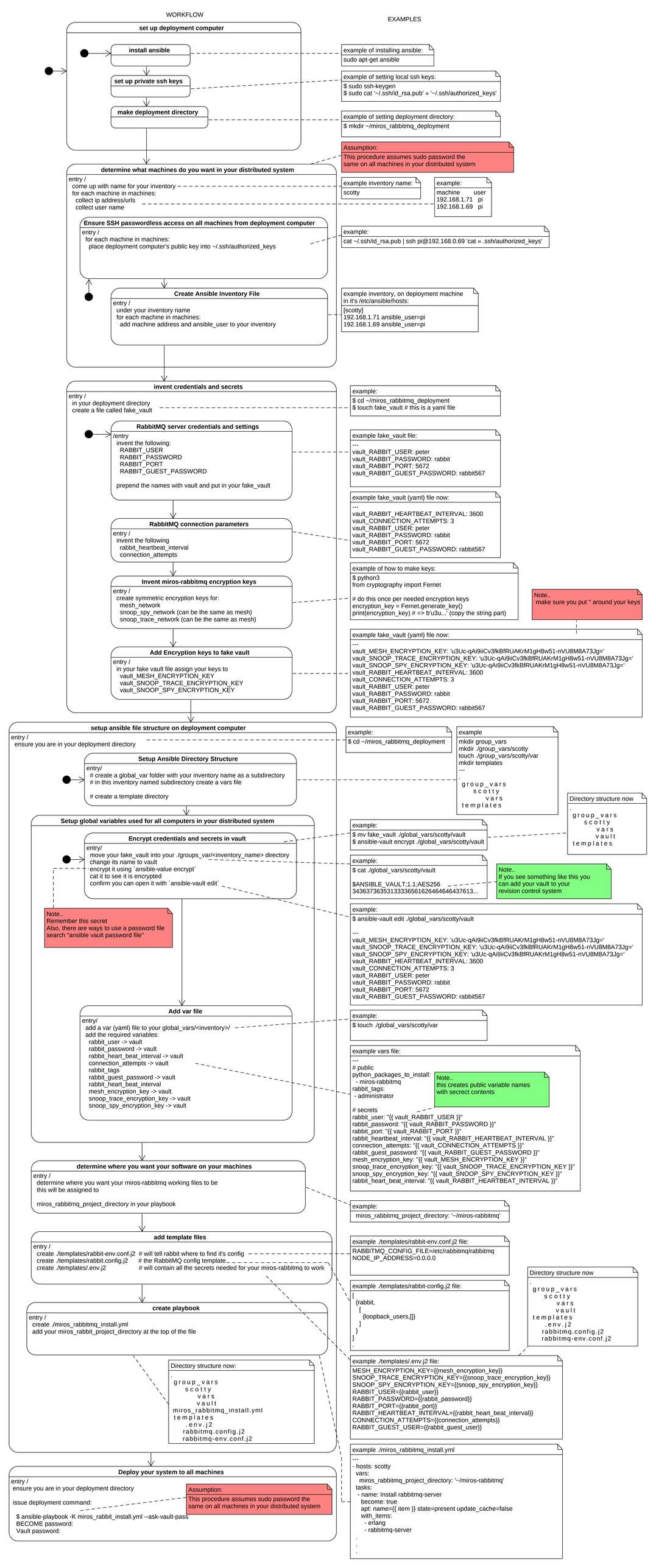
WORKFLOW
EXAMPLES
set up deployment computer
install ansible
set up private ssh keys
make deployment directory
example of installing ansible:
sudo apt-get ansible
example of setting local ssh keys:
$ sudo ssh-keygen
$ sudo cat '~/.ssh/id_rsa.pub' » '~/.ssh/authorized_keys'
example of setting deployment directory:
$ mkdir ~/miros_rabbitmq_deployment
Assumption:
This procedure assumes sudo password the
same on all machines in your distributed system
determine what machines do you want in your distributed system
entry /
come up with name for your inventory
for each machine in machines:
collect ip address/urls
collect user name
example inventory name:
scotty
example:
machine user
192.168.1.71 pi
192.168.1.69 pi
Ensure SSH passwordless access on all machines from deployment computer
entry /
for each machine in machines:
place deployment computer's public key into ~/.ssh/authorized_keys
example:
cat ~/.ssh/id_rsa.pub | ssh pi@192.168.0.69 'cat » .ssh/authorized_keys'
Create Ansible Inventory File
entry /
under your inventory name
for each machine in machines:
add machine address and ansible_user to your inventory
example inventory, on deployment machine
in it's /etc/ansible/hosts:
[scotty]
192.168.1.71 ansible_user=pi
192.168.1.69 ansible_user=pi
invent credentials and secrets
entry /
in your deployment directory
create a file called fake_vault
example:
$ cd ~/miros_rabbitmq_deployment
$ touch fake_vault # this is a yaml file
RabbitMQ server credentials and settings
/entry
invent the following:
RABBIT_USER
RABBIT_PASSWORD
RABBIT_PORT
RABBIT_GUEST_PASSWORD
prepend the names with vault and put in your fake_vault
example fake_vault file:
---
vault_RABBIT_USER: peter
vault_RABBIT_PASSWORD: rabbit
vault_RABBIT_PORT: 5672
vault_RABBIT_GUEST_PASSWORD: rabbit567
RabbitMQ connection parameters
entry /
invent the following
rabbit_heartbeat_interval
connection_attempts
example fake_vault (yaml) file now:
---
vault_RABBIT_HEARTBEAT_INTERVAL: 3600
vault_CONNECTION_ATTEMPTS: 3
vault_RABBIT_USER: peter
vault_RABBIT_PASSWORD: rabbit
vault_RABBIT_PORT: 5672
vault_RABBIT_GUEST_PASSWORD: rabbit567
Invent miros-rabbitmq encryption keys
