Hybrid Operations with Ansible

Ansible Engine & Tower for Cloud & On-Prem

Ryan Scott Brown
Senior Software Engineer
May 2018

#rhsummit #ansible

AGENDA

Automate the world

- About Ansible
- AWS, RHV, and GCP provider support
- Demo: provisioning callbacks
- AWS best practices
- Multi-provider planning and automation
- Ansible best practices
- Demo: Application Load Balancing (time permitting)

ANSIBLE ENGINE

- YAML-driven orchestration tool
- 1600+ modules covering Linux, Windows, and network device hosts
- Simple, powerful language for automating work

ANSIBLE TOWER

- Web UI on top of Engine
- Role-based access controls
- Scheduled jobs
- Inventory management

AUTOMATION NEVER SLEEPS

On any platform

- Automating dull work reduces risk
- IT pros don’t want to repeat the same tasks over and over
- Handle more projects, more safely with automated deployments
- Ansible is a force multiplier on any team

APIS SUPERCHARGE INFRASTRUCTURE

- Take full advantage of provider flexibility
- Amazon Web Services and Google Cloud Platform provide per-minute VM billing
- New instances can start in 60 seconds
- APIs let you gather information and send commands

HYBRID

RHV ADDS AN API FOR ON-PREM

- Existing datacenters
- Colo/IaaS deployments
- DR locations

The Dalles, Oregon. Google Data Center.
Photo: Google/Connie Zhou.

SENSIBLE HYBRID CLOUD

Mix of on-prem, colo, and *aaS

- Insurance policy for provider-specific downtime, pricing, or regionality
- Splitting individual workloads is often more difficult than moving the whole workload between providers

DATA HEAVY APPS

Don’t let a single point-of-presence form a data black hole

- Transfer costs
- WAN/leased line speeds
- Site-to-site encryption
- Daily transfer volume (GB/day or GB/hour)

AUTOMATING HYBRID CLOUD

Mix of on-prem, colo, and *aaS

- Host-layer automation can be shared between clouds
- Use playbooks/roles to smooth provider differences

HYBRID != HOMOGENOUS

Take advantage of best-of-breed services everywhere

- Different apps have different requirements
- Providers each have strengths and weaknesses

HYBRID PRINCIPLES

Provider-specific awareness

- Prefer open platforms like OpenShift and Kubernetes
- Prefer open operations tools like Ansible
- Provider APIs build into applications are a tradeoff
    - Velocity vs. portability
- Testing in multiple clouds pays dividends
    - Encourages good practices for tooling and automation
    - Avoids surprise cloud-specific features later in the process

ANSIBLE AND PROVIDERS

INVENTORY

Finding nouns to verb

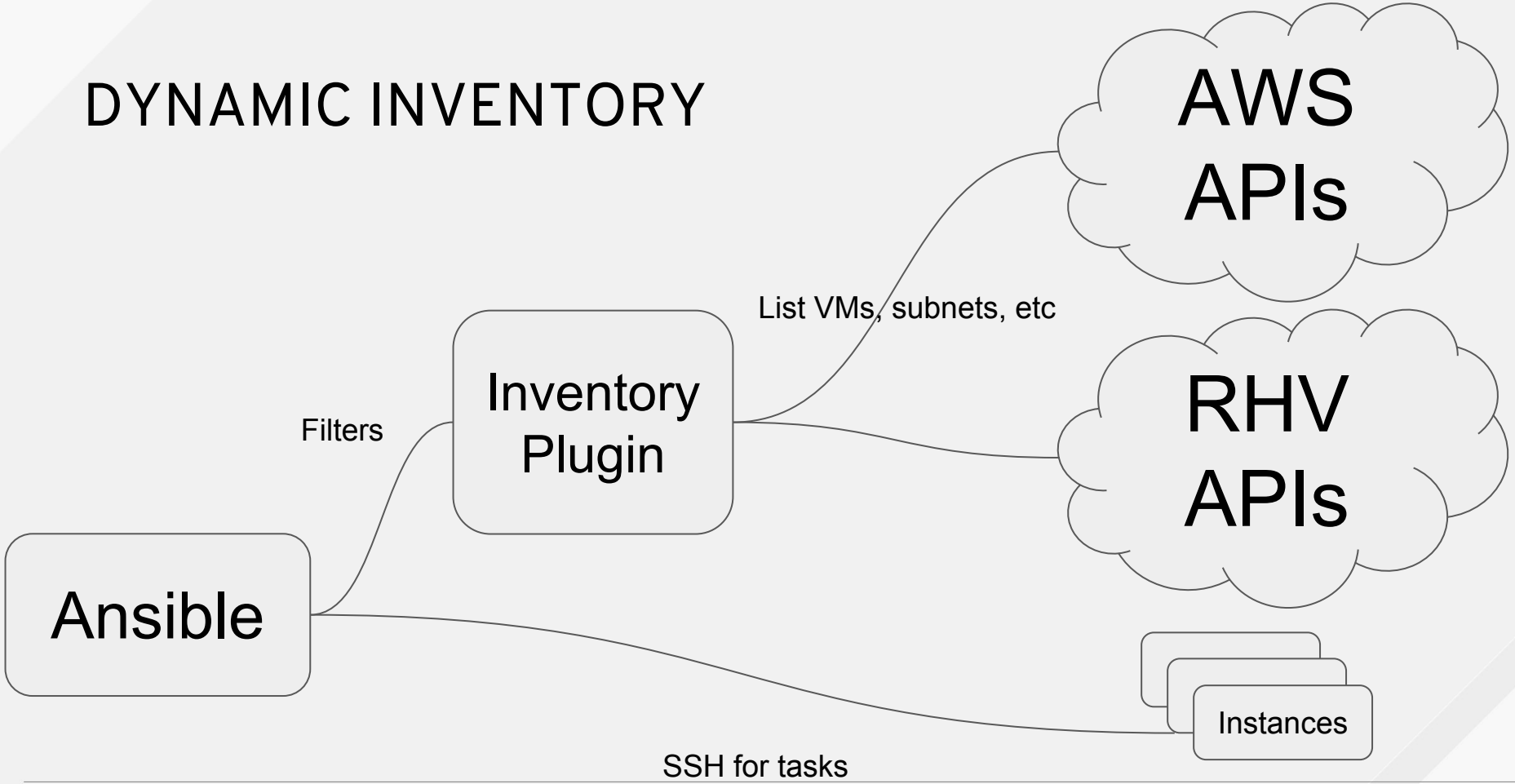
- APIs aren't just for provisioning
- Plugins for common cloud providers ship with Ansible
- For non-host resources use `_facts` modules

