Welcome to the Ansible for Linux & Windows Hands-on Lab!

The url of the Ansible Tower cluster where you work on is: https://tower-{guid}.rhpds.opentlc.com

Where {guid} is the 4 character code you received at the start of the LAB.

You have the following accounts available on your Tower machine:

windowsadmin
This persona maintains all windows playbooks. He has admin rights on the Tower project for Windows playbooks. If you want to use his playbooks, you must ask him nicely.

linuxadmin
This persona maintains all linux playbooks. He has admin rights on the Tower project for Linux playbooks. If you want to use his playbooks, you must ask him even more nicely.

webadmin
This persona's job is to build and maintain web stacks. He needs the playbooks from the other persona's to do her or his job.

admin
The tower administrator maintains the Azure playbooks. He is a nice guy by default. He already gave permission to use them, including the credentials you need to be able to use them. Neat!

user
This is the persona who is able to run but not maintain workflows.

These accounts are all member of the Ansible Tower Organization "ACME Corporation"

All these accounts have password: r3dh4t1! (that includes the !)

The url of the gitlab where your repositories live is: https://control-{guid}.rhpds.opentlc.com

Where {guid} is the 4 character code you received at the start of the LAB.
Username: git
Password: r3dh4t1!

Notes: a trusted certificate is not installed, so you need to accept an exception for this in your browser.

This lab uses Azure. When creating resources in Azure there are restrictions on VM names, usernames and passwords. Read https://docs.microsoft.com/en-us/azure/virtual-machines/windows/faq before you start if you do not know these restrictions!

Now make your choice for an exercise (you probably don't have time to do both, unless you are really good.. ;-)

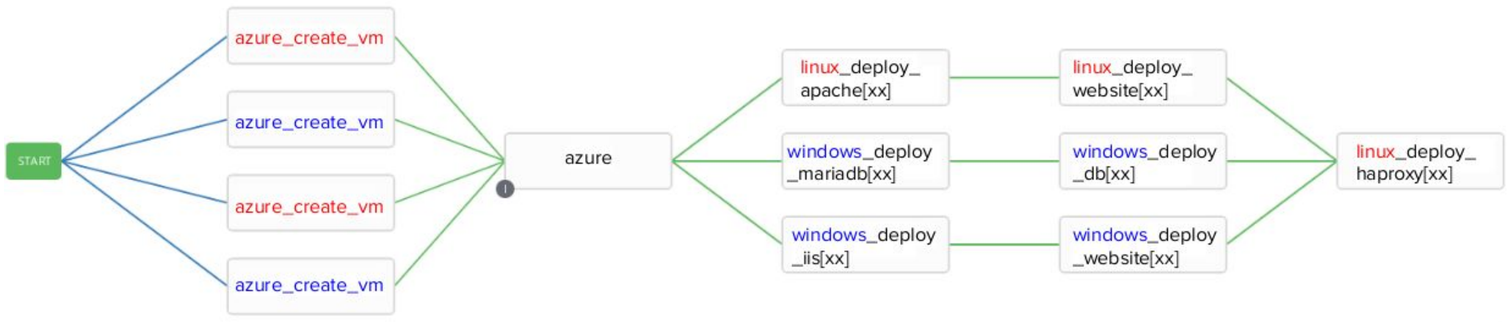
Exercise 1: Mixed Web Stack (mix it any way you want) -> next page

Exercise 2: Windows Domain with Linux member(s) -> page 13

and start playing around :-)

Exercise 1: Mixed Web Stack

Here is (again) an example of what kind of workflow you are tasked to build:



Remember, this is just an example!

- If you choose to use Linux for your web server, you must use the playbooks *linux_deploy_apache* and *linux_deploy_website* as your building blocks of choice
- If you choose to use Windows for your web server, you must use the playbooks *windows_deploy_iis* and *windows_deploy_website.*
- You can deploy multiple web servers. Even, like the example shows, one (or more) Windows based web servers and one (or more) Linux based web servers. They will all serve the same example website.
- The load balancer will do round-robin load balancing across all web servers, independent of the underlying technology stack. So, sometimes you are served using a Linux web server and sometimes using a Windows web server. And you won't be able to tell the difference in the deployed app. How neat is that!
- Same thing for the database: You either choose the playbooks *windows_deploy_mariadb* and *windows_deploy_db* for a Windows based database server or *linux_deploy_mariadb* and *linux_deploy_db* for a Linux based one.
- You actually deploy a real working website where you can maintain users that are stored in the database.

notes:

1. *If you deploy multiple database servers, which does not make sense at all, the web servers will only use the first one in the inventory.*
2. *The load balancer is always Linux based. This is because the Windows Network Load Balancer (WNLB) is not supported on Azure and haproxy is not available for Windows. Using the Azure load balancer would have been a solution, but defeats the scope of this lab.*
3. *You are free to deploy a stack with just one web server and then a load balancer is not needed and you can thus ignore the instructions on the load balancer part, but that would be less fun!*

1. Give Permissions

“webadmin” currently only has access to the Azure job templates as shared by the Ansible Tower admin. So, you need to grant access to the Windows and/or Linux projects, depending on how you want to build your stack.

