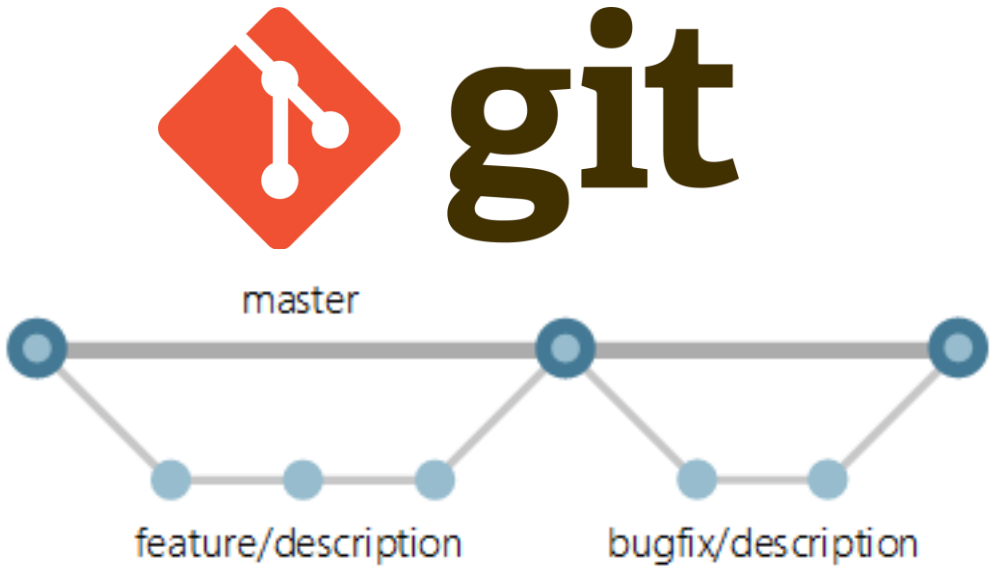
- **git init**
  - Initialization of local repository
- **git status**
  - Checks status of current branch
- **git commit**
  - Makes a snapshot
- **git push**
  - Pushes version of code to remote repository, e.g., on GitHub
- **git pull**
  - Pulls the latest version of code from remote repository, e.g., on GitHub
- **.gitignore**
  - Adds list of files that won't be added to remote repository, e.g., on GitHub (e.g., libraries, IDE auto-generated files, etc)

Basic commands (2/3)



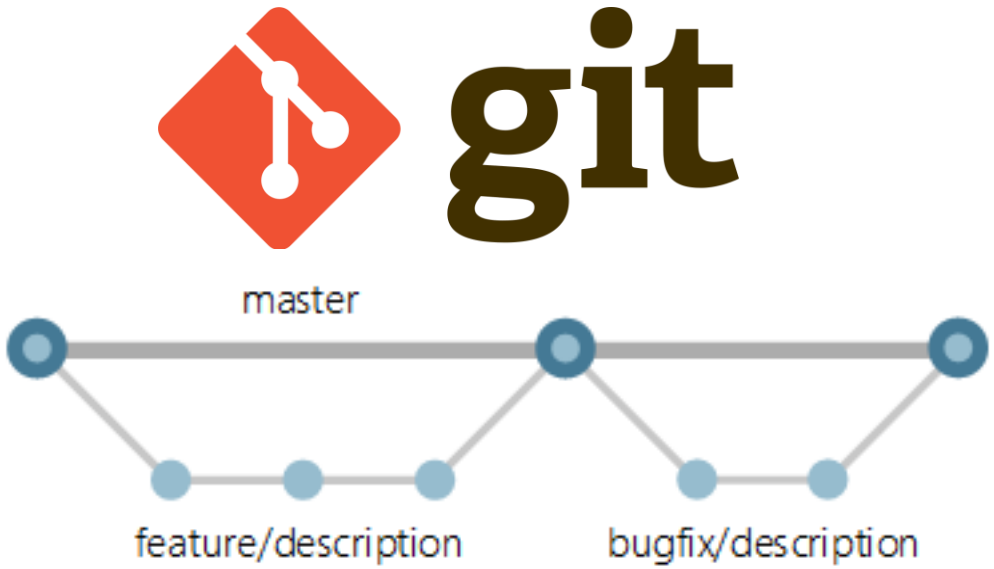
- **git branch feature1**
  - Creates branch named “feature1”
- **git checkout feature1**
  - Switches to branch “feature1” (update according to the latest code committed to the branch)
- **git add .**
  - Add all new files to local repository
- **git commit –m “changes message”**
  - Commits changes to local repository
- **git checkout master**
  - Switches to master branch
- **git merge feature1**
  - Merges branch feature1 to master

