Welcome to the Ansible for Windows Hands-on Lab!

To be able to execute playbooks on an end-point you need a few things:

1. A Playbook
2. An Inventory
3. A Credential

We have prepared a set of nice Windows playbooks for you to play with (and modify!). However, we have not created an inventory nor a credential, because we don’t know upfront what you name your VM, your username and your password. So, you need to create these in the following 2 exercises.

1. Create an inventory

- Open a new tab on your browser and go to https://tower.redhatworkshops.nl
- Login into Ansible Tower using your admin account wsadmin{x} (where {x} is your unique assigned number. The password is **R3dH4tW0rkSh0p** (sorry for the strong password, but this is a publicly reachable environment)

- Choose in the main menu (the grey vertical bar at the left of the window) for “Jobs”.

  This is a list of all the executed jobs that the logged in user has access to. Find the last executed job with “azure_provision_windws_vm” in the name. If there is a green colored dot in front of it, that job finished successfully. If it’s red, call the instructor.

- Click on the last executed job with “azure_provision_windows_vm” in it’s name

  Now you see the job details pane. On the right side you see the logging of all tasks. The last task (called “debug”) shows the public IP address that has been assigned to your provisioned VM by Azure. Write down this IP Address (or copy it into your clipboard)

- Choose in the main menu (the grey vertical bar on the left of the main window) for “Inventories”
- Click on the green “+” sign to create a new “inventory”
- Give it a name and a description of your choosing and click “save”
- Within your inventory click “Hosts” and then the green “+” sign to add a Host to this inventory manually.
- As the name, use the Public IP Address you got from the logging. Choose a description yourself and click “save: again.
- Within your inventory choose “permissions” and give the **team** “team{x}” *use* permissions. To do this click the green “+” sign, choose the “teams” tab and select team{x} (you can use the search bar or go to the right page yourself). Next, choose the role “use” and click save. See how that looks like in the permissions page. You see that wsuser70, being member of a team, has “use” rights on this inventory, which means it is able to run playbooks against hosts in this inventory. If you want to know the actual team where this is coming from, hover the mouse over the “use” permission in the “Team Roles” column.

Note: While you are here.. You can imagine manually updating an inventory is rather tedious. That’s why there is such a thing called dynamic inventories. Here, Tower simply asks an external source (such as Azure) to give it the hosts. These are called “sources”. A single inventory can have multiple sources. To be able to sync from a source you need credentials. With the inventory itself along comes all kinds of metadata on those end-points that can be used in playbooks. Cool, huh?

2. Create a machine credential

There are a lot of different kind of credentials: Credentials for syncing inventories and/or source control repositories, credentials to private or public clouds and, of course, credentials to get into the end-points themselves to do nifty stuff. In this exercise you are going to create a “machine” credential for the VM you just provisioned in Azure. A machine credential gives you the ability to remotely login into the end-point. Remember the admin username and admin password you had to specify when creating the VM? Those are needed now..

- Choose “Credentials” in the main menu. There you already see 2 credentials. Press the green colored “+” sign to add a credential.
- Give it a name and a description and choose your own organisation (org{x}). You can click the magnifying glass icon to get a list.
- Now choose the “Credential Type”. This must be “**Machine Credential**”.
- Based on this choice a lot of new fields appear. As you can see there are a lot of possibilities to specify the machine credential and also to configure a privilege escalation method if needed. For this lab this is not needed, because the configured admin username already has admin rights in the VM.
- In the field “Username” fill in the Admin Username you specified when creating the VM
- In the field “Password” fill in the Admin Password you specified when creating the VM
- click “save”.
- Within your credential choose “permissions” and give the **team** “team{x}” the *use* role.
- Your done!

Note: have you noticed those little question marks all around the user interface in a little grey ball? If you press them you get context sensitive help!

Another note: have you noticed the “Prompt on Launch” checkbox. If you check that, it will prompt you for the password when you run a playbook that uses this credential.

3. Familiarize yourself with the playbooks

A separate set of playbooks are stored for you on gitlab. If you know github, you know gitlab. If not: it is a web-based repository for files to maintain and share code (extremely short summary). Let's have a look at your playbooks:

- Open a new tab on your browser and go to https://gitlab.redhatworkshops.nl
- Login into gitlab using your account wsadmin{x} (where {x} is your unique assigned number. The password is, again, **R3dH4tW0rkSh0p**
- You are presented with a list of "projects". This is currently one project, called Ansible4Windows. Click on this project.
- You now see a list of files. The playbooks in this list are the files with extension .yml.
- Clicking on one of those files will "open" the file and you will see it's content. Have a look around. You now see actual executable Ansible code for Windows type end-points. Do you recognize (some of) the concepts we explained to you earlier?
- Suppose you like to edit one of these files. Simply "open" the file and press the "edit" button in the toolbar. You can change both the content and the filename of the file.
- When your done editing, you need to store the changes. You do this by pressing the green "commit changes" button at the bottom of the screen. Press the "cancel" button to cancel all edits.

4. Projects

How do these playbooks get into Tower? By defining a “Project” in Tower. A project is a set of playbooks from a common repository. This repository can, in theory, be al local (to Tower) directory, but way more common is to reference an external SCM (Source Control Management) tool for that. Gitlab is an implementation of an SCM which is based on git.

- In Tower, choose “Projects” in the main menu. You get presented a list of projects. At this time you only have one project called “MyFirstProject{x}”. We made that one for you already. Click on it!
- Here you see that this project is configured to sync playbooks from git and if you look at the URL, you see it points at your repository in gitlab.
- When does it sync? You can do it manually (press the 🔄 button in the list beneath the details pane). However, do you notice the “Update revision on launch” checkbox is checked? That means that whenever a playbook from this project is executed, it will first sync with its source to make sure it has the latest greatest version. Neat!
- Have a look around the different options here. If the SCM needs a credential to access the repository you can define one in Credentials and reference it here. SCM typically have controls to select a specific version of a repository. You can specify that in the “SCM BRANCH/TAG/COMMIT” field. It’s empty now, which means use the default. For git this is the “master” branch.

