

Let's talk about

Ansible Facts

Jan-Piet Mens

May 2023

(No AI was used to generate the content of these slides.)

[@jpmens@mastodon.social](https://mstdn.social/@jpmens)

Moo?

If you see any cows in this presentation,
I'm still making amends.

Not for cow lovers: optionally disable cowsay #1246

 Merged mpdehaan merged 1 commit into `ansible:devel` from `jpmens:nocows`  on Oct 7, 2012

 Conversation 3

 Commits 1

 Checks 0

 Files changed 1



jpmens commented on Oct 6, 2012

Cow-lovers look away now!

This cow patch disables use of installed cowsay if `ANSIBLE_NOCOWS=1` is set in the environment.

Facts

- variables related to remote systems
- obtained automatically
- can be set at runtime
- can be cached
- modules can produce them
- local facts

examples

```
ansible_architecture: "x86_64"  
ansible_default_ipv4.address : "192.0.2.43"  
ansible_distribution: "Debian"  
ansible_windows_domain: "WORKGROUP"  
ansible_hostname: "rabbit"  
ansible_local.nluug.voorjaarsconferentie: "Utrecht"
```

gather facts

```
- name: "Obtain all the facts"
  hosts: unix
  gather_facts: true
  tasks:
    - copy:
      ...

  # re-gather if needed
  - setup:

  # rm *
  - meta:
      clear_facts
```

not all but **some**

```
$ ansible localhost \  
    -m setup \  
    -a gather_subset='!min,!all,dns'
```

```
localhost | SUCCESS => {  
  "ansible_facts": {  
    "ansible_dns": {  
      "domain": "example.com",  
      "nameservers": [  
        "192.0.2.82",  
        "192.0.2.1"  
      ],  
      "search": [  
        "ww.example.com"  
      ]  
    }  
  }  
}
```

subsets

all, all_ipv4_addresses, all_ipv6_addresses,
apparmor, architecture, caps, chroot, cmdline,
date_time, default_ipv4, default_ipv6, devices,
distribution, distribution_major_version,
distribution_release, distribution_version, dns,
effective_group_ids, effective_user_id, env, facter,
fips, hardware, interfaces, is_chroot, iscsi,
kernel, local, lsb, machine, machine_id, mounts,
network, ohai, os_family, pkg_mgr, platform,
processor, processor_cores, processor_count, python,
python_version, real_user_id, selinux, service_mgr,
ssh_host_key_dsa_public, ssh_host_key_ecdsa_public,
ssh_host_key_ed25519_public,
ssh_host_key_rsa_public, ssh_host_pub_keys,
ssh_pub_keys, system, system_capabilities,
system_capabilities_enforced, **user, user_dir,**
user_gecos, user_gid, user_id, user_shell, user_uid,
virtual, virtualization_role, virtualization_type

minimal

```
apparmor, caps, cmdline, date_time,  
distribution, dns, env, fips, local,  
lsb, pkg_mgr, platform, python,  
selinux, service_mgr, ssh_pub_keys,  
user
```


printing



- copy:

```
content: "{{ ansible_facts | to_nice_json }}"
dest: "/tmp/facts.{{ inventory_hostname }}"
mode: "0444"
delegate_to: localhost
```

```
{
  "all_ipv4_addresses": [
    "192.168.1.140",
    "192.168.1.170"
  ],
  "all_ipv6_addresses": [
    REDACTED
  ],
  "ansible_local": {
    "hungry": {
      "dish": "rijsttafel",

