

Automating IPAM In Cloud

Ansible + Netbox

William Collins

Lead Cloud Architect

What is *IPAM*?

IP Address Management (*IPAM*) is the critical component that organizes your IP Addresses and Networks in one place. Responsible management of IP Addressing drives efficient, repeatable, and reliable network automation.



IP Address

Single device on a network



Virtual Machine

Network Prefix

Aggregation of IP Addresses



Virtual Network

SuperNet

Multiple networks combined into a bigger network



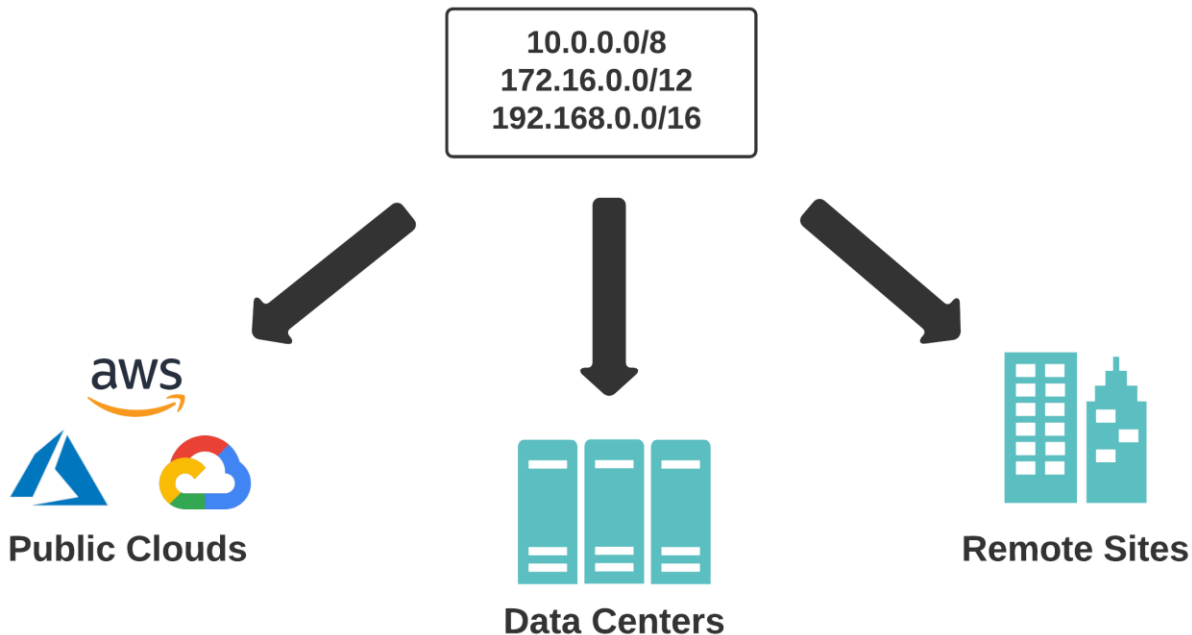
Cloud Region

The Problem

As Hybrid Multi-Cloud becomes reality, private IP space becomes shared across cloud(s) and on-premises

Developers leverage CI/CD as they deploy, migrate, and maintain applications in the cloud; network can't keep up

Some developers also do not have a good understanding of how IP Addressing works and how to responsibly consume it



The Problem (cont.)

1.) Data required for efficient automation is often dispersed across many tools and platforms



CMDB



Data



Inventory



Second IPAM



Third IPAM

3.) Tools and platforms may be owned and managed by different teams with different directions

2.) Oftentimes, overlap can exist between tools and platforms for a given domain of data

4.) Good automation is dependent on data accuracy, consistency, and ability to be consumed

What Do We Want?

Automated IP Address Management

- Minimal human intervention
- Repeatable with other cloud providers
- Compatible across traditional + cloud networking
- Reliable but flexible agility (Go fast but responsible)

Outcome = **Good User Experience**
for those that design, consume,
maintain, and support cloud
networking!



