Workshop Handout Lecture „Config Management & Continuous Delivery”

Your Group name for the rest of the workshop:

GG = _____

Requirements:

Laptop with SSH Client, e.g. Putty or MobaXTerm. You can get them free here:
Putty: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html or
MobaXTerm: https://mobaxterm.mobatek.net/download-home-edition.html

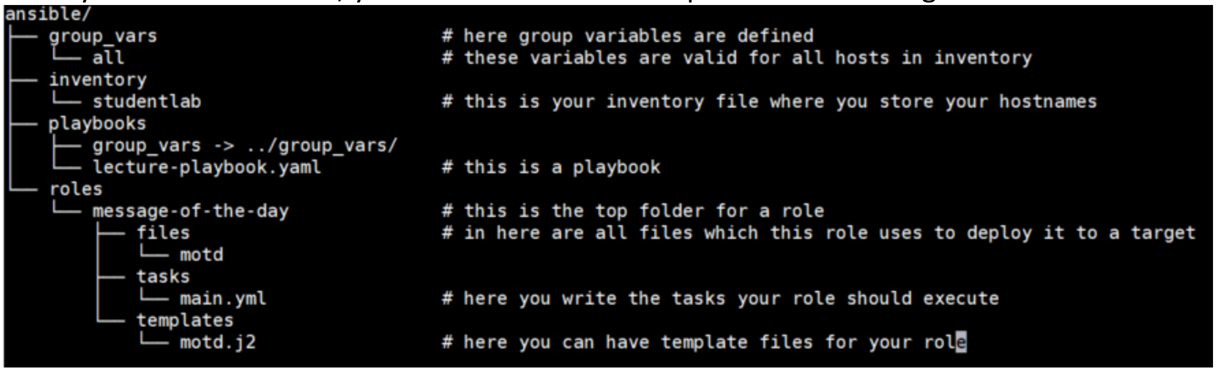
Scenario:

You are IT Admin in a company, and you are tasked to setup and configure multiple new PCs in an autonomous driving vehicle. Until now, you always configured your servers manually by hand, but this would mean to do the same thing on multiple machines. Thus, you decided to use the config management tool Ansible to deploy the configurations to all PCs simultaneously.

In the following tasks, you will:

- See if your PCs are up & running & reachable in the network
- Make them known and reachable to Ansible
- Use Ansible to configure a message-of-the-day on all PCs with just one playbook

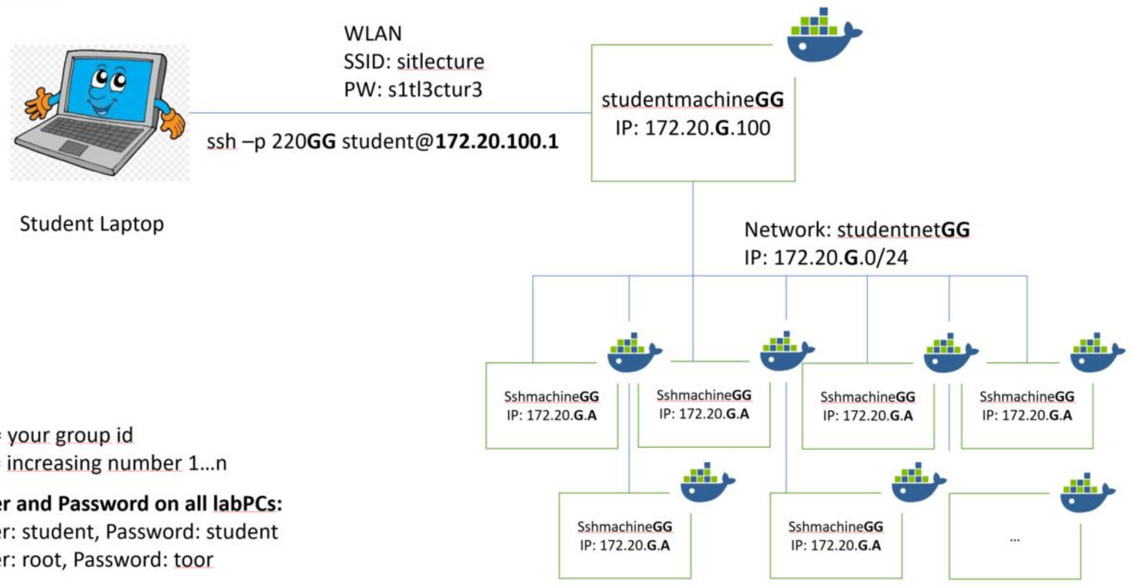
From you Ansible Tutorial, you know that Ansible requires the following file structure:



```
ansible/
├── group_vars                              # here group variables are defined
│   └── all                                 # these variables are valid for all hosts in inventory
├── inventory
│   └── studentlab                          # this is your inventory file where you store your hostnames
├── playbooks
│   ├── group_vars -> ../group_vars/
│   └── lecture-playbook.yaml               # this is a playbook
└── roles
    └── message-of-the-day                  # this is the top folder for a role
        ├── files                           # in here are all files which this role uses to deploy it to a target
        │   └── motd
        ├── tasks
        │   └── main.yml                    # here you write the tasks your role should execute
        └── templates
            └── motd.j2                     # here you can have template files for your role
```

