10. ročník Žilina 2. – 4. 7. 2018
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
osssconf.soit.sk
OSS Conf

OSSConf 2019: 1–8

AUTOMATION OF MASS COMPUTER CLASSROOMS MANAGEMENT USING ANSIBLE

MATEJ MADEJA (SK) AND MIROSLAV BIŇAS (SK)

Abstract. Managing a large number of computers has been generally a major challenge for many years, especially at universities. Linux systems administrators are mostly familiar with scripting that is quite reliable, but managing Windows–based devices is still an issue. Many tools tries to help with the management of such devices, e.g. using group policies of the operation system or external tools. This paper describes the use of IT automation tool *Ansible* for Windows–based computer classrooms administration at the Technical University of Košice. The article describes 2 years of practical experience at university and presents particular implementations within the project *anstall*.

Key words and phrases. automatisation, ansible, mass computer management, administration, installation.

AUTOMATIZÁCIA MASOVEJ SPRÁVY POČÍTAČOVÝCH UČEBNÍ POMOCOU ANSIBLE

Abstrakt. Správa veľkého počtu počítačov je vo všeobecnosti veľkou výzvou už dlhé roky, a to najmä v univerzitnom prostredí. Administrátori linuxových systémov sú väčšinou zvyknutí používať skriptovanie, ktoré je relatívne spoľahlivé, problémom je však správa zariadení s operačným systémom (OS) Windows. Rôzne skupiny nástrojov pomáhajú so správou takýchto zariadení, napríklad využitím skupinovej politiky systému alebo externými nástrojmi. V článku popisujeme využitie automatizačného nástroja *Ansible* pre správu počítačových učební s OS Windows na Technickej univerzite v Košiciach. Článok popisuje získané skúsenosti počas jeho 2 ročného používania v praxi a prezentuje aj konkrétne riešenia v rámci projektu *anstall*.

Kľúčové slová. automatizácia, ansible, masová správa počítačov, administrácia, inštalácia.

Introduction

Software in today's operating systems is a large part of the user experience and end-user motivation to work with the device. In particular, for user are important the simplicity and purpose of the software. The user uses the application in order to simplify or speed up a particular task, meaning software brings a benefit to him/her. To use the software, it is also necessary to install, regularly update and, in general, manage the application. Most of mentioned tasks are often covered by an operating system (OS). According to Bacchus [3] in today's OS can be seen an effort to simplify the software maintenance process so it is as automated as possible and relieve the end user from unnecessary tasks. At the same time, it is an effort to perform software maintenance activities at the time when the device is not used or when the user does not mind.

Software management of one device is relatively sufficient covers the operating systems with end-user cooperation. The issue is management of multiple devices at the same time, such as at large companies or schools that manage hundreds to thousands of devices at once. The reality is that devices are often divided between multiple administrators who manage a particular group of devices. The software update is usually handled by the operating system itself, but not always in the desired time. At universities it often happens that an OS update starts during the test when the device usage time is limited. A similar situation can also occur in a business environment.

The biggest issue is installing new and configuring already installed software. If the administrator manages 50 computers, it is necessary to perform the same operations on each device, which is very time consuming. In addition, errors can occur during manual configuration , therefore, different devices can be differently set up and device inconsistencies may occur with subsequent difficult troubleshooting. The variety of target operating systems also makes installation difficult.

This paper describes the project *Anstall*[1], used for mass management of computer classrooms at *Department of Computers and Informatics* of *Technical university of Košice*. Project is available at `https://github.com/madeja/anstall` and is based on open–source IT automation tool *Ansible*, focusing mainly on OS *Windows*. In the following sections, some approaches to support mass management of devices, the design of the project structure and practical usage experiences are described.

