Unleashing Full Potential of Ansible Framework: University Labs Administration

Pavel Masek, Martin Stusek, Jan Krejci, Krystof Zeman, Jiri Pokorny, Marek Kudlacek
Brno University of Technology
Brno, Czech Republic
masekpavel@vutbr.cz

***Abstract*—The proliferation of virtualization coupled with the increasing power of industry-standard servers and the availability of cloud computing has led to a significant increase in the number of servers and end stations that need to be managed with or without an organization. At this point, data center orchestration and configuration management tools come into play as in many cases. System administrators manage groups of identical servers or end stations (physical hosts or virtual machines) which run identical applications and services. They are deployed on virtualization frameworks within the organization, or running as cloud or hosted instances in data centers. In this paper, we consider building up new *layer* for the utilized Ansible orchestration tool. In the realized scenario, more than 10 laboratories at Brno University of Technology were utilized to test our developed framework in case of remote management. To bring the introduced functionality closer to system administrators, an universal web application enabling them to do the configuration changes via the smart devices (e.g., smart phones and tablets) was created and thoroughly tested.**

I. Introduction

While it is still not completely clear, what the Internet of Things (IoT), Machine-to-Machine (M2M), Vehicle-to-Vehicle (V2V), and other types of communication in future wired and mobile systems will become in the end [1], [2], one point is apparent – in the current Information and Communication Technologies (ICT) market [3], [4], the need for agile and reliable techniques capable of shortening the software development cycle as well as multi-platform software development is a must [5]. Software Defined Networks (SDNs) and Network Function Virtualization (NFV) enable *virtualization* and *cloudification* to execute (virtualized) functions as software modules (plugins) under the control of a single orchestrating entity or multiple in case of hierarchical deployment [6], [7].

The above mentioned trend, where the software is used for planning that reduce the space, time and efforts between the software development and operations in real scenarios as well as the technical and organizational gap between mentioned types of research teams has been introduced as DevOps [5], [8]. The DevOps aims to improve communication, collaboration, and integration between software developers (Dev) and IT operations professionals (Ops). As the part of the DevOps, certain actions take over standard tools from software development area e.g., code version systems or code-revision management to manage what is these days known as *Infrastructure-as-Code (IaC)* [9], [10], [11] (Supporting technologies for IaC known as Topology and Orchestration Specification for Cloud Applications (TOSCA) i.e., industrial programming languages for automated deployment of technologically independent and multi-platform/multi-cloud compliant applications are not in the scope of this paper.). Therefore, the goal of the IaC is to provide system administrators with the ability to manage knowledge and experiences of plenty of subsystems from one place instead of traditional approach where each sub-system has its own dedicated administrator.

Against the background, in this paper, we introduce a platform for efficiently managing large-scale university labs infrastructure, with minimal input from developers or local administrators. In our case, the Ansible tool which was developed to simplify complex orchestration and configuration management tasks has been chosen. The platform (framework) itself has been written in Python and allows users to script commands in YAML Ain't Markup Language (YAML) as an imperative programming paradigm. To unleash the full potential of Ansible platform, we build upon the available sources and extend Ansible by an extra layer which allows to manage the university labs from both local and public network infrastructure. To bring the developed functionality closer to sysadmins, a universal web application has been created, which enables to do the configuration changes via using the smart devices (e.g., smart phones and tablets). At the end, the whole platform was extensively tested at Brno University of Technology (BUT), Czech Republic in selected university laboratories where physical hosts and virtual machines with different operating systems and software configurations were utilized.

The remainder of this paper is structured as follows. In Section II, we take closer look at the description of Infrastructure as Code. Going further, Section III presents university lab scenario located at BUT, Czech Republic. Within this section, the utilized configuration management tool Ansible is described together with examples of used syntax. Newly created platform for remote management of university labs is detailed in Section IV. Finally, concluding remarks together with lessons learned are drawn in Section V.

