Cloud Provisioning with Ansible

Magnus Glantz (@mglantz)

Senior Solution Architect

Content contributions by:

Bianca Henderson (@bizonks)

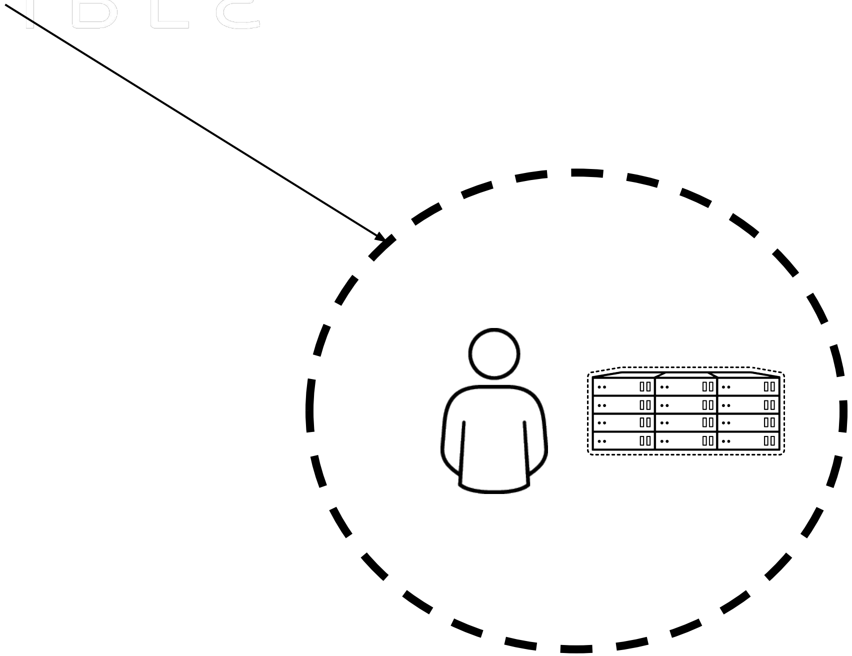
John Lieske (@johnlieske)

Jake Jackson (@thedoubl3j)

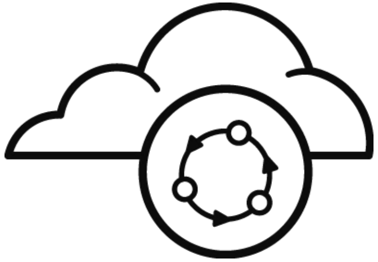
Sr. Product Field Engineers, Getting Started Team

Magnus Glantz (@mglantz), Senior Cloud|Infra SA

Automation context



Automation context: the cloud



Automation context: the cloud

Computers Storage Network devices Some security Complex systems

Automation context: the cloud

Automation context: the cloud

Automation context: the cloud

Automation context: the cloud

Automation context: the cloud

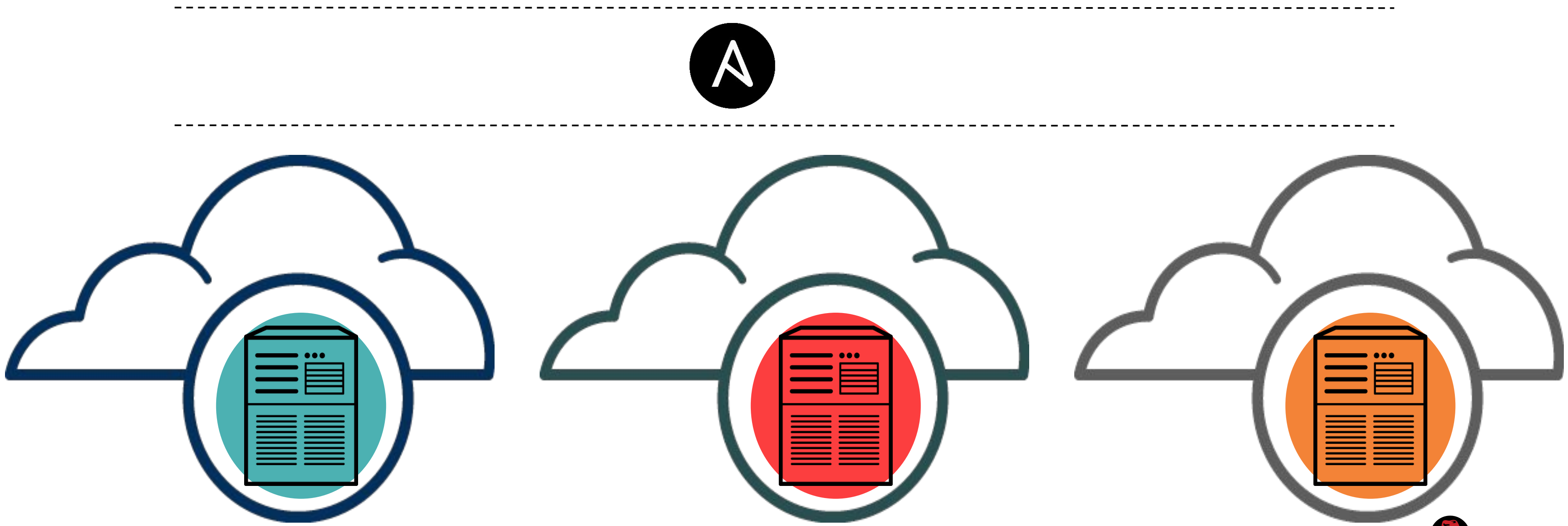
Automation context: the cloud

Automation context: the cloud

“Only **12%** of the **Fortune 500** firms in **1955** existed in **2015.** Close to **9/10** have been **eliminated**”