The Technology

Automating the network at scale means, automating across multiple vendors and environments

Tools



Ansible Tower

Acts as the central orchestration layer for network automation tasks



Netbox

Serves as a Source of Truth for desired network state; many use cases beyond IPAM

Environments



Colocation DCs

Provides fast on-ramps to cloud providers; contains bare metal network gear



Public Cloud

Cloud services owned and operated by a third-party and delivered over the internet



What Are We Automating?

7

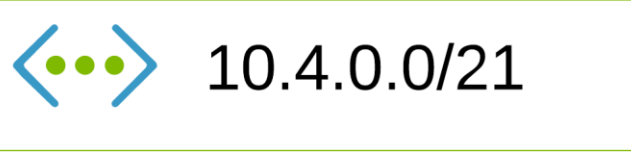
Cloud Environment

/16 Allocated for
Azure Region

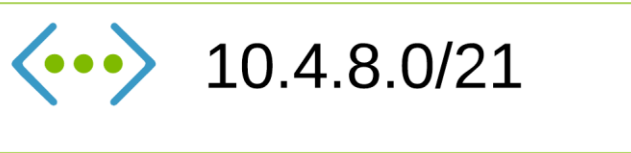


10.4.0.0/16
(65K - IPs)

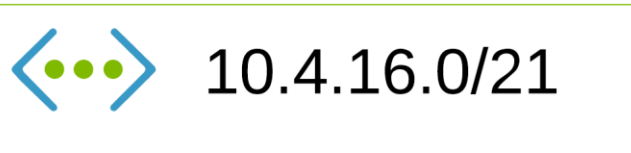
Shared Services



Analytics

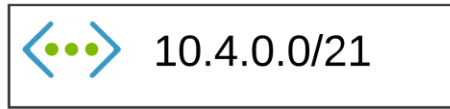


Finance

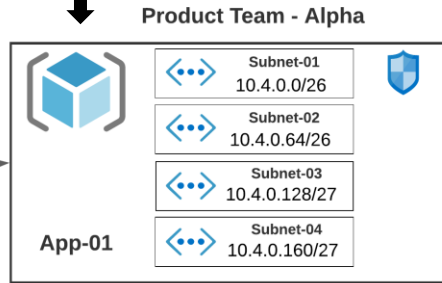


VNets may be shared
across multiple apps,
while being in scope
across many teams

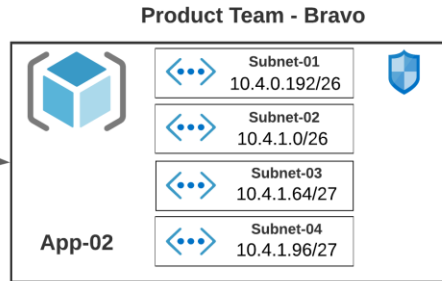
Cloud Environment (cont.)