Basic commands (3/3)



- **git remote**
  - List all remote repositories
- **git remote add origin *url***
  - Adds remote repository
- **git push –u origin master**
  - Pushes changes to origin master

Basic commands (Example)



- touch index.html
- touch test.css
- touch other.css

- git init
- git add index.html
- git status
- git add *.css
- git commit –m “initial commit”

TCP/UDP

TCP/UDP communication

- **Socket connection means that two machines have information about each other's IP address and TCP/UDP port**
  - [Suggestion] What is the difference between "bind" and "connect" -> Lab report
  - [Suggestion] What are TCP/UDP ports? -> Lab report
- **Java API networking package (java.net) enables network programming in Java.**
- **Java.net.Socket (TCP)**
  - Socket socket = new Socket("127.0.0.1", 5000);
  - 127.0.0.1 is IP address of server (localhost), and 5000 is a number of port (0 to 65535, e.g., HTTP 80)
- **Java.net.DatagramSocket(UDP)**
  - DatagramSocket ds = new DatagramSocket(1234);
  - Socket listens on port 1234
  - DatagramSocket is a Java mechanism for network communication over UDP
  - [Suggestion] What is the difference between TCP and UDP? -> lab report

TCP/UDP communication

- **Prerequisites for lab session:**
  - Knowledge of Java programming language
  - Knowledge of TCP and UDP transport protocols
  - Work with files and I/O
  - Work with exceptions
  - Work with threads
- **Rules**
  - Work in groups of two
  - All applications developed during the lab sessions should be tested and run in the remote cloud environment (once available)

Task for students (Work in groups of two)

- **Create a Java Maven project in your preferable IDE**
- **Create a basic Hello world application**
- **Test the following git commands on your applications:**
  - Create a local repository
  - Create a public repository on GitHub (one repo per group)
  - Create feature branches and test switching between branches
  - Push code to public repository
  - Roll back to previous commits
  - Include conflicts on the remote repository
  - Merge the conflicts manually and in IDE

- **TCP/UDP communication**
  - Develop an application using TCP sockets, with both client and server side
    - The functionality of application includes sharing files over network based up on a request coming from the client
  - Modify the aforementioned application by enabling multithreading at server site, so it becomes able to serve multiple clients at the same time
  - Modify the existing application by enabling UDP communication instead of TCP

TCP/UDP communication

- https://guides.github.com/activities/hello-world/
- https://www.youtube.com/watch?v=SWYqp7iY_Tc
- https://ieeexplore.ieee.org/document/8908548

IDLab cloud environment

How to reach it?

IDLab cloud environment

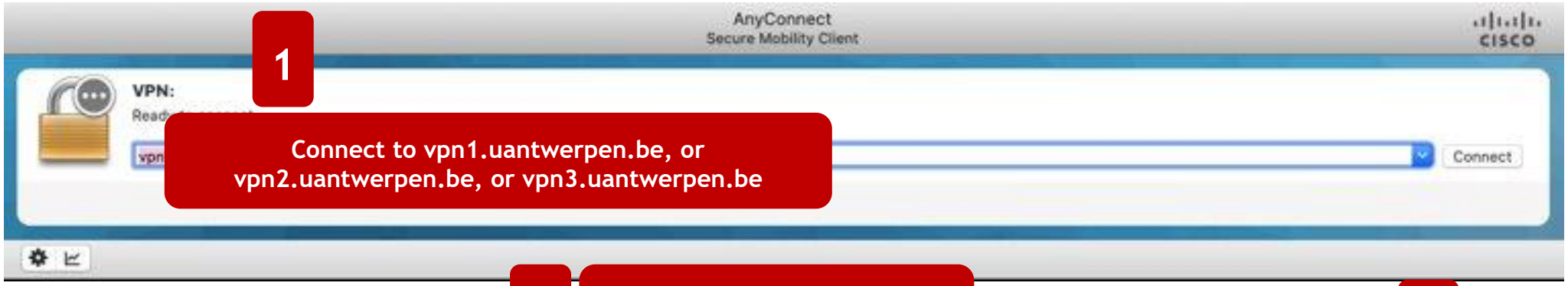
1. **Remote access to nodes, these will be used in project implementation**
2. **All nodes will be accessible 24/7**
   - Advantage, students can work on the labs and the project whenever needed
   - Disadvantage, persistent storage is not guaranteed
3. **Each group will have 5 containers/machines with Ubuntu**
4. **To access these machines students, need to be connected to University Network!**
   - Thus, at home use the UA VPN
     - vpn1.uantwerpen.be
     - vpn2.uantwerpen.be
     - vpn3.uantwerpen.be

