Lab 6

Ansible plugins

Go to `~/course/ansible/plugins/output-callback` and run the playbook:

```
$ ansible-playbook main.yaml
```

Notice the error? Don't try to solve it (yet).

How could you improve the readability:

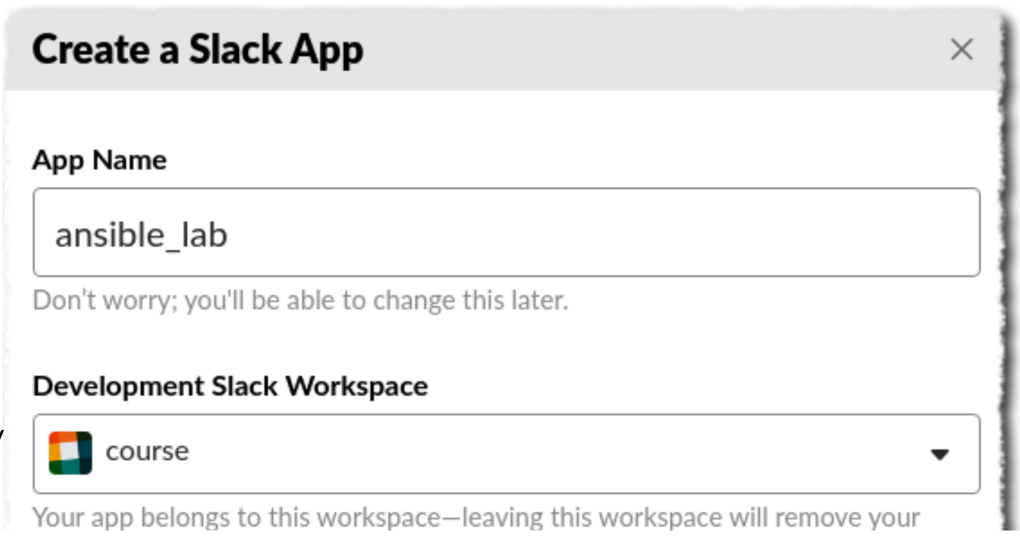
- `ANSIBLE_STDOUT_CALLBACK debug`
- try some of these:
    - yaml
    - json
- Fix the problem and run with the default output callback
- Then again try debug and yaml
- Then fix the second problem by allowing the task to create the file
- Now run it with:
    - dense
    - unixy
    - actionable

Let's use a non-stdout plugin. We will explore the slack plugin. You will need a slack account for this. You can sign up for free if you don't have one at https://slack.com (https://slack.com)

Setup a slack webhook

Login to your slack account and after that go to https://api.slack.com/apps (https://api.slack.com/apps) and create a new App

Name your app and set the workspace:

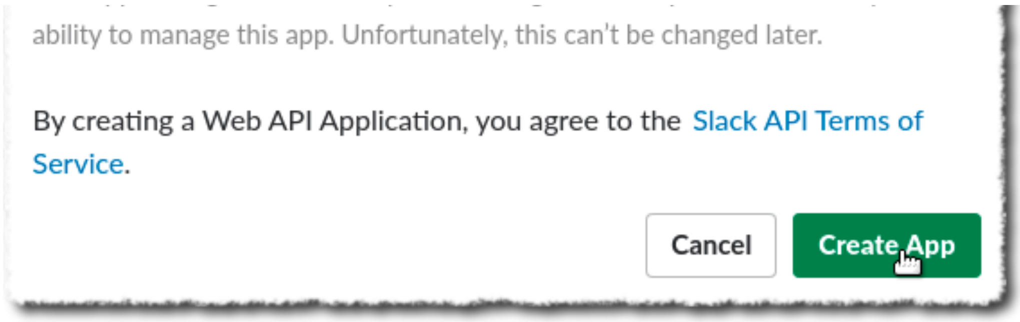

Create a Slack App

App Name: ansible_lab

Don't worry; you'll be able to change this later.