- If you need Linux playbooks to build your stack:
  a. Log into Ansible Tower with the linuxadmin account
  b. Go to “Projects” in the left main menu. You see the projects this account has access to.
  c. Select “Linux”. You see all the details of this project, including the url of the playbook repository on GiTea.
  d. In the projects menu, select PERMISSIONS and click the + button.
  e. Select the “Web Admin” user
  f. In the lower menu, choose “*Use*” as assigned role (click in the input field for a list)
  g. In the lower menu, choose “*Update*” as assigned role (click in the input field for a list).
  h. Click Save
  i. Log out of Tower. That is the “switch” icon in the upper right corner of the Tower UI, next to the “book” icon.
- If you (also) need Windows playbooks to build your stack, repeat the previous step where you replace Linux with Windows.

notes:

1. *If you want to assign roles to teams instead of users, like you did just now (e.g. Web Team), you need to be Tower admin.*

2. Deploy a test VM

Nothing stops us from already deploying a VM in Azure. You can use that VM in the later workflow. So, let's execute a playbook that will do just that and see how that works:

- Log into Ansible Tower as webadmin
- Go to "Templates" in the left main menu. Here, you see a list of job templates that begin with azure_ and are shared with you and you are allowed to execute.
- Next to each template you see a "run" icon in the form of a flying rocket. Click on the one next to the job template called "azure_create_vm".
- You will be presented with a *survey*. A survey is a simple form to ask for input parameters for this run. Each field is self explanatory. Fill all fields and press NEXT. Choose values that you can later use in your workflow, so use a *role* you need. Remember all input you use here. You need them later! Also make sure you enter valid values. Now click NEXT.
- You get to (p)review what you entered. You also see the var names that will be populated with these values in the playbooks. Press LAUNCH when all is good.
- You now see the JOBS window appear with logging in the right pane as the job executes. The left pane shows all metadata of the job.

Notes:

1. *You can safely ignore the purple warning message at the beginning of the logging.*
2. *The job will take some time to execute. A Windows VM deployment will typically take longer than a Linux VM (on Azure). You can continue with the next exercise or take a look at the playbooks in your git server while you wait and go back to this job at any time by going to the "My View" main menu item and look up this job's details. As you can see each Job has been given a unique number. Select "My Jobs" to only see your own jobs.*
3. *When the job completes it will report in the logging the public IP address of the VM. You can log into this VM with either RDP or SSH (depending on the OS you've chosen) and the username /password you entered in the survey.*
4. *If, for some reason you want to remove this VM, you can execute the job template called azure_delete_vm. It will ask for the VM Name and will delete that VM from Azure.*
5. *Have a look at the playbooks in the ansible4azure repository in your git server. They showcase how you can use Ansible with Azure.*
6. *If the browser window is too small, the interface will automatically switch to mobile view. LEft and Right becomes Up and Down.*
7. *You have write access to the git repositories. Feel free to customize the playbooks to your own liking. Remember though that this server will be deleted after the workshop..*

3. Create a Machine Credential

When you deploy machines, you needed to specify credentials to be able to log in to it. You also need credentials to be able to run Ansible playbooks _in_ the VM's OS. Although the Windows admin and/or Linux admin could share their credentials to do that, that would mean you can log into their machines as well. Better to create your own machine credential then. Here we go:

- Log into Ansible Tower as "webadmin" (if you haven't already)
- Go to "Credentials" in the left main menu. You see some credentials already exist in your account:
    - *You see one Azure credential, given to you to be able to authenticate against Azure to be able to manage Azure resources, like VM's*
    - *There are also two "Vault" credentials. What are those? A vault in Ansible is a file that has been encrypted using the ansible-vault tool. You can encrypt any file with this. In this Lab we encrypted 2 text files which hold credentials. One for accessing the database for write, the other one for configuring the access to the haproxy stats webpage. The encrypted files are protected with a password. We defined vault credentials in Tower with these passwords to be able to decrypt and use the vaulted files during playbook execution.*
- Click the + button.
- Give your credential a name (for example MyCred), optionally a description and select your organization "ACME Corporation".
- Select "Machine Credential" as the credential type. This will present you with some additional fields to fill in. As you can see there are lots of different types of credentials which show you the power of Ansible Tower in its ability to handle all these types.
- For username and password use the EXACT same username and password you used in the previous exercise.
- If you plan to deploy Linux based building blocks in your workflow then specify "sudo" for "Privilege Escalation Method", "root" for "Privilege Escalation Username" and the same password you used earlier for "Privilege Escalation Password".
- Click SAVE