entry /
create symmetric encryption keys for:
mesh_network
snoop_spy_network (can be the same as mesh)
snoop_trace_network (can be the same as mesh)
example of how to make keys:
$ python3
from cryptography import Fernet
# do this once per needed encryption keys
encryption_key = Fernet.generate_key()
print(encryption_key) # => b'u3u...' (copy the string part)
Note..
make sure you put '' around your keys
Add Encryption keys to fake vault
entry /
in your fake vault file assign your keys to
vault_MESH_ENCRYPTION_KEY
vault_SNOOP_TRACE_ENCRYPTION_KEY
vault_SNOOP_SPY_ENCRYPTION_KEY
example fake_vault (yaml) file now:
---
vault_MESH_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_SNOOP_TRACE_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_SNOOP_SPY_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_RABBIT_HEARTBEAT_INTERVAL: 3600
vault_CONNECTION_ATTEMPTS: 3
vault_RABBIT_USER: peter
vault_RABBIT_PASSWORD: rabbit
vault_RABBIT_PORT: 5672
vault_RABBIT_GUEST_PASSWORD: rabbit567
setup ansible file structure on deployment computer
entry /
ensure you are in your deployment directory
example:
$ cd ~/miros_rabbitmq_deployment
example
mkdir group_vars
mkdir ./group_vars/scotty
touch ./group_vars/scotty/var
mkdir templates

.
group_vars
scotty
vars
templates
Setup Ansible Directory Structure
entry/
create a global_var folder with your inventory name as a subdirectory
in this inventory named subdirectory create a vars file
create a template directory
Directory structure now
.
group_vars
scotty
vars
vault
templates
Setup global variables used for all computers in your distributed system
Encrypt credentials and secrets in vault
entry/
move your fake_vault into your ./groups_var/<inventory_name> directory
change its name to vault
encrypt it using `ansible-value encrypt`
cat it to see it is encrypted
confirm you can open it with `ansible-vault edit`
example:
$ mv fake_vault ./global_vars/scotty/vault
$ ansible-vault encrypt ./global_vars/scotty/vault
example:
$ cat ./global_vars/scotty/vault
$ANSIBLE_VAULT;1.1;AES256
34363736353133333656162646464637613...
Note..
If you see something like this you
can add your vault to your
revision control system
Note..
