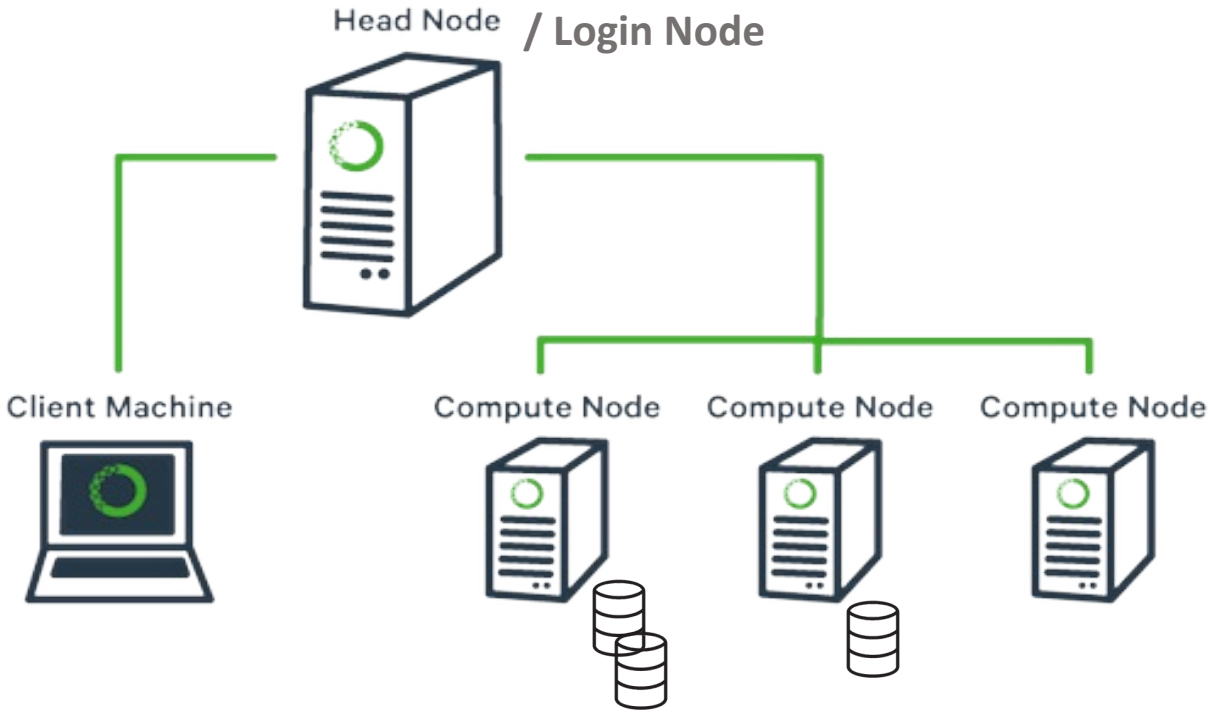
Part 2.
Create a “cluster” on the cloud
Terraform and Ansible