Lecture Lab Network Layout

Part 1: Ansible



G = your group id
A = increasing number 1…n

User and Password on all labPCs:
User: student, Password: student
User: root, Password: toor

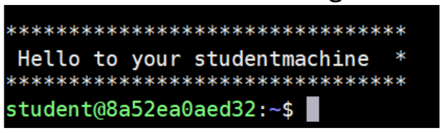
Task 1: Connect to your group studentmachine via ssh and try to ping your "sshmachine"

SSH command to access your studentmachine:

ssh -p 220***GG*** student@172.20.100.1

Example: Group **01**:
Studentmachine1 IP: 172.20.**1**.2
SSH on Studentmachine is listening on port: 220**01**

You should see something similar to this:



```
*********************************
 Hello to your studentmachine  *
*********************************
student@8a52ea0aed32:~$
```

T1.1: Ping your corresponding sshmachines.
ping sshmachine0101
ping sshmachine0102
…

Task 2: Ansible ping

Enter the directory "/home/student/ansible".
Here you can find a basic ansible directory structure. Look a bit around to see what's here.

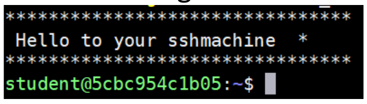
T2.1: Update the inventory file "studentlab", with your "sshmachine**GGAA"** hosts in the inventory group **"studentlab"**

Try ansible -m ping <hostname> to validate that Ansible can reach and access each of your sshmachine.

T2.2: Try to use ansible –m ping –i <inventory> <inventory_group> to reach your whole group "studentlab" from your inventory.

Task 3: Deploy "message of the day"

T3.1: Ssh into your sshmachine**GG** and look at the prompt how you are greeted. You should see something like this:



```
*********************************
 Hello to your sshmachine  *
*********************************
student@5cbc954c1b05:~$
```

After you saw it, **log out** to be in your studentmachine again.

Hint: you can use the Linux cmd:
`figlet <some text>`
To create some ASCII Arts for your motd file ;-)

T3.2: Read what a Linux "Message of the Day (MOTD) is:
https://linuxconfig.org/how-to-set-a-custom-message-of-the-day-on-linux

T3.3: Now you want to create your own motd file and **deploy** it to all your sshmachines.
Create a folder for a role "message-of-the-day" with subfolders "files" and "tasks".
Create a file with name motd under /roles/message-of-the-day/files/motd with the following content:

```
*******************************************
* sshmachine: group <YOUR GROUP Number> *
*******************************************
```