Note: Tower has the ability to send notification out on success or failure of executing playbooks. We have preconfigured a notification template for slack. You can find it in “Notifications” in the main menu. To enable it go to “Organizations” in the main menu, Edit your own org{x} and choose “Notifications. Enable Success and/or Failure notification by moving the sliders from off to on. We have a webpage open on the slack channel so we can monitor what is happening.

5. Create a job template.

A job template is an object that brings it all together: Playbook, Inventory, Credentials, etc. Once you have one, you can start a "Job" based on it, which means you execute the specified playbook from the specified project using the specified inventory and the specified set of credentials. So, let's create one:

- In Tower choose "Templates" from the main menu and click on the green " + " button and choose "Job Template". Don't get overwhelmed with all the options and fields. We only need a few for this lab.
- In the exercise we are going to deploy an IIS webserver and a website, so make the name of the template something like "windows_install_webserver" and give it a description of your own choosing.
- In the field Inventory, select the inventory you created previously
- In the field Project, select the "MyFirstProject" project (it likely to be already filled in)
- In the field Playbook, choose the playbook "windows_install_webserver.yml"
- In the field Credential, choose the machine credential you previously created. Now suppose this playbook does stuff on different end-points. You can then specify multiple credentials here. Also remember that credentials are part of the RBAC controls, so you only get to use the credentials you are allowed to by the (org) admin.
- Ignore all other fields and press the green "save"
- Go to the permissions tab and give the **team** "team{x}" *execute* role.
- You're done!

6. Run a job template.

And now, Ladies and Gentlemen, the proof is in the pudding! Let's run the thing:

- **Log out** and log in again as wsuser{x}. (you can also run the job template as wsadmin, but we have created a wsuser{%} account that only has read/use rights to showcase RBAC).
- Go to "Templates" in the main menu again and find your freshly baked job template.
- There is a launch button there:

  🚀
- press it!
- You have now started a "job". Each job has a unique job-number (that is prefixed to the jobname). You now have the job details window in front of you. On the left are job details and on the right is the logging so you can see what is happening. (if you want more verbose logging, you can configure that in the job template under "verbosity".
- This job will take some time (7-10 minutes). You earned a coffee or tea, I would say…
- Also, now is a good time to study the code in the yaml file on gitlab you are now executing.
- When it is finished you should find a website at the public IP address of the VM. ..

1. And now: Hack!

So, now that you have all the bits in place, have some fun! Go and play around with all the playbooks we gave you on gitlab. That is, in our opinion, the best way to learn it. We are around if you have questions. Here are some things to try out:

- When you have created a job template, there is a button called “add survey”. When you press that you can add a survey to the playbook. Then, when you run it, it comes up first and you get to enter some data (remember the job template you used to deploy the VM). To make this useful for a playbook, you need to edit the playbook and use the variable name from the survey somewhere in there with double accolades around it. Like {{ my_var_name }}. Great playbooks to try this with are the playbooks windows_files.yml, windows_create_user.yml, windows_install_packages.yml, or create a brand new playbook!
- You can also create workflows! A workflow is a sequence of playbooks. To create one, go to templates and create (using the “ + ” button) a “workflow template”. After you created and saved it, you can use the workflow visualizer to create a workflow of previously created job templates. Just press “start” and you get the hang of it. You can add job templates based on success, error, or always from a previous template. If a job_template has a survey you can pre-populate it as well
- If you want to see what is actually happening in the end-point itself, you can connect to it using an RDP tool. Just use the public IP address the admin username and the admin password. (or, if you have promoted your server to a domain controller, use the username, password and domain name you filled in its survey). We opened the firewall port for RDP.
- Did you screw up beyond repair? No worry, you can make a job template for the “azure_clean.yml” playbook. You need to add a survey for the var vm_name. When you run it, it will remove the Azure Resource Group with the VM in it. Then you can run the job template “azure_provision_windows_vm” again. I must warn you that it takes about 10-15 minutes to deploy a new Windows VM in Azure, so it would be a good time to take a break. Also, remember that your public IP address will then change so you need to update the inventory as well.
- Do you need help? Usually that is for a module. Simply google “ansible module <modulename>”. If you need a list of windows modules, look here: https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html
- Try the windows_update_website.yml to refresh the website. If you do, you must update the yml file windows_update_website.yml first to change the path where the content comes from from wsadmin{x} to wsadmin{yournumber}.
- If you feel up to it, create a dynamic inventory for Azure. This will sync everything in the account, so you will see a lot more end-points than your own VM. To do this, create a new inventory. Then inside this inventory create a new source. Choose Azure Resource manager as the type and choose the AzureCredential{x} as the credential for Azure. Then, sync the source and look at the list of hosts in the inventory. Choose a host and look at all the variables that came along with the syncing.

Thank you!

We hope you got a good sense of the basics around Ansible for Windows and Ansible Tower. We do invite you to continue learning Ansible. Here are some great resources you can use to further develop you Ansible skills:

Quick Start Video:
https://www.ansible.com/quick-start-video

Ansible Documentation:
https://docs.ansible.com/

Ansible Galaxy:
https://galaxy.ansible.com/

List of Windows modules:
https://docs.ansible.com/ansible/latest/modules/list_of_windows_modules.html

Get a Ansible Tower Trial license:
https://www.ansible.com/products/tower/trial

Manage Windows like Linux with Ansible: https://www.youtube.com/watch?v=FEdXUv02Dbg

Happy automating! (with Ansible!)