Learn how to create your own computer cluster on

Norwegian Research and Education Cloud (NREC) Openstack cloud

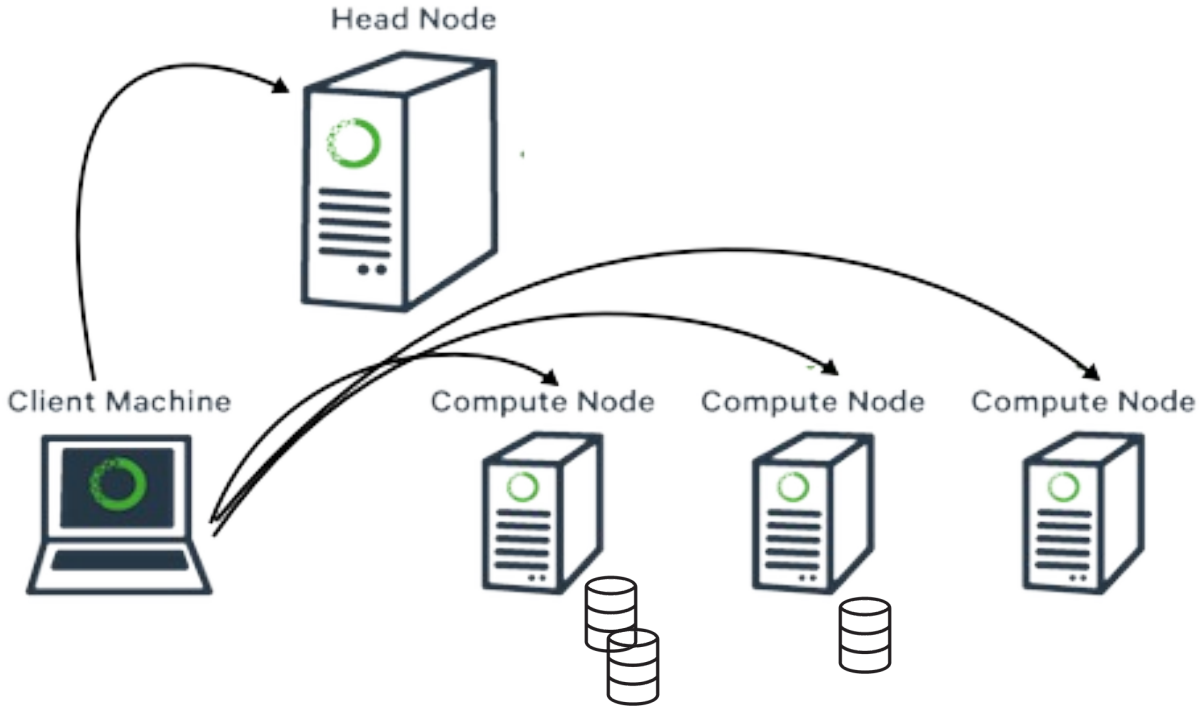
Simplified view of a computing cluster

- A cluster is usually comprised of
  - A master machine and/or login-node
  - A set of compute-nodes/worker-nodes
  - Optional:
    - If many users: A cluster management and job scheduling system orchestrating the jobs that come in e.g. SLURM



But you don’t have to obey the head-node setup – not a cluster in the traditional sense

- We will in this tutorial create 3 servers
- 1 will act as an admin server, but also do computation
  - This machine is already created for you
- They will not be connected together other than being in the same network (no slurm)
- For practical reasons – the admin machine will share some storage to the other machines