Notes:

- *Try to look up the password you just entered. As you will notice you can't anymore. You can only replace it. Nice and safely stored :-)*
- *If you want to check whether you typed the password correctly, you can do that only before you click SAVE.*

4. Create an Inventory

As you know by now you need an inventory of machines you want your playbooks to run on. In this exercise we are going to create a *dynamic* inventory. This is an inventory that will populate itself from an external *source*, in this case Azure. This means that as you create VM's in Azure the inventory will be populated with the VM's that you have created once you sync it. You need credentials for Azure to be able to do this, but these have been shared with you by the Tower Admin to use, remember?

- Log into Ansible Tower as "webadmin" (if you haven't already)
- Go to "Inventories" in the left main menu, click the + button and choose "Inventory".
- Give your inventory a NAME (ie MyInventory), optionally a DESCRIPTION, "ACME Corporation for ORGANIZATION and click SAVE
- Choose SOURCES in the inventory menu and click the + button.
- Give the source a NAME(suggestion: Azure) and optionally a DESCRIPTION. Choose "Microsoft Azure Resource Manager" as the SOURCE (note that the CREDENTIAL is auto populated to the Azure CREDENTIAL, because that is the only one with that type you have access to).
- Select "Overwrite" in the UPDATE OPTIONS (click the ? next to it to get an explanation on this option)
- **Important**: In SOURCE VARIABLES, just below the "---", so on line 2, type in the following:
    - *resource_groups: ansible_workshop_{guid}*
    This little trick will filter the hosts in Azure to those that are in your Azure resource group.
- Click SAVE
- Click the SYNC ALL button. It will now start syncing your current VM's in Azure into your inventory.
- When the synchronization is done (the cloud icon next to the source stops blinking and is green), click the HOSTS submenu item within the inventory. If all is well you see the VM you deployed in the previous step. **If you see none or more, that is not good! Talk to the instructor!**
- When you select the host you see the DETAILS pane where you see vars that are generated from the Azure SOURCE and that you can use in your playbooks.
- In the GROUPS tab you also see that the host is automatically added to various groups that represent different types of Azure metadata. The resource_group is one, The OS is another, Also, the role you specified during the creation you now see come back here as well. We will use these groups to run the different playbooks against.

5. Create Job Templates

Depending on how you want to deploy your version of the stack, you now add job templates for each node in your workflow.

- Log into Ansible Tower as webadmin (if you haven't already)
- Go to "Templates" in the left main menu and click the + button and choose "Job Template"
  - NAME: choose the name of the playbook, for example "linux_deploy_apache"
  - DESCRIPTION: is optional and you can choose anything
  - INVENTORY: The Inventory you just created
  - PROJECT: Choose Windows or Linux, depending on the building block.
  - PLAYBOOK: Choose the correct playbook from the list.
  - CREDENTIAL: Choose the Machine Credential you just created.

  Important! Certain playbooks use vaulted files that contain resource credentials, as explained above. Those vaults are encrypted using ansible-vault. To be able to decrypt them a vault credential needs to be specified when used in a playbook. The **_website* and the **_db* playbooks need the "Database Access" vault credential and the *linux_deploy_haproxy* playbook needs the "Load Balancers" vault credential. You can specify multiple credentials for a playbook (makes sense, right?). To do this open the searchbox of the Credential field again and choose "Vault" in the dropdown. Now you see the Vault credentials you have access to and you can choose the one you need for the playbook at hand.
- Click SAVE.

Repeat this exercise for each building block in your workflow. If might seem like a lot of work, but once you've done one or two, it becomes second nature and you can make them quite fast :-)

6. **Create workflow template**
Now we finally have all the needed components to build a workflow. Phew! A workflow is a set of job templates (and thus playbooks) that are linked together in a certain specific way such that multiple job templates are executed in a predictable order. Each job template in a workflow is called a “node”.

