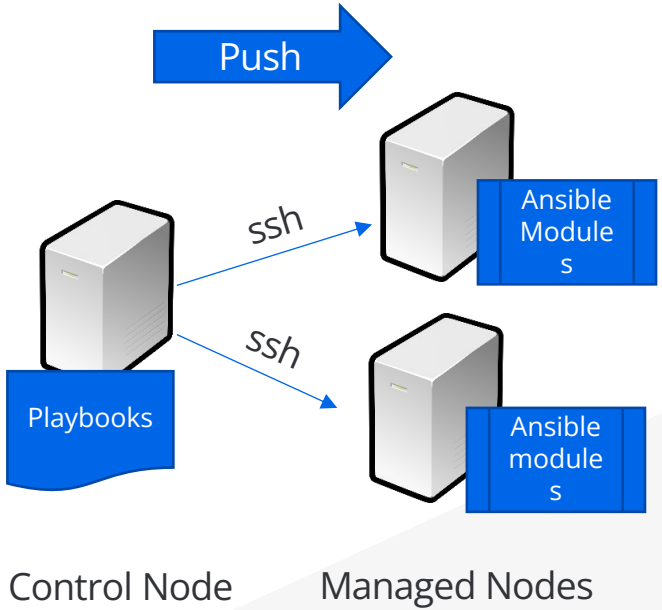
OPITZ CONSULTING

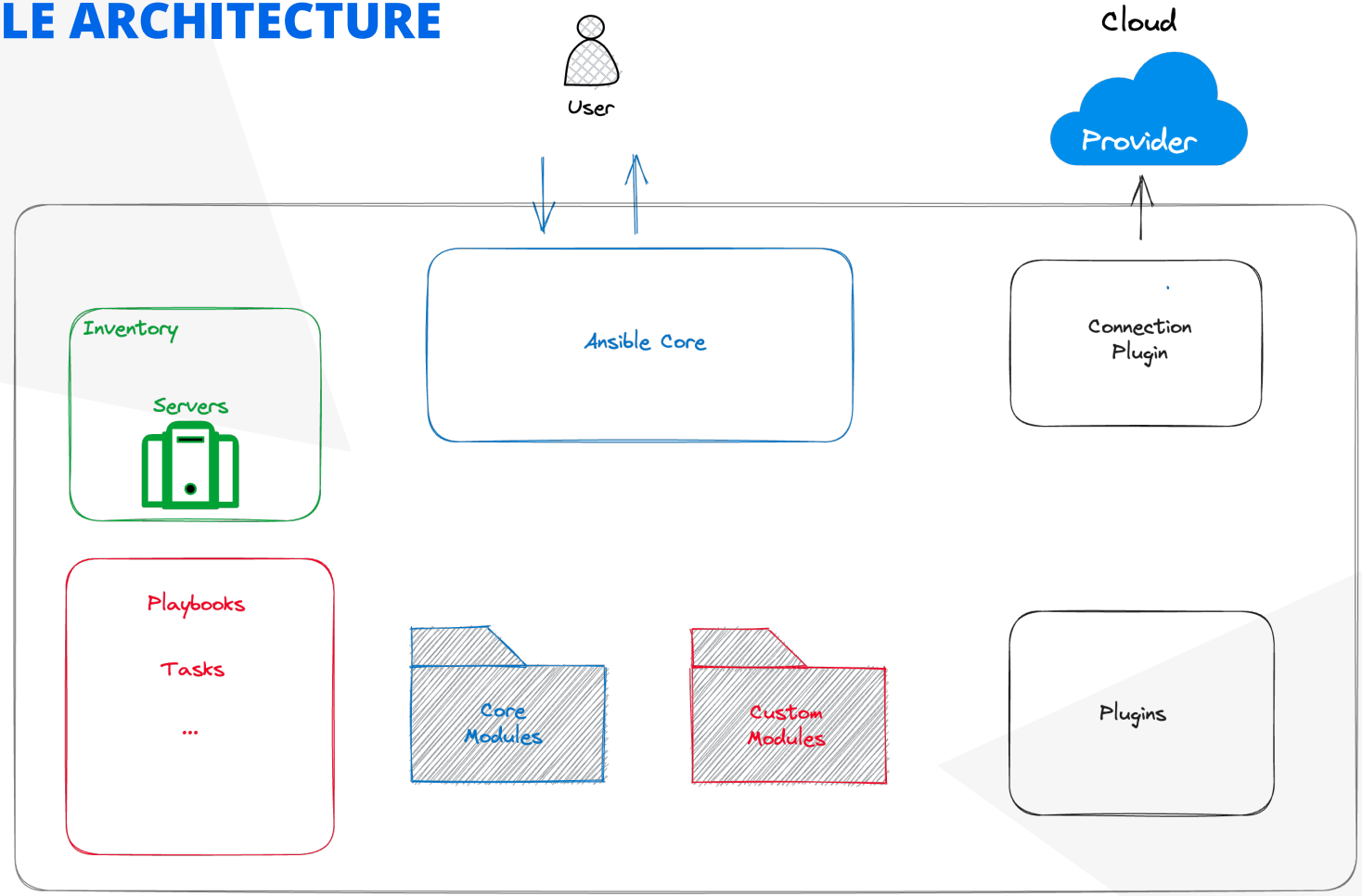
INTRODUCTION TO ANSIBLE

FEATURES OF ANSIBLE

- Open source project (from RedHat)
- Push-based (agent-less)
  - Remote execution of Python code (Ansible Modules)
  - Communication via SSH
- Many ready-made modules
  - Currently over 2000 modules
- YAML as a description language for playbooks
- Implemented in Python (modules)



ANSIBLE ARCHITECTURE



NEWS FROM ANSIBLE 2.9

- New versioning after version 2.9
- Ansible Community specifies the dependency to Ansible Core
- Division into Ansible Community (2.10 / 3.0.0) and Ansible-Core (Ansible-Base 2.10 for short)
- Ansible core continues with versions 2.11, 2.12, 2.13,...
  - Currently 2.15.6 (as of Nov. 2023)
  - Minimal set of runtime and built-in modules
- Ansible Community with new versions 3.0.0, 4.0.0, 5.0.0, ...
  - Currently 8.6.0 (as of Nov. 2023)
  - Includes collections and many other community modules

SETUP REQUIREMENTS OF ANSIBLE

Requirements

- Control Node
  - **Python**
  - **Ansible software package**
  - **SSH connection to all nodes**
    - Via OpenSSH or paramiko (Python-Lib)
  - + typically Git
    - Repository of the infrastructure code
- Managed Nodes
  - **Python**
  - **SSH user** (with Sudo rights)
    - SSH keys recommended
  - Windows Nodes: WinRM

- One advantage of Ansible is that **no adjustments** are **usually necessary** on the managed nodes
  - Python and SSH mostly available
- The push approach means that no agent daemon needs to run on the nodes
- Windows currently only supported as a managed node

WHY ANSIBLE?