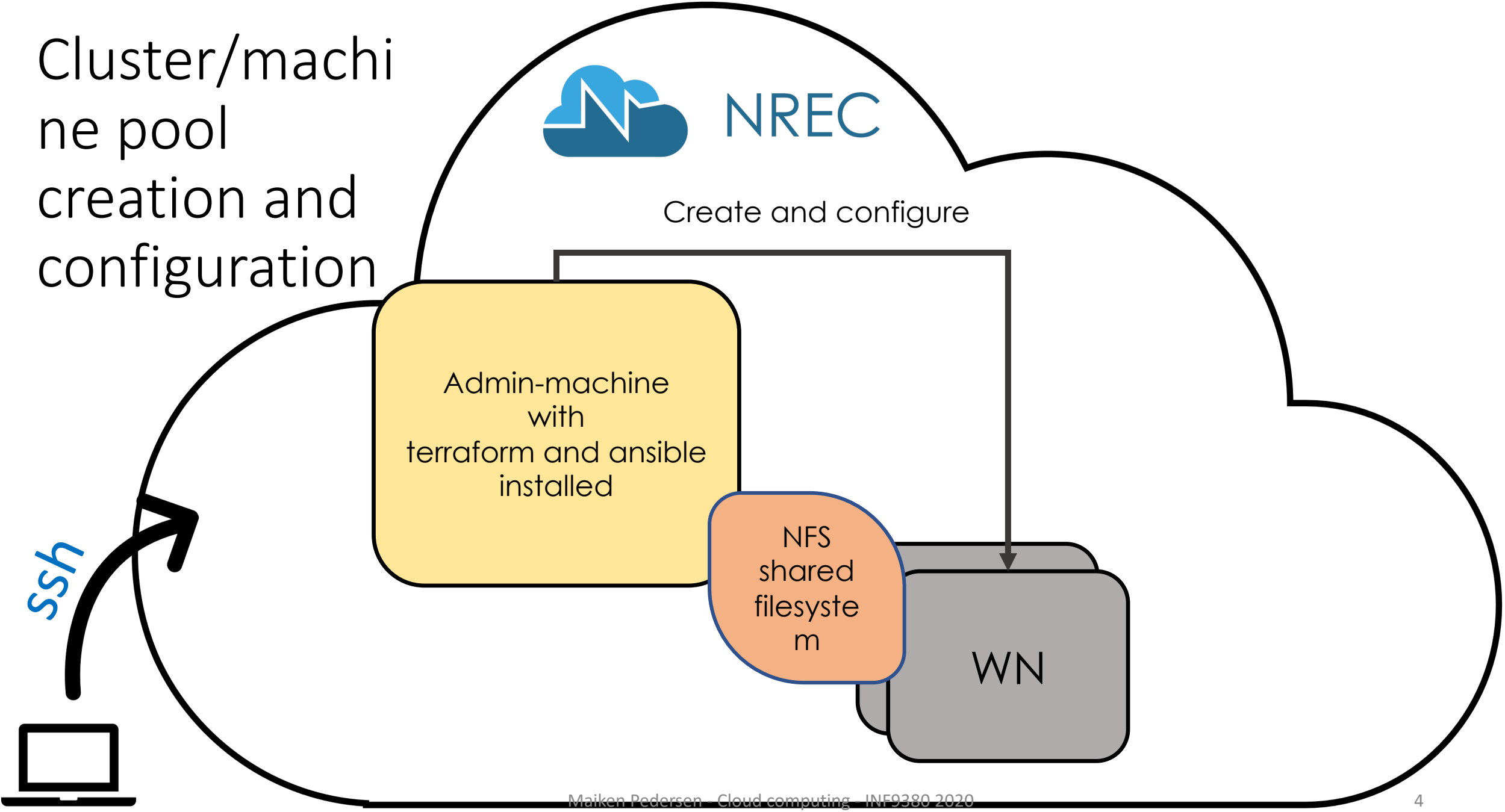
Steps needed to create your own cluster or set of machines

1. To get you going (and since you do not have access to NREC web interface) 1 admin machine is available for each of you
   - You will be using this machine to perform the rest of the tasks
2. Create the virtual machines you need
3. Configure the machines
   - Install needed software
   - Adding any additional storage
   - Set up any needed shared storage
4. Log into master/admin machine and submit a job

Your mini-cluster/pool of machines

- One admin node
- Two compute/work nodes
- Shared filesystem between them

Cluster/machine pool creation and configuration



Terraform

- *"Terraform is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files."*
- Terraform tutorial NREC user-documentation
  - We will be using just the very basic simple parts of Terraform
    - https://docs.nrec.no/terraform-part1.html
    - https://docs.nrec.no/terraform-part2.html (only parts of this)
  - Much more can be done, and to a higher level of sophistication – for you to explore

Basic terraform config file

```
# Define required providers
terraform {
  required_version = ">= 1.0"
  required_providers {
    openstack = {
      source  = "terraform-provider-openstack/openstack"
    }
  }
}
```

a.

```
# Configure the OpenStack Provider
# Empty means using environment variables "OS_*".
# I.e. that you have a keystone file with necessary parameters, and have sourced it
#More info: https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs
provider "openstack" {}
```

b.

```
# Create two compute nodes
resource "openstack_compute_instance_v2" "compute" {

  count = 2
  name = "student02-compute${count.index}"

  image_name = "GOLD CentOS Stream 8"
  flavor_name = "m1.small"

  key_pair = "inf9380-2022-ssh"
  security_groups = [ "default", "inf9380", "slurm_cluster" ]

  network {
    name = "dualStack"
  }
}
```

c.