INVENTORY SOURCES

Supported Dynamic Inventories

- AWS
- Azure
- Google Cloud
- OpenStack
- RHV/oVirt
- VMWare
- Custom scripts

DYNAMIC INVENTORY



Configuring Hostnames

- Precedence-based names
- Tower connects via VPN if available
- Group by tag, host type, and much more
- Meaningful names are lifesavers

```yaml
hostnames:
  - tag:DNS
  - dns-name
  - private-dns-name

keyed_groups:
  - prefix: arch
    key: architecture
  - prefix: zone
    key: placement.availability_zone
  - prefix: instance_type
    key: instance_type
  - prefix: tag
    key: tags
```

Dynamic Features

- Refresh as you provision
- Complex grouping
- Add “serial: 30%” to do 1/3rd of hosts at a time (great for ASG’s)

```
1 groups:
2   # simple name matching
3   webservers: inventory_hostname.startswith('web-')
4   # match on attributes existing (or not)
5   in_vpc: public_dns_name is undefined
```

All Together Now

- Add configs for multi-account
- Aggregate hosts with multiple inventories
- Cache groups that change infrequently

```yaml
plugin: aws_ec2
boto_profile: summit-18
# Cache hosts on whatever lifetime you like
cache: no
regions:
  - us-east-2
  - us-west-2
hostnames:
  ...
groups:
  ...
keyed_groups:
  ...
```