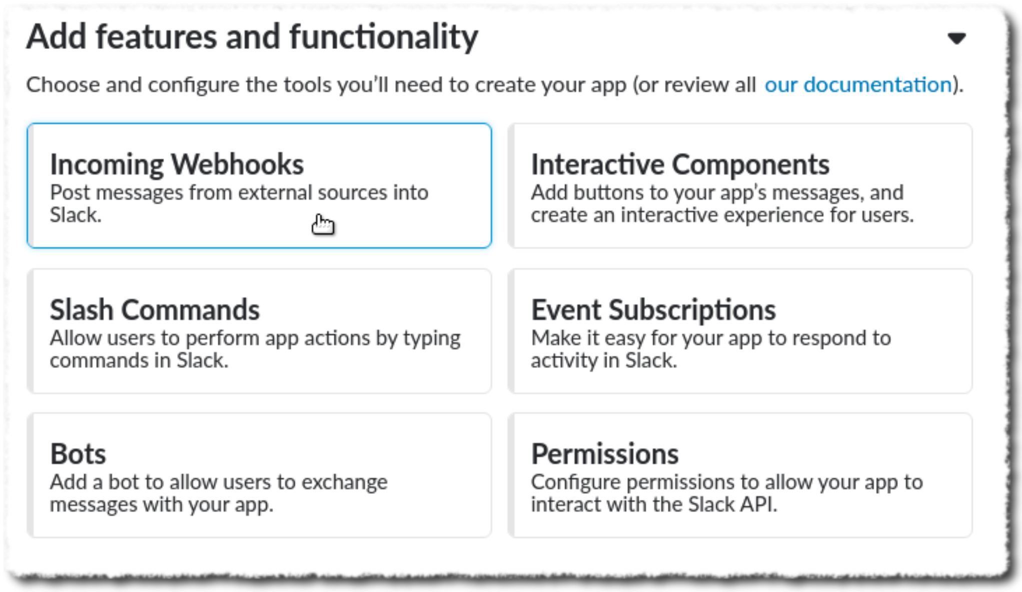
Development Slack Workspace: course

Your app belongs to this workspace—leaving this workspace will remove your


ability to manage this app. Unfortunately, this can't be changed later.

By creating a Web API Application, you agree to the Slack API Terms of Service.

Then select **Incoming Webhooks**


Add features and functionality

Choose and configure the tools you'll need to create your app (or review all our documentation).

Incoming Webhooks
Post messages from external sources into Slack.

Interactive Components
Add buttons to your app's messages, and create an interactive experience for users.

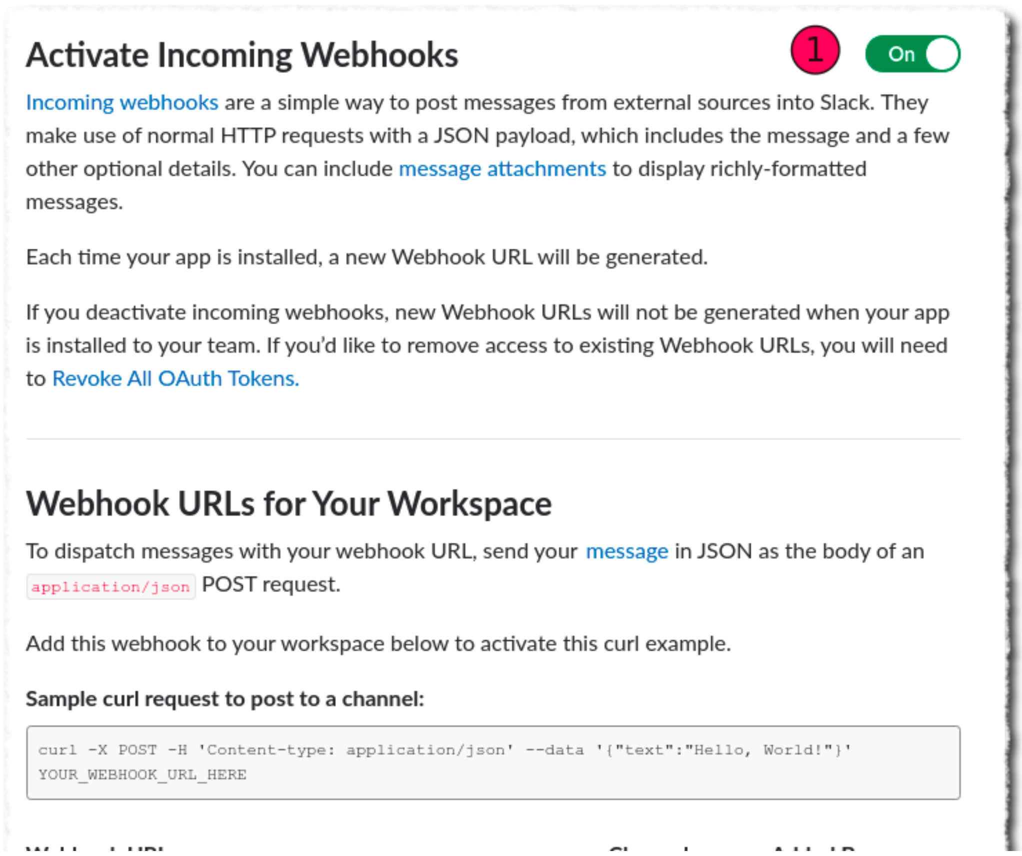
Slash Commands
Allow users to perform app actions by typing commands in Slack.