- Log into Ansible Tower as webadmin (if you haven’t already).
- Go to “Templates” in the left main menu and click the + button and choose “Workflow Template”
    - NAME: name the workflow “Deploy Web Stack ”
    - DESCRIPTION: is optional and you can choose anything
    - ORGANIZATION: Choose “ACME Corporation”
    - INVENTORY: Choose the Azure inventory you made previously
    - Click SAVE

  You are now brought to the *Workflow Visualizer*. Here you can graphically build out your workflow.
- Click on the START node. A dialog will open where you can choose one of your job templates to add to the workflow.
- Choose “azure_create_vm” (it is greyed-out, but you can still select it), scroll down and click PROMPT. You can now fill in all the fields needed for this node.
    - For the first azure_create_vm node select the same field values you used to create the test VM in the previous step. It will then reuse that VM.
    - Click SELECT. A node has been added to the workflow.
- You can now click the START node again to add a step that runs *parallel* to the previous created step. This way you can, for example in this workflow, create multiple VM’s in parallel.
- Now first create all Azure VM’s you need and specify the correct OS, role and credentials. (*Suggested vm names: winweb{x}, linweb{x}, windb, lindb, linlb).*
- When you hover over the just created node 3 small icons appear:
    - A green icon you can click to add a node *after* this node
    - A red icon to *remove* this node
    - A blue icon to create a *link* from this node to another existing node. You create the link by clicking on the node to link to. You select if this path should be followed “*Always*”, “*On Success*” or “*On Failure*” and you click SAVE to establish the link.
- **Important**! After you have added the nodes that create the needed VM’s in Azure for your stack you want an inventory sync to happen because otherwise the playbooks that must operate on those VM’s have an empty inventory. To do this, add a node after the node(s) that create the VM(s) and, in the top of the dialog that appears, choose “INVENTORY SYNC”. Then you can choose what inventory source to sync and in what condition (Success, Failure, Always). See the example workflow above to see how that

looks like. You first add an “inventory sync” node after the first “azure_create_vm” node and then you link the other “azure_create_vm” nodes to it. Easy!

- When done with building the workflow, click SAVE.

With this simple (...) set of instructions, you should be able to build out the workflow the way you want it, so have a go at it!

Notes:

- *You can move the workflow diagram on your screen by click-and-drag on an empty spot in the workflow area.*
- *You can zoom in or out by doing a two-finger drag (without click!) in de workflow area up or down.*
- *You can also do both these actions by clicking on the gear icon in the top right of the workflow area.*

7. **Finalize workflow template**

We are almost there! Three things left to do, and one is to add a survey to the workflow template. Why would we need that? The web app needs a name! This is optional and if you don't create a survey, the website playbooks are smart enough to define a default name for the app. If your curious, just have a look in the specific playbooks in the git repo to see how that is done.

- Go to the workflow template details that you have created and click ADD SURVEY.
- Create a field:
  - Prompt: *Name*
  - Description: *Application Name*
  - Answer variable name: *appname*
  - Answer type: *text*
  - Minimum length: *4*, Maximum length: *40*
  - Default answer: whatever you like
  - Required: *no*
- Click ADD and then SAVE

Now, add a notification to the workflow. Notifications are a way to get notified of the result of a job run: Success or Failure. There are many notification methods and one of them is slack. We have already set up a notification to a slack channel for you to use. You can set various notifications on various levels (project, org, job template, etc) so it's quite a powerful feature. Let's add a slack notification to the workflow you've made:

- Go to the workflow template details that you have created and click on NOTIFICATIONS. You see the list of notifications is currently one: the defined slack channel.
- Enable this notification for both *SUCCESS* and *FAILURE*.

Note: the slack channel is displayed on the presentation screen. If it's not, notify the instructor: he forgot..

Last (but not least), assign the "*Execute*" role to the user named "user" in PERMISSIONS. As you are already experienced on assigning roles, we will not elaborate on the details :-).

8. Execute the workflow

Now we finally are ready to give the workflow a run for it's money!

You can run this workflow either as user "webadmin" or as user "user". The difference is that "webadmin" has access to the logging, the inventory, etc, so if anything goes wrong you can immediately see what it is. When you execute the workflow as "user", you only see the workflow and the ability to execute it. You can follow the execution, see if it succeeds or fails, but you have no access to the logging.

