



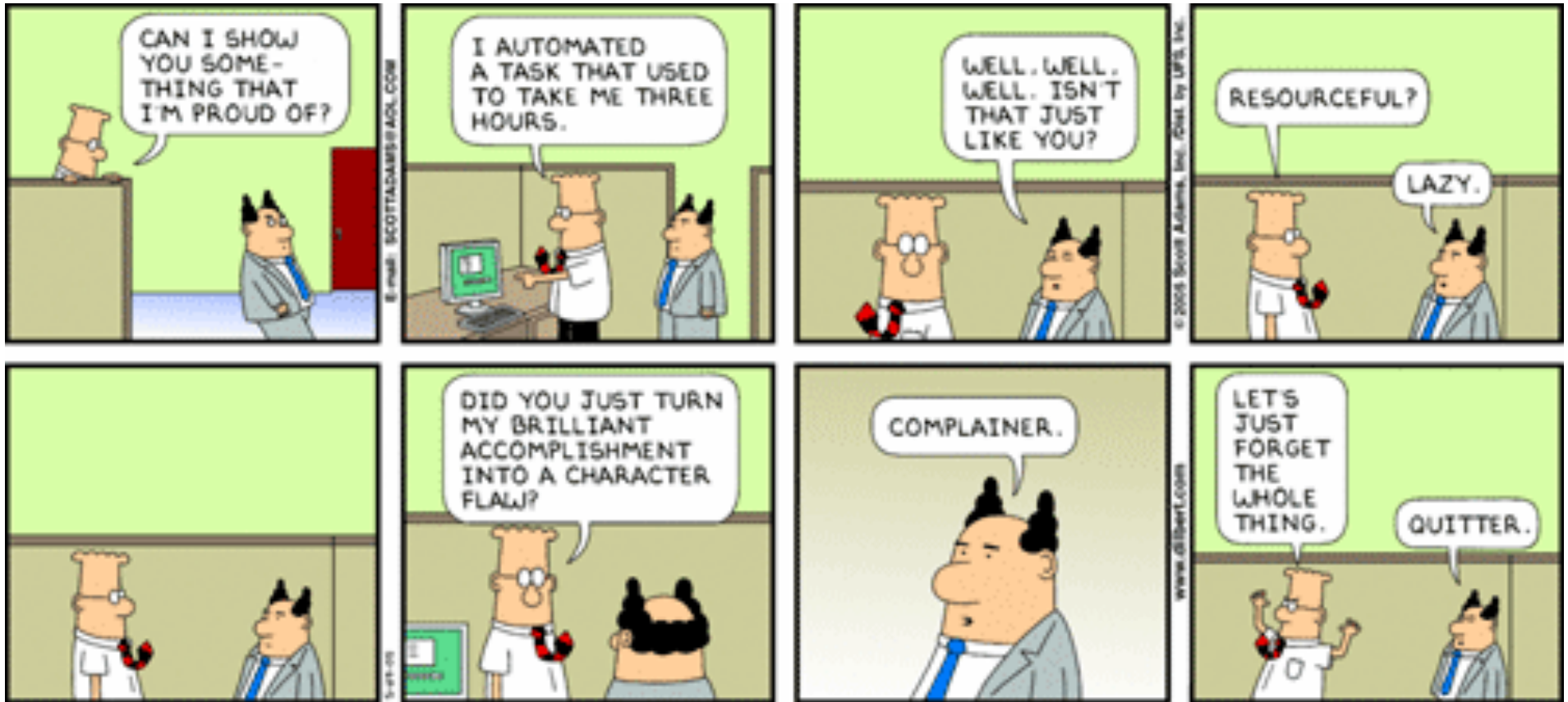
BIG-IP Automation Workflows in AWS with Ansible Tower

Nate Helfrey/Roy Shaver

nate.helfrey@siriuscom.com

roy.shaver@siriuscom.com


Dilbert Cartoon



Other Immersion Days



- Ansible Tower
 - Linux/Windows / Tower 101
- Network Automation
 - Basics / Ansible 101
- F5 Automation
 - Rescue logic, tower workflows / Ansible 102
- Cisco ACI Automation
 - Ansible 103
- Palo Alto Automation
 - Collections and Roles / Ansible 201

**AUTOMATION IMMERSION
DAY AGENDAS**

Summary

Sirius offers 4 separate automation immersion days that teach Ansible Engine and Ansible Tower topics to our customers. These Immersion days are each designed to last 4 hours, are short on presentation and heavy on hands on lab. We provide each student access to a jump station and their own virtual equipment in a shared virtual environment to complete the labs.