Applications are separated by Resource Groups

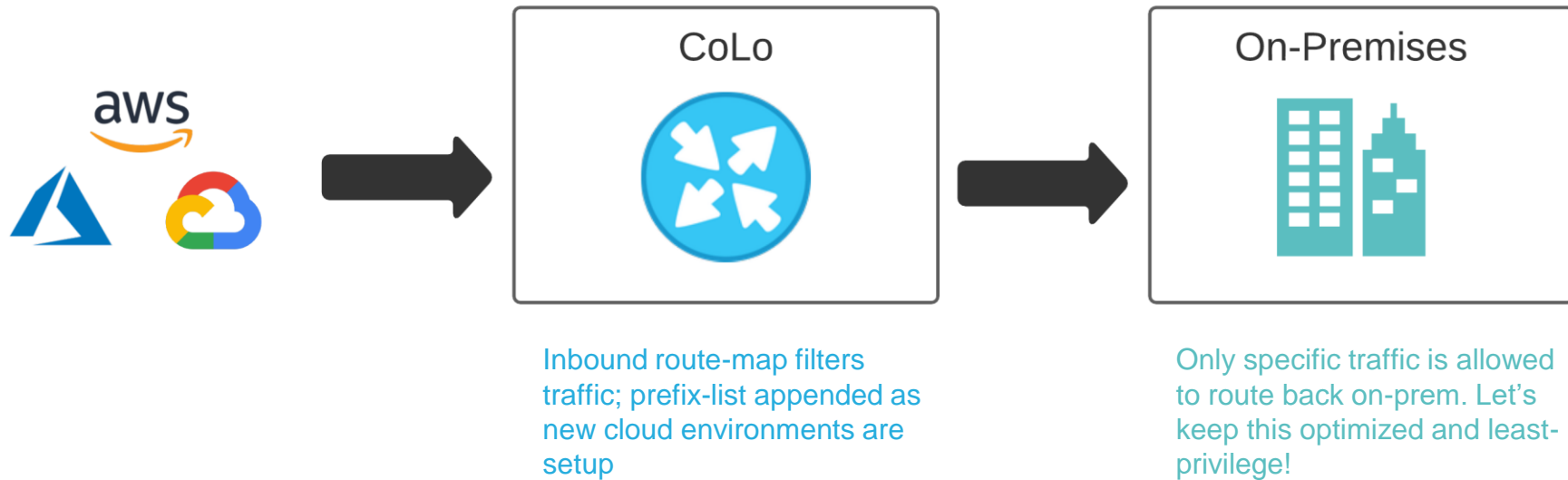


NSGs with standard policy defined



Subnets segment application tiers

CoLo Environment



Touchpoints

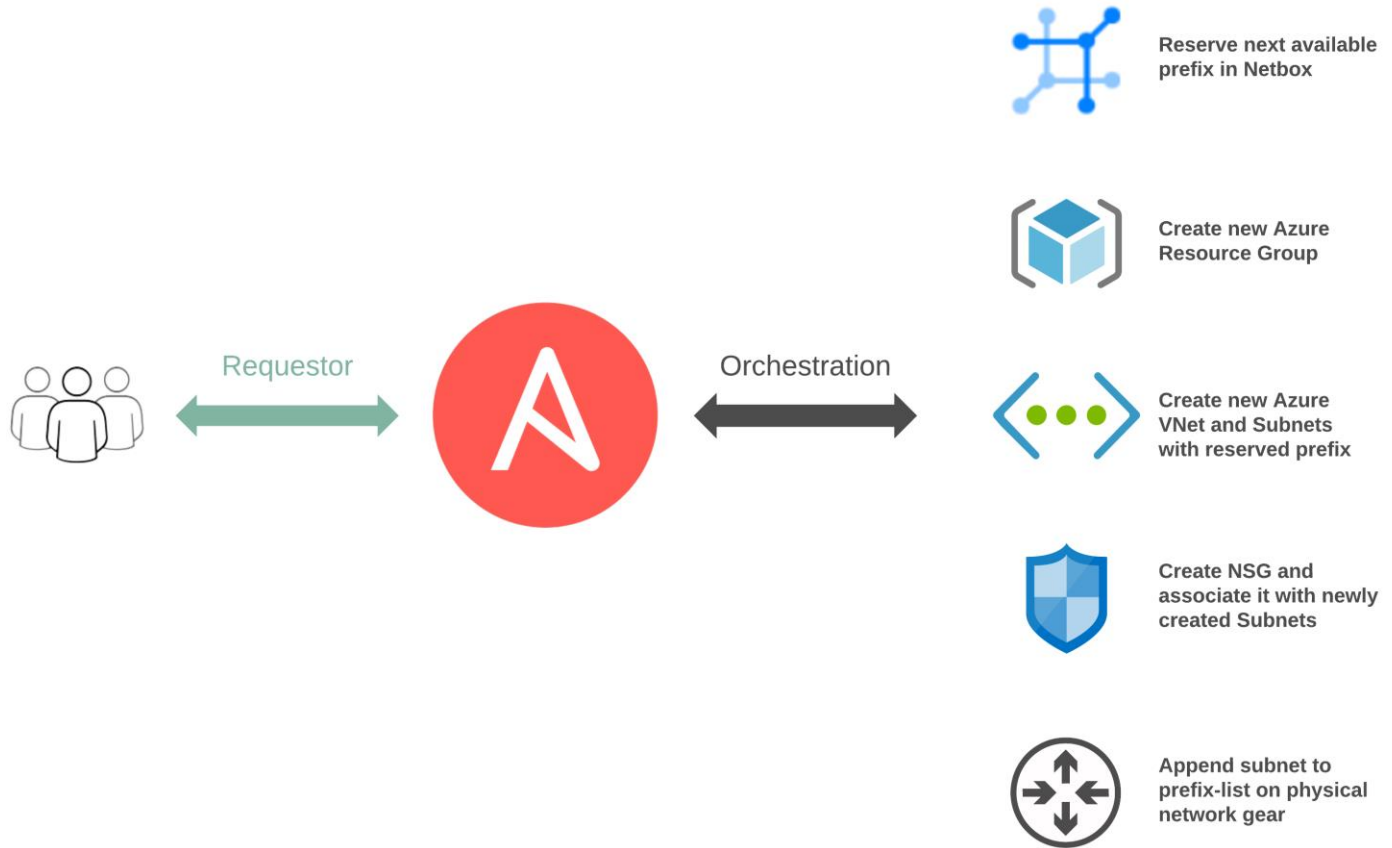
1.) Source Of Truth



2.) Cloud Infrastructure



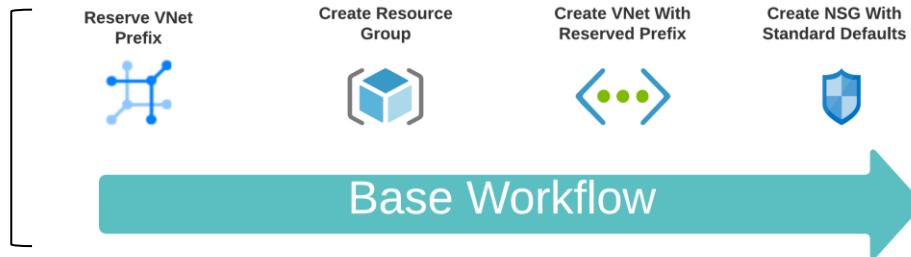
2.) Network Gear



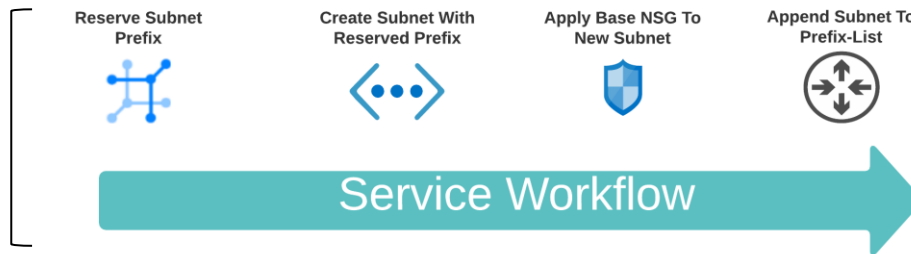
Design Drives Automation

Thinking through a given design is a key element for how you approach the automation. For this demo, we will leverage two different workflows for automating IPAM and building cloud infrastructure.

Builds *foundational* components that can be shared across business units, teams, applications, and tools.



Builds *service* oriented components specific to a given team, service, and application environment.



Tagging All The Things!

Applying tags to resources in cloud has a lot of benefits.

What if we tagged our IP reservations with the same tags that we use when we deploy new cloud infrastructure?



Now we can start treating our pets like cattle!

10.4.0.0/27

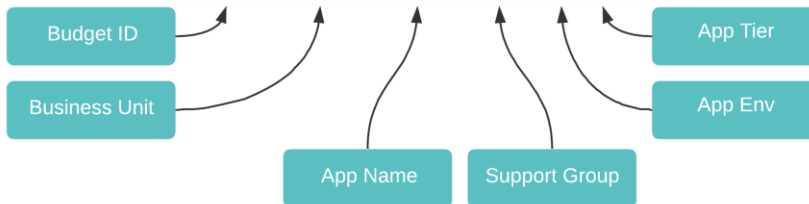
Created Sept. 11, 2020 · Updated 2 minutes ago

Prefix: [Child Prefixes](#) 0 [IP Addresses](#) 0 [Change Log](#)