1. Software distribution approaches

There are currently several approaches that help distribute software and can be categorized into the following groups:

Cloning: The initial installation is done on one device and then a system or a disk image template is created. Finally the image is distributed to other computers. This is only possible if the hardware features of all computers are the same, otherwise deployment problems may occur, for example, not all OS can adjust the hardware change or the disk image will not be usable due to lack of space on the target device. This is quite useless for deploying software after it is already installed, but it's very useful for the initial setup. Examples are *Clonezilla*[2], *HDCone*[3] or *Macrium Reflect*[4].

[1]Title created by joining words *ansible* and *install*.

[2]`https://clonezilla.org/`

[3]`https://www.miray.de/products/sat.hdclone.html`

[4]`https://www.macrium.com/reflectfree`

Scripting: It is possible to prepare scripts and/or logon scripts to run actions (e.g. install new software) on a system. This solution is platform dependent and assumes scripting knowledge for a particular O shell. For example, for Unix-based systems it is possible to use *bash* scripts, for OS Windows using Powershell *cmdlets* [2]. The disadvantage of this solution is the high complexity and the necessity of programming relatively complex programs, where it is often necessary to wait for the operation completion, wait for resources, verify the availability, etc.

Group Policies: Accoding to [5] if a windows installer file (MSI) is available, so the software is packaged, installing is pretty easy. However, this solution is very limited to MSI files and cannot guarantee the same environment for each device. For example, OS settings may be manually changed and the current configuration may cause installation failure, thereby failure identification and special attention to each device is required.

MS Intune: Microsoft [5] offers the ability to manage computers and mobile devices through the cloud and also offers *Intune for Education*[6], which was created exclusively for school purposes, but both are proprietary and charged. The functionality is very similar to our solution but the ability to manage all devices from a browser provides a more user friendly environment. Since *Ansible* is primarily targeted on Unix-based OS, it cannot provide all the benefits of the *Intune*. On the other hand, it provides a more global system configuration, such as service control, registers, etc.

Other vendors: Specialized tools that are mostly proprietary and provide remote computer management. However, these solutions are more business oriented and often not adapted to the university environment.

Very often solution are so-called *thin clients* that have a different architecture organization in the network, which consists of a server and clients. In recent years thin clients have proven to be very advantageous in the educational environment (see [1, 4, 6]), especially for easy client management, high availability and energy efficiency. Despite mentioned facts these solutions are often not enough powerful for programming courses, so classical classrooms with multiple computers are often built today in technical schools.

2. *Anstall* as a general purpose management solution

Because funds for such tools, mentioned in Section 1, mostly are not reserved in the education institutions, we were looking for an open-source solution with the ability to manage multiple Windows-based nodes. Since 2017, when the *Ansible* 1.8 released with Windows modules support, we started use it as the main computer management platform. *Anstall* project is the result of our two-year

[5]`https://www.microsoft.com`

[6]`https://www.microsoft.com/en-ca/education/intune/default.aspx`

experience of installing various software for different courses at our university. All directory paths presented in the following sections refer to the *Anstall* project.

2.1. Task groups

For grouping relevant tasks to be performed on target devices we use ansible playbooks. In its basis, ansible uses adhoc task execution mode, using playbooks (`./playbooks`) tasks can be performed in a specific order, which is very well suited to deploying complex setup. In our solution, a playbook represents one classroom and it can contain multiple plays in a YAML file (see Listing 1). Particular play executes different roles where a role represents a group of tasks with a specific goal, e.g. in Listing 1 *general* role includes base manipulation with computers (reboot, messaging user, services management, etc.) and *courses* role includes tasks focused on software installation for courses taught in the classroom (install new software, configuration, etc.). Sometimes it is not necessary to run all plays of the playbook, so we use tags that distinguish a particular play or play group, that creates a set of related logical units.

Listing 1. Example of play with 2 roles for classroom B512 playboook.

```
- name: General manipulation with computers and courses install
  hosts: b512
  roles:
    - general
    - courses
  gather_facts: no
  tags:
    - general_courses
```

Each play performs defined roles, which are contained in `./playbooks/roles` directory. The role is a separate unit for which variables, templates, necessary files and the like can be defined. The most important part is the `tasks` in particular role directory, where the tasks to be performed are located. The tasks are performed by modules[7], which creates a base of ansible automation. We did not use templates in our case because we did not implemented such a complex configuration. The necessary files can be distributed using the `files` directory placed in particular role using classical copying or by downloading using http protocol.