a.

```
# Define required providers
terraform {
  required_version = ">= 1.0"
  required_providers {
    openstack = {
      source  = "terraform-provider-openstack/openstack"
    }
  }
}
```

b.

```
# Configure the OpenStack Provider
# Empty means using environment variables "OS_*".
# I.e. that you have a keystone file with necessary parameters, and have sourced it
#More info: https://registry.terraform.io/providers/terraform-provider-openstack/openstack/latest/docs
provider "openstack" {}
```

keystonerc file:

Bash script that defines some variables
sourcing the script sets the environment variables

```
## Fill in your username and password below
## Leave all the rest as is
export OS_USERNAME=
export OS_PASSWORD=
export OS_PROJECT_NAME=uio-itf-inf9380
export OS_AUTH_URL=https://api.nrec.no:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_USER_DOMAIN_NAME=dataporten
export OS_PROJECT_DOMAIN_NAME=dataporten
export OS_REGION_NAME=osl
export OS_NO_CACHE=1
```

```
[centos@admin-student01 create]$ printenv | grep OS_
[centos@admin-student01 create]$ source keystonerc_noauth
[centos@admin-student01 create]$ printenv | grep OS_
OS_AUTH_URL=https://api.nrec.no:5000/v3
OS_REGION_NAME=osl
OS_PROJECT_NAME=uio-itf-inf9380
OS_PROJECT_DOMAIN_NAME=dataporten
OS_USER_DOMAIN_NAME=dataporten
OS_IDENTITY_API_VERSION=3
OS_NO_CACHE=1
OS_PASSWORD=
OS_USERNAME=
[centos@admin-student01 create]$ echo $OS_REGION_NAME
osl
```

c.

```
# Create two compute nodes
resource "openstack_compute_instance_v2" "compute" {

  count = 2
  name = "student02-compute${count.index}"

  image_name = "GOLD CentOS Stream 8"
  flavor_name = "m1.small"

  key_pair = "inf9380-2022-ssh"
  security_groups = [ "default", "inf9380", "slurm_cluster" ]

  network {
    name = "dualStack"
  }
}
```

NREC uio-itf-inf9380 • osl

Project
Compute
Overview
Instances
Images
Key Pairs
Server Groups
Volumes
Network
DNS
Identity

Launch Instance

- Details *
- Source *
- Flavor *
- Networks *
- Network Ports
- Security Groups
- Key Pair
- Configuration
- Server Groups
- Scheduler Hints
- Metadata

Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *

Description

Availability Zone

osl-default-1

Count *

1

Total Instances
(160 Max)