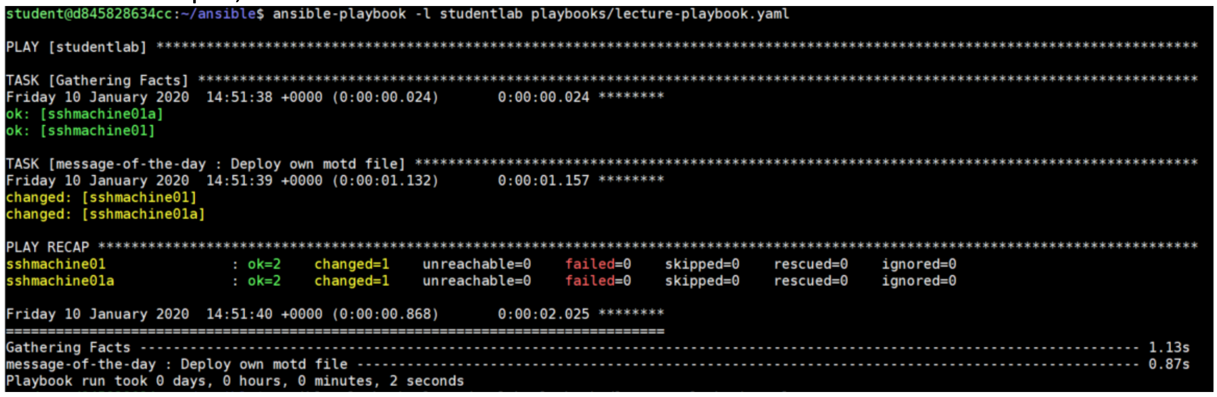
T3.4: Now create a file /roles/message-of-the-day/tasks/main.yml, where you use the ansible module “copy” to deploy your motd file to your sshmachines to the folder /etc/motd.

Ansible copy module: https://docs.ansible.com/ansible/latest/modules/copy_module.html

T3.5: Write a simple Ansible playbook “lecture-playbook.yaml”, which executes your role “message-of-the-day” to your ssh machine. Use the already prepared example file under ansible/playbook as a help.

Run your ansible playbook to deploy your motd file to your sshmachines:
ansible-playbook -i studentlab playbooks/lecture-playbook.yaml

Watch the output, it should look like this:

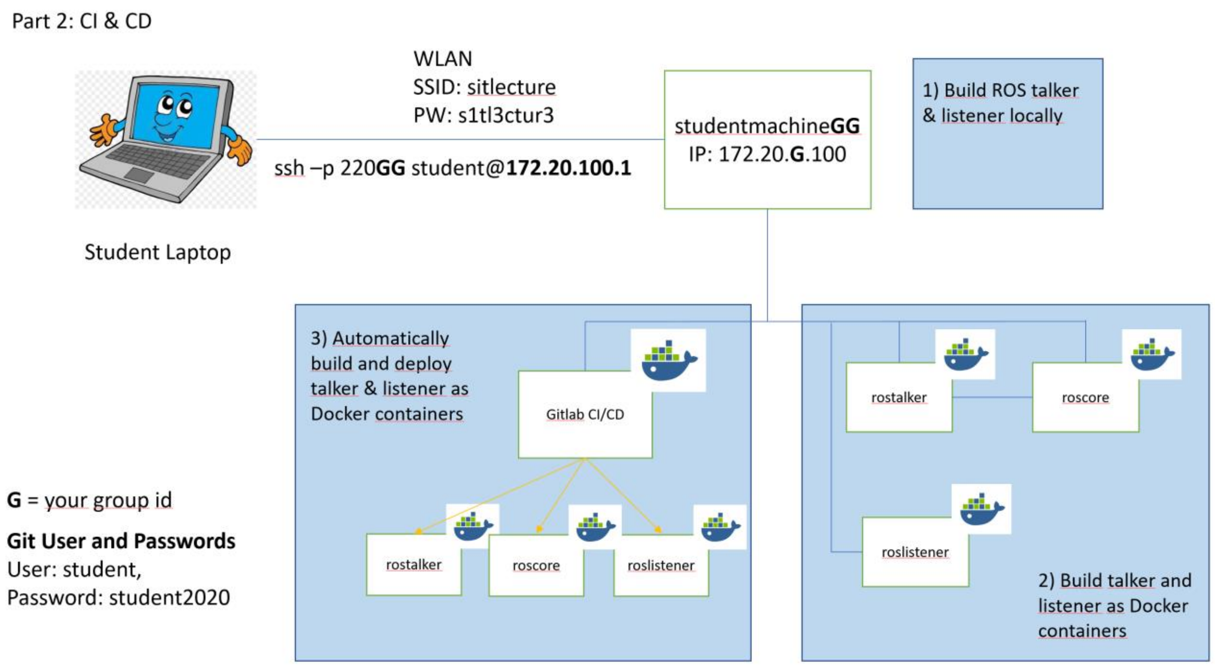


```
student@d845828634cc:~/ansible$ ansible-playbook -l studentlab playbooks/lecture-playbook.yaml

PLAY [studentlab] *****************************************************************************************************************************

TASK [Gathering Facts] ************************************************************************************************************************
Friday 10 January 2020  14:51:38 +0000 (0:00:00.024)       0:00:00.024 ********
ok: [sshmachine01a]
ok: [sshmachine01]

TASK [message-of-the-day : Deploy own motd file] **********************************************************************************************
Friday 10 January 2020  14:51:39 +0000 (0:00:01.132)       0:00:01.157 ********
changed: [sshmachine01]
changed: [sshmachine01a]

PLAY RECAP ************************************************************************************************************************************
sshmachine01               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
sshmachine01a              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

Friday 10 January 2020  14:51:40 +0000 (0:00:00.868)       0:00:02.025 ********
===============================================================================
Gathering Facts ------------------------------------------------------------------------------------------------------------------------ 1.13s
message-of-the-day : Deploy own motd file ---------------------------------------------------------------------------------------------- 0.87s
Playbook run took 0 days, 0 hours, 0 minutes, 2 seconds
```