- Log in as either "webadmin" or user "user". (you now know the difference)
- Find the workflow in the list of templates and press the rocket icon next to it. If all is well it should ask for the application name because you just created that survey. Enter (optionally) a nice description, click NEXT and then, finally, click LAUNCH !!!
- The screen will now move to a graphical representation of the workflow and you see continued feedback where the execution is and what the result of each node's execution is: a green outline means SUCCESS. A red outline FAILURE. The black dot means the node is currently executing. You can click on the "*DETAILS*" and it will bring you to the logging if you are logged in as "webadmin".
- Hopefully your workflow will finish in one go. If it does, in the logging of the last step (the haproxy node) you find the details to access the website (basically, the public IP of the load balancer VM).
- Well Done! Have some fun with te website!
- You can also access the load-balancer statics page (see logging for the details). It will ask you for a password. That password you already know.

Notes:

- *If something has gone wrong, probably a node is displayed outlined in red. See what's wrong in the logging, fix it, and restart the workflow job, just like you can restart a template job. You do this by going into the workflow jobs details page and press the run icon (so, NOT the job template details!). This will restart the workflow with the previously entered survey. You don't need to worry about creating VMs multiple times. Ansible will take care of just creating the VMs that do not exist yet. It is omnipotent in that regard.*

9. Cleanup

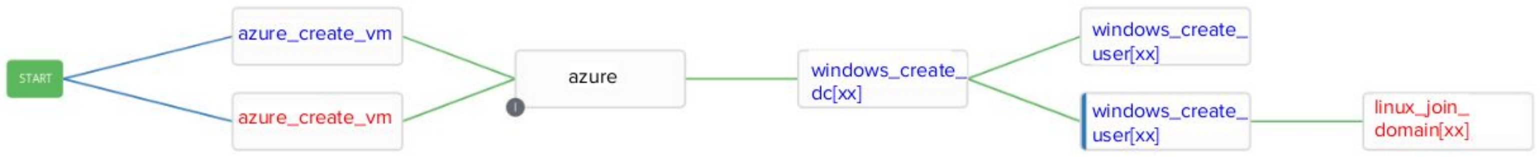
After you have recovered from the amazement of your result it is time to clean up. We made a special job template for that:

- Log in as "webadmin"
- Find the job template called "azure_delete_all" and execute it. It will ask for a confirmation (duh!) and if you confirm, it will remove all the VM's *you* have created in Azure.

Note: If you plan to also do the other exercise, it is mandatory that you do this.

Exercise 2: Windows Domain with Linux member(s)

Here is (again) an example of what kind of workflow you need to build:



You are tasked to build this workflow as the user "*linuxadmin"* with the added ability to have user "*user*" to execute it as well.

Summary of tasks:

- You create one Windows and one Linux VM in Azure
- You sync the inventory to import the newly created VM's in Azure into the inventory
- You create a new AD domain on the Windows machine
- You create a Domain Admin user and a Normal user
- You join the Linux machine to the AD domain
- You log into the Linux machine with either SSH or Cockpit using the Windows accounts in the Active Directory.

Note: If you use cockpit, a trusted certificate is not installed, so you need to accept an exception for this in your browser.

1. Give Permissions

linuxadmin currently has only access to the Linux Job Templates and the Azure job templates as shared by the Ansible Tower admin. So, you need to grant access to the Windows projects

- Log into Ansible Tower with the windowsadmin account
- Go to “Projects” in the left main menu. You see the projects this account has access to.
- Select “Windows”
- In the projects menu, select PERMISSIONS and click the + button.
- Select “ linuxadmin”
- In the lower menu, choose “Use” as the role.
- Click Save
- Log out of Tower. That is the “switch” icon in the upper right corner of the Tower UI.

notes:

1. *If you want to assign roles to teams (e.g. Web Team), you need to be Tower admin.*

2. Deploy a test VM

Nothing stops us from already deploying a VM in Azure. You can use that VM in the later workflow. So, let's execute a playbook that will do just that and see how that works:

- Log into Ansible Tower as "linuxadmin"
- Go to "Templates" in the left main menu. Here, you see a list of job templates that begin with azure_ and are shared with you and you are allowed to execute.
- Next to each template you see a "run" icon in the form of a flying rocket. Click on the one next to the job template called "azure_create"_vm".
- You will be presented with a survey. A survey is a simple form to ask for input parameters for this run. Fill each field as follow:
    - Name: <whatever you like>
    - OS Type: Linux
    - Username: whatever you like
    - Password: make it strong, Luke! (and remember it because you need it later).
    - Role: Webserver
    - Now click NEXT.
- You now get to (p)review what you entered. You also see the var names that will be populated with these values. Press LAUNCH when all is right.
- You now see the JOBS window appear with logging in the right pane as the job executes. The left pane shows all metadata of this job.