28%

43 Current Usage
1 Added
116 Remaining

Cancel | Back | Next | Launch Instance

Configuration done, run terraform to create the resources

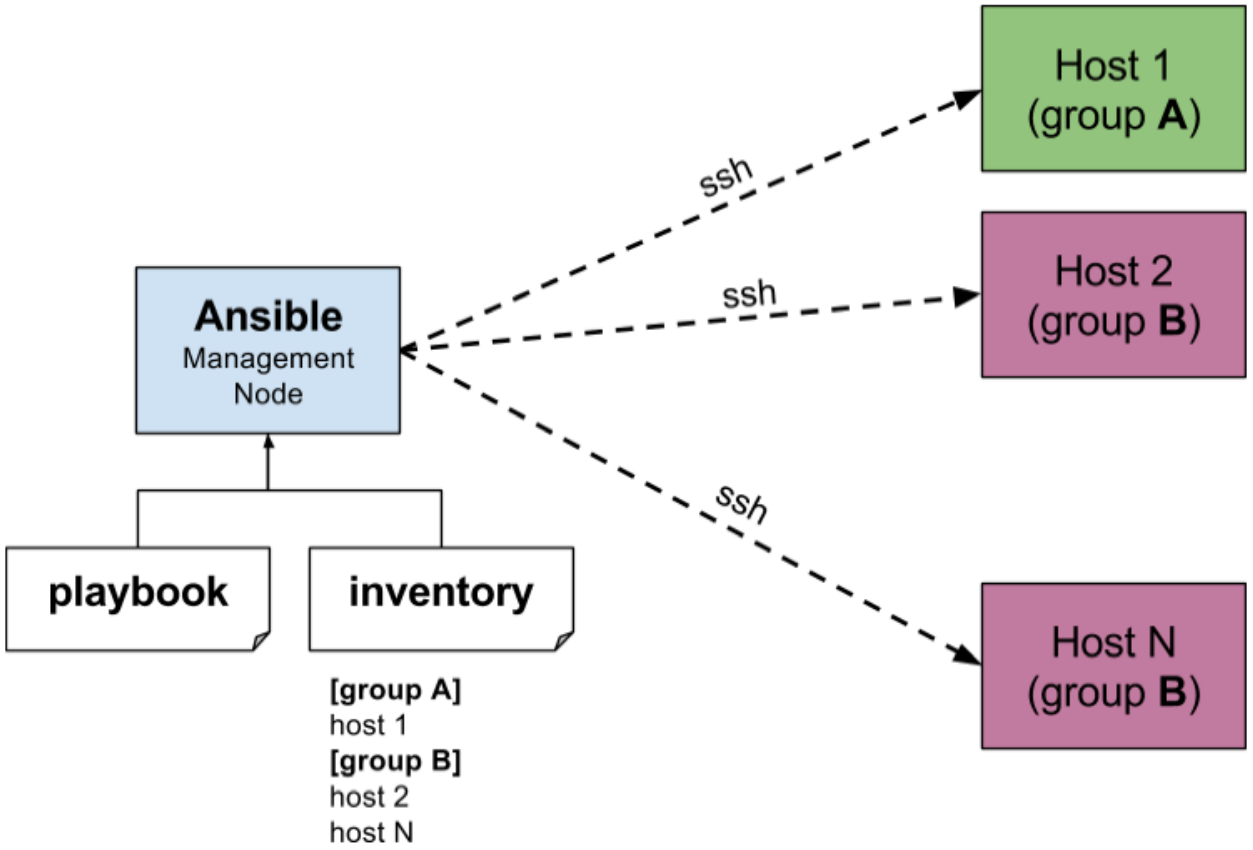
```
terraform init
terraform plan
terraform apply –auto-approve
```

That is it! (At least for this very basic application)

Hands-on Terraform

Ansible

- We will configure the machines using Ansible
- Ansible is an automation tool
  - Instructions are written in yaml-format
  - Skips steps already done, has various fault handling features, and options to run some tasks only by using tags etc, etc.
- You could just write a set of python scripts or a bash scripts that do the same, but that quickly becomes more time-consuming



Ansible terminology

- Ansible inventory file: A list of hostnames and/or ip-addresses in groups
- Ansible playbook: Set of instructions to perform: *"Playbooks are Ansible's configuration, deployment, and orchestration language."*
- Ansible roles: framework for breaking down a playbook into a file-structure
  - Each role (can) contain(s) a set of variables, templates, files and tasks
  - Example: one role for common tasks for all servers being configured, a separate for specific tasks to perform on the webservers
    - Or a more HPC-type example: 1 role for a compute node, another for a login node.

site.yml - playbook

```yaml
---
- hosts: webservers
  roles:
    - common
    - webservers
```

Example inventory file

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Example folder structure for role

```
site.yml
webservers.yml
fooservers.yml
roles/
    common/
        tasks/
        handlers/
        files/
        templates/
        vars/
        defaults/
        meta/
    webservers/
        tasks/
        defaults/
        meta/
```