Event Subscriptions
Make it easy for your app to respond to activity in Slack.

Bots
Add a bot to allow users to exchange messages with your app.

Permissions
Configure permissions to allow your app to interact with the Slack API.

Enable it and a new webhook to workspace


Activate Incoming Webhooks

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include message attachments to display richly-formatted messages.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to Revoke All OAuth Tokens.

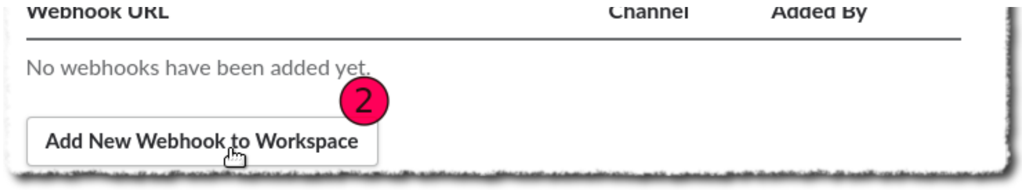
Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your message in JSON as the body of an `application/json` POST request.

Add this webhook to your workspace below to activate this curl example.

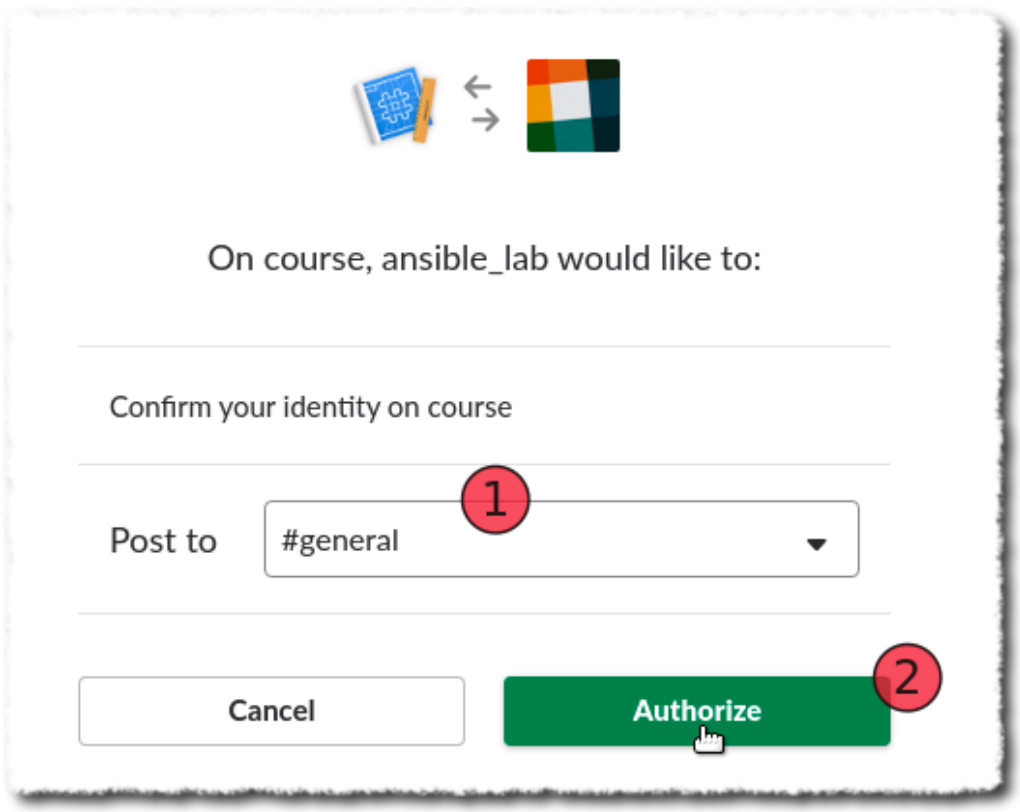
Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}'
YOUR_WEBHOOK_URL_HERE
```



Webhook URL	Channel	Added By
No webhooks have been added yet.		

Then select a channel and click authorise

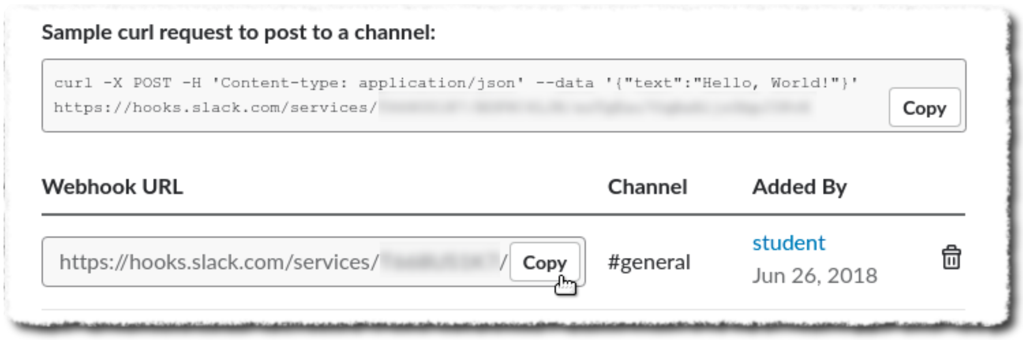


On course, ansible_lab would like to:

Confirm your identity on course

Post to: #general

Then copy your webhook url to your clipboard



Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}'
https://hooks.slack.com/services/[redacted]
```