Notes:

- *The job will take some time to execute. You can continue with the next exercise or take a look at the playbooks in your gitlab while you wait and go back to this job at any time by going to the "My View" main menu item and look up this job details. Each Job has a unique job number assigned to it. Select "My Jobs" to only see your own jobs.*
- *When the job completes it will report in the logging the public IP address of the VM. You can log into this VM with either RDP or SSH (depending on the OS type) and the username/password you specified.*
- *If, for some reason, you want to remove this VM, you can execute the template called azure_delete_vm. It will ask for the VM Name and will delete that VM from Azure.*
- *Have a look at the playbooks in the ansible4azure repository in your gitlab. They showcase how you can use Ansible with Azure.*
- *If the browser window is too small, the interface will automatically switch to mobile view. LEft and Right becomes Up and Down.*

3. Create a Machine Credential

When you deploy machines, you need to specify a credential to be able to log in to it. You also need credentials to be able to run Ansible playbooks _in_ the VM's OS. Although the Windows admin and/or Linux admin could share their credentials to do that, that would mean you can log into their machines as well. Better to create your own machine credential then. Here we go:

- Log into Ansible Tower as linuxadmin (if you haven't already)
- Go to "Credentials" in the left main menu. You see some credentials already exist in your account:
  - *You see one Azure credential, given to you to be able to authenticate against Azure to be able to deploy VM's*
  - *There are also 2 "Vault" credentials. What are those? A vault in Ansible is a file that has been encrypted using the ansible-vault tool. You can encrypt any file with this. In this Lab we encrypted 2 files which hold credentials. One for accessing the database for write, the other one for configuring the access to the haproxy stats webpage. The encrypted files are protected with a password. You define a vault credential in Tower with this password to by able to decrypt and use the vaulted files during playbook execution. These credentials are not used by this exercise but by exercise 1.*
- Click the + button.
- Give your credential a name (for example: MyCred) and optionally a description and select your organization "ACME Corporation".
- Select "Machine Credential" as the credential type. This will present you with some additional fields to fill in. As you can see there are lots of different types of credentials which show you the power of Ansible Tower in its ability to handle all these types.
- For username and password use the EXACT same username and password you used in the previous exercise.
- Specify "sudo" for "Privilege Escalation Method", "root" for "Privilege Escalation username" and the EXACT same password you used earlier for "Privilege Escalation Password".
- Click SAVE

Notes:

- *Try to look up the password you just entered. As you will notice you can't anymore. You can only replace it. Nice and safely stored :-)*
- *If you want to check whether you typed the password correctly, you can do that only before you click SAVE.*

4. Create an Inventory

As you know by now you need an inventory of machines you want your playbooks to run on. In this exercise we are going to create a *dynamic* inventory. This is an inventory that will populate itself from an external source, in this case Azure. This means that as you create VM's in Azure the inventory will be populated with the VM's that you have created once you sync it. You need credentials for Azure to be able to do this, but these have been shared with you by the Tower Admin to use, remember?

- Log into Ansible Tower as "linuxadmin" (if you haven't already)
- Go to "Inventories" in the left main menu, click the + button and choose "Inventory".
- Give your inventory a NAME (ie MyInventory), optionally a DESCRIPTION, "ACME Corporation for ORGANIZATION and click SAVE
- Choose SOURCES in the inventory menu and click the + button.
- Give the source a NAME(suggestion: Azure) and optionally a DESCRIPTION. Choose "Microsoft Azure Resource Manager" as the SOURCE (note that the CREDENTIAL is auto populated to the Azure CREDENTIAL, because that is the only one with that type you have access to).
- Select "Overwrite" in the UPDATE OPTIONS (click the ? next to it to get an explanation on this option)
- In SOURCE VARIABLES, just below the "---", so on line 2, type in the following:
    - *resource_groups: ansible_workshop_{guid}*
      This little trick will filter the hosts in Azure to those that are in your Azure resource group.
- Click SAVE
- Click the SYNC ALL button. It will now start syncing your current VM's in Azure into your inventory.
- When the synchronization is done (the cloud icon next to the source stops blinking and is green), click the HOSTS submenu item within the inventory. If all is well you see the VM you deployed in the previous step.
- When you select the host you see the DETAILS pane. Here you see vars that are generated from the Azure SOURCE and that you can use on your playbooks.
- In the GROUPS menu you also see that the host is automatically added to various groups that represent different types of Azure metadata. The resource_group is one, The OS is another, Also, the role you specified during the creation, you now see come back here as well. We will use these groups to run the different playbooks against.