ANSIBLE

Automation context: the cloud



Automation context: processes

Provisioning Configuring Troubleshooting

Automation context: processes

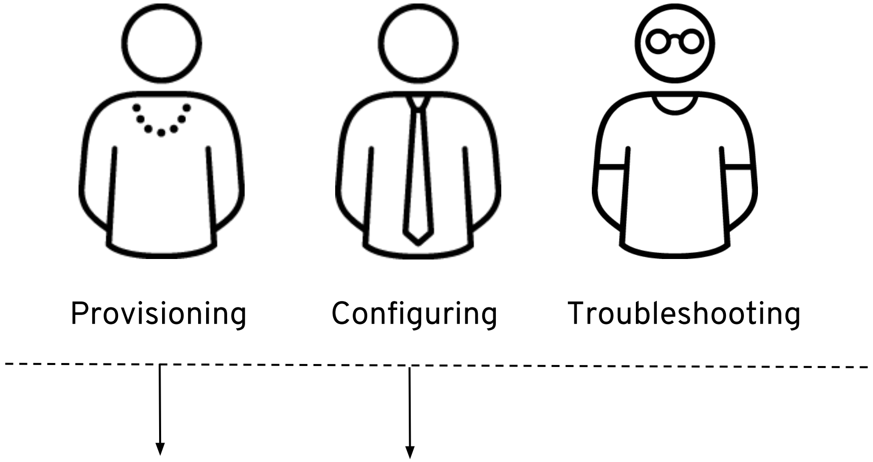
Provisioning

Configuring

Troubleshooting

Put in place
new things

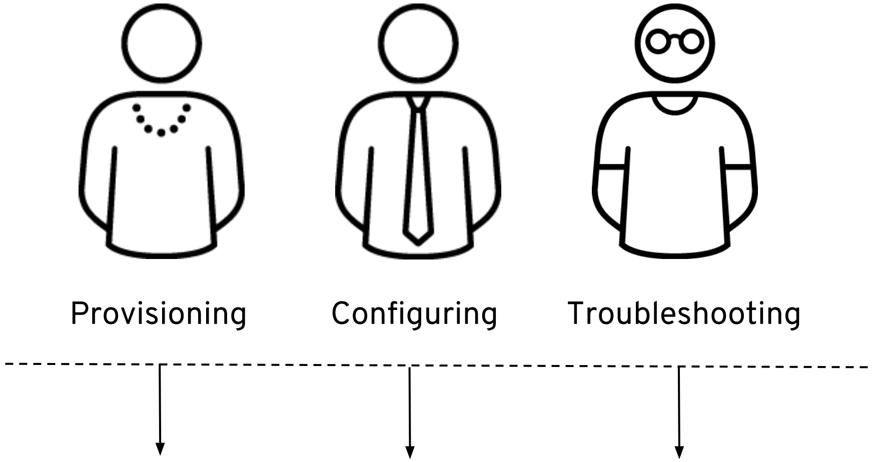
Automation context: processes



Put in place new things

Change existing things

Automation context: processes



Put in place new things

Change existing things

Find out why things doesn't work

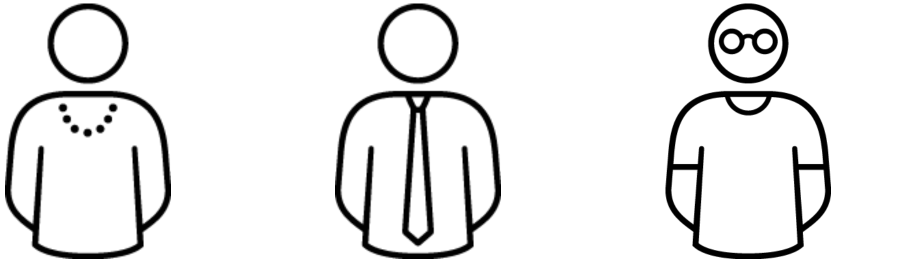
Automation context: processes

Provisioning Configuring Troubleshooting

ANSIBLE

Ansible: what can you automate in a cloud

Ansible: what can you automate in a cloud



Provisioning Configuring Troubleshooting

ALL THE THINGS

Ansible: what can you automate in a cloud

Provisioning

Configuring

Troubleshooting

ALL THE THINGS

The end.

Ansible: what can you automate in a cloud

Provisioning Configuring Troubleshooting

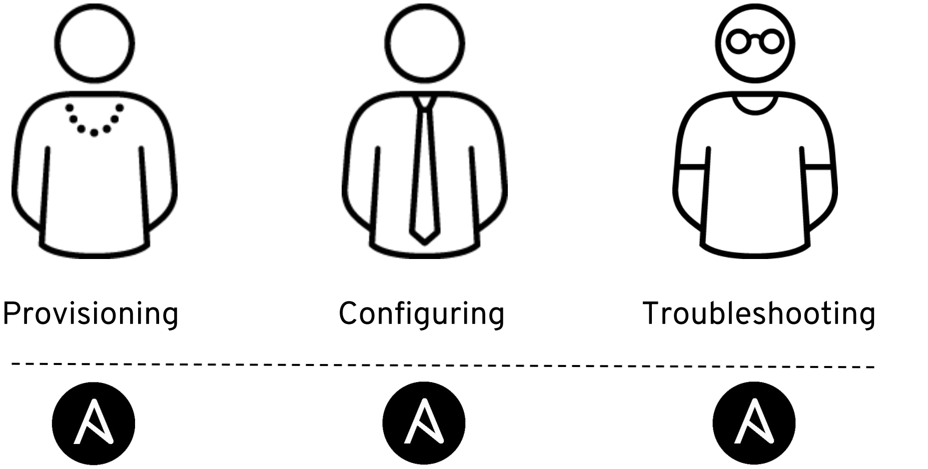
ANSIBLE

Ansible: Common complimenting tools

Provisioning | Configuring | Troubleshooting

- Terraform
- AWS CloudFormation
- Azure Resource Templates
- Google Cloud Deployment Manager

Ansible: Common complimenting provisioning tools

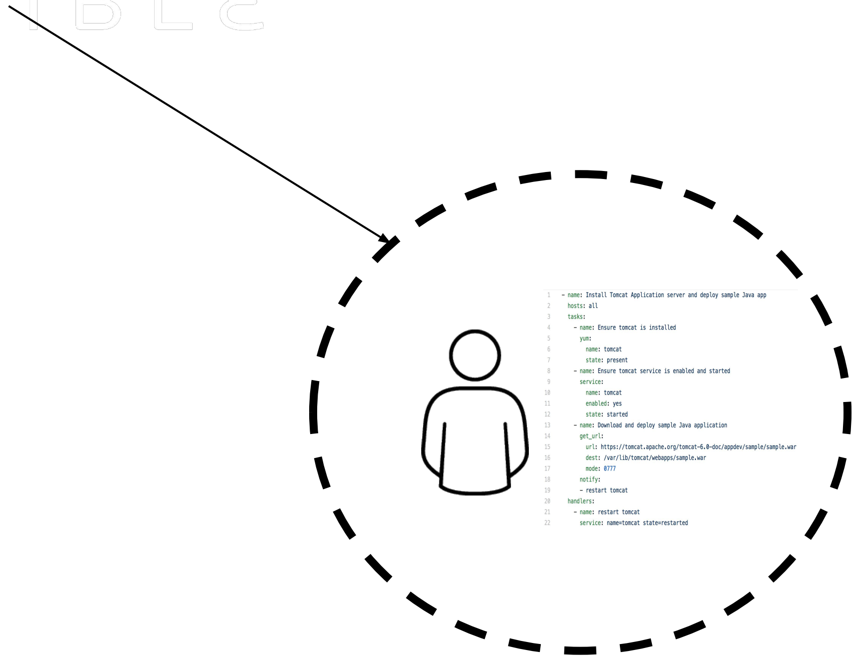


Often different depending on platform

Often the same no matter the platform

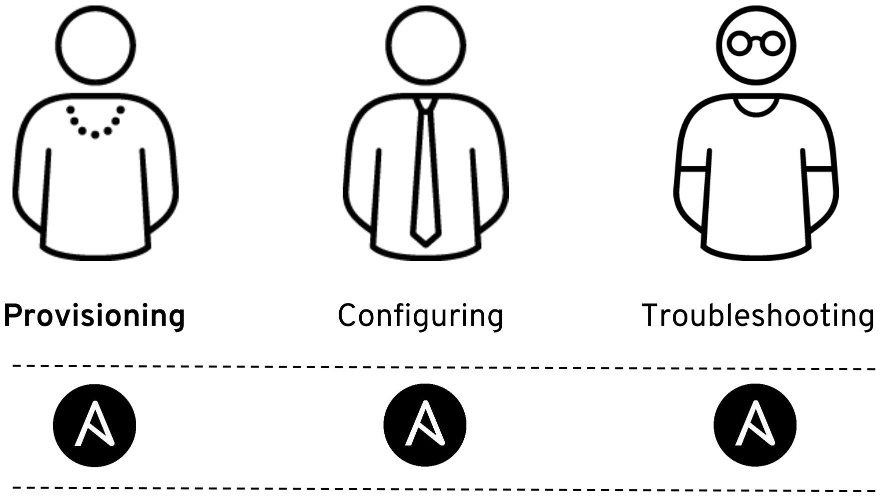
Often the same no matter the platform

Automation specifics



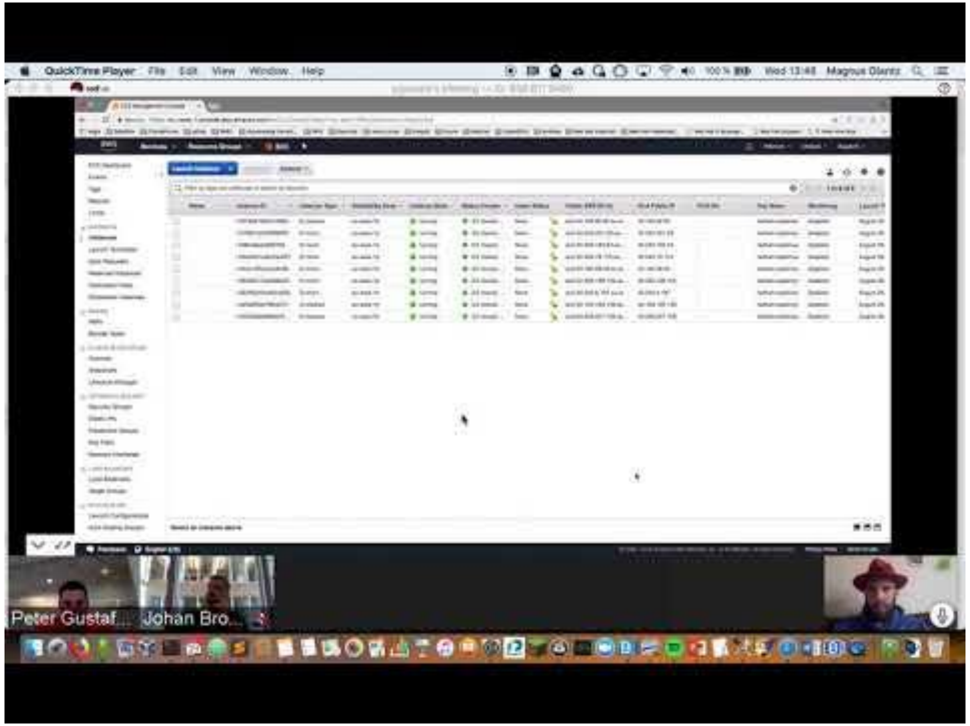
```
- name: Install Tomcat Application server and deploy sample Java app
  hosts: all
  tasks:
    - name: Ensure tomcat is installed
      yum:
        name: tomcat
        state: present
    - name: Ensure tomcat service is enabled and started
      service:
        name: tomcat
        enabled: yes
        state: started
    - name: Download and deploy sample Java application
      get_url:
        url: https://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/sample.war
        dest: /var/lib/tomcat/webapps/sample.war
        mode: 0777
      notify:
      - restart tomcat
  handlers:
    - name: restart tomcat
      service: name=tomcat state=restarted
```