Below is a detailed agenda for each of the immersion days.

Ansible Tower Immersion Day
Agenda

- Provision an AWS EC2 instance
- Minimally configure the instance once it has been provisioned.
- Log into the instance once configuration is complete to take a look around.
- Clean up, de-provisioning the instance once the lab is complete.
- Take a quick tour of Ansible Tower.
- Set up a project and a dynamic inventory.
- Create job templates to provision a simple multi-tier web application.
- Clean up, deprovisioning the application once the lab is complete
- Create the wait_for_ssh auxiliary job template.
- Create a workflow template that combines the plays from the previous lab into a single, executable job run.

Network Automation Immersion Day
Agenda

- Getting Setup: Fork Git Repo, SSH to jump station, downloading forked Git Repo
- Explore the lab environment with shell and Ansible ad-hoc commands
- Investigate Ansible's configuration and inventory
- Connect to the Routers using Ansible
- Directory structure review
- Understand and use facts modules
- Run initial playbook to put basic configuration on Router
- Gather banner information and reconfigure the banner
- Backup router configuration
- Restore router configuration from files
- Configure DNS and loopback interface
- Create GRE tunnel and setup routing

- Ping another student router from loopback interface
- Access Ansible Tower
- Build a backup job with scheduling
- Interact with a job containing a Survey

Palo Alto Automation Immersion Day

- Getting setup: Fork Git Repo, SSH to jump station, Connecting to Palo FW
- Understanding Roles and Collections
- Explore the lab environment
- Gather data from Palo Alto Firewall
- Using Ansible Vault
- Initializing the Palo Alto Firewall
- Setting up Syslog
- Adding a simple security rule
- Adding multiple rules using a CSV
- Removing an address object from rules
- Saving/Restoring Configuration

F5 Automation Immersion Day

- Getting Setup: Fork Git Repo, SSH to jump station, Connecting to F5
- Explore the lab environment
- Gather data from F5
- Adding nodes to F5
- Adding a load balancing pool
- Adding members to a pool
- Adding a virtual server
- Adding and attaching an iRule to a virtual server
- Save the running configuration
- Disabling a pool member
- Deleting F5 configuration
- Error Handling
- Intro to AS3
- Operational change with AS3
- Deleting a web application
- Explore Ansible Tower
- Create an Ansible Tower job template

Requirements

Internet Connection with outbound ports 80, 443, 8443 and 22 allowed through firewall, SSH Client such as Putty or iTerm, Chrome or Firefox browser, Personal Github.com account with known username and password. Some attendees were required to disable corporate VPN for access into the lab environment.