```

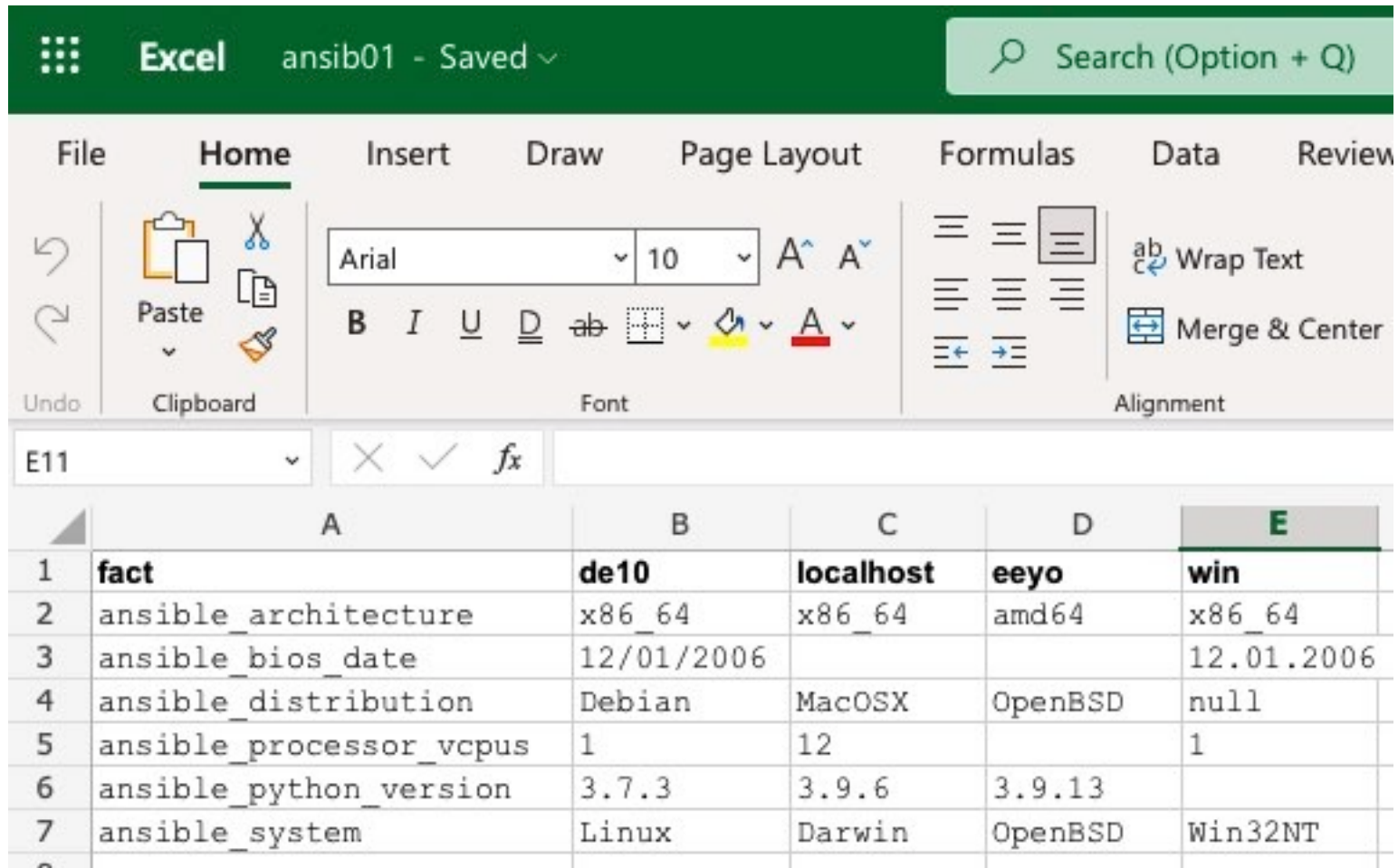


Please consider the environment before printing facts

Fact caching

- store gathered facts for reuse
- default cache is `memory`
- enabled per `ansible.cfg`
- plugins: `jsonfile`, `memcached`, `pickle`, `redis`, `mongodb`, `yaml`, and your own
- plugins can be in a collection (FQCN)
- use a cache to feed your CMDB

seriously?