SSH into your sshmachines and check if your deployment worked.

T3.3. Run your playbook again and look at the Ansible output. Notice, that it should not change the /etc/motd file, since it is still correct. Congrats, you reached a reproducible target state!

Part 2 – CI / CD



Scenario

Google hired you as an SRE (Site-Reliability Engineer). You are supposed to set up a Continuous Delivery pipeline for a set of distributed applications (talker, listener and middleware) such that the development team can make changes and deliver them to the customer in a more efficient way. You plan to get developer, tester and operations people together and first figure out how they build, test and deploy their applications now. Then you start automating things based on Docker and GitLab CI.

Task 4: Manually checkout, build and run applications

```
# checkout git repository
mkdir workspace && cd workspace
git clone http://172.17.0.1:8080/root/ad_software.git src
# source environment and run build commands
source /opt/ros/melodic/setup.bash
catkin_make talker
catkin_make listener
# run it
source devel/setup.bash
rosrun talker talker
# kill it
CTRL + c
```

Task 5: Automate the build

Now we want start automating above steps using Docker. There are two Dockerfiles that we need to adapt for that (add build commands)

1. add build command for talker application in src/talker/Dockerfile
2. *cd src/talker && sudo docker build –t talker_**GG**:latest **.***
3. add build command for talker application in src/listener/Dockerfile
4. cd src/listener && sudo docker build –t listener_**GG**:latest .

Task 6: Automate deployment

To prepare deployment of your applications, we want to implement a docker-compose.yml file that describes how our applications inside docker will be executed.

1. Rename the services described in workspace/src/docker-compose.yml to make them unique (see ToDo in docker-compose.yml)
2. Use your build docker images (e.g. talker_**GG**:latest) in docker-compose.yml
3. Check if applications start properly
4. ***sudo docker-compose up***

Task 7: Implement Continuous Integration (CI)

1. Create a feature branch for your team, add, commit and push changes made in the previous tasks:
   a) git checkout -b feature/team-**GG**
   b) git add *
   c) git commit -m "my commit message"
   d) git push --set-upstream origin feature/team-**GG**
2. Add docker build commands from Task 5 to .gitlab-ci.yml file
3. Commit and push change of .gitlab-ci.yaml file

Task 8: Implement Continuous Delivery (CD)

1. Add docker-compose command from Task 6 to .gitlab-ci.yml file
2. Commit and push change of .gitlab-ci.yml file
3. Verify your pipeline builds properly and runs your services
   a. sudo docker ps | grep –e core –e _GG