Webhook URL	Channel	Added By
https://hooks.slack.com/services/[redacted]/	#general	student Jun 26, 2018

Install required software

Install the required python module

```
$ pip install prettytable --user
```

Run your playbook

Now run a playbook. (use for example `~/course/ansible/worlddb` or `~/course/ansible/frontend`) We will use environment variables to configure the plugin Check the slack plugin (https://docs.ansible.com/ansible/2.5/plugins/callback/slack.html) documentation:

- Enable the slack plugin `ANSIBLE_CALLBACK_WHITELIST=slack`
- Specify your webook plugin `SLACK_WEBHOOK_URL=https://hooks.slack.com/services/…/…/…`

```
$ ANSIBLE_CALLBACK_WHITELIST=slack  \
     SLACK_WEBHOOK_URL=https://hooks.slack.com/services/…/…/…  \
     ansible-playbook main.yaml --vault-id @prompt
```

Let's cache the facts for `~/course/ansible/worldstack`. You will change the `ansible.cfg` that you used before in a previous lab.

json file

You will first use the jsonfile (https://docs.ansible.com/ansible/2.6/plugins/cache/jsonfile.html) cache plugin.

Open the `ansible.cfg`:

- Set the fact caching strategy to `jsonfile` `fact_caching=…`
- You will need to specify the directory to save the json files to. Set this to `$HOME/course/ansible/.cache` `fact_caching_connection`
- For this lab we will set the cache timeout to 2 minutes `fact_caching_timeout=120`
- ensure the cache is used when available `gathering=smart`

Run a playbook (e.g, only `rest-server.yaml`). Notice it gathers facts. After the playbook has completed run it again, notice this time it does not gather facts. also check check the contents of the `~/course/ansible/.cache` directory.

After 2 minutes run the playbook again. It should gather facts again as the timeout has expired.

Also try to run and force gathering of the facts add `--flush-cache`

Redis

This task does not add much value, but it might be nice to use redis as this is often used as a cache backend.

First start a redis docker container on your control machine

```
$ docker run -p 6379:6379 -d   --name redis -d redis
```

Then change the `ansible.cfg` so that it will use the redis (https://docs.ansible.com/ansible/2.6/plugins/cache/redis.html) cache provider:

- the connection URI is just 'localhost' (we are using the default 6379 port and no username/password)
- For now also disable the timeout and revert back it to its default value of 24 hours.