House Keeping

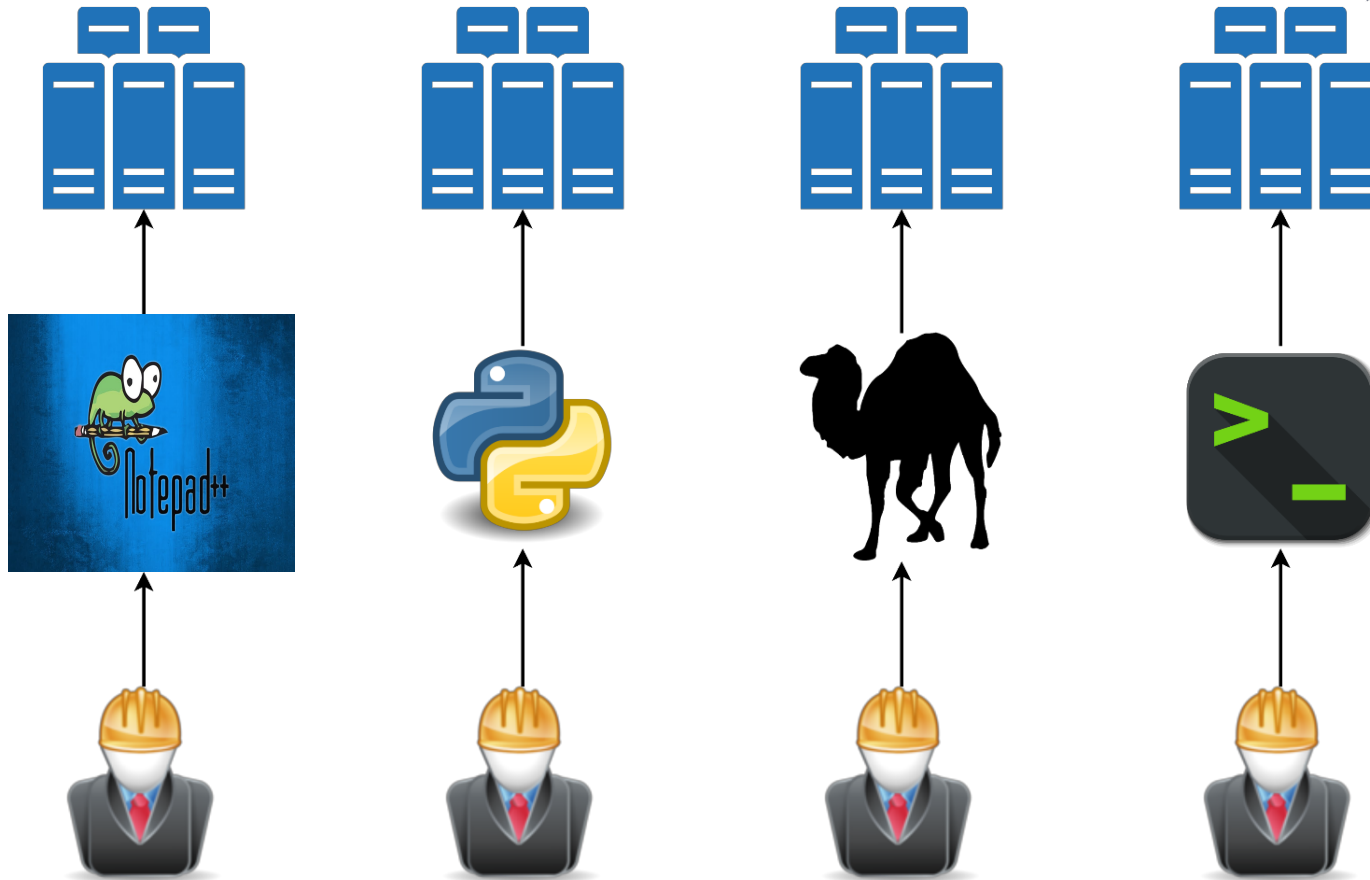
- › Introductions
- › Workshop
 - 2 Hours
- › Lab Requirements
 - GitHub Account
 - <https://github.com>
 - Web Browser
 - Chrome or Firefox
 - HTTP/HTTPS, port 8443 access through firewall
 - Lab Guide Download
 - Optional - 2 Screens