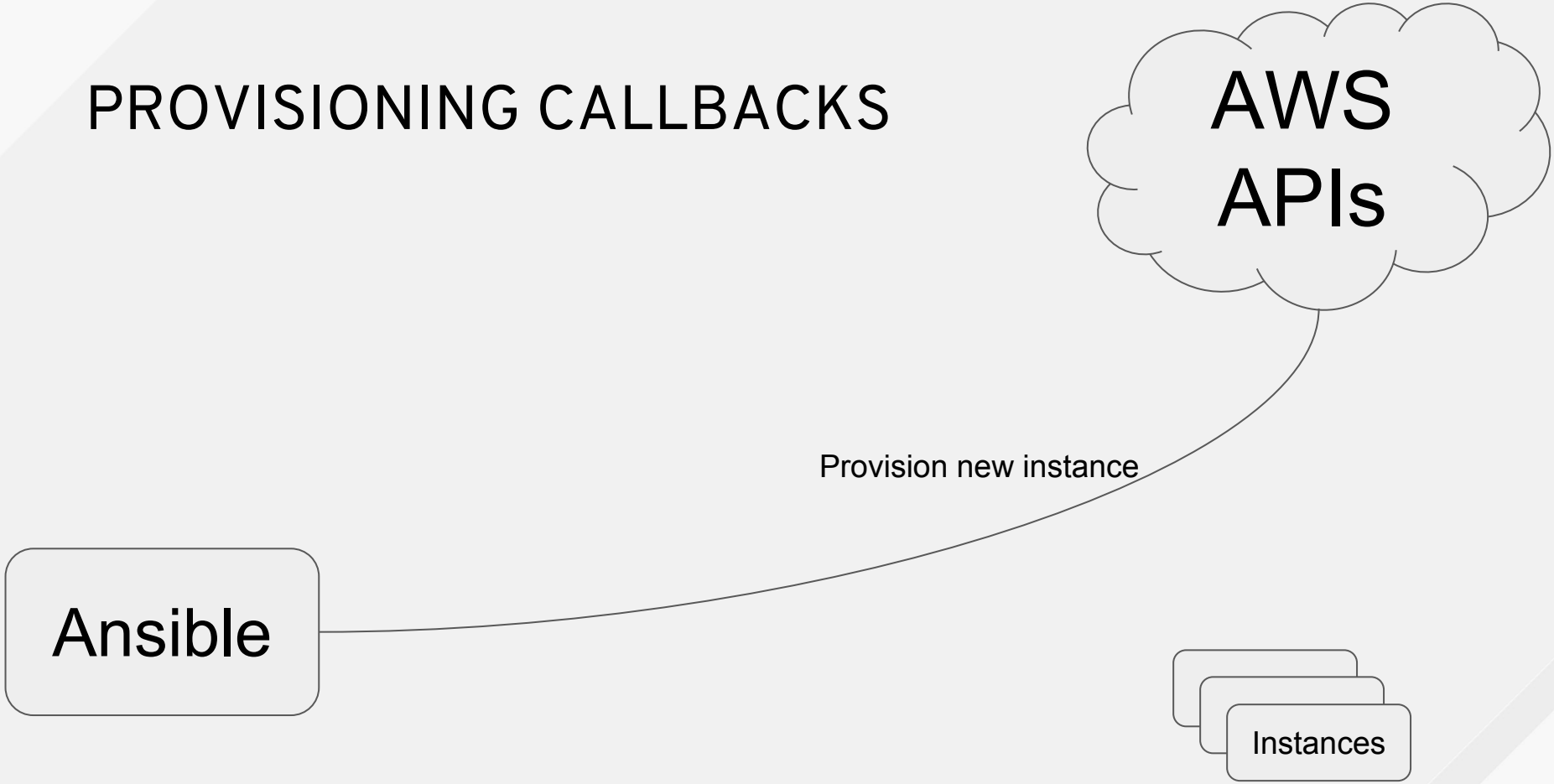
Refresh Dynamically

- Dynamic inventories can be refreshed anytime
- Use with “wait_for_connection” and “delegate_to” to get on the new host immediately
- Tower can refresh hosts on a schedule or every time a job is run

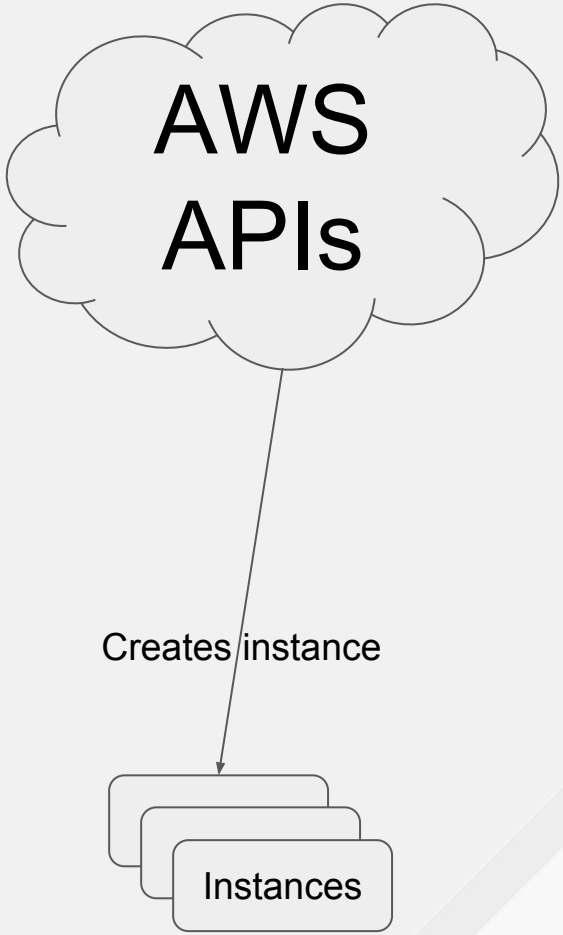
```yaml
- name: Create a few new instances
  ec2_instance:
    name: "{{ item }}"
    vpc_subnet_id: ...
    ....
  with_items:
    - test1-server
    - test2-server
    - test3-server
- meta: refresh_inventory
```