This cache plugin requires the redis python package, ensure this is installed:

```
$ pip install "redis==2.10.6" --user
```

Now run your playbook and notice it gathers facts again. Run it again and it should be using the cache. Let's have a look at the cache so you feel more comfortable redis is actually used.

Run `redis-cli` using your container and check the available keys:

```
$ docker exec -it redis redis-cli
127.0.0.1:6379> keys *
…
```

If you wanted to see the values, you could use:

```
$ mget ansible_facts10.20.1.5
```

Let's crank up a windows server. Navigate to `~/course/ansible/plugins/connections/windows` :

```
$ vagrant up
```

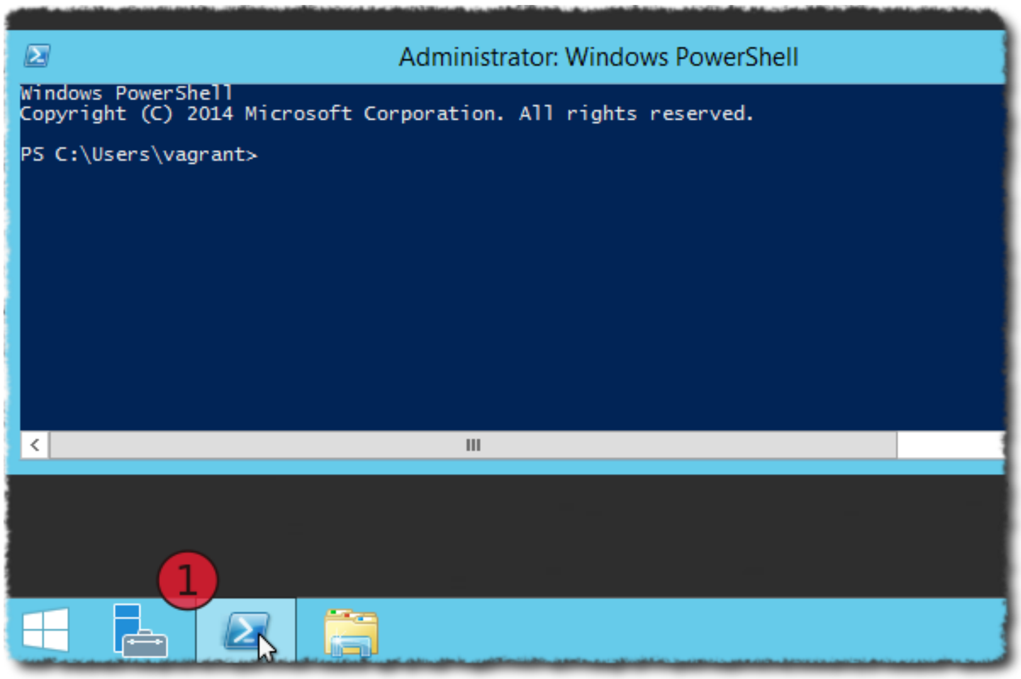
This might take a while.

You will also need to install the pywinrm (https://pypi.org/project/pywinrm/) python module:

```
$ pip install "pywinrm>=0.3.0" --user
```

Configure winrm

After the server has started up, open it with Oracle VirtualBox manager and open a powershell:



```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS C:\Users\vagrant>
```

Winrm has already been installed, but is only listening to an unsecured HTTP port:

```
PS c:\Users\vagrant> winrm e winrm/config/listener
Listener
    Address = *
    Transport = HTTP
    Port = 5985
    Hostname
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint
    ListeningOn = 10.0.2.15, 10.20.1.10,…
```

To define a HTTPS endpoint, we will first need to generate a certificate:

```
New-SelfSignedCertificate -DnsName "mybox" -CertStoreLocation Cert:\LocalMachin
e\My
```

Copy the thumbprint to your clipboard and use it when defining the HTTPS endpoint:

```
winrm create winrm/config/Listener?Address=*+Transport=HTTPS '@{Hostname="mybox
";
    CertificateThumbprint="…"}'
```

We need to open the firewall for port 5986

```
netsh advfirewall firewall add rule name="winrm (https)" dir=in action=allow pr
otocol=TCP localport=5986
```

Define an Inventory

Now define a new inventory file named `windows-hosts` :

- Define a group named `windows`
- In it define a single host named for example mybox for host 10.20.1.10 `ansible_host`

Then create a file to set group host variables for your group `group_vars/windows.yaml`

- Use winrm for the connection `ansible_connection`
- set the username/password both to "vagrant" `ansible_user` / `ansible_password`
- set to ignore certification validation `ansible_winrm_server_cert_validation: ignore`

This last setting is required as we are not using a proper domain name ("mybox") nor are we using a CA or are otherwise establishing trust.