II. Infrastructure as Code

Infrastructure as Code represents the management of network infrastructure (e.g., virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as DevOps team uses for source code [13]. Like the principle where the same source code generates the same binary, an IaC model generates the same environment every time it is applied, see Fig. 1. Therefore, IaC stand for a key DevOps practice and is heavily used in conjunction with continuous delivery.

IaC evolved to solve the problem of environment drift in the release pipeline. Without IaC, research or system teams must maintain the settings of individual deployment environments/sub-systems. Over time, each environment be-

TABLE I. A COMPARISON OF POPULAR *INFRASTRUCTURE-AS-CODE* PLATFORMS [12]

	Chef	**Puppet**	**Ansible**	**SaltStack**	**CloudFormation**	**Terraform**
Code	Open	Open	Open	Open	Closed	Open
Cloud	All	All	All	All	AWS only	All
Type	Config Mgmt	Config Mgmt	Config Mgmt	Config Mgmt	Orchestration	Orchestration
Infrastructure	Mutable	Mutable	Mutable	Mutable	Immutable	Immutable
Language	Procedural	Declarative	Procedural	Declarative	Declarative	Declarative
Architecture	Client/Server	Client/Server	Client-only	Client/Server	Client-only	Client-only

comes a *snowflake*, that is, a unique configuration that cannot be reproduced automatically. Inconsistency among environments leads to issues during deployments. With snowflakes, administration and maintenance of infrastructure involves manual processes which are hard to track and contributes to errors. Going further, idempotency is a principle of Infrastructure as Code – it is the property that a deployment command always sets the target environment into the same configuration, regardless of the environment's starting state. Idempotency is achieved by either automatically configuring an existing target or by discarding the existing target and recreating a fresh environment.

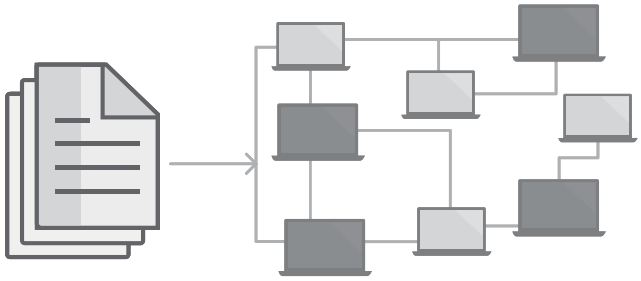

Fig. 1. Infrastructure as Code (IaC) data distribution

Accordingly, with IaC, maintenance teams make changes to the environment description and version the configuration model, which is typically in well-documented code formats e.g., JavaScript Object Notation (JSON). The release pipeline executes the model to configure target environments. If the team needs to make changes, they edit the source, not the target. Therefore, IaC enables DevOps teams to test an applications in production-like environments early in the development cycle. These teams are expected to provision multiple test environments reliably and on demand. Infrastructure represented as code can also be validated and tested to prevent common deployment issues before it is rolled out to the production environment.

A. Orchestration Tools

Today, if you search for "infrastructure as a code" you will probably end up with a list of the most popular configuration management tools i.e., Chef, Puppet, Ansible, SaltStack, CloudFormation, and Terraform [12]. All of the mentioned tools are open source, backed by large communities of contributors, and work with many different cloud providers (Notable exception is the CloudFormation, which is closed source and Amazon Web Services (AWS)-only.).

Chef, Puppet, Ansible, and SaltStack are all "configuration management" tools, which means they are designed to install and manage software on existing servers/end stations. Furthermore, CloudFormation and Terraform are "orchestration tools", which means they are designed to provision the servers themselves – leaving the job of configuring those servers to other tools. These two categories are not mutually exclusive, as most configuration management tools can do some degree of provisioning and most orchestration tools can do some degree of configuration management. But the focus on configuration management or orchestration means that some of the tools are going to be a better fit for certain types of tasks. Putting it all together, Table I shows how the most popular IaC tools sum up.

III. UNIVERSITY LAB SCENARIO

This section details the practical usability of the Ansible framework for the orchestration of computer labs located at the Faculty of Electrical Engineering, Department of Telecommunications, BUT, Czech Republic. The basic options (modules, playbooks, roles) will be described, followed by a specific example we have created.