Ansible: Automation specifics: provisioning



Often different depending on platform

Demo: 960 VMs on AWS in seconds with Ansible



Ansible (2.7): Cloud related Modules

Total number of modules: 2078
Total number of cloud modules: 768
Total number of providers: 34

Cloud related solutions: dynamic inventories

```
ansible -i ec2.py all -m ping
```

```
ansible -i azure_rm.py all -m ping
```

```
ansible -i gce.py all -m ping
```

READ MORE: https://docs.ansible.com/ansible/2.7/user_guide/intro_dynamic_inventory.html

Cloud related solutions: inventory plugin (aws)

```
# Fetch all hosts in us-east-1, the hostname is the public DNS if it exists, otherwise the private IP
address
plugin: aws_ec2
regions:
  - us-east-1
```

READ MORE: https://docs.ansible.com/ansible/2.7/plugins/inventory/aws_ec2.html

Cloud related solutions: inventory plugin (azure)

```
# required for all azure_rm inventory plugin configs
plugin: azure_rm

# forces this plugin to use a CLI auth session instead of the automatic auth source selection (eg,
prevents the
# presence of 'ANSIBLE_AZURE_RM_X' environment variables from overriding CLI auth)
auth_source: cli

# fetches VMs from an explicit list of resource groups instead of default all (- '*')
include_vm_resource_groups:
- myrg1
```

READ MORE: https://docs.ansible.com/ansible/2.7/plugins/inventory/azure_rm.html

Cloud related solutions: inventory plugin (gce)

```
plugin: gcp_compute
zones: # populate inventory with instances in these regions
  - us-east1-a
projects:
  - gcp-prod-gke-100
  - gcp-cicd-101
```

READ MORE: https://docs.ansible.com/ansible/2.7/plugins/inventory/gcp_compute.html

Cloud related solutions: add_host

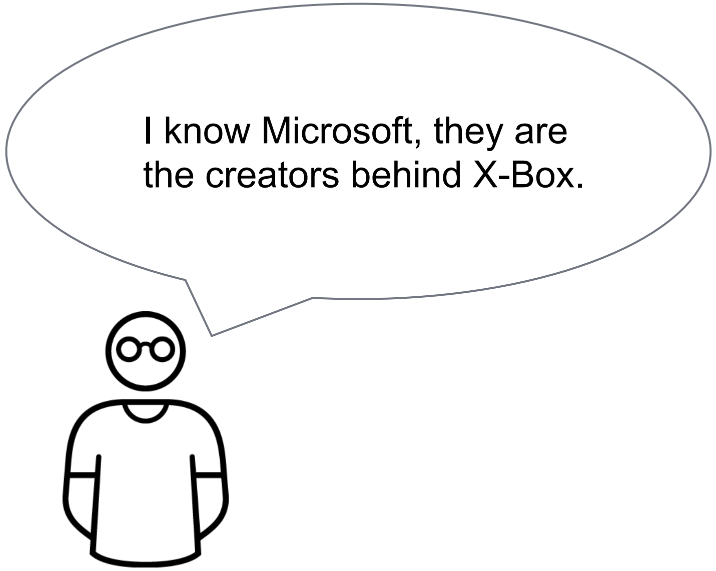
```
---
- name: "[Play 1] Deploy VMs in Amazon EC2"
  hosts: localhost
  gather_facts: false
  connection: local

  tasks:
  - name: Include vars to be used
    include_vars: vars/vars.yml

  - name: Provision Ansible Tower VMs
    ec2:
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      key_name: "{{ec2_key}}"
      region: "{{ ec2_region }}"
      group: "{{ ec2_security_group_tower }}"
      instance_type: t2.medium
      image: "{{ ami_id }}"
      user_data: "{{ lookup('file', '/tmp/tower-prep.sh') }}"
      wait: true
      exact_count: "{{ number_of_tower_systems }}"
      count_tag:
        identity: tower
      instance_tags:
        identity: tower
    register: ec2large

  - name: Setup in-memory inventory for just created VMs
    add_host:
      name: hostname={{ item.public_ip }}
      groups: just_created_vms
    with_items: "{{ ec2large.tagged_instances }}"
```

```
- name: "[Play 2] Post Configuration of VMs, put Ansible Tower software in place"
  hosts: just_created_vms

  tasks:
  - name: Ensure /opt/tower is created
    file:
      path: /opt/tower
      state: directory

  - name: Unzip the latest tower software
    unarchive:
      src: "https://releases.ansible.com/ansible-tower/setup/ansible-tower-setup-latest.tar.gz"
      dest: /opt/tower
      remote_src: yes
```

READ MORE:
https://docs.ansible.com/ansible/latest/modules/add_host_module.html

ANSIBLE

Provisioning Azure VMs with Ansible



Make Sure Ansible is Installed

Azure CLI will need to be version 2.0.4 or later.
Run the `az --version` command to find the version. If the CLI command is named `azure` instead of `az` then it's too old.

Acquire Azure Credentials

For a development environment, create a *credentials* file for Ansible on your Cloud Shell. First, type this command:

```
az ad sp create-for-rbac
```

Acquire Azure Credentials (cont.)

To find out what your subscription ID is, type in:

```
az account show --query "{ subscription_id: id }"
```

Output like this should show up; copy this information into a text file so that you can copy/paste it later:

```
{
  "subscription_id": "854c5e9a-ed49-687e-bc7a-96ed7315095"
}
```

Acquire Azure Credentials (cont.)

Then, type this command in:

```
az ad sp create-for-rbac --query '{"client_id": appId,
"secret": password, "tenant": tenant}'
```

Output like this should show up:

```
{
  "client_id": "eec5624a-90f8-4386-8a87-02730b5410d5",
  "secret": "531dcffa-3aff-4488-99bb-4816c395ea3f",
  "tenant": "72f988bf-86f1-41af-91ab-2d7cd011db47"
}
```

Configure Ansible to Use Azure Credentials

```
cd ~/.azure
vi ~/.azure/credentials
```

Format for the credentials file:

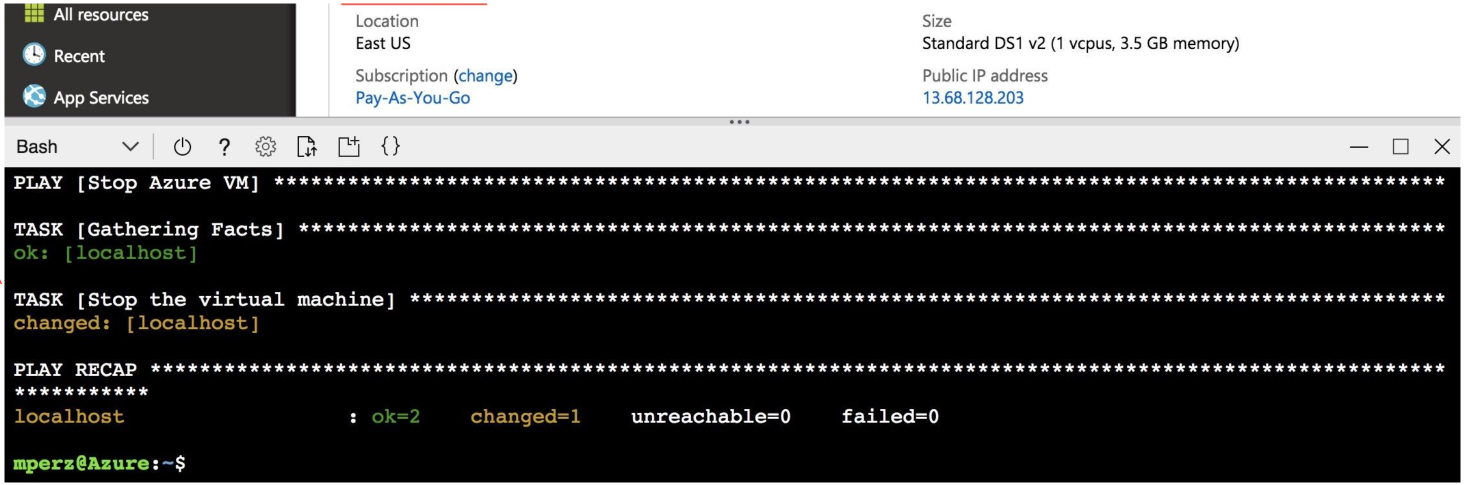
```
[default]
subscription_id=<your-subscription_id>
client_id=<security-principal-appid>
secret=<security-principal-password>
tenant=<security-principal-tenant>
```

