OPITZ CONSULTING

CONFIGURATION

INVENTORIES

- Contain **all hosts** that Ansible should know
- Hosts can be assigned to **groups**
- Can contain **variables**
  (Best practice: outsource variables)
- Static Inventory Format
  - INI, YAML or JSON
- Inventory parameter can point to directory with many files

Default: /etc/ansible/hosts

```
localhost ansible_connection=local


[webservers]

node1.example.com ; Some comment

node2.example.com www_root=/var/http ; host var

node[3-10].example.com ;


[webservers:vars]

www_root=/var/www/ ; var for all webservers
```

USE OF GROUPS

- Primary addressing of hosts
  - How do I want to address my hosts?
- Meaningful groupings
  - Function (database, webserver, ...)
  - Location (dc1, dc2, frankfurt, berlin, ...)
  - Stage (develop, production)
  - Infrastructure type (vms, aws, ...)
- Multiple grouping possible
- Group hierarchies possible

```
[web]
web1 ansible_host=w0102.acme.com
web2 ansible_host=w0103.acme.com

[db]
db1 ansible_host=w0104.acme.com
db2 ansible_host=w0105.acme.com

[dev]
db1
web1

[prod]
db2
web2
```

SELECTION OF HOSTS

Example: `ansible [selection] -m ping`

Selection	Hosts
all	web1, web2, db1, db2
webservers1	web1, web2
webservers1:db	web1, web2, db1, db2
webservers1:&prod	web2
webservers1:db:&prod	web2, db2
webservers1[0]	web1
webservers1:! webservers2	web1

```
[webservers1]
web1 ansible_host=w0101.acme.com
web2 ansible_host=w0102.acme.com

[webservers2]
web2 ansible_host=w0102.acme.com
web3 ansible_host=w0103.acme.com

[db]
db1 ansible_host=w0104.acme.com
db2 ansible_host=w0105.acme.com

[dev]
web1
db1

[prod]
web2
db2
```

INI-Sample

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
halon_system_timeout=30
self_destruct_countdown=60
escape_pods=2

[usa:children]
southeast
northeast
southwest
northwest
```

YAML-Sample

```
usa:
children:
southeast:
children:
atlanta:
hosts:
host1:
host2:
raleigh:
hosts:
host2:
host3:
vars:
some_server: foo.southeast.example.com
halon_system_timeout: 30
self_destruct_countdown: 60
escape_pods: 2
northeast:
northwest:
southwest:
```

DYNAMIC INVENTORIES

- Inventory is determined dynamically
- Useful for
  - Dynamic infrastructure (e.g. cloud provider, VMware)
  - Externally managed infrastructure (CMDB)

- Variant 1: Own inventory script (e.g. bash, python, ruby, ...)
  - Standardized interface
  - Understands parameters --list and --host [host]
  - Provides JSON-formatted inventory

- Variant 2: Inventory plug-in
  - Plugins deactivated by default (enable_plugins in Config)
  - Many available (`ansible-doc -t inventory -l`)

INVENTORY PLUGIN

```
ansible-doc -t inventory -l
```

nmap Uses nmap to find hosts to target
host_list Parses a 'host list' string
hcloud Ansible dynamic inventory plugin for the Hetzner Cloud
openstack OpenStack inventory source
vultr Vultr inventory source
aws_ec2 EC2 inventory source
cloudscale cloudscale.ch inventory source
virtualbox virtualbox inventory source
constructed Uses Jinja2 to construct vars and groups based on existing inventory
k8s Kubernetes (K8s) inventory source
azure_rm Azure Resource Manager inventory plugin
script Executes an inventory script that returns JSON
vmware_vm_inventory VMware Guest inventory source
openshift OpenShift inventory source
docker_machine Docker Machine inventory source
yaml Uses a specific YAML file as an inventory source

CONFIGURATION OF ANSIBLE

- What can be configured (among other things)
  - Path to inventory, path to additional roles
  - Number of forks (parallelism)
  - SSH options (e.g. sudo user, pipelining)
  - Dealing with "facts”
  - Etc.

CONFIGURATION OF ANSIBLE

- Sources for configuration (by ranking, will be aggregated)
  - Configuration settings
    - Configuration file
    - Environment variables
  - Command-line options
  - Playbook keywords
  - Variables

- https://docs.ansible.com/ansible/latest/reference_appendices/general_precedence.html#general-precedence-rules

PRECEDENCE EXAMPLE – BECOME (SUPERUSER)

- What did we learn:
  - Configuration settings
  - Command-line options
  - Playbook keywords
  - Variables

```
[privilege_escalation]
# (boolean) Toggles the use of privilege escalation, allowing you to 'become' another user after login.
become=True
```

```
ansible-playbook -b site.yml
```

```yaml
---
- hosts: testkiste
  gather_facts: false
  become: false
  tasks:
    - name: test
      ansible.builtin.apt:
        name: unzip
```

```yaml
---
- hosts: testkiste
  gather_facts: false
  become: true
  become_user: sysadmin
  vars:
    ansible_become_user: admin
  tasks:
    - name: test
      ansible.builtin.apt:
        name: unzip
```

- Result: become is set to false in the playbook run

ANSIBLE.CFG EXAMPLE VALUES

- Configuration file (Ansible picks the first it finds)
  - ANSIBLE_CONFIG=/path/to/file
  - ./ansible.cfg
  - ~/.ansible.cfg
  - /etc/ansible/ansible.cfg

```
[defaults]
# (pathlist) Comma separated list of Ansible inventory sources
;inventory=/etc/ansible/hosts

# (path) Option for connections using a certificate or key file to authenticate, rather than an age
;private_key_file=

# (integer) Port to use in remote connections, when blank it will use the connection plugin default
;remote_port=

# (string) Sets the login user for the target machines
# When blank it uses the connection plugin's default, normally the user currently executing Ansible
;remote_user=

# (pathspec) Colon separated paths in which Ansible will search for Roles.
;roles_path={{ ANSIBLE_HOME ~ "/roles:/usr/share/ansible/roles:/etc/ansible/roles" }}

[privilege_escalation]
# (boolean) Toggles the use of privilege escalation, allowing you to 'become' another user after lo
;become=False
```

- Default cfg ist generated with:
  - ansible-config init --disabled > ansible.cfg