A. Intended Scenario

In the faculty's premises, a large number of computer classrooms is used for the teaching and research, which according to the purpose of teaching includes sets of physical and virtual machines with different operating systems and different software equipment and settings. However, some settings and applications are common to all of these cases. We developed a scenario that would unify the administration of such classrooms and save considerable amount of time spent on the maintenance of these classrooms. The scenario must be capable of orchestrating all the stations located in the selected laboratory, which is composed of stations with different mainstream operating systems: (i) Windows 10; (ii) Windows 7; (iii) Ubuntu 16.04; (iv) Debian 8. It also has to be able to install and uninstall various programs: (i) Microsoft Office; (ii) Putty; (iii) Wireshark; (iv) Mozilla Firefox; (v) Google Chrome; (vi) Microsoft Visual Studio, and (vii) Total Commander accompanied by their latest updates. In addition, several system operations must be enabled: (i) Displaying information and status of stations; (ii) restarting and shutting down stations; (iii) creating users; (iv) creating and deleting a file; (v) installing system updates. The selected settings will be implemented with the usage of a remote management framework Ansible.

Logical schema of site is depicted in Fig. 2, where the communication infrastructure is specified. Stations are disseminated according to the membership to laboratory, operating system and classroom type. Depending on the design, the cor-

responding application software package and its configuration is defined.

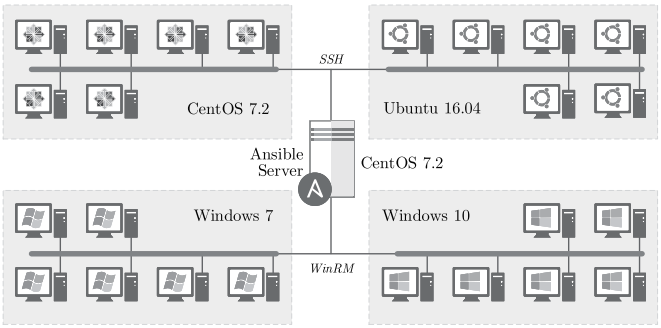

Fig. 2. Scenario of management laboratories with different operating systems

B. Implemented features

The requirements for installation of different types of application were specified as well as operations the intended scenario has to meet. This subsection details the configuration steps that were used to fulfill these needs.

As mentioned above, Ansible framework works on client-server topology, therefore the Ansible server application needs to be installed. To do so, a system with a Unix or Linux operating system that contains Python in version 2.6 and higher (Linux CentOS 7.2 with Python 3 has been chosen in intended scenario.) is necessary. The installation itself was done from a general repository which contains appropriate packages in all major distributions. Further it was necessary to configure Ansible Server to communicate with remote hosts:

- Because the Ansible was primarily developed for the Linux machines management, no additional configuration was needed on Linux hosts and the communication settings was rather simplified. The SSH (Secure Shell) protocol was used. To establish communication between Linux machines, a pair of ssh keys (private and public) was required. Ansible server has a pair of these keys, which represents the identity of the given server and allows the initialization of encrypted connection. The *ssh-keygen* command is used to generate this key.
- To establish communication with Windows hosts, the *group_vars* directory with the windows.yml file was created on the server specifying the configuration for SSH service required to establish server connection with Windows operating systems. It was assumed that each host station has an administrator account with the same name and password. This step is related to the configuration of remote stations towards Windows. These can be managed if WinRM (Windows Remoting) service is properly configured and a valid self-signed certificates are created. The service itself provides communication between the server and the managed station. The HTTPS protocol (Hypertext Transfer Protocol Secure) is connected to port 5986, which enables encrypted communication. These necessary settings can be executed by running the configuration script located on Ansible web pages [14]. The most important aspect of Windows systems configuration is the presence of Powershell console. The minimum version of Powershell compatible with Ansible is 3.0, which is supported on Windows 7 and newer.