Verify the Configuration

In CloudShell, create a file named `rg.yml`:

```
vi rg.yml
```

Paste the code found on the next slide into the editor, keeping in mind that the `name` variable underneath `azure_rm_resourcegroup` can be anything you want.



Location
East US

Subscription (change)
Pay-As-You-Go

Size
Standard DS1 v2 (1 vcpus, 3.5 GB memory)

Public IP address
13.68.128.203

```
PLAY [Stop Azure VM] ***********************************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [localhost]

TASK [Stop the virtual machine] ************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************
***********
localhost                  : ok=2    changed=1    unreachable=0    failed=0

mperz@Azure:~$
```

Verify the Configuration (cont.)

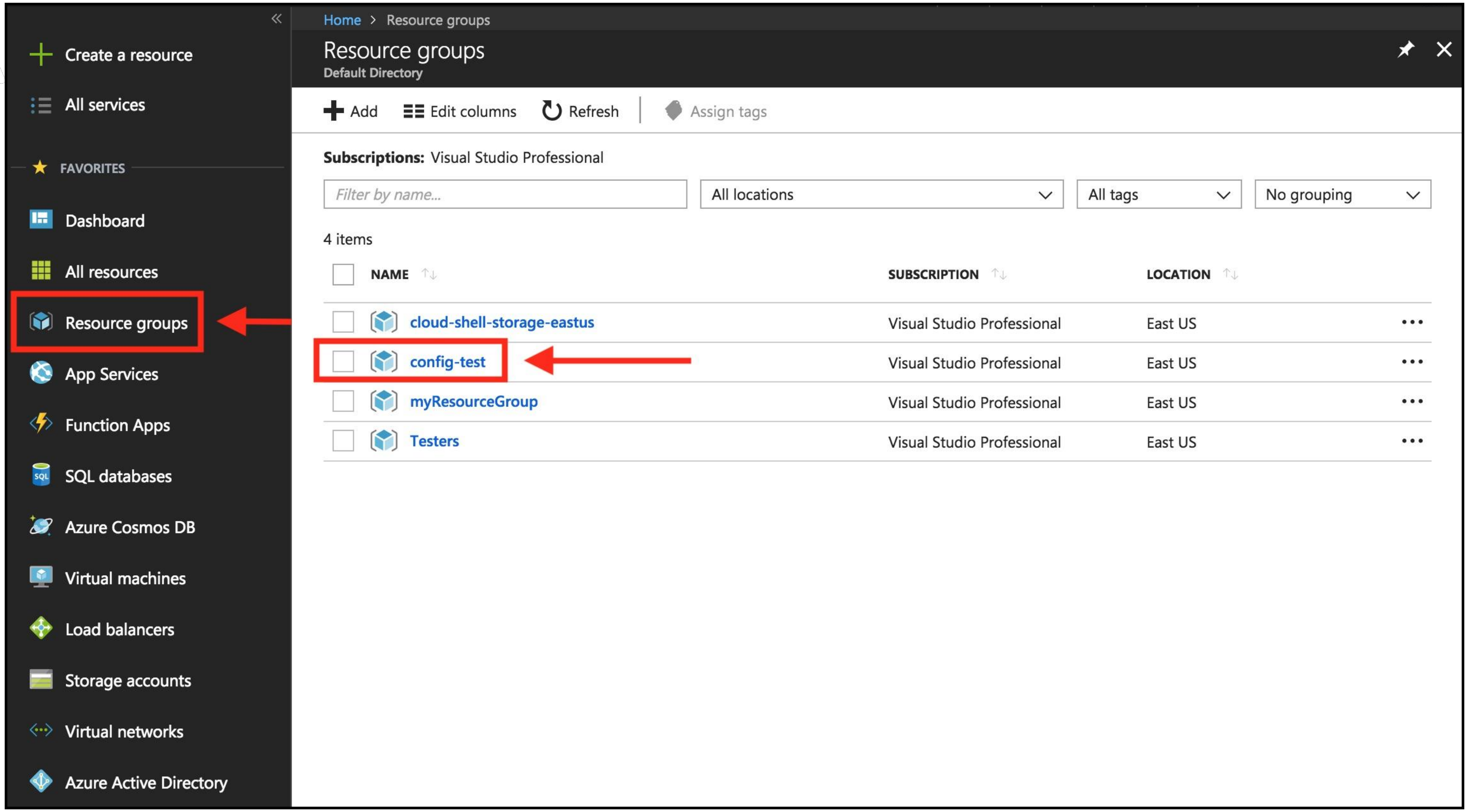
```
---
- hosts: localhost
  connection: local
  tasks:
    - name: Create resource group
      azure_rm_resourcegroup:
        name: config-test
        location: eastus
      register: rg
    - debug:
        var: rg
```

Verify the Configuration (cont.)

Run the playbook `rg.yml` with the following command:

```
ansible-playbook rg.yml
```

Navigate to the Resource Groups tab on the left side of the Azure user interface to see your newly created resource group!



Home > Resource groups

Resource groups
Default Directory

Subscriptions: Visual Studio Professional

Filter by name... | All locations | All tags | No grouping

4 items

NAME	SUBSCRIPTION	LOCATION
cloud-shell-storage-eastus	Visual Studio Professional	East US
config-test	Visual Studio Professional	East US
myResourceGroup	Visual Studio Professional	East US
Testers	Visual Studio Professional	East US

Create a Complete VM Environment in Azure

ANSIBLE

SSH Key

First, make sure to create an SSH key pair (if you don't have one already) by typing:

```
ssh-keygen
```

Copy the output from the following command:

```
cat ~/.ssh/id_rsa.pub
```

...into a text file so that you can paste it into the ssh_public_keys part of

```
azure_create_vm.yml
```

The Playbook

Create an Ansible playbook named `azure_create_vm.yml`

The following slides will show you the content that should be in that playbook and how it works.

Note: The text in **`red`** *indicate arbitrary names for things that you can change/customize.*

Create a Resource Group

```
---
- name: Create Azure VM
  hosts: localhost
  connection: local
  tasks:
    - name: Create resource group
      azure_rm_resourcegroup:
        name: webinar-test
        location: eastus
      register: rg
    - debug:
        var: rg
```

Create a Virtual Network

```
- name: Create virtual network
  azure_rm_virtualnetwork:
    resource_group: webinar-test
    name: webinarVnet
    address_prefixes: "10.0.0.0/16"
```

Add a Subnet to the Virtual Network

```
- name: Add subnet
  azure_rm_subnet:
    resource_group: webinar-test
    name: webinarSubnet
    address_prefix: "10.0.1.0/24"
    virtual_network: webinarVnet
```

Access Resources and Assign Public IP to the VM

```
- name: Create public IP address
  azure_rm_publicipaddress:
    resource_group: webinar-test
    allocation_method: Static
    name: myPublicIP
```

Create a Network Security Group

```
- name: Create Network Security Group that allows SSH
  azure_rm_securitygroup:
    resource_group: webinar-test
    name: webinarNetworkSecurityGroup
    rules:
      - name: SSH
        protocol: Tcp
        destination_port_range: 22
        access: Allow
        priority: 1001
        direction: Inbound
```