5. **Create Job Templates**

   You now add job templates for each node in your workflow.
   - Log into Ansible Tower as linuxadmin(if you haven't already)
   - Go to "Templates" in the left main menu and click the + button and choose "Job Template"
     - NAME: choose the name of the playbook, for example "windows_create_user"
     - DESCRIPTION: is optional and you can choose anything
     - INVENTORY: The Inventory you just created
     - PROJECT: Choose Windows or Linux, depending on the workflow node you are working on.
     - PLAYBOOK: Choose the correct playbook from the list.
     - CREDENTIAL: Choose the Machine Credential you just created.
   - ONLY for the job template *windows_create_user*, enable the option "Enable Concurrent Jobs". What this does is enabling the possibility to have multiple jobs running from this template concurrently. As you can see from the workflow diagram, we want this. The shared job template "azure_create_vm" has the same capability.
   - Click SAVE.

   Repeat this exercise for each building block in your workflow. If might seem like a lot of work, but once you've done one or two, it becomes second nature and you can make them quite fast.

6. **Build Surveys**
The used playbooks need a survey defined on the job template to be able to work. This way, the template will provide the playbook with user defined var's. You create a survey by going to the job template details and click on ADD SURVEY. Here you can enter the specification of each field in the survey. When you're done, click SAVE. Here is the specification for the survey of each job template that you need to make:

Survey for job template **windows_create_domain**

PROMPT	DESCRIPTION	ANSWER VARIABLE NAME	ANSWER TYPE	MIN. LEN	MAX LEN	REQUIRED
Name	*Domain Name*	***domain_name***	*text*	*4*	*30*	*Yes*
Password	*Password*	***password***	*Password*	*8*	*20*	*Yes*

Survey for job template **windows_create_user**

PROMPT	DESCRIPTION	ANSWER VARIABLE NAME	ANSWER TYPE	MIN. LEN	MAX LEN	REQ
Domain	*Domain Name*	***domain***	*text*	*4*	*30*	**No**
Name	*Username*	***username***	*Text*	*4*	*30*	*Yes*
Password	*Password*	***password***	*Password*	*8*	*20*	*Yes*
Group	*Group Name*	***group***	*Multiple Choice (Single Select)*	*Domain Admins Users*		*Yes*

Survey for job template **linux_join_domain**

PROMPT	DESCRIPTION	ANSWER VARIABLE NAME	ANSWER TYPE	MIN. LEN	MAX LEN	REQ
Domain	*Domain Name*	***domain***	*text*	*4*	*30*	***No***
Name	*Admin username*	***ad_user***	*Text*	*4*	*30*	***No***
Password	*Admin Password*	***ad_password***	*Password*	*8*	*20*	***No***

Notes:

- Did you notice some fields in the survey specification are optional (REQ: **No**)? So what would happen if you keep them empty in the workflow? Here, a nice feature is demonstrated: The ability to pass data from one node in a workflow to nodes that follow it. How that works? Go look at the underlying playbooks in gitlab and search for the method “set_stats”. That should explain it. You can test it by actually leave all fields that are optional empty, which we will do below.

6. Create workflow template

Now we finally have all the needed components to build a workflow. Phew! A workflow is a set of job templates (and thus playbooks) that are linked together in a certain specific way such that multiple job templates are executed in a predictable order. Each job template in a workflow is called a "node".

- Log into Ansible Tower as "linuxadmin" (if you haven't already).
- Go to "Templates" in the left main menu and click the + button and choose "Workflow Template"
- NAME: name the workflow *"create_windows_domain_with_linux_client*" (just to show you can create really long template names ;-)
    - DESCRIPTION: is optional and you can choose anything
    - ORGANIZATION: Choose ACME Division
    - INVENTORY: Choose the Azure inventory you made previously
- Click SAVE
- You are now brought to the Workflow Visualizer. Here you can graphically build out your workflow.
- Click on the START node. A dialog will open where you can choose one of your job templates to add to the workflow.
- Choose "azure_create_vm" (it is greyed-out (a known bug!), but you can still select it), scroll down and click PROMPT. You can now fill in all the fields needed for this node. For the first azure_create_vm node select the same field values you used to create the test VM in the previous step. It will then reuse that VM.
- Click SELECT.
- You can now click the START step again to add a step that runs *parallel* to the previous created step.
- When you hover over the just created node 3 small icons appear:
    - A green icon you can click to add a node *after* this node
    - A red icon to *remove* this node
    - A blue icon to create a *link* from this node to another existing node. You create the link by clicking on the node to link to. You select if this path should be followed "Always", "On Success" or "On Failure" and you click SAVE to establish the link.
- You need to create 2 VM's: One Windows VM with role "dcserver" and one VM with role "webserver" (the one you already made).
- **Important**! After you have added the nodes that create the needed VM's in Azure for your workflow you want an inventory sync to happen because otherwise the playbooks that must operate on those VM's have an empty inventory. To do this, add a node after the node(s) that create the VM(s) and, in the top of the dialog that appears, choose "INVENTORY SYNC". Then you can choose what inventory source to sync and in what condition (Success, Failure, Always). See the example workflow above to see how that looks like. You first add an "inventory sync"