2.2. Different classrooms installation and permissions

Roles can be used multiple times and for different classrooms, so they should be implemented for general use because their main advantage is simple distribution by copying. Devices in each classroom are defined by thier IP addresses in the

[7]`https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html`

`./hosts` inventory file and can be grouped, too (Listing 2). One device can also be included in multiple groups. It is possible to create multiple inventory hosts files and use a specific one when executing ansible (using `-i` switch of `ansible-playbook` utility), but in our case each group is represented by a specific playbook. This approach has proven to be more useful for sleeping than to define separate inventory files. This approach has proven more useful in managing classrooms than defining separate inventory files.

Listing 2. Hosts file defining groups by classrooms.

```
[b512]                    # classroom B512
147.232.34.215
147.232.34.205

[a537]                    # classroom A537
147.232.34.227
147.232.34.228
```

Each computer has both administrator (admin) and user account (student). The administrator account is considered main maintenance account, e.g. for software installation, so for remote control it is necessary to provide login and password for individual classrooms (groups). Variables of this type can be defined directly in the inventory files, but for clarity we keep them in separate files for every single classroom (`./group_vars`), example in Listing 3. If it is necessary to install a user specific software, admin account installation must be copied for the student account, so it is not possible to install under another account. For storing passwords it is recommended to use Ansible Vault[8], for presentation purposes we store password as plain text.

Listing 3. Example of classroom specific variables.

```
ansible_user: admin
ansible_password: mypwd
ansible_port: 5986
ansible_connection: winrm
ansible_winrm_server_cert_validation: ignore
```

2.3. Package manager for Windows

There is the *Chocolatey*[9] package manager for Windows that makes it easy to install new software. In the past, the package manager had to be pre-installed and subsequently ansible could use it. In the current version, ansible offers automatic installation of *Chocolatey* when using the `win_chocolatey` module.

[8]https://docs.ansible.com/ansible/2.4/vault

[9]https://chocolatey.org/

2.4. Running playbooks

The best way how to execute an ansible playbook is using a server that is accessible by all nodes that need to be configured. This way it is possible to make a remote installation from anywhere. It is possible to use following statement to run a specific playbook:

```
ansible-playbook playbooks/b512.yml \
        --tags "prog-block,programming-course"
```

Defining tags only a specific logical group of tasks or plays can be performed. If you need to run multiple playbooks at the same time, you can create a new playbook, which includes other playbooks (Listing 4).

Listing 4. Multiple playbook execution, using playbook of playbooks.

```
- include: playbook-one.yml
- include: playbook-two.yml
```

3. Special solutions from practice

The academic environment has led us to special solutions to the problems that have occurred. In the sample examples of *anstall* project following solutions are partially implemented.

3.1. Daily cleanup

When working with computers students often leave changes to the file system that are not needed for others. In general, most file-spam folders in Windows OS are *Downloads*, *Desktop*, *Documents*, etc. In Windows there is possible to create a temporary user account with automatic deletion of changes made. However, this method is too computationally difficult because the account is created every time the user log in and deleted immediately after user logout. The other issue is account configuration, which needs to be implemented after every recreation.

Our solution is to define the particular structure and files to be included in the folders. E.g. *Downloads* and *Documents* directories can be empty for out purposes, on the other hand, *Desktop* directory will contain links to specific applications (see role `./playbooks/roles/cleanup`). Other files can be removed periodically, in our case every day at night.

3.2. Faster licence key deployment

Some tools, e.g. integration development environment (IDE), require manual license confirmation on each device. In this case, it is difficult to automate this action and needs to be done manually. Also in this case ansible can be partially helpfull, e.g. copying the license key file into clipboard using windows Powershell (module `win_shell` and executing `type license-key.txt | Set-Clipbard`).