After that, groups of managed computers were defined. As Ansible did not use its pre-installed guest agents to perform its tasks, no further installation was required. The hosts intended to be managed must be specified in the inventory located in */etc/ansible/hosts* file which may look like as follows in Listing 1. The listing shows four machines in two classroom groups independent of the used operating systems. Each machine can be a member of several groups.

Listing 1: Ansible inventory example

```
[laboratory-networking]
lab1host1 ansible_ssh_host=192.168.1.1
lab1host2 ansible_ssh_host=192.168.1.2

[laboratory-cryptography]
lab2host1 ansible_ssh_host=192.168.2.1
lab2host1 ansible_ssh_host=192.168.2.2
```

Once the communication chain is set, scripts for remote management could be created. The rudiments of these scripts are modules, playbooks and roles as depicted in Fig. 3.

- Playbook is the file containing the order of the commands, composed of aforementioned modules and is represented by a YAML file.
- Roles are complex structures that include tasks, modules, handlers and files. Role is composed of a directory that has sub-directories which contain a main.yml file, which specifies the sequence of operations to be performed.
- Modules are simple scripts focused on simple system actions. They can be run as a single command or as a part of more complex scripts, called playbooks.

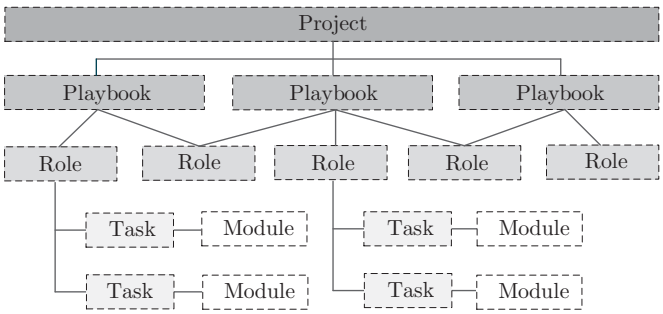

Fig. 3. Structure of Ansible scripts

Playbook is a file containing the commands to be executed on that host. It also defines the host guest group in which the tasks are run. Tasks utilize modules for unattended installation of applications on target operation systems. The following command executes the Playbook.

```
ansible-playbook run-example-playbook.yml
```

The Playbook can have a tremendous amount of information about the host station. If the playbook has to be as simple as possible, you can use a role. Ansible roles help to divide repeatable scripts into separate units. This makes it easy to read and allow the use of other necessary files, such as .sh, .ps, .conf, .msi and other scripts.

Roles are ways of automatically loading certain files, tasks, and handlers based on a known file structure. Certain role is composed of a directory that has a sub-directories which contain a main.yml file. It, same as a playbook, specifies the sequence of operations to be performed. Before the installation itself, it is necessary to determine which type of the operating system will be installed. Another requirement is the availability of the repository application at the manufacturer's site. The last condition is to select available modules from [15] to meet the set of defined requirements.

Modules allow to control machine system resources, such as services, packages, files, permissions, etc. An example of a module for getting system information from a guest host may be as follows

```
ansible -i hosts windows -m setup
```

The most important module used in intended scenario, is module *win_chocolatey* [16], which allows to install and uninstall any application for Windows. Structure of module for Mozilla Firefox installation is shown in Listing 2.

Listing 2: Install Win app via Chocolatey

```
win_chocolatey: # module name
   name: firefox # parameter name
   state: present # parameter state
```

The name parameter specifies the name of the application to be installed. The state parameter represents desired state of the application. Here, the *present* parameter defines that the application should be installed. A summary of the remaining parameters is available on the Chocolatey web pages [16]. The *Chocolatey* module allows installation of large applications but some of them are available only on the manufacturer's pages. This is also the case when applications from Microsoft are installed. For such applications it is necessary to define the exact URL on which the *.msi* or *.exe* file is needed to install the application. To install the application on Windows from the web or local store, the *win_package* module is used. The showcase [17] playbook for installing applications from different sources is given on Listing 3.