node after the first “azure_create_vm” node and then you link the other “azure_create_vm” node to it. Easy!

- *Make sure you use a 2-part domain name for the node windows_create_domain. Something like blah.local and not blah.*
- The node “windows_create_user” that precedes the node “linux_join_domain” must create a user that is a member of the domain_admins group.
- When done with building the workflow, click SAVE.
- Do NOT fill out the survey for the linux_join_domain node. Why not? Did you notice some fields in the survey specification are optional (REQ: NO)? So what would happen if you keep them empty? Here, a nice feature is demonstrated: The ability to pass data from one node to nodes that follow it. How that works? Go look at the underlying playbooks in gitlab and search for the method “set_stats”. That should explain it. You can test it by actually leave all field that are optional empty, which we do here.

With this simple (...) set of instructions, you should be able to build out the workflow for this exercise, so have a go at it!

Notes:

- *You can move the workflow diagram on you screen by click-and-drag on an empty spot in the workflow area.*
- *You can zoom in or out by doing a two-finger drag (without click!) in de workflow area up or down.*
- *You can use both these actions also by clicking on the gear icon in the top right of the workflow area.*

7. Finalize workflow template

Now, add a notification to the workflow. Notifications are a way to get notified of the result of a job run: Success or Failure. There are many notification methods and one of them is slack. We have already set up a notification to a slack channel for you to use. You can also set various notifications on various levels (project, org, job template, etc) so it’s quite a powerful feature. Let’s add a slack notification to the workflow you made:

- Log in as “admin”
- Go to the workflow template details that you have created and click on NOTIFICATIONS. You see the list of notifications is currently one: the defined slack channel.
- Enable this notification for both SUCCESS and FAILURE.

Note: the slack channel is displayed on the presentation screen. If it’s not, notify the instructor: he forgot ;-).

Last (but not least), assign the “Execute” role to the user named “user” in PERMISSIONS. As you are already experienced on assigning roles, we will not elaborate on the details :-).

8. Execute workflow template

Now we finally are ready to give the workflow a run for its money!
You can run this workflow either as user "linuxadmin" or as user "user". The difference is that "linuxadmin" has access to the logging, the inventory, etc, so if anything goes wrong you can immediately see what it is. When you execute the workflow as "user", you only see the workflow and the ability to execute it. You can follow the execution, see if it succeeds or fails, but you have no access to the logging.

- Log in as either "linuxadmin" or user "user".
- Find the workflow in the list of templates and press the rocket icon next to it. If all is well it should ask for the application name because you just created that survey. Enter a nice description, click NEXT and then, finally, click LAUNCH !!!
- The screen will move to a graphical representation of the workflow and you see continued feedback where the execution is and what the result of each node's execution is: a green outline means SUCCESS. A red outline FAILURE. You can click on the "DETAILS" and it will bring you to the logging if you are logged in as "linuxadmin".
- Hopefully your workflow will finish in one go. If it does, in the logging of the last step (the haproxy node) you find the details to access the website (basically, the public IP of the load balancer VM).
- Well Done!

Notes:

- *If something has gone wrong, probably a node is displayed in red. See what's wrong, fix it, and restart the workflow job, just like you can restart a template job. You do this by going into the workflow jobs details page and press the run icon. This will restart the workflow with the previously entered survey. You don't need to worry about creating VMs multiple times. Ansible will take care of just creating the VMs that do not exist yet.*

9. Cleanup

After you have recovered from the amazement of your result it is time to clean up. We made a special job template for that:

- Log in as "linuxadmin"
- Find the job template called "azure_delete_all" and execute it. It will ask for a confirmation (duh!) and if you confirm, it will remove all the VM's you have created in Azure.

Note: If you plan to also do the other exercise, it is mandatory that you do this.

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 all modules:
https://docs.ansible.com/ansible/latest/modules/modules_by_category.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

All playbooks are permanently available on https://github.com/fvzwieten

Happy automating! (with Ansible Tower!)