Demo of running a small ansible script

- Task: Install nano on all the admin nodes

1. Get a list of machines (here called hosts) to run on
   - Can be grouped by servers, like e.g. a set of login machines, compute machines, all machines etc etc
   - Can be created in a variety of ways
     - Manually
     - Command-like client for the cloud architecture you are using – for us: openstack client
       - openstack server list
       - Needs installation, see e.g. https://docs.openstack.org/mitaka/user-guide/common/cli_install_openstack_command_line_clients.html

```
[admin]
student00 ansible_host=158.39.48.136
student01 ansible_host=158.39.75.98
student02 ansible_host=158.37.63.228
student03 ansible_host=158.39.48.48
student04 ansible_host=158.37.63.242
student05 ansible_host=158.39.48.10
student06 ansible_host=158.39.48.68
student07 ansible_host=158.39.48.18
student08 ansible_host=158.39.48.75
student09 ansible_host=158.39.48.29
student10 ansible_host=158.39.48.54
```

Ansible demo contd.

- Our demo will install the nano text editor using yum
  - Example: ansible yum module doc: https://docs.ansible.com/ansible/latest/modules/yum_module.html

1. Simplest: ad-hoc command
   - the simplest way of performing ansible tasks

what hosts to run on

```
ansible -i hosts_inf9380_2022 student00 --private-key=~/.ssh/inf9380-2022-ssh  -m yum -a "name=nano state=present" --become
```

we need to become root to perform yum install

m for module

a for arguments

Ansible demo contd.

2. Put the instructions in a playbook

- This is the recipe for ansible to follow – it contains everything that ansible needs to perform the tasks you define
  - In this playbook the tasks are directly defined in the playbook file

l for limit – run only on student00 host of all the hosts in admin

Run the playbook:

```
ansible-playbook -i hosts_inf9380_2022 -l student00 --private-key=~/.ssh/inf9380-ssh-2022  demo_playbook.yml
```

ansible-playbook instead of ansible

```
- hosts: admin
  become: true

  tasks:
  - name: Output hostname
    debug:
        msg:
         - Machine ansible alias is  {{ inventory_hostname }}
         - Machine hostname is  {{ ansible_hostname }}

  - name: Install nano
    yum:
      name: nano
      state: present
```

which hosts to run on (admin group)

become root

task instructions directly in the playbook

Ansible demo contd.

- Perform the same things, but now using a playbook which uses a simple role (not tasks directly in the playbook)

```
- hosts: admin
  become: true


  roles:
   - demo
```

```
maikenp:ansible_demo$ ls roles/
total 0
drwxr-xr-x  5 maikenp  staff  160 Mar 23 15:56 demo
maikenp:ansible_demo$ ls roles/demo/
total 0
drwxr-xr-x  3 maikenp  staff  96 Mar 23 15:56 defaults
drwxr-xr-x  3 maikenp  staff  96 Mar 23 15:56 files
drwxr-xr-x  3 maikenp  staff  96 Mar 23 15:56 tasks
maikenp:ansible_demo$ ls roles/demo/tasks/
total 8
-rw-r--r--  1 maikenp  staff  481 Mar 23 15:56 main.yml
maikenp:ansible_demo$ ls roles/demo/files/
total 8
-rw-r--r--  1 maikenp  staff  38 Mar 23 15:56 copytestfile.txt
maikenp:ansible_demo$ ls roles/demo/defaults/main.yml
-rw-r--r--  1 maikenp  staff  22 Mar 23 15:56 roles/demo/defaults/main.yml
```

roles/demo/tasks/main.yml

```
- name: Output hostname
  debug:
      msg:
       - Machine ansible alias is  {{ inventory_hostname }}
       - Machine hostname is  {{ ansible_hostname }}

- name: Install nano
  yum:
    name: nano
    state: present


- name: Output default variable
  debug:
    msg: "The default variable testcase contains the string {{ testcase }}"


- name: Copy file from local to remote host
  copy:
    src: files/copytestfile.txt
    dest: /tmp/copiedtestfile.txt
    mode: 0400
```

```
ansible-playbook -i hosts_inf9380_2022 -l student00 --private-key=~/.ssh/inf9380-2022-ssh  demo_role.yml
```

Hands on ansible