Prefix	
Family	IPv4
VRF	Global
Tenant	Azure
Aggregate	10.0.0.0/8 (RFC1918)
Site	Southeast > EastUS2
VLAN	None
Status	Active
Role	Prod
Description	Created by Ansible
Is a pool	✗
Utilization	<div style="width: 0%;"></div> 0%

Tags

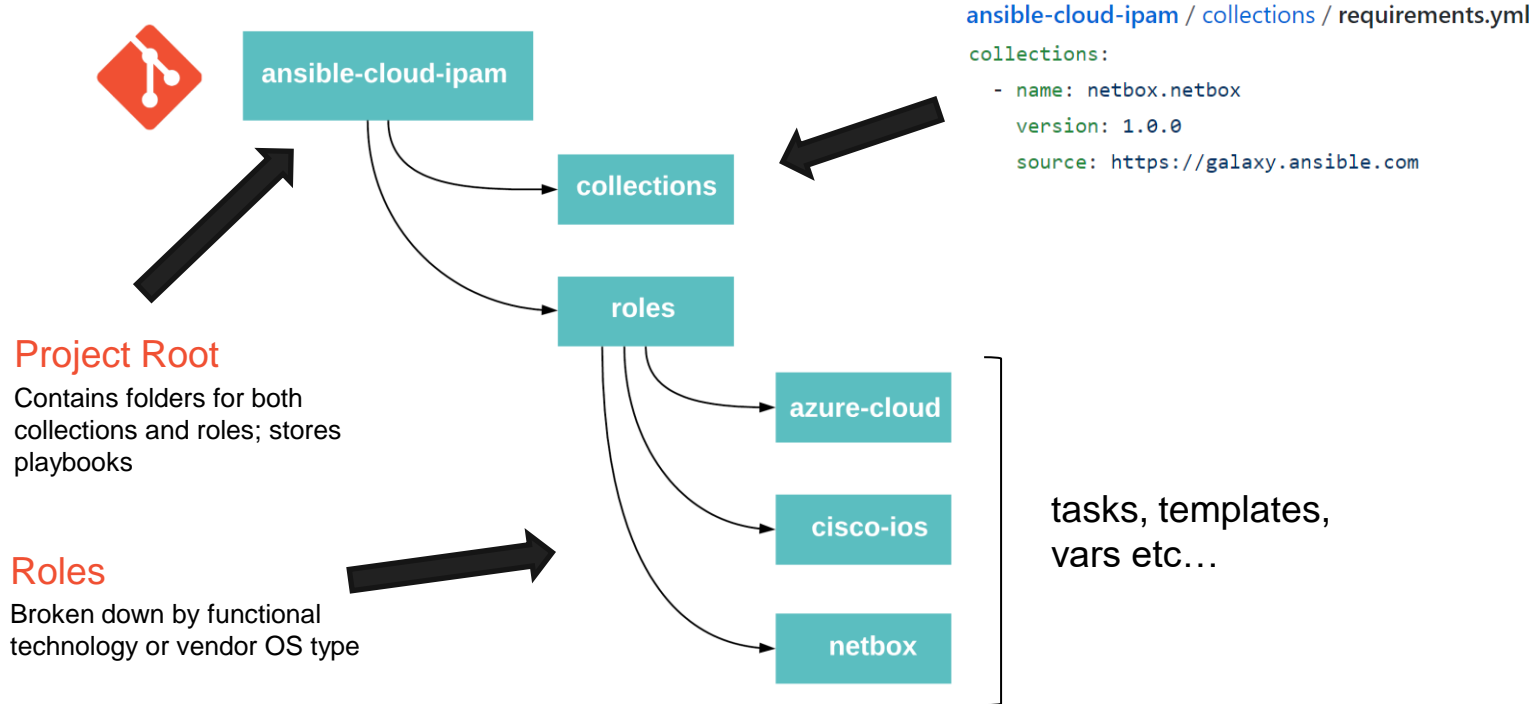
br123456 Shared Services FancyApp Cloud Ops Prod DB



Breaking Down The Logic

14

Project Structure



The Logic – High Level Plays

Playbooks sitting in our root project folder are simple. They are primarily used as the entry point for Ansible and execute based on a specific condition.