Listing 3: Install Win apps from different sources

```
name: Install applications
hosts: windows
tasks:
   # Install app via Chocolatey
   - name: Install Wireshark
      win_chocolatey:
      name: wireshark
      state: present

   # Install app via MS Store
   - name: Install MS Visual Studio
      win_package:
      path: "https://.../vstudio.exe"
      product_id: "{DE0-...8B-3638}"
      state: present

   # Download app from URL
   - name: Download the MS Office
      win_get_url:
      url: http://.../office2010.msi
      dest: C:\....\office2010.msi

   # Install app from local source
   - name: Install MS Office
      win_msi:
      path: C:\...\office2010.msi
      state: present
```

Registers are checked before installing the application. If the keys within the registry entry were set, the application will not be installed as it proves that the same version of the application is already installed. Applications with different versions can be installed adjacently.

As for Windows and Linux, not all applications are executable on the target operating system. For Linux distribution, an application such as Microsoft Office or Visual Studio IDE (Visual Studio IDE (Windows only) and Visual Studio Code (installable on Windows, Linux and Mac) are different software.) will not be installed. Before installing an application on Linux systems, it is necessary to determine which distributions will be used. Each distribution uses different installation tool or package. To install the application on CentOS, the *apt* module is used, which is similar to *win chocolatey* for Windows. The *apt* module allows the administrator to install the application, the .rpm package, and also update the distribution. The *apt* module for installing Mozilla Firefox follows in Listing 4, syntax is similar to *win chocolatey* module.

Listing 4: Install Linux app via apt

```
apt:
   name: firefox-esr-l10n-cs
   state: present
```

After application installation or during regular maintenance, remote reboot or shutdown might be required. For this feature *win_reboot* and *win_shell* modules are utilized for Windows and Linux, respectively. The *win_reboot* module represents the Powershell command to restart the station. To restart Linux stations, the shell module is used to directly transfer shell commands for restarting or shutting down machines.

When all the required applications are installed on the system, user accounts creation is necessary. Ansible is able to create user accounts for both Windows and Linux. It utilizes *win_user* for Windows and *user* for Linux. Both modules have a number of settings, such as ensuring that the user is assigned to the required group, or the password is changed or new one is set. On Linux systems it is possible to create a private ssh key for the created user.

In order to remove and uninstall the applications, it is not necessary to use new module but the *absent* parameter in configuration file is used, as follows in Listing 5.

Listing 5: Remove installed applications

```
   # Remove Linux app via apt
   apt:
      name: firefox-esr-l10n-cs
      state: absent

   # Remove Win app via Chocolatey
   win_chocolatey:
      name: wireshark
      state: absent
```

This setting performs desired operations i.e., application uninstallation and even deletion of files, folders or users.

The necessary step in managing the computer lab is installation of system updates. Updating the system is the most important operation as the up to date system ensures robustness and compatibility of HW with SW. For updates, Ansible uses the *win_update* module for Windows and the *apt* module for Linux. Windows OS allows to install only specific updates such

as security patches or application-specific hotfixes. If updates are not specified, all of them will be installed.

IV. Remote Management of IaC

In the previous section, remote management of host stations via Ansible was tested. It was verified that Ansible is powerful undemanding tool ensuring reliable management of small but also extensive networks with stations utilizing various operating systems, i.e., Linux or Windows.

Despite the fact that Ansible is easy to learn it still enforces the user to use a terminal window for creating playbooks and launching commands. Over the time, with growing number of stations, this attitude becomes time-consuming and ineffective. Moreover, the user is limited to connect via local network. Otherwise the connection via VPN (Virtual Private Network) is needed. It brings additional steps to the network management like the need for another password and login procedure.

These drawbacks could be solved by web interface accessible through the external network providing management over GUI (Graphical User Interface). Such user interface is accessible via standard web browser through the specific URL.

The aforementioned approach requires the cooperation of several technologies, see Fig. 4 including Ansible, web server ensuring remote access through the HTTPS and web application providing user interface.

A. MVC framework

Majority of the modern web application is written in JAVA, ASP.NET (Active Server Pages) or PHP (Hypertext PreProcessor) enhanced by framework extending their basic capabilities [18]. Authors decided not to break the paradigm and built the application on top of Spring framework written in JAVA.