The screenshot shows the Microsoft Excel interface with the 'Home' tab selected. The ribbon includes options for File, Home, Insert, Draw, Page Layout, Formulas, Data, and Review. The 'Font' group shows 'Arial' font and '10' size. The 'Alignment' group shows 'Wrap Text' and 'Merge & Center' options. The active cell is E11. Below the ribbon is a table with the following data:

	A	B	C	D	E
1	fact	de10	localhost	eeyo	win
2	ansible_architecture	x86_64	x86_64	amd64	x86_64
3	ansible_bios_date	12/01/2006			12.01.2006
4	ansible_distribution	Debian	MacOSX	OpenBSD	null
5	ansible_processor_vcpus	1	12		1
6	ansible_python_version	3.7.3	3.9.6	3.9.13	
7	ansible_system	Linux	Darwin	OpenBSD	Win32NT

Le cache nouveau

```
$ cat ansible.cfg
[defaults]
gathering = smart
fact_caching = jsonfile
fact_caching_timeout = 600
fact_caching_connection = /tmp/fcache

$ ansible-playbook ...

$ jq -r .ansible_local.nluug.lunch < /tmp/fcache/localhost
karnemelk
```

set_fact

Create variables (and facts) on the fly

```
- set_fact:
    sound: moo

- set_fact:
    hotel:
        street: "Winthontlaan 4-6"
        city: "Utrecht"
        tel: 030 8000 800
    cacheable: true
```

```
$ jq .hotel < /tmp/fcache/localhost
{
  "city": "Utrecht",
  "street": "Winthontlaan 4-6",
  "tel": "030 8000 800"
}
```

custom fact gathering

```
#!/usr/bin/env bash

unbound=$( [ -x /usr/sbin/unbound ] && echo true || echo false )
cat <<EOF
{
    "ansible_facts" : {
        "resolver" : $unbound
    }
}
EOF
```

... and configure to use

```
[defaults]
facts_modules = jpfact
```

module facts

Any module can (additionally) produce facts

```
$ ansible localhost -m who
localhost | SUCCESS => {
  "ansible_facts": {
    "who": [
      "jpm",
      "root"
    ]
  },
  "changed": false,
  "fromage": "la 🐮 qui rit"
}
```

who.py

```
from ansible.module_utils.basic import AnsibleModule

if __name__ == '__main__':
    result = dict(changed = False)

    module = AnsibleModule(argument_spec=dict())

    cmd = "who | awk '{print $1}' | sort -u"
    users = module.run_command(cmd, use_unsafe_shell=True)[1]

    result["fromage"] = "la 🐮 qui rit"
    result["ansible_facts"] = {
        "who" : users.split(),
    }

    module.exit_json(**result)
```


Local facts a.k.a **facts.d**

- files in `/etc/ansible/facts.d`
- **MUST** be named `*.fact` on Unix
- can contain INI or JSON
- executables emit JSON to *stdout*

INI facts

```
$ cat /etc/ansible/facts.d/machine.fact
[info]
location=rack49
contact=Jane
```

```
$ ansible localhost -m setup -a filter=ansible_local
localhost | SUCCESS => {
  "ansible_facts": {
    "ansible_local": {
      "machine": {
        "info": {
          "contact": "Jane",
          "location": "rack49"
        }
      }
    }
  }
}
```

Please contact `{{ ansible_local.machine.info.contact }}` for information.

groups from facts

- name: "Construct group from machine/info/location facts"
hosts: unix
gather_subset: ["!min", "!all", "local"]
tasks:
 - **group_by**:
key: "{{ ansible_local.machine.info.location }}"
 - meta:
clear_facts # if cached
- name: "Run actual play on the group"
hosts: **rack49**
gather_subset: ["min", "all_ipv4_addresses"]
tasks:
 - debug:
msg: "{{ ansible_default_ipv4.address }}"

constructed hosts

```
$ cat ansible.cfg
[defaults]
inventory = inventories/

[inventory]
enable_plugins = ini, constructed

$ cat inventories/zz-constr.config
---
plugin: constructed
strict: false
groups:
  zsrv: ansible_local.system.zabbix.role == "server"
  zcli: ansible_local.system.zabbix.role == "agent"
  staging: '-stag-' in inventory_hostname

keyed_groups:
  # this creates a group per machine rack
  - prefix: loc
    key: ansible_local.machine.info.location
```

facts on **Windows**

- no standard `facts.d` path
- **MUST** be `*.ps1` or `*.json` files
- PowerShell scripts output raw
hashtables, arrays, or primitive objects
- alongside other Ansible facts
- no INI (?!)