`ansible-cloud-ipam / play.netbox_reserve_prefix_snet.yml`

```
- hosts: all
  connection: local
  hosts: localhost
  gather_facts: False
  vars:
```

```
    play_action: netbox_reserve_prefix_snet
```

← Let's define our Play Action folks!

```
roles:
```

```
- netbox
```

← Include the netbox role

...



The Logic – Functional Roles

Roles are organized by either a given platform or OS type. Each role's main file will include specific tasks to use based on the condition defined in the root level play.

```
ansible-cloud-ipam / roles / netbox / tasks / main.yml
```

```
---
```

```
- include_tasks: task.netbox_reserve_prefix_vnet.yml  
  when: play_action == 'netbox_reserve_prefix_vnet'
```

```
- include_tasks: task.netbox_reserve_prefix_snet.yml  
  when: play_action == 'netbox_reserve_prefix_snet'
```



Only include this task when...
Play Action is exactly this!

```
...
```

The Logic – Purpose Built Tasks

The real logic is handled inside a given role's tasks. Each task is built to do a very specific thing. The idea is to keep things **DRY (Don't Repeat Yourself)** to maximize reuse.

In using Workflows with Ansible Tower, we generally want to perform some action, retrieve some data, and then provide as an artifact to be used in forthcoming tasks in the Workflow.

ansible-cloud-ipam / roles / netbox / tasks / task.netbox_reserve_prefix_snet.yml

```
- name: Add next prefix in Netbox
  netbox.netbox.netbox_prefix:
    netbox_url: "{{ netbox_url }}"
    netbox_token: "{{ netbox_token }}"
    data:

    # Define criteria for prefix
    parent: "{{ prefix_parent }}"
    prefix_length: "{{ prefix_length }}"

    # Define criteria for site
    tenant: "{{ tenant_name }}"
    site: "{{ azure_location }}"

    # Metadata
    prefix_role: "{{ prefix_role }}"
    description: "{{ prefix_desc }}"

    # Tags
    tags:
      - "{{ app_name }}"
      - "{{ app_tier }}"
      - "{{ app_env }}"
      - "{{ billing_id }}"
      - "{{ business_unit }}"
      - "{{ support_group }}"

    # State
    state: present
    first_available: yes

    # Register output
    register: first_available

    # Set stat for prefix string
  - name: Set stats
    set_stats:
      data:
        net_prefix: "{{ first_available.prefix.prefix }}"
```

Enforced Tagging!
Let's keep track of stuff



set_stats == AWESOME



The Logic – Physical Network

For the physical Cisco ASR, we will leverage Ansible's `network_cli` for communication and the `ios_config` module to push the configuration.

Each time the Service Workflow is run, each subnet reservation will be appended to a prefix-list living on the ASR.

The configuration is rendered with Jinja2 with the variables being passed in from Ansible Tower.



[ansible-cloud-ipam / roles / cisco-ios / templates / prefix_list_append.j2](#)

```
ip prefix-list {{ list_name }} {{ list_action }} {{ net_prefix }}
```



[ansible-cloud-ipam / roles / cisco-ios / tasks / task.ios_prefix_list_append.yml](#)

```
---

# Block - Append prefix-list on IOS
- name: Append prefix-list on IOS
  block:

    # Create temp staging directory
    - name: Stage config directory
      file:
        path: "staging/{{ inventory_hostname }}"
        state: directory
        mode: '0777'

    # Stage configuration for IOS prefix-list
    - name: Stage configuration for IOS prefix-list
      template:
        src: prefix_list_append.j2
        dest: "staging/{{ inventory_hostname }}/prefix_list_append.cfg"

    # Push configuration to IOS
    - name: Push configuration to IOS
      ios_config:
        src: "staging/{{ inventory_hostname }}/prefix_list_append.cfg"

    # Clean up temp staging directory
    - name: Clean staging directory
      file:
        path: "staging/{{ inventory_hostname }}"
        state: absent

...
```