Agenda

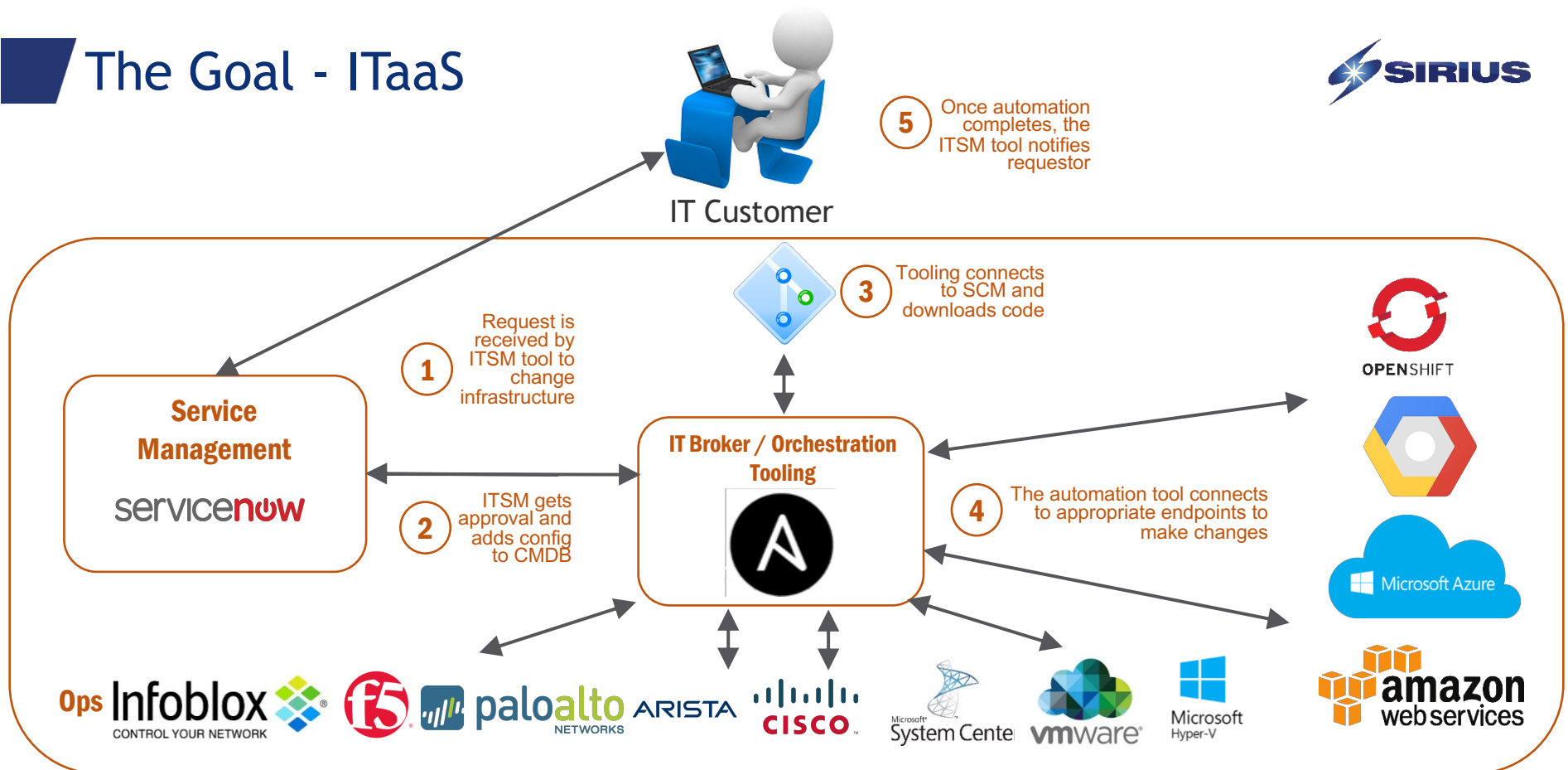


- › Automation Journey
- › Ansible overview
- › Jinja2
- › JSON
- › F5 Modules
- › F5 AS3
- › Cloud-Init
- › Labs