The key is automatically copied to clipboard and maintainer only needs to paste it into the desired field.

3.3. Automatic updates

Automatic updates are often a nightmare, as they often start to execute at system boot or shutdown, which is the worst time period in a university environment. Our solution is to disable Windows update service and enable it only if it is needed or periodically in the night hours. During updates, maintainers should be specially careful because ansible configuration on destination device can be disrupted, e.g. the Window 10 version 1809 update caused ansible to malfunction and manual re-configuration has been required. For updating Windows there is a `win_update` module which is very helpful. An example of our solutions is presented in `./playbooks/roles/general`.

3.4. Exam restrictions

Restricting the internet to prevent cheating during exams is a common academic issue. We use the classic `win_shell` module to modify firewall settings causing blocking/unblocking necessary websites in a second. Using `win_msg`, we are able to show instructions for student before the exam, where they can find materials they are allowed to use and how to behave during the exam. If a student is using a prohibited application it is possible to alert him/her without disturbing other participants or terminate the application. An example of internet restriction is presented in `./playbooks/roles/general`.

4. Conclusions and future work

This paper presented a mass management solution used for computer classrooms at the *Technical University of Košice* using automation IT tool *Ansible.* We created project *Anstall*, whose structure was developed over two years of experimentation. The authors described the current possibilities of software distribution approaches and subsequently described the possibilities of the *Ansible* tool and its unrivaled possibilities suitable not only for the university environment.

The solutions for interesting issues, often arising during computer management of multiple computers at universities, has been described, such as daily user account cleanup, optimization of license key deployment, controlling the automated updates and setting internet restrictions for exams. The project is also publicly available and ready to use by other universities or companies.

Ansible offers much more possibilities than described in this paper, in which we highlighted the most important in terms of use at a university. The structure of the project is continuously changes and is optimized depending on the teachers' software requirements, used for teaching. In the future, we would like to fully automate the management of computers with Wake-on-LAN support, because currently it is necessary to turn on computers manually.

Acknowledgment. This work was supported by project KEGA No. 053TUKE-4/2019: Learning Software Engineering via Continues Challenges and Competitions.

References

[1] Ahmed, S., Ghashem, I. A., Aalsalem, M. Y., and Khan, W. Z. Thin client technology for higher education at universities of saudi arabia: Implementation, challenges and lesson learned. In *2017 International Conference on Computer and Applications (ICCA)* (Sep. 2017), pp. 195–199.

[2] Aiello, J., Schonning, N., Marramaque, R., and Wheeler, S. Cmdlet overview, 09 2016. URL: `https://docs.microsoft.com/en-us/powershell/developer/cmdlet/cmdlet-overview`.

[3] Bacchus, A. Microsoft to make “easier, faster” updates starting with windows 10 version 1809, 08 2018. URL: `https://www.onmsft.com/news/microsoft-to-make-easier-faster-updates-starting-with-windows-10-version-1809`.

[4] Mahdzar, R. Using technology to improve education for cambodian children, 2015. URL: `http://geeksincambodia.com/using-technology-to-improve-education-for-cambodian-children/`.

[5] Microsoft Corporation. How to use group policy to remotely install software in windows server 2008 and in windows server 2003, 06 2018. URL: `https://support.microsoft.com/en-us/help/816102/how-to-use-group-policy-to-remotely-install-software-in-windows-server`.

[6] World Education, Inc. Technical brief, using computer technologies to improve basic education in cambodia: Thin client labs, 2013. URL: `https://www.worlded.org/WEIInternet/inc/common/_download_pub.cfm?id=13309&lid=3`.

Contact addresses

Ing. Matej Madeja, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia, *E-mail address*: `matej.madeja@tuke.sk`, `http://madeja.github.io/`

Ing. Miroslav Biňas, PhD., Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia, *E-mail address*: `miroslav.binas@tuke.sk`