Spring is JEE (Java Enterprise Edition) framework designed to relieve development of enterprise applications. Framework is based on the modern principle of MVC (Model-View-Controller) architectural pattern. It allows structuring of the created application into logical and functional units in accordance with its functionality. Each part of MVC model ensures specific task [19].

- **Model** – represents Java object containing applications data, i.e., username and password. Model is the only part of the application directly communicating with the database. Other parts of the system communicate with the database through the model's public functions. This approach allows a high level of data abstraction provided by the model.
- **View** – represents Java object which contains information about graphical layout. From the perspective of end-user, it is the only visible part of the system as view provides the interpretation of data in the user interface. Spring framework utilizes special files called JSP (Java Server Pages) which allow transformation of code written in Java into HTML web pages accessible through the web browser.
- **Controller** – represents a bridge between model and view thus it is the most important part of the MVC pattern. When an incoming request occurs, the controller has to acquire data from the model and provide them to the view, which then ensures graphical interpretation.

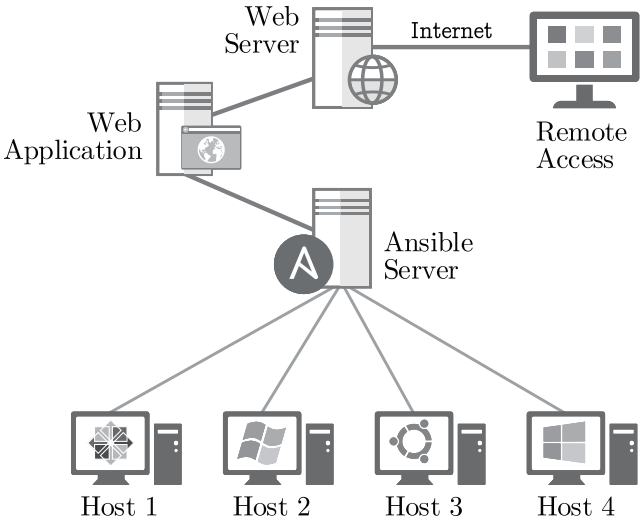

Fig. 4. Architecture of Ansible remote interface management

Fig. 5 depicts architecture of the Spring framework. The example shows processing of HTTP request on */home* page. The main parts of the architecture are [20]:

- **Dispatcher Servlet** – represents main control point of the application. Servlet ensures communication within the application, process incoming HTTP requests and generates HTTP responses.
- **Handler Mapping** – interface with a mechanism that allows finding of proper controller attached to the request address. In the aforementioned example the handler is looking for controller attached to the */home* servlet.
- **View Resolver** – receives the name of the requested view to find and return the view object to render.

All components above are native part of the Spring framework. Such an approach eases developer's work who implements only model, view, and controller objects bundled with the container via the Annotations. The remainder is handled by Spring framework without developer intervention.

B. Created web application

Ansible in its free version does not provide users with GUI (Graphical User Interface) therefore configuration is done completely in a text console. However, this solution requires perfect knowledge of all configuration commands. Beside this, administrator has to be on the same network as Ansible server or VPN (Virtual Private Network) connection to the local network is needed. Those are the main reasons why the web application with user interface was created.

The application is available through the web browser, and it is divided into several subcategories according to its functionality.

Access to the application is restricted via the login dialog. It is not possible to enter the application without valid credentials. To ensure a high level of security, the application is accessible only via HTTPS connection. It protects system against MITM (Man in the Middle) attacks because transferred data are encrypted with TLS (Transport Layer Security) protocol. After successful login, user is redirected to the home page