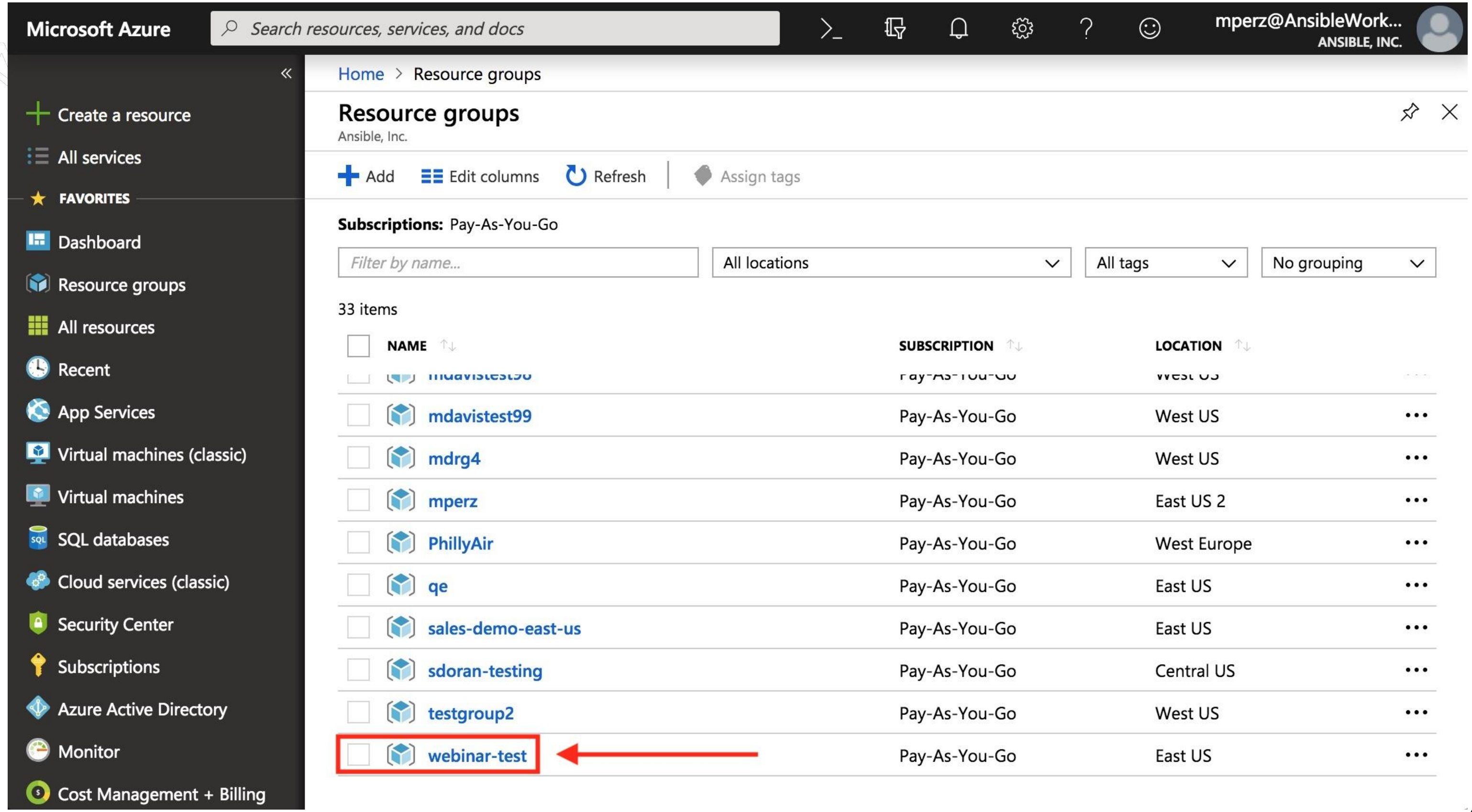
Create a Virtual Network Interface Card (NIC)

```
- name: Create virtual network interface card
  azure_rm_networkinterface:
    resource_group: webinar-test
    name: myNIC
    virtual_network: webinarVnet
    subnet: webinarSubnet
    public_ip_name: myPublicIP
    security_group: webinarNetworkSecurityGroup
```

Create the VM

```
- name: Create VM
  azure_rm_virtualmachine:
    resource_group: webinar-test
    name: WebinarVM
    vm_size: Standard_DS1_v2
    admin_username: azureuser
    ssh_password_enabled: false
    ssh_public_keys:
      - path: /home/azureuser/.ssh/authorized_keys
        key_data: " ssh-rsa AAAAB3Nz{snip}hwhqT9h "
    network_interfaces: myNIC
    image:
      offer: RHEL
      publisher: RedHat
      sku: '7-raw'
      version: latest
```

ANSIBLE



Microsoft Azure

Search resources, services, and docs

mperz@AnsibleWork...
ANSIBLE, INC.

Create a resource
All services
FAVORITES
Dashboard
Resource groups
All resources
Recent
App Services
Virtual machines (classic)
Virtual machines
SQL databases
Cloud services (classic)
Security Center
Subscriptions
Azure Active Directory
Monitor
Cost Management + Billing

Home > Resource groups

Resource groups
Ansible, Inc.

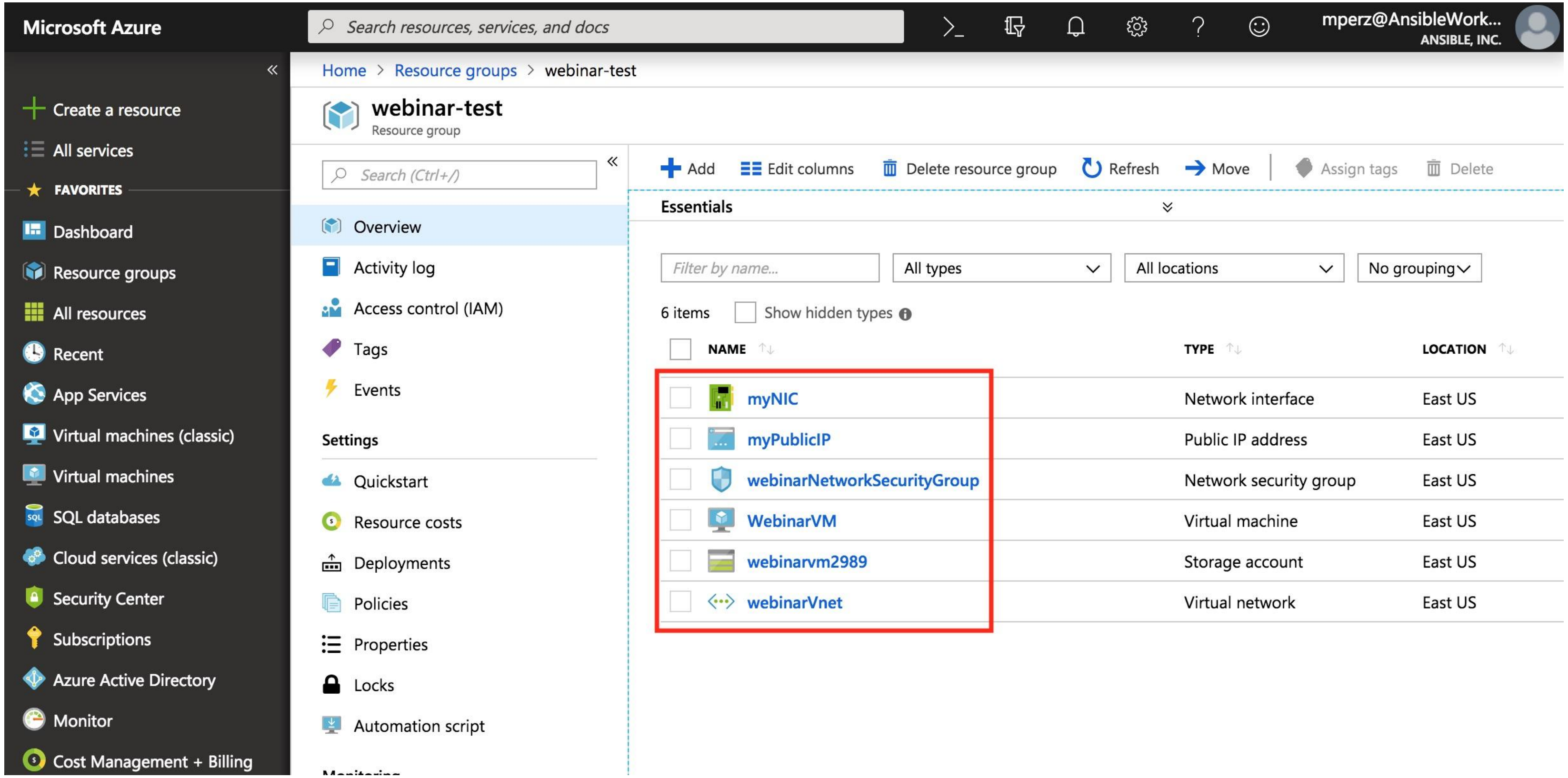
Add | Edit columns | Refresh | Assign tags