DEMO BREAK: CALLBACKS

PROVISIONING CALLBACKS

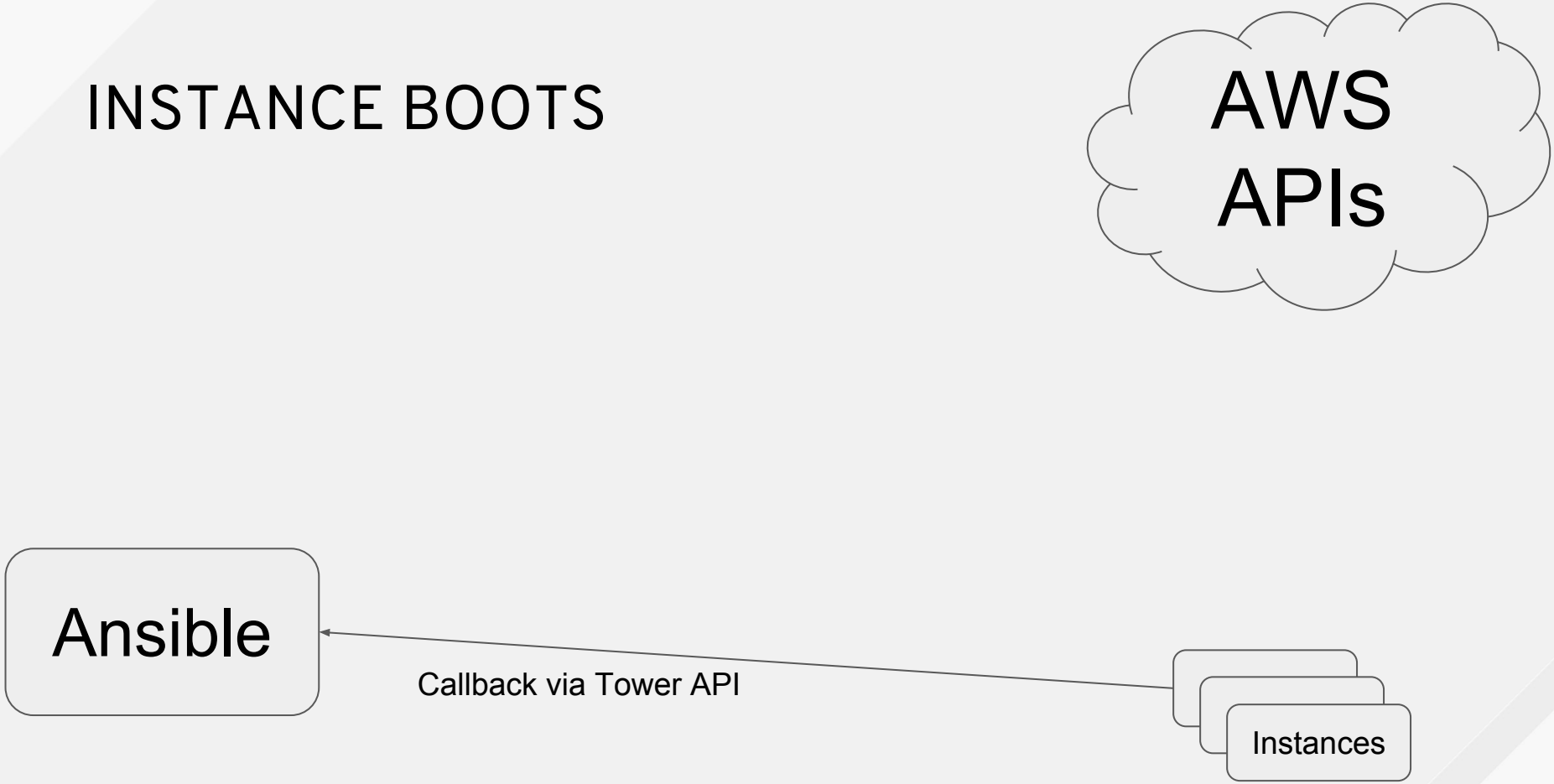


NEW INSTANCE CREATED

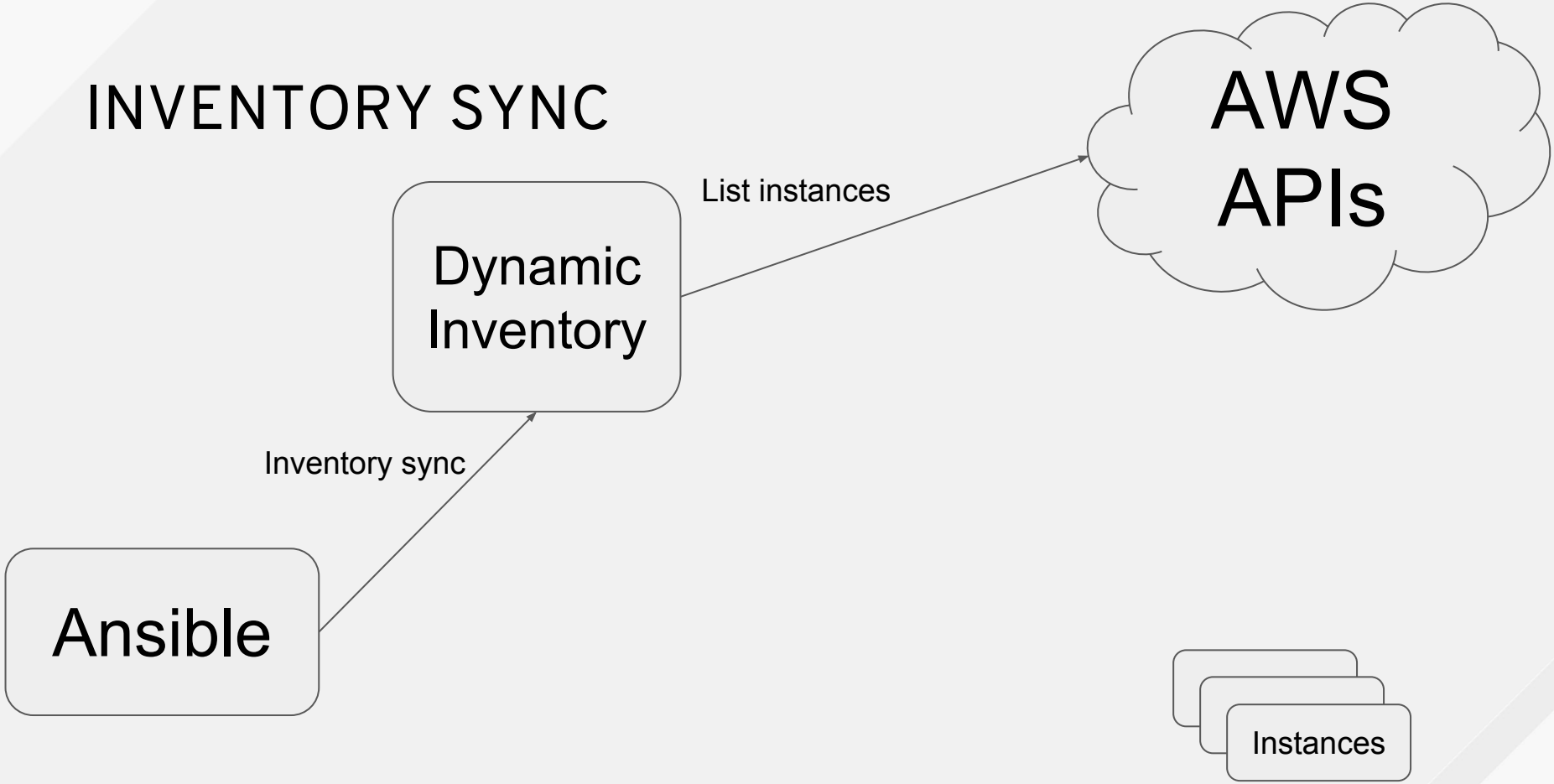


Ansible

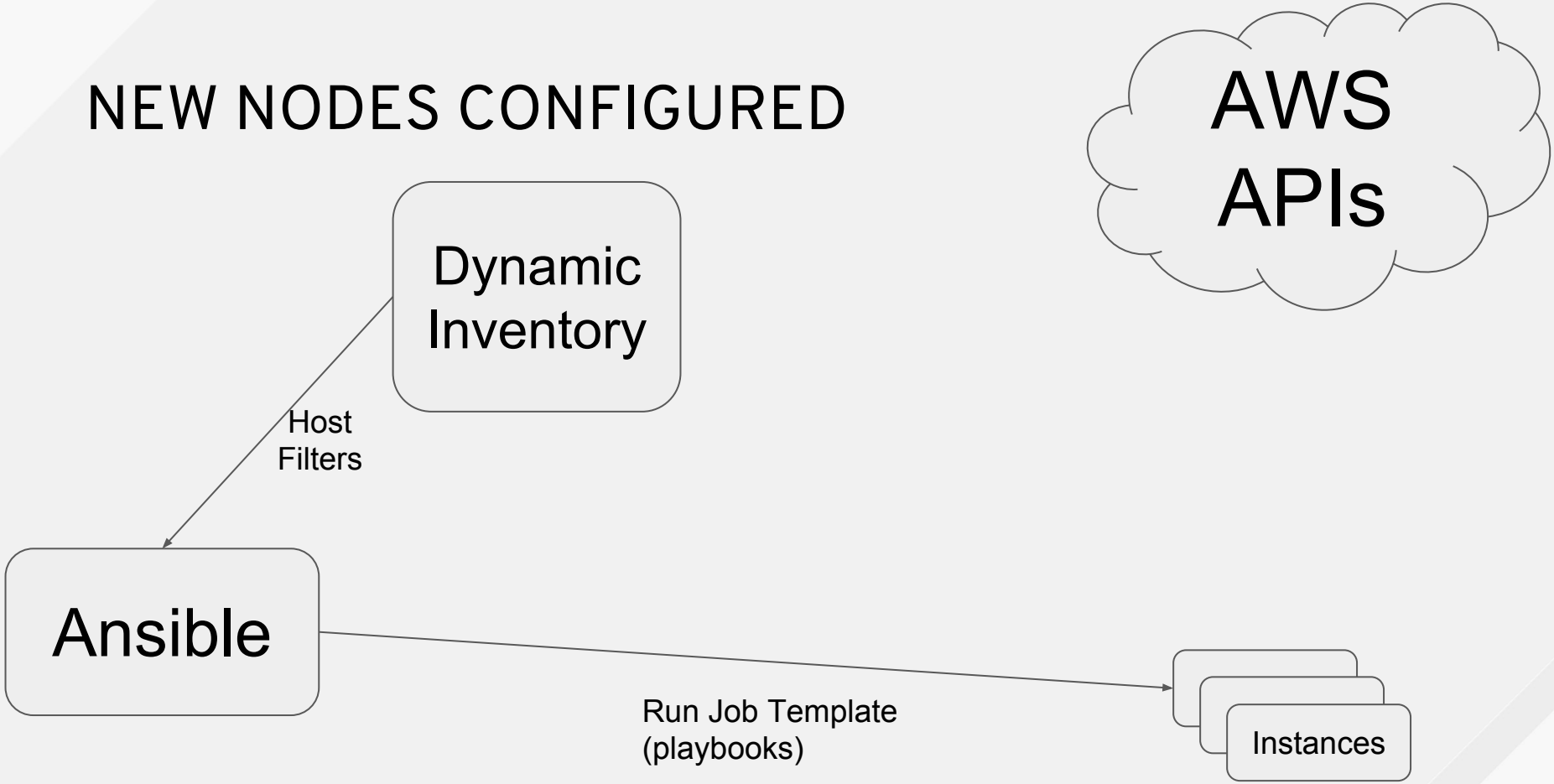
INSTANCE BOOTS



INVENTORY SYNC



NEW NODES CONFIGURED



INVENTORIES / AWS presence / SOURCES / INVENTORY SOURCES

AWS us-east-2

DETAILS | NOTIFICATIONS

* NAME: AWS us-east-2

DESCRIPTION: AWS presence in us-east-2

* SOURCE: Amazon EC2

SOURCE DETAILS

CREDENTIAL

REGIONS: US East (Ohio)