Remember this secret
Also, there are ways to use a password file
search "ansible vault password file"
example:
$ ansible-vault edit ./global_vars/scotty/vault
---
vault_MESH_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_SNOOP_TRACE_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_SNOOP_SPY_ENCRYPTION_KEY: 'u3Uc-qAi9iiCv3fkBfRUAKrM1gH8w51-nVU8M8A73Jg='
vault_RABBIT_HEARTBEAT_INTERVAL: 3600
vault_CONNECTION_ATTEMPTS: 3
vault_RABBIT_USER: peter
vault_RABBIT_PASSWORD: rabbit
vault_RABBIT_PORT: 5672
vault_RABBIT_GUEST_PASSWORD: rabbit567
Add var file
entry/
add a var (yaml) file to your global_vars/<inventory>/
add the required variables:
rabbit_user -> vault
rabbit_password -> vault
rabbit_heart_beat_interval -> vault
connection_attempts -> vault
rabbit_tags
rabbit_guest_password -> vault
rabbit_heart_beat_interval
mesh_encryption_key -> vault
snoop_trace_encryption_key -> vault
snoop_spy_encryption_key -> vault
example:
$ touch ./global_vars/scotty/var
example vars file:
---
public
python_packages_to_install:
- miros-rabbitmq
rabbit_tags:
- administrator
secrets
rabbit_user: "{{ vault_RABBIT_USER }}"
rabbit_password: "{{ vault_RABBIT_PASSWORD }}"
rabbit_port: "{{ vault_RABBIT_PORT }}"
rabbit_heartbeat_interval: "{{ vault_RABBIT_HEARTBEAT_INTERVAL }}"
connection_attempts: "{{ vault_CONNECTION_ATTEMPTS }}"
rabbit_guest_password: "{{ vault_RABBIT_GUEST_PASSWORD }}"
mesh_encryption_key: "{{ vault_MESH_ENCRYPTION_KEY }}"
snoop_trace_encryption_key: "{{ vault_SNOOP_TRACE_ENCRYPTION_KEY }}"
snoop_spy_encryption_key: "{{ vault_SNOOP_SPY_ENCRYPTION_KEY }}"
rabbit_heart_beat_interval: "{{ vault_RABBIT_HEARTBEAT_INTERVAL }}"
Note..
this creates public variable names
with secrect contents
determine where you want your software on your machines
entry /
determine where you want your miros-rabbitmq working files to be
this will be assigned to
miros_rabbitmq_project_directory in your playbook
example:
miros_rabbitmq_project_directory: '~/miros-rabbitmq'
add template files
entry /
create ./templates/rabbit-env.conf.j2 # will tell rabbit where to find it's config
create ./templates/rabbit.config.j2 # the RabbitMQ config template
create ./templates/.env.j2 # will contain all the secrets needed for your miros-rabbitmq to work
example ./templates/rabbit-env.conf.j2 file:
RABBITMQ_CONFIG_FILE=/etc/rabbitmq/rabbitmq
NODE_IP_ADDRESS=0.0.0.0
example ./templates/rabbit-config.j2 file:
[
{rabbit,
[
{loopback_users,[]}
]
}
].
Directory structure now
.
group_vars
scotty
vars
vault
templates
.env.j2
rabbitmq.config.j2
rabbitmq-env.conf.j2
create playbook
entry /
create ./miros_rabbitmq_install.yml
add your miros_rabbit_project_directory at the top of the file
Directory structure now:
.
group_vars
scotty
vars
vault
miros_rabbitmq_install.yml
templates
.env.j2
rabbitmq.config.j2
rabbitmq-env.conf.j2
example ./templates/.env.j2 file:
MESH_ENCRYPTION_KEY={{mesh_encryption_key}}
SNOOP_TRACE_ENCRYPTION_KEY={{snoop_trace_encryption_key}}
SNOOP_SPY_ENCRYPTION_KEY={{snoop_spy_encryption_key}}
RABBIT_USER={{rabbit_user}}
RABBIT_PASSWORD={{rabbit_password}}
RABBIT_PORT={{rabbit_port}}
RABBIT_HEARTBEAT_INTERVAL={{rabbit_heart_beat_interval}}
CONNECTION_ATTEMPTS={{connection_attempts}}
RABBIT_GUEST_USER={{rabbit_guest_user}}
example ./miros_rabbitmq_install.yml

- hosts: scotty
vars:
miros_rabbitmq_project_directory: '~/miros-rabbitmq'
tasks:
- name: Install rabbitmq-server
become: true
apt: name={{ item }} state=present update_cache=false
with_items:
- erlang
- rabbitmq-server
.
.
.
Deploy your system to all machines
entry /
ensure you are in your deployment directory
issue deployment command:
$ ansible-playbook -K miros_rabbit_install.yml --ask-vault-pass
BECOME password:
Vault password:
Assumption:
This procedure assumes sudo password the
same on all machines in your distributed system