- Flat learning curve
  - Quick to learn with a little scripting knowledge
  - System administrators are quickly productive
- Large community
  - Very active GitHub project
  - Many ready-made solutions (roles & collections)
  - Active further development
- Backing by RedHat
  - Also Enterprise Support (Ansible Engine)
  - Ansible Tower (AWX): UI attachment with roles/rights concept

FIELDS OF APPLICATION OF ANSIBLE

- Provisioning of server farms (automated installation)
- Remote control of server farms (ad-hoc tasks)
- Automation of a local setup
  - E.g. set up developer workstation
- Application deployments
  - Rolling updates possible through serial processing
- Self-service for users
  - E.g. provisioning of a test environment
  - through Ansible-Tower / AWX

COMPONENTS OF ANSIBLE

- Playbooks: Configuration code (YAML format)
  - Tasks (here **Install vim**) call Ansible modules (here **package**)
  - Modules are transferred to nodes and executed there
- Inventory: List of hosts known to Ansible (INI format)
  - Also yaml or json possible

```yaml
---
- hosts: localhost
  become: yes
  tasks:
    - name: Install vim
      package:
        name: vim
        state: present
```

```
localhost

[nodes]
node1
node2
```

STARTING THE EXERCISE ENVIRONMENT

- Starting a terminal
- Connect to the shared server
  - `ssh {{ user }}@oc-ansible-training-all.centralus.cloudapp.azure.com`
  - PW: `Welcome{{ USER }}`
- Clone the Git repo there in your own home directory
  - `cd git`
  - `git clone https://github.com/schwenne/ansible-schulung.git`
- Switch to task 0 and install Ansible.

- Demo: Working with VSCode on the server.