INSTANCE FILTERS

ONLY GROUP BY: Availability Zone, Region, Tags, VPC ID

VERBOSITY: 1 (INFO)

UPDATE OPTIONS

- [ ] Overwrite
- [ ] Overwrite Variables
- [x] Update on Launch

CACHE TIMEOUT (SECONDS): 0

15	configure-web-server	Playbook Run	4/24/2018 10:34:54 AM
16	Demo Project	SCM Update	4/24/2018 10:34:45 AM
14	AWS presence	Inventory Sync	4/24/2018 10:34:40 AM

configure-web-server

DETAILS | PERMISSIONS | NOTIFICATIONS | COMPLETED JOBS | ADD SURVEY

* NAME
configure-web-server

DESCRIPTION

* JOB TYPE — PROMPT ON LAUNCH
Run

* INVENTORY — PROMPT ON LAUNCH
AWS presence

* PROJECT
Demo Project

* PLAYBOOK
hello_world.yml

* CREDENTIAL — PROMPT ON LAUNCH
MACHINE: Instance Keys

FORKS
DEFAULT

LIMIT — PROMPT ON LAUNCH

* VERBOSITY — PROMPT ON LAUNCH
0 (Normal)

INSTANCE GROUPS

JOB TAGS — PROMPT ON LAUNCH

SKIP TAGS — PROMPT ON LAUNCH

LABELS

SHOW CHANGES — PROMPT ON LAUNCH
ON

OPTIONS
- [x] Enable Privilege Escalation
- [x] Allow Provisioning Callbacks
- [ ] Enable Concurrent Jobs
- [ ] Use Fact Cache

PROVISIONING CALLBACK URL
https://tower.glad.news:443/api/v2/job_templates/7/c

* HOST CONFIG KEY
5b5ba30417bd723d5e10e954047bf6b3

AWS PRACTICES

USE MARKETPLACE PRODUCTS

Cloud providers have pre-built appliances - do you use them when you can?

```
- ec2_ami_facts:
    owners: 900854079004
    filters:
      architecture: x86_64
      name: 'ansible-tower-*'
  register: amis
- set_fact:
    latest_image: >
      {{ amis.images | sort(attribute='creation_date') | last }}
```

USE MARKETPLACE PRODUCTS

Cloud providers have pre-built appliances - do you use them when you can?

```
- ec2_instance:
    image: {id: "{{ latest_image.image_id }}"}
    key_name: hornet_2018
    instance_type: m5.large
    name: tower-instance
    security_groups:
      - "{{ group.group_id }}"
    network:
      assign_public_ip: true
    vpc_subnet_id: subnet-4b36143d
```

BUT WHAT ABOUT KEYS

Access and privilege separation

- Ansible can operate with restricted permissions
  - Policies are specific to the cloud provider
  - Subject to the same limits as any user account in a cloud
- Tower encrypts credentials internally
- Can make use of instance roles to obviate keys entirely

AUDIT AND REVIEW

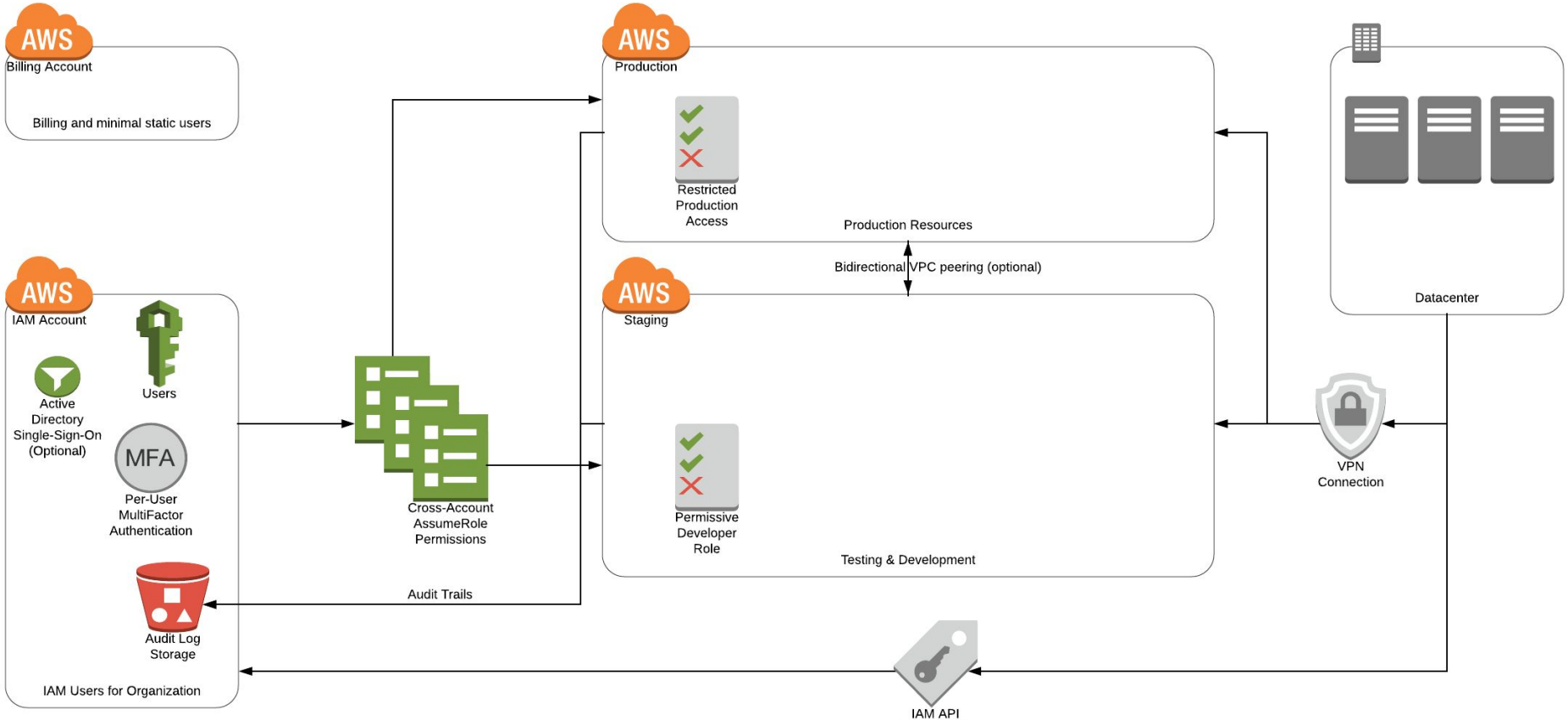
- Tower has its own logging and compliance support
- Cloud actions can be logged by the provider
- Special read-only Tower Auditing role