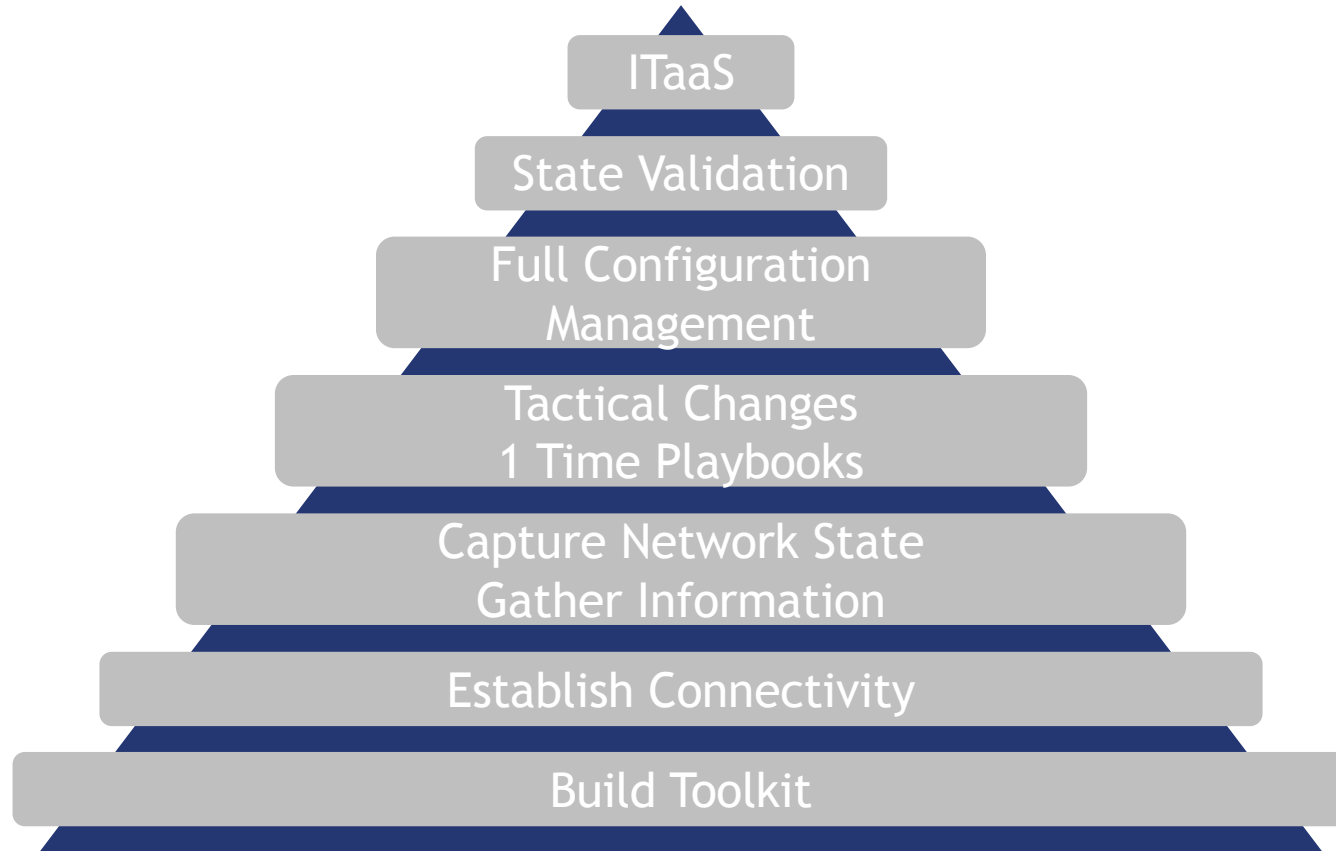
Chaotic Automation



The Goal - ITaaS



Automation Progression Pyramid



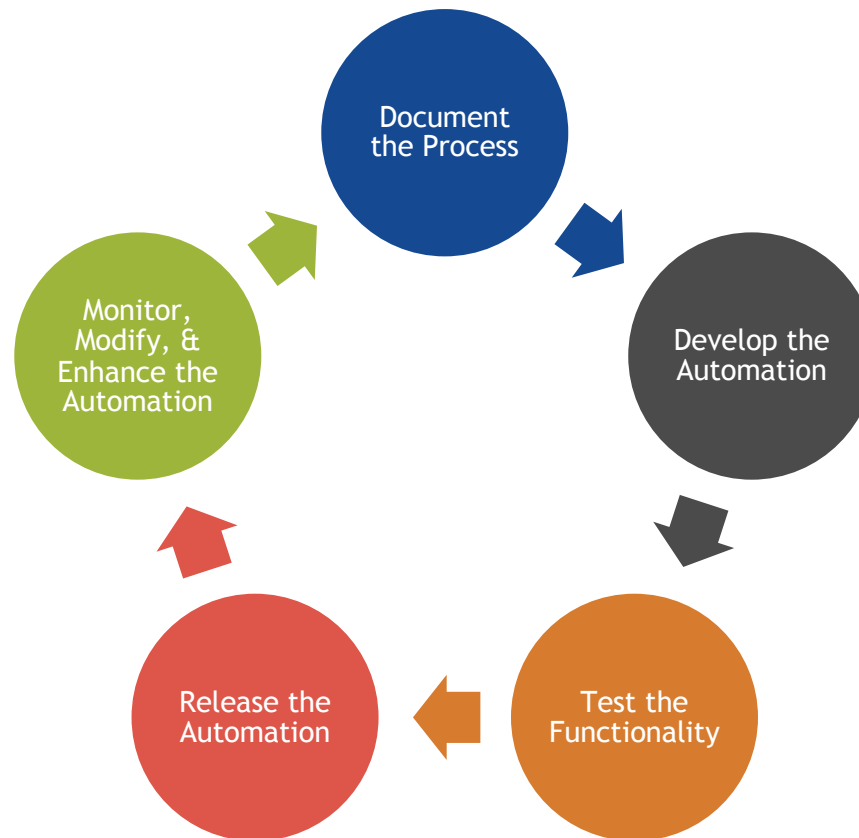
Automation Toolkit Components



- › **Automation Engine**
 - Ansible, Puppet, Chef, Saltstack, Terraform
- › **Orchestration Tool**
 - Cloudbolt, Morpheus, Tower
- › **Source Code Mananagement**
 - Gitlab, Github, Mercurial, BitBucket
- › **Credential Vault**
 - Cyberark, Hashicorp Vault
- › **IPAM/DDI/CMDB**
 - Bluecat, Infoblox, NetBox, ITSM
- › **Test/Dev Environment**





Automation Life Cycle



Why Ansible?

- › Easy to use and learn
- › Agentless
- › Extensible
- › Common tool across organizational silos
- › Centralized or Decentralized

...and 
BOB'S
 your
UNCLE!

```
---
- name: BIG-IP SETUP
  hosts: tag_lb
  connection: local
  gather_facts: false

  tasks:
    - name: Setting up provider values
      set_fact:
        provider:
          server: "{{ ansible_host }}"
          server_port: "8443"
          validate_certs: "False"

    - name: ADD VIRTUAL SERVER
      bigip_virtual_server:
        provider: "{{ provider }}"
        name: "{{ vipName }}"
        destination: "{{ private_ip_address }}"
        port: "443"
        enabled_vlans: "all"
        all_profiles: ['http', 'clientssl', 'oneconnect']
        pool: "{{ poolName }}"
        snat: "Automap"
```

What is Ansible Tower?

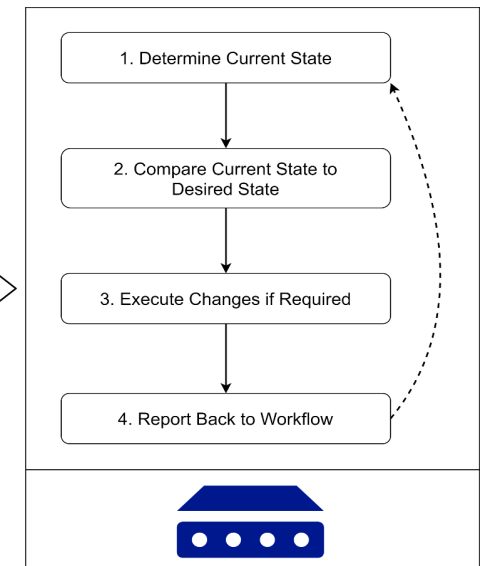


Key features of Ansible

- › Idempotency