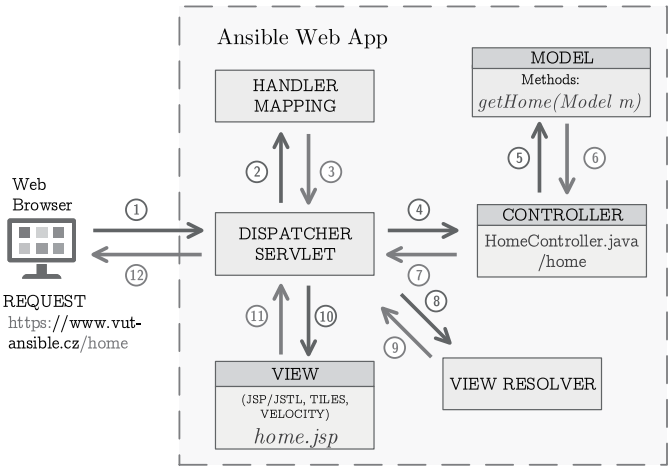

Fig. 5. Architecture of Spring MVC framework

that serves as a signpost to the next sections of the application as it is depicted in Fig 6. Via the application, it is possible to create new playbooks, edit playbooks, host files or delete the records.

However, the crucial part of the developed application is its ability to execute created playbooks on available hosts. The system allows launching the whole playbook or even a single command. Every playbook can be firstly launched in check mode which serves as a test of syntax and functionality. Application screen output is the same as from console, see Fig. 7; thus the user is fully informed about current progress and results. If it is not required to launch the whole playbook, a single command can be launched from the menu.

As company or university could have more than one administrator, it is better to distinguish their access to the application. Therefore, the created platform allows management of user accounts. Security of user credentials is ensured by hashing algorithm Bcrypt [21].

V. Conclusion

Today, the infrastructure design is the software life-cycle phase that defines and configures the software infrastructure needs for that software as well as the number and type of physical hosts or virtual machines required. Infrastructure design typically consists of many installation and configuration scripts needed to, among others: (i) instantiate and link the required machines (either physical or virtual) for the software to run, (ii) install and configure the required software and middle-ware for those virtual machines, (iii) instantiate and run the needed ancillary services for the software to be operated.

Following the current trends known as Infrastructure-as-Code, we have created new layer for the Ansible framework which stands for the orchestration and configuration management framework. Newly created layer for Ansible offers management of university labs at Brno University of Technology, Czech Republic from local and public network. Also, the new web interface has been created to simplify the configuration of tasks which are performed on daily basis – the created application is written in Java programming language utilizing the Spring framework.

Fig. 6. The showcase of implemented actions within the main menu of created BUT Ansible application

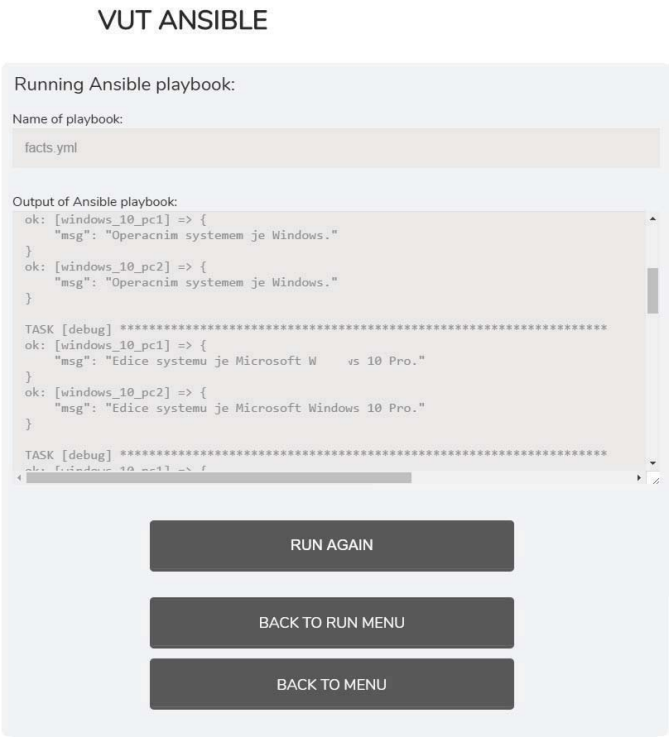

Fig. 7. Output of executed playbook which was chosen from predefined list. As the playbooks are designed to be human-readable, information about the operating systems are pro-vided

With the outlined goals accomplished, the management of machines (physical or virtual) from both the internal university network and public network was thoroughly tested via the web-interface and as an output, it can be stated all required software changes were successfully accommodated. The created framework represents a convenient tool enabling secured remote management and configuration of the selected network or network parts.