RESTRICTION OPTIONS

Implementing “least privilege” with Ansible Tower

- Restrict specific workflows/jobs
- Restrict access to hosts
- Isolate users to single SCM projects

MULTI ACCOUNT



ASSUMING ESCALATED PERMISSIONS

Or, sudo in the cloud

```yaml
- sts_assume_role:
    role_arn: 'arn:aws:iam::1234567890:role/superAdmin'
    session_name: ansible-escalated
    profile: base_creds
  register: assumed_role
- cloudfront_distribution:
    aws_access_key: "{{ assumed_role.sts_creds.access_key }}"
    aws_secret_key: "{{ assumed_role.sts_creds.secret_key }}"
    security_token: "{{ assumed_role.sts_creds.session_token }}"
    alias: foo.bar.com
    comment: My CDN distribution
```

ANSIBLE AND HYBRID CLOUD

OPENSTACK

On and off site

- Modules prefixed “os_*”
- Nova, Neutron, Keystone, Heat, and many more modules
- Dynamic inventory support

RHV AND OVIRT

Two faces of virtualization

- Modules are prefixed “ovirt_*”
- Supports network, host, disk, and firewall configuration
- Dynamic inventory support

VMWARE

60+ modules

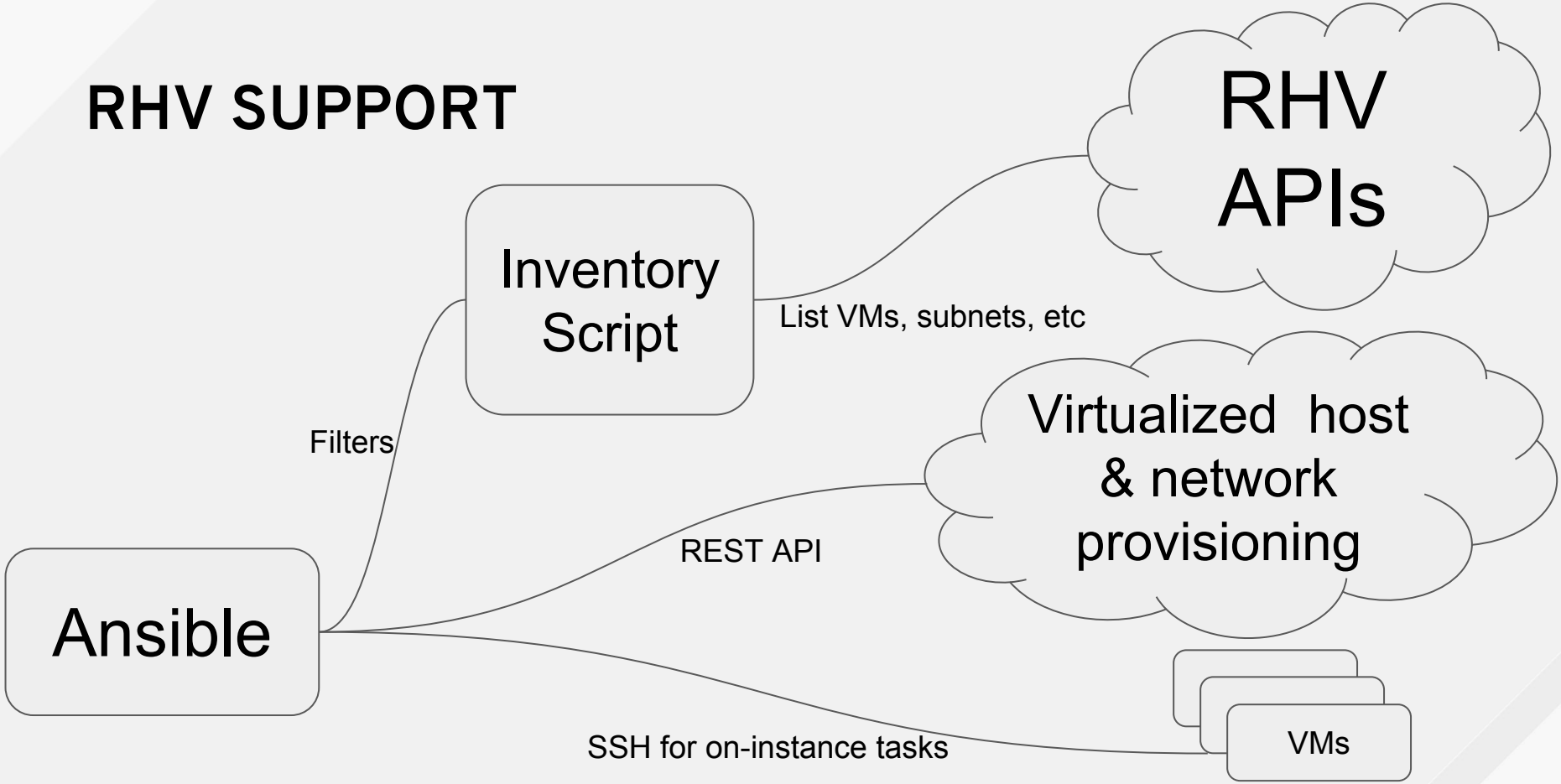
- Supports network, host, disk, and firewall configuration
- Dynamic inventory support