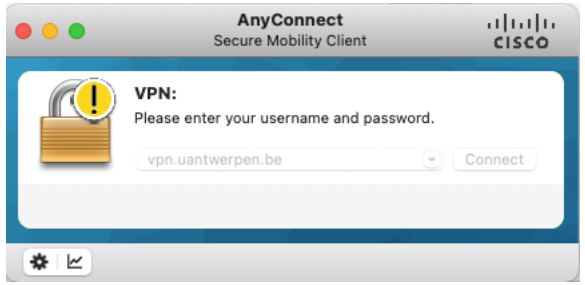
IDLab cloud environment

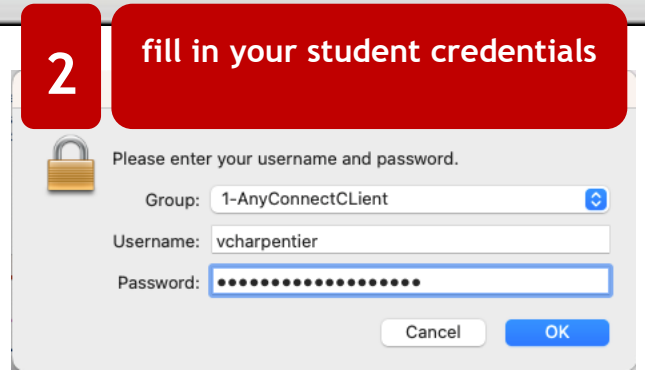
5. **SSH connection to remote nodes**
   - See BB, ”Cloud access” document
     - Open the file with your group number
6. **Brief recap**
   1. Connect to the university network VPN
   2. ssh to remote nodes via **terminal** on your local machine (laptop, pc) by typing the command stated in the the “Cloud access” document on bb

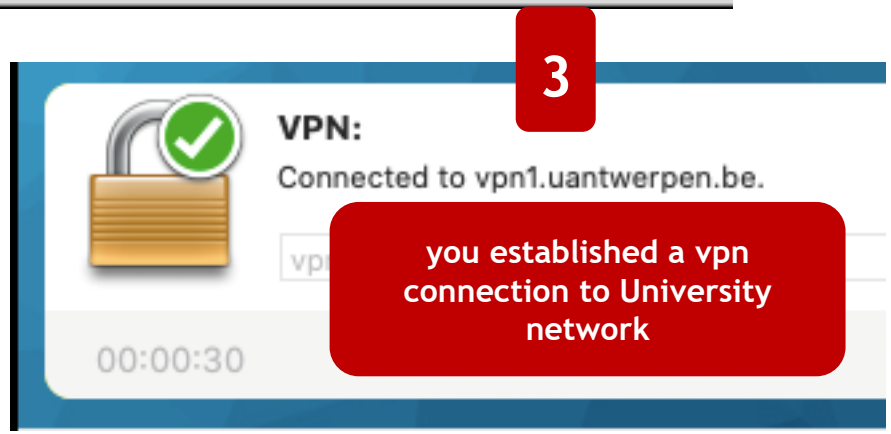
IDLab cloud environment

7. How to connect to UA VPN?









Plan of today:

1. **Each group, if not yet done, choses a nice group name**
   - I will provide your group a group number –> see also BB Google Doc "Students groups"
     - "Async Avengers" -> number 1
     - "??????" -> number 2
     - "NitendoDS" -> number 3
     - "??????" -> number 4
     - "Distribute and conquer" -> number 5
2. **Every group member creates its own public key**
   1. How? With the following command -> "ssh-keygen –t rsa –b 2048" (if you don't already have a OpenSSH key)
      - This is the OpenSSH key format
   2. Each group collects all public keys
      - Then each group creates one zip file (containing the four public keys of each group member)
      - Send this zipfile (named: group_<groupNumber>, e.g., group_1) -> to raul.cuervo@uantwerpen.be
3. **Once I received from each group a zip with all ssh keys -> you will have access to the remote nodes (soon)**
4. **Open the "Cloud access" document for your group**
   - Read the document
   - All steps are explained to reach your nodes (are 24/7 available)
5. **Have fun!**
   1. Start with your first assignments
      1. First on your local machine, once you can reach the remote nodes
   2. I recommended IntelliJ as IDEA, and as framework Spring Boot
      - Especially once you start with the project