Subscriptions: Pay-As-You-Go

Filter by name... | All locations | All tags | No grouping

33 items

NAME	SUBSCRIPTION	LOCATION
mdavistest98	Pay-As-You-Go	West US
mdavistest99	Pay-As-You-Go	West US
mdrg4	Pay-As-You-Go	West US
mperz	Pay-As-You-Go	East US 2
PhillyAir	Pay-As-You-Go	West Europe
qe	Pay-As-You-Go	East US
sales-demo-east-us	Pay-As-You-Go	East US
sdoran-testing	Pay-As-You-Go	Central US
testgroup2	Pay-As-You-Go	West US
webinar-test	Pay-As-You-Go	East US

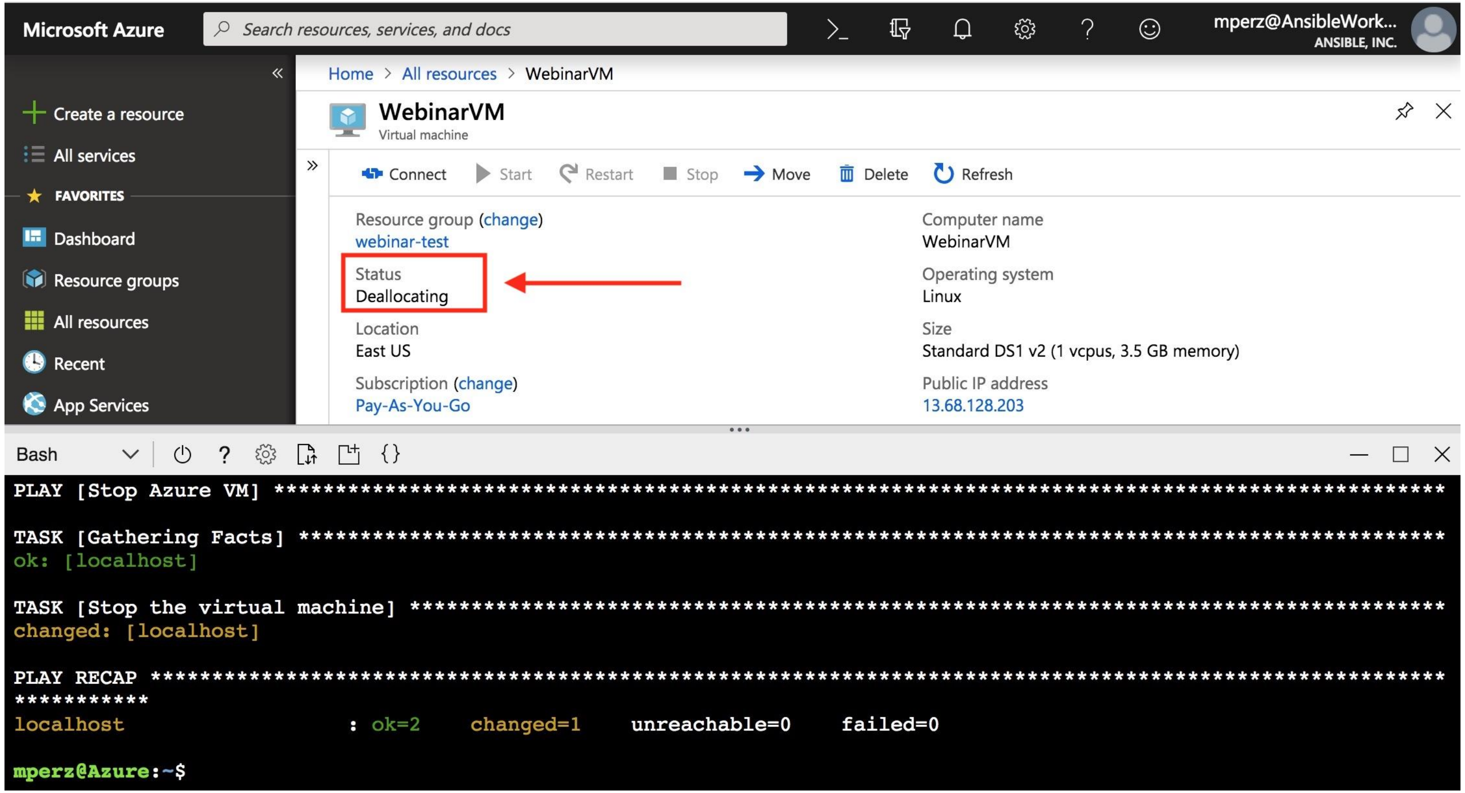


NAME	TYPE	LOCATION
myNIC	Network interface	East US
myPublicIP	Public IP address	East US
webinarNetworkSecurityGroup	Network security group	East US
WebinarVM	Virtual machine	East US
webinarvm2989	Storage account	East US
webinarVnet	Virtual network	East US

Manage VMs in Azure Using Ansible

Stop a VM Using a Playbook

```
---
- name: Stop Azure VM
  hosts: localhost
  connection: local

  tasks:
  - name: Stop the virtual machine
    azure_rm_virtualmachine:
      resource_group: webinar-test
      name: WebinarVM
      allocated: no
```



WebinarVM
Virtual machine

Resource group (change)
webinar-test

Status
Deallocating

Location
East US

Subscription (change)
Pay-As-You-Go

Computer name
WebinarVM

Operating system
Linux

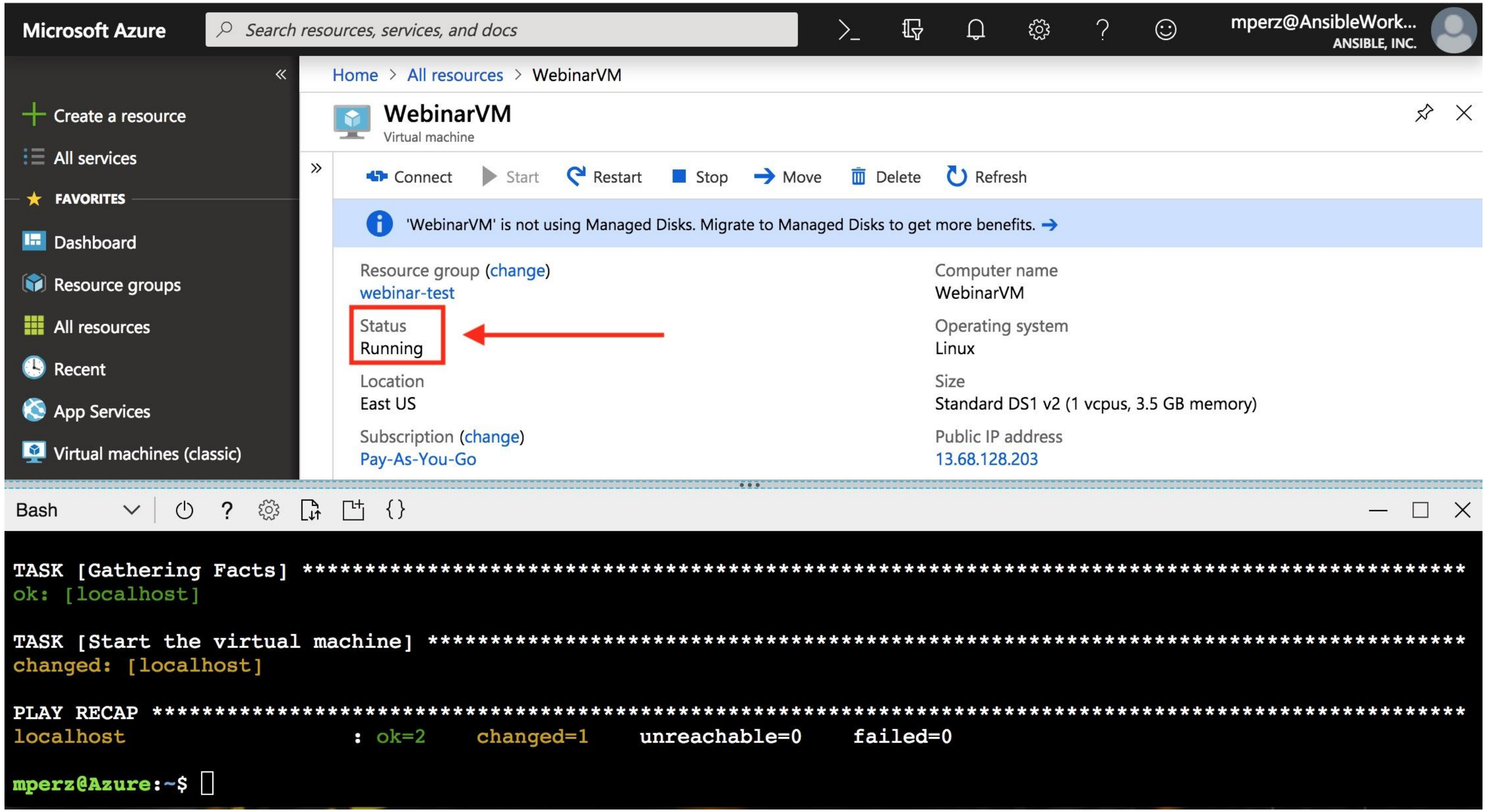
Size
Standard DS1 v2 (1 vcpus, 3.5 GB memory)