ACKNOWLEDGMENT

The described research was supported by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Center was used.

REFERENCES

[1] J. Hosek, P. Masek, S. Andreev, O. Galinina, A. Ometov, F. Kropfl, W. Wiedermann, and Y. Koucheryavy, "A SyMPHOnY of Integrated IoT Businesses: Closing the Gap between Availability and Adoption," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 156–164, 2017.

[2] S. Andreev, D. Moltchanov, O. Galinina, A. Pyattaev, A. Ometov, and Y. Koucheryavy, "Network-assisted device-to-device connectivity: contemporary vision and open challenges," in *European Wireless 2015; 21th European Wireless Conference; Proceedings of*, pp. 1–8, VDE, 2015.

[3] E. Olshannikova, A. Ometov, Y. Koucheryavy, and T. Olsson, "Visualizing Big Data with augmented and virtual reality: challenges and research agenda," *Journal of Big Data*, vol. 2, no. 1, p. 22, 2015.

[4] I. Farris, A. Orsino, L. Militano, A. Iera, and G. Araniti, "Federated IoT services leveraging 5G technologies at the edge," *Ad Hoc Networks*, vol. 68, pp. 58–69, 2018.

[5] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015.

[6] Q. Duan, N. Ansari, and M. Toy, "Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks," *IEEE Network*, vol. 30, no. 5, pp. 10–16, 2016.

[7] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1061–1075, 2017.

[8] G. X. Kolometsos, C. Xilouris, A. Rocha, J. F. Hidalgo, S. Siddiqui, S. Castro, F. Vicens, and J. Martrat, "DevOps based service orchestration in 5G virtualised Networks.,"

[9] K. Morris, *Infrastructure as code: managing servers in the cloud.* " O'Reilly Media, Inc.", 2016.

[10] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "DevOps: introducing infrastructure-as-code," in *Software Engineering Companion (ICSE-C), 2017 IEEE/ACM 39th International Conference on*, pp. 497–498, IEEE, 2017.

[11] D. Palma and T. Spatzier, "Topology and orchestration specification for cloud applications (TOSCA)," *Organization for the Advancement of Structured Information Standards (OASIS), Tech. Rep*, 2013.

[12] J. O. Benson, J. J. Prevost, and P. Rad, "Survey of automated software deployment for computational and engineering research," in *Systems Conference (SysCon), 2016 Annual IEEE*, pp. 1–6, IEEE, 2016.

[13] Y. Jiang and B. Adams, "Co-evolution of infrastructure and source code: An empirical study," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, pp. 45–55, IEEE Press, 2015.

[14] S. Thakur, S. C. Gupta, N. Singh, and S. Geddam, "Mitigating and patching system vulnerabilities using ansible: A comparative study of various configuration management tools for iaas cloud," in *Information Systems Design and Intelligent Applications*, pp. 21–29, Springer, 2016.

[15] L. Hochstein and R. Moser, *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way.* " O'Reilly Media, Inc.", 2017.

[16] M. Balliauw and X. Decoster, "Automated Delivery," in *Pro NuGet*, pp. 179–214, Springer, 2013.

[17] M. Mohaan and R. Raithatha, *Learning Ansible*. Packt Publishing Ltd, 2014.

[18] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[19] D.-P. Pop and A. Altar, "Designing an MVC Model for Rapid Web Application Development," *Procedia Engineering*, vol. 69, pp. 1172 – 1179, 2014. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

[20] J. Sharma and A. Sarin, *Getting started with Spring Framework: a hands-on guide to begin developing applications using Spring Framework*. [S.l.]: CreateSpace, 2016. The book can be consulted by contacting: EN-ACE-AMM: Pater, Lukasz Piotr.

[21] N. Provos and D. Mazieres, "Bcrypt algorithm," USENIX, 1999.