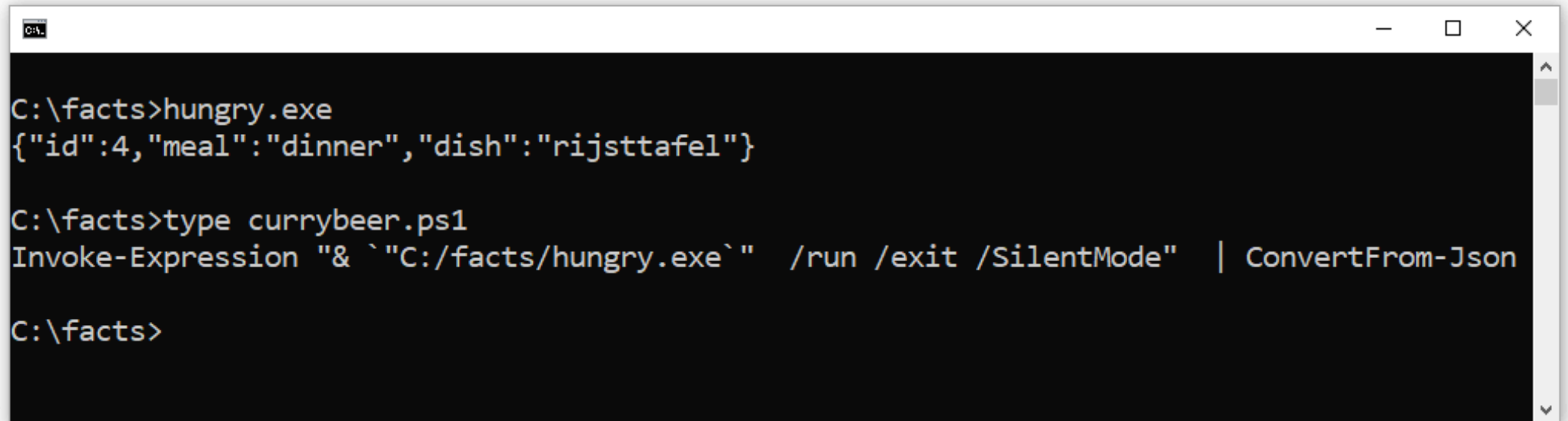
ENOINI ?

```
Subsets = 'local'
Code = {
    if (Test-Path -Path $factPath) {
        $factFiles = Get-ChildItem -Path $factpath | Where-Object {
            -not $_.PSIsContainer -and @('.ps1', '.json') -ccontains
            $_.Extension
        }

        foreach ($factsFile in $factFiles) {
            $out = if ($factsFile.Extension -eq '.ps1') {
                & $($factsFile.FullName)
            }
            elseif ($factsFile.Extension -eq '.json') {
                Get-Content -Raw -Path $factsFile.FullName |
                ConvertFrom-Json
            }
            if ($null -ne $out) {
                $ansibleFacts."ansible_{$factsFile.BaseName}" = $out
            }
        }
    }
}
```

ansible/windows/plugins/modules/setup.ps1

facts on **Windows**



```
C:\facts>hungry.exe
{"id":4,"meal":"dinner","dish":"rijsttafel"}

C:\facts>type currybeer.ps1
Invoke-Expression "& `\"C:/facts/hungry.exe`\" /run /exit /SilentMode" | ConvertFrom-Json

C:\facts>
```

Changing **fact_path**

```
- hosts: all
gather_facts: true
module_defaults:
  setup:
    fact_path: "{{ (os == 'win') |
      ternary('c:/facts', '/etc/ansible/facts.d') }}"
    fact_path: "{{ fpath }}"
tasks:
- debug:
  msg: "NLUUG = {{ ansible_facts['nluug'] |
    default(ansible_local['nluug']) }}"

- set_fact:
  dinner: "{{ (os == 'win') |
    ternary(ansible_currybeer, ansible_local.hungry) }}"

- debug:
  msg: "{{ dinner.dish }} for {{ dinner.meal }}"
```