Public IP address
13.68.128.203

```
PLAY [Stop Azure VM] ***************************************************************************

TASK [Gathering Facts] *************************************************************************
ok: [localhost]

TASK [Stop the virtual machine] ****************************************************************
changed: [localhost]

PLAY RECAP *************************************************************************************
***********
localhost                  : ok=2    changed=1    unreachable=0    failed=0

mperz@Azure:~$
```

Start a Previously Stopped VM Using a Playbook

```
---
- name: Start Azure VM
  hosts: localhost
  connection: local

  tasks:
  - name: Start the virtual machine
    azure_rm_virtualmachine:
      resource_group: webinar-test
      name: WebinarVM
```



WebinarVM
Virtual machine

'WebinarVM' is not using Managed Disks. Migrate to Managed Disks to get more benefits. →

Resource group (change)	Computer name
webinar-test	WebinarVM
Status	Operating system
Running	Linux
Location	Size
East US	Standard DS1 v2 (1 vcpus, 3.5 GB memory)
Subscription (change)	Public IP address
Pay-As-You-Go	13.68.128.203

```
TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [Start the virtual machine] ***********************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0

mperz@Azure:~$ 
```

Helpful Resources

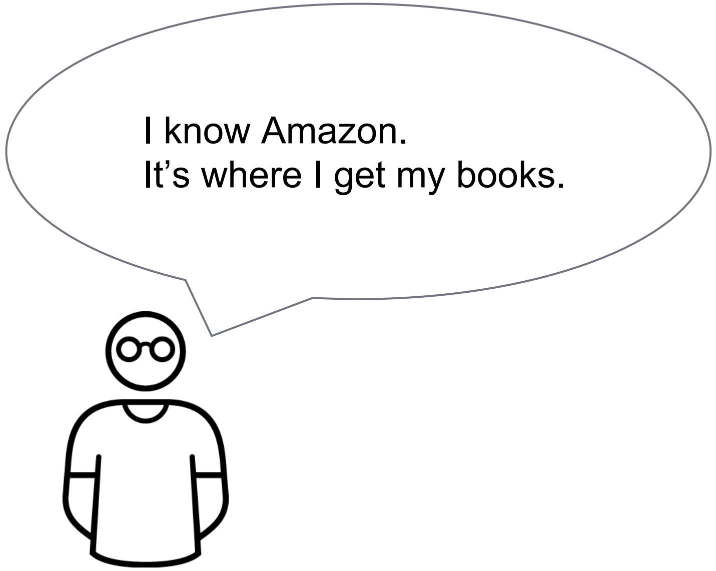
GitHub Repo (with instructions):
https://github.com/Ansible-Getting-Started/Provision-Azure

Ansible Docs:
https://docs.ansible.com/ansible/latest/scenario_guides/guide_azure.html

ANSIBLE

ANSIBLE

AWS Cloud Provisioning with Ansible



Requirements:

- AWS credentials (Access Key ID + Secret Access Key)
- Install AWS boto Python module: https://docs.ansible.com/ansible/latest/scenario_guides/guide_aws.html
- Ansible 2.6+ and git
- git clone https://github.com/mglantz/ansible-aws

Creating Credentials

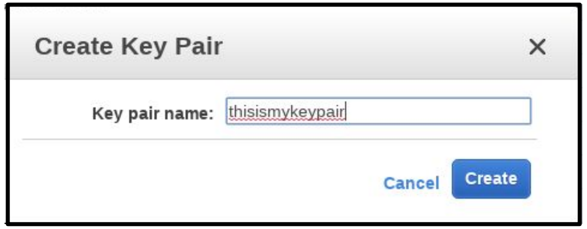
1. Log into your AWS account, go to your user in Identity and Access Management.
2. Navigate to Security Credentials. Click “Create Access Key. You should receive something like:

```
Access Key ID: PDMQMTIB1L1LGTFO2
Secret Access Key: 0SILWO5DSJ6IN8OJF8UZ3PQ2FKU
```

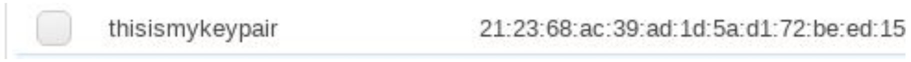
3. Copy `ansibe-aws/vars/vars-example.yml` to `ansible-aws/vars/vars.yml` and enter in the access key id and secret access key.

Creating Key Pair

1. Log into your AWS account, go to your user in EC2 Management Console.
2. Scroll to “Key Pairs” (grouped under Network & Security). Click “Create Access Key. You will be asked a name.
3. Download the key pair to the project directory and run `ssh-add ./name-of-key.pem`


Create Key Pair

Key pair name: thisismykeypair


thisismykeypair 21:23:68:ac:39:ad:1d:5a:d1:72:be:ed:15

Variables

Variables for the playbook

```
vars/vars.yml
---
ec2_access_key: the-access-key-id
ec2_secret_key: secret-key
ec2_key: name-of-your-key
ec2_region: eu-central-1 # AWS region
ec2_security_group_vms: arbitary-name-of-security-group
ami_id: ami-c86c3f23 # AMI ID for RHEL 7.5 in eu-central-1. Can be replaced with what you want.
number_of_systems: 1 # Number of systems to spin up
```

Creating the security group (deploy-server.yml)

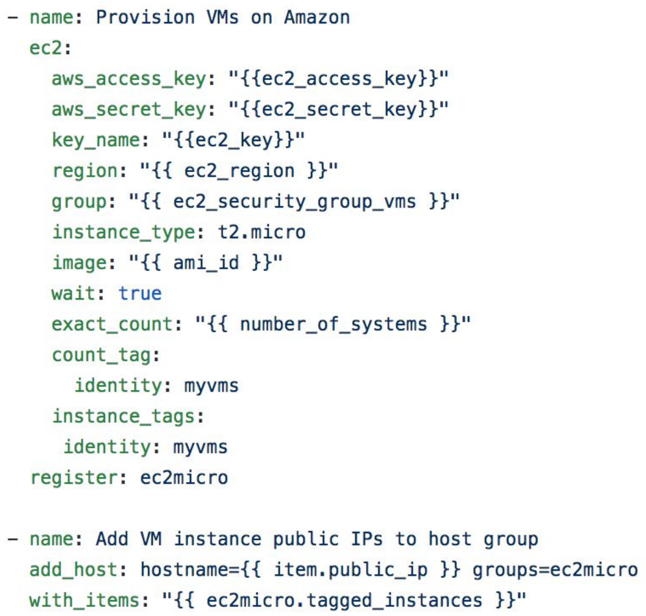
```
- name: "[Play 1] Create VMs in Amazon EC2"
  hosts: localhost
  connection: local
  gather_facts: False
  vars_files:
    - vars/vars.yml
  tasks:
  - name: Ensure a security group for VMs servers is in place
    ec2_group:
      name: "{{ ec2_security_group_vms }}"
      description: Security Group for my VMs servers
      region: "{{ ec2_region }}"
      aws_access_key: "{{ec2_access_key}}"
      aws_secret_key: "{{ec2_secret_key}}"
      rules:
        - proto: tcp
          from_port: 22
          to_port: 22
          cidr_ip: 0.0.0.0/0
      rules_egress:
        - proto: all
          cidr_ip: 0.0.0.0/0
```