ANSIBLE AS AN APPLIANCE INSTALLER

For CloudForms

```yaml
- name: Hotplug {{ item }} disk for CFME
  ovirt_disk:
    auth: "{{ ovirt_auth }}"
    name: "{{ miq_vm_disks[item].name }}"
    vm_name: "{{ miq_vm_name }}"
    interface: "{{ miq_vm_disks[item].interface | default(omit) }}"
    size: "{{ miq_vm_disks[item].size | default(omit) }}"
    format: "{{ miq_vm_disks[item].format | default(omit) }}"
```

RHV SUPPORT



MANAGEIQ/CLOUDFORMS INSTALLATION

On RHV/oVirt

https://github.com/oVirt/ovirt-ansible-manageiq

REMEMBER THE TRIMMINGS

What's compute without storage and network?

- Red Hat Storage (Ceph)
- Virtual networking in RHV
- Arista, Cisco, VyOS, F5, Juniper hardware devices supported

ANSIBLE AND MULTI-CLOUD

THE BIG 3

Amazon, Azure, and Google

- Compute
- Networking
- Storage

GCP MODULES

- PEM and JSON token auth supported
- Kubernetes modules can run against GKE
- Dynamic inventory support
- 40+ services included

```yaml
- name: create managedzone for dns
  gcp_dns_managed_zone:
      state: present
      name: 'app-redhatsummit-2018'
      description: 'Mangaged Zone for RHS 2018 talk'
      dns_name: 'app1.mysite.rocks.'
      project: "{{ project }}"
  register: managedzone
- name: create resource record set
  gcp_dns_resource_record_set:
      state: present
      name: 'www.app1.mysite.rocks.'
      managed_zone: "{{ managedzone }}"
      type: 'A'
      ttl: 600
      target:
        - 1.2.3.4
```

AZURE MODULES

- Azure Resource Manager backs modules for consistent groupings
- Dynamic inventory support
- 40+ services included

```yaml
- name: Create security group that allows SSH
  azure_rm_securitygroup:
    resource_group: Testing
    name: secgroup001
    rules:
      - name: SSH
        protocol: Tcp
        destination_port_range: 22
        access: Allow
        priority: 101
        direction: Inbound

- name: Create NIC
  azure_rm_networkinterface:
    resource_group: Testing
    name: testnic001
    virtual_network: testvn001
    subnet: subnet001
    public_ip_name: publicip001
    security_group: secgroup001
```

ANSIBLE GOOD PRACTICE

PRACTICES

- Roles & directory structures
- Variables and tagging
- Think declaratively
- Using cloud APIs
- Dynamic inventories

http://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

KEEP IT SIMPLE, SYSADMINS

Strive to make your playbooks...

- Simple
- Readable
- Documented

SHARE PROD AND STAGE CONTENT

Sometimes, conditional love is what you need

```
- name: Set up a production-only service
  some_module:
    arg1: abc
  when: environment == “production”
```

ROLE STRUCTURE

```
mysite-automation/
  vars/
    ...yml
  playbooks/
    ci_deploy_webapp.yml
    roll_dep_updates.yml
  roles/
    myco.netsec/
      tasks/
        ...
```

SPLIT PROVIDER TASKS

```
# Create separate tasks for
# provision_gcp.yml and provision_aws.yml
- include: “provision_{{ provider }}.yml”
```

THE “I” WORD

- Modules aren’t always consistent
  - shell
  - command
- Check status of these resources **before** changing state
- Use **`changed_when`** to avoid extra “changed” counts when running plays
- Tower keeps track of changed/failed/ok tasks for every job

AWS APPLICATION LOAD BALANCER DEMO

GOT QUESTIONS? GET ANSWERS

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos

FAQ

Q: Will slides be available?
A: Yes, when talks are posted

Q: Are code samples public?
A: Yes, https://da.gd/hybrid-ops