Test

Use the win_ping module to check your inventory:

```
$ ansible all -i windows-hosts -m win_ping
  mybox | SUCCESS => {
      "changed": false,
      "ping": "pong"
  }
```

Run a local container for httpbin (https://httpbin.org/). This is a python/flask test server to test various HTTP request/response interactions. We won't be really using it, we will just run tasks against in using a playbook:

```
$ docker run -p 8080:80 -d --name httpbin  kennethreitz/httpbin
```

Now navigate to the `~/course/ansible/plugins/connections` folder. In it you should find:

- a `hosts` inventory file, for you to complete,
- a `main.yaml` playbook, also for you to complete.

The playbook

Let's start with the playbook, this will just be exercising your playbook writing skills and is not specific to the subject at hand.

The objective is to get the version of flask (http://flask.pocoo.org/) installed on the host:

- Use the correct module to run the following command `command`

  ```
  flask --version
  ```

- For this command set the following environment variable `environment:`

  ```
  LC_ALL: C.UTF-8
  ```

- Save the result in a variable named `version` `register:`
- Then use a `debug` task to show the value of the version

Create the inventory

Now we will run this against our container. You therefore will need to change the `hosts` file. Open it.

Add a host:

- the name of the host must be the name of the container (which in our case is `httpbin`, you can check with `docker ps`)
- You won't be connecting over ssh, but you should use docker instead `ansible_connection=docker`
- You won't have to specify any further connection information as we'll be using the defaults
- Also the host is running python3, how can you tell ansible to use a different python interpreter `ansible_python_interpreter=/usr/bin/python3`

Test

You can now run your playbook. Make sure to pass in the inventory file:

```
$  ansible-playbook -i hosts main.yaml
```

You should see the version of flask (which should be 1.0.2)

Navigate to `~/course/ansible/plugins/inventory`. There are no files yet in this directory.

Virtualbox

Earlier in the course you wrote a dynamic inventory for virtualbox. You will now use the virtualbox inventory plugin: virtualbox (https://docs.ansible.com/ansible/latest/plugins/inventory/virtualbox.html)

- Create a configuration file named `all.vbox.yaml`.
- Configure that you want to use the plugin `plugin: virtualbox`

Now let's have a look at this inventory.

- For now just set the environment variable to whitelist the plugin `ANSIBLE_INVENTORY_ENABLED`
- Use `ansible-inventory` to show a graph of the inventory

  ```
  $ ansible-inventory -i all.vbox.yaml --graph
  ```

- Use the same tool to output all host info in ini format `--list`

Let's use some additional features:

- Limit the hosts only to those *running* `running_only: yes`
- Define two vars from the following virtualbox properties `query:`

variable	property
`vbox_version`	`/VirtualBox/HostInfo/VBoxVerExt`
`ansible_host`	`/VirtualBox/GuestInfo/Net/1/V4/IP`

Check the values of these vars using the `debug` module using ansible ad-hoc:

```
$ ansible all -i all.vbox.yaml -m debug -a var=…
```

The virtual machines you are running have an additional guestproperty named `stack` have a look at `~/course/ansible/machines/Vagrantfile` at how we add this (`v.customize ["guestproperty", "set", :id, "/stack","worldstack"]`). Let's use this to define a group for our servers:

- Define a third variable named `stack`
- Use this to group servers into "worldstack" use `groups` or `keyed_groups`

  ```
  #groups:
  #  worldstack: stack == "worldstack"
  keyed_groups:
    - prefix: ""
      key: stack
      separator: ""
  ```

Check your inventory using a graph, and after that ping each host (you'll need to pass the vagrant user to the command line)

```
$ ansible worldstack -i all.vbox.yaml -m ping -u vagrant
```