Creates incoming and outgoing security rules

- ➢ Uses key pair we created previously
- ➢ Uses the variables out of the vars/vars.yml file
- ➢ We only allow incoming SSH traffic and allow all outgoing traffic
- ➢ Review the parameters on the module index:

https://docs.ansible.com/ansible/latest/modules/ec2_module.html

Creating VMs (deploy-server.yml)



```
- name: Provision VMs on Amazon
  ec2:
    aws_access_key: "{{ec2_access_key}}"
    aws_secret_key: "{{ec2_secret_key}}"
    key_name: "{{ec2_key}}"
    region: "{{ ec2_region }}"
    group: "{{ ec2_security_group_vms }}"
    instance_type: t2.micro
    image: "{{ ami_id }}"
    wait: true
    exact_count: "{{ number_of_systems }}"
    count_tag:
      identity: myvms
    instance_tags:
     identity: myvms
  register: ec2micro

- name: Add VM instance public IPs to host group
  add_host: hostname={{ item.public_ip }} groups=ec2micro
  with_items: "{{ ec2micro.tagged_instances }}"
```

Creates the virtual machine

- ➢ Uses key pair we created previously
- ➢ Uses the variables out of the vars/vars.yml file
- ➢ The instance here has t2.micro specified (it is the free tier level).
- ➢ We store data from the VM creation, things such as IP in ec2micro, then use that to add all public IPs to an in-memory inventory for usage in further plays
- ➢ Review the parameters on the module index:

https://docs.ansible.com/ansible/latest/modules/ec2_module.html

Print debug and wait (deploy-server.yml)

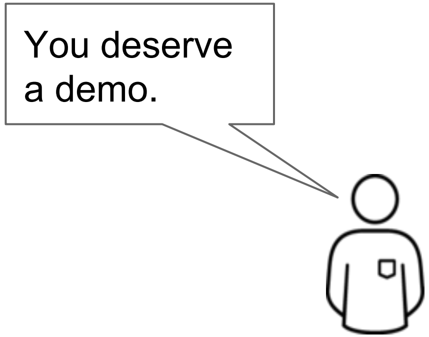
```
- name: Print IP address of VMs
  debug:
    msg: "{{ groups['ec2micro'] }}"

- name: Wait for SSH to come up
  wait_for:
    host: "{{ item.public_ip }}"
    port: 22
    delay: 30
    timeout: 120
    state: started
  with_items: "{{ ec2micro.tagged_instances }}"
```

Print IPs of all VMs created and wait until they boot up

➢ Loops through all created VMs and waits for them to be reachable via SSH
➢ After this, you may add your own play and use the previously created in-memory inventory or the dynamic inventory available in inventory/ec2.py

Run the playbook (deploy-server.yml)

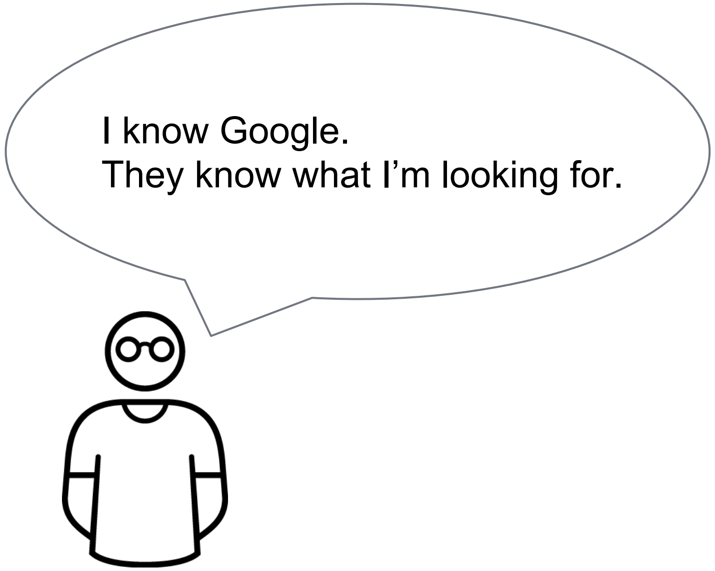
```
$ cd ansible-aws
$ ansible-playbook -i inventory/ec2.py -u ec2-user deploy-server.yml
```


You deserve a demo.

Links

Ansible AWS demo: https://github.com/mglantz/ansible-aws

Google Cloud Provisioning with Ansible



Requirements

- The Google Cloud Platform (GCP) modules require both the requests and the google-auth libraries to be installed. (Can be installed via pip)

- Credentials (Service account or machine accounts)

- Ansible v 2.6 (for the particular modules I will talk about)

Credentials

- Service Account (JSON) or Machine Account
- JSON Credentials Recommended

Creating JSON Credentials

- Open the Cloud Platform Console Credentials page.
- If it's not already selected, select the project that you're creating credentials for.
- To set up a new service account, click New credentials and then select Service account key.
- Choose the service account to use for the key.
- Choose whether to download the service account's public/private key as a standard P12 file, or as a JSON file that can be loaded by a Google API client library.

Using the Credentials with Ansible

- Specifying them directly as module parameters
- Setting environment variables

Module Parameters

```
vars:
    service_account_file: /home/my_account.json
    project: my-project
    auth_kind: serviceaccount
    scopes:
      - www.googleapis.com/auth/compute
```

Setting them as Environment Variables

```
GCP_AUTH_KIND
GCP_SERVICE_ACCOUNT_EMAIL
GCP_SERVICE_ACCOUNT_FILE
GCP_SCOPES
```

The Good Stuff... Creating Instances

- New Modules under the naming scheme “**gcp_***”
- Using the new GCP modules found in 2.6.x

The Top Level

```
- name: Create an instance
  hosts: localhost
  gather_facts: no
  connection: local
  vars:
      project: my-project
      auth_kind: serviceaccount
      service_account_file: /home/my_account.json
      zone: "us-central1-a"
      region: "us-central1"
```

Creating the Disk

```
tasks:
   - name: create a disk
     gcp_compute_disk:
         name: 'disk-instance'
         size_gb: 50
         source_image:
'projects/ubuntu-os-cloud/global/images/family/ubuntu-1604-lts'
         zone: "{{ zone }}"
         project: "{{ gcp_project }}"
         auth_kind: "{{ gcp_cred_kind }}"
         service_account_file: "{{ gcp_cred_file }}"
         scopes:
           - https://www.googleapis.com/auth/compute
         state: present
    register: disk
```

Creating the Network

```
- name: create a network
    gcp_compute_network:
        name: 'network-instance'
        project: "{{ gcp_project }}"
        auth_kind: "{{ gcp_cred_kind }}"
        service_account_file: "{{ gcp_cred_file }}"
        scopes:
          - https://www.googleapis.com/auth/compute
        state: present
    register: network
```

Creating an Address

```
- name: create a address
    gcp_compute_address:
        name: 'address-instance'
        region: "{{ region }}"
        project: "{{ gcp_project }}"
        auth_kind: "{{ gcp_cred_kind }}"
        service_account_file: "{{ gcp_cred_file }}"
        scopes:
          - https://www.googleapis.com/auth/compute
        state: present
    register: address
```

Creating the VM

```
- name: create a instance
  gcp_compute_instance:
      state: present
      name: test-vm
      machine_type: n1-standard-1
      disks:
        - auto_delete: true
          boot: true
          source: "{{ disk }}"
      network_interfaces:
          - network: "{{ network }}"
            access_configs:
              - name: 'External NAT'
                nat_ip: "{{ address }}"
                type: 'ONE_TO_ONE_NAT'
      zone: "{{ zone }}"
      project: "{{ gcp_project }}"
      auth_kind: "{{ gcp_cred_kind }}"
      service_account_file: "{{ gcp_cred_file }}"
      scopes:
        - https://www.googleapis.com/auth/compute
  register: instance
```

Helpful Links

https://docs.ansible.com/ansible/latest/scenario_guides/guide_gce.html

https://docs.ansible.com/ansible/latest/modules/list_of_cloud_modules.html#google

https://support.google.com/cloud/answer/6158849?hl=en&ref_topic=6262490#serviceaccounts

Overview of getting things done in Cloud

1. There are dependencies, such as a CLI or library which implements the cloud API
2. You need to fetch credentials to an account beforehand, try to limit the access of the account due to security concerns
3. Playbooks required to create assets in cloud are simple :-)