Bringing It All Together

20

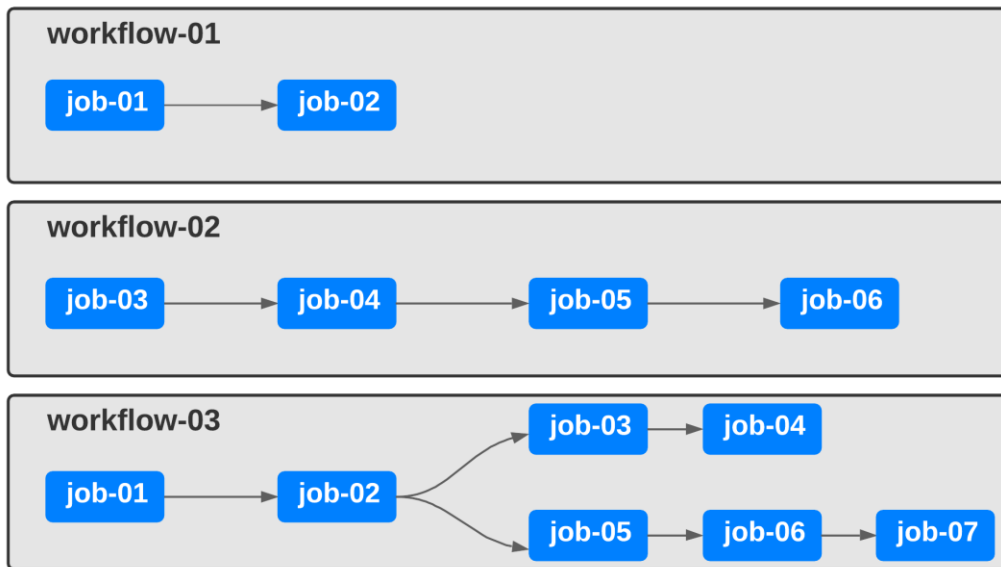
Jobs And Workflows

Workflows group our job templates together to enable different combinations / outcomes



Tower

Jobs run our high-level plays we have living in source control



Surveys And Data

WORKFLOW.BASE_CLOUD_ENV ✕

SURVEY

PREVIEW

* BUSINESS UNIT

Shared Services

* BILLING ID

br123456

* SUPPORT GROUP

Cloud Ops

* AZURE RESOURCE GROUP NAME

shared-services-rg

* AZURE VNET NAME

shared-services-vnet

* AZURE NSG NAME

shared-services-nsg

* VNET PREFIX LENGTH

20

CANCEL

NEXT

WORKFLOW.SERVICE_CLOUD_ENV ✕

SURVEY

PREVIEW

* BUSINESS UNIT

Shared Services

* BILLING ID

br123456

* SUPPORT GROUP

Cloud Ops

* APP NAME

FancyApp

* APP TIER

DB

* APP ENV

Prod

* AZURE SUBNET NAME

shared-services-snet

* PREFIX LENGTH

24

CANCEL

NEXT



Workflows In Action

The infrastructure's build + release pipeline!

DETAILS

STATUS ● Successful

STARTED 9/13/2020 1:56:59 PM

FINISHED 9/13/2020 2:01:56 PM

INVENTORY localhost

TEMPLATE workflow.base_cloud_env

LAUNCHED BY admin

EXTRA VARIABLES YAML JSON EXPAND

```
1 azure_location: eastus2
2 azure_nsg_name: shared-services-nsg
3 azure_rg_name: shared-services-rg
4 azure_vnet_name: shared-services-vnet
5 billing_id: br123456
6 business_unit: Shared Services
```



DETAILS

STATUS ● Successful

STARTED 9/13/2020 2:04:37 PM

FINISHED 9/13/2020 2:07:54 PM

TEMPLATE workflow.service_cloud_env

LAUNCHED BY admin

EXTRA VARIABLES YAML JSON EXPAND

```
1 app_env: Prod
2 app_name: FancyApp
3 app_tier: DB
4 azure_location: eastus2
5 azure_snet_name: shared-services-snet
6 billing_id: br123456
```





twitter.com/wcollins502



wcollins.github.io



linkedin.com/in/william-collins



github.com/wcollins/ansible-cloud-ipam



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



youtube.com/user/RedHatVideos



linkedin.com/company/Red-Hat



facebook.com/ansibleautomation



twitter.com/ansible