Windows & Unix

```
TASK [debug] *****
ok: [localhost] => {
    "msg": "NLUUG = {'voorjaarsconferentie': 'Utrecht', 'lunch': 'karnemelk'}"
}
ok: [win] => {
    "msg": "NLUUG = {'voorjaarsconferentie': 'Utrecht', 'lunch': 'karnemelk'}"
}

TASK [set_fact] *****
ok: [localhost]
ok: [win]

TASK [debug] *****
ok: [localhost] => {
    "msg": "rijsttafel for dinner"
}
ok: [win] => {
    "msg": "rijsttafel for dinner"
}
```

possible solution

Put all local facts in a `local.ps1` file on Windows

```
C:\facts > type local.ps1
@{
    hungry = & "C:/facts/hungry.exe" | ConvertFrom-Json
}

$ ansible win -m setup -a 'fact_path=c:/facts'

$ jq .ansible_local.hungry /tmp/fcache/win
{
  "dish": "rijsttafel",
  "id": 4,
  "meal": "dinner"
}
```

ansible-cmdb

```
$ pip install ansible-cmdb
$ ansible all -m setup --tree o/
$ ansible-cmdb o/ > cmdb.html
```

The screenshot shows a web browser window displaying the 'Host Overview' page of the ansible-cmdb tool. The page header indicates it was generated on Mon Mar 20 16:25:52 2023 by jpm @ rabbit.wm.mens.de. Below the header, there is a section for 'Shown columns' with various filterable columns like Name, OS, vCPUs, and Mem Usage. The main content area is titled 'Host overview' and contains a table with 3 entries. The table has columns for Name, OS, Kernel, vCPUs, and Mem Usage. The 'wooz' entry shows a progress bar for memory usage at 0.3 / 1.0 GiB.

Host Overview Generated on Mon Mar 20 16:25:52 2023 by jpm @ rabbit.wm.mens.de [Clear settings](#)

Shown columns

Name Groups Cust DTAP Comment Ext ID FQDN Main IP All IPv4 All IPv6 OS
Kernel Arch Virt CPU type vCPUs RAM [GiB] Mem Usage Swap Usage Disk usage
PhysDisk size Nr of Ifaces Timestamp Product Name Product Serial

Host overview Search:

Name	OS	Kernel	vCPUs	Mem Usage
localhost	MacOSX 12.6.3	21.6.0	12	n/a
win	None 10.0.19044.0	10.0.19044.0	1	n/a
wooz	Debian 11	5.10.0-20-amd64	1	<div style="width: 30%;"><div style="width: 30%;"></div></div> (0.3 / 1.0 GiB)

Showing 1 to 3 of 3 entries

Generated by **ansible-cmdb** v1.31 - © Ferry Boender 2017

ansible-cmdb

The screenshot shows a web browser window displaying the ansible-cmdb interface. The browser's address bar shows a file path. The page header includes the text 'localhost', a 'Back to top' link, the generation date and time 'Generated on Mon Mar 20 16:25:52 2023 by jpm @ rabbit.wv.mens.de', and a 'Clear settings' button.

The main content area is organized into sections:

- General**
- Host local facts**

These are variables read from the host's /etc/ansible/facts.d directory.

hungry	dish	rijsttafel
	id	4
	meal	dinner
nluug	lunch	karnemelk
	voorjaarsconferentie	Utrecht
- Hardware**

Vendor	
Product name	Macmini8,1
Product serial	
Architecture	x86_64
Form factor	
Virtualization role	

At the bottom right of the page, it says 'Generated by ansible-cmdb v1.31 - © Ferry Boender 2017'.

the **good** and the ugly

- Fact caching enabled per `ansible.cfg`
- Beware fact decay on high cache TTL
e.g. `ansible_date_time`
- Incompatibilities Unix/Windows:
missing `*.INI`, facts in different keys
- Feed Ansible facts to CMDB
- Wot? `"epoch": "1679577220,95027"`

aim for compatibility

- Use JSON fact files for Unix & Windows

```
- name: "Install JSON fact file"
  copy:
    content:
      forecast:
        day: Tuesday
        temperature: 17.3
    dest: "{{ weather_file }}"
```

- INI on Windows maybe with Get-IniFile via <https://stackoverflow.com/questions/43690336/>

that's a ~~fact~~ **wrap**

one last **cowch**