 - The ability to run an operation which produces the same result whether run once or multiple times
- › Check-Mode
 - Validate code before making the change
- › Diff-Mode
 - Not all modules support but those that do use in combination with check-mode to review before applying a change
- › Limit
 - Dynamically adjust the scope of playbook at run time



Connect to Managed Device



How Does Ansible Work?

- › Inventory - ini file or dynamic
 - Contains hosts & groups
- › Playbooks - YAML with Jinja
 - Ansible tools include:
 - Plays, Tasks, Loops, Handlers, Blocks, Tags
- › Templates - Jinja2
 - Configurations and Documentation
- › Plugins
 - Python code that plugs into the core engine
- › Modules - Python and PowerShell
 - Can be any language that returns JSON
 - Code that ansible uses to interact with devices

Jinja2



- › Jinja2 is a template engine for the Python language.
- › Jinja2 can be used to template any text-based file such as web pages
- › All templating happens on the Ansible controller before the task is sent and executed on the target machine.
- › Jinja2 can use variables

```
"web_app": {
  "class": "Application",
  "template": "http",
  "serviceMain": {
    "class": "Service_HTTP",
    "virtualAddresses": [
      "{{private_ip}}"
    ],
    "pool": "app_pool"
  },
  "app_pool": {
    "class": "Pool",
    "monitors": [
      "http"
    ],
    "members": [
      {
        "servicePort": 443,
        "serverAddresses": [
          {% set comma = joiner(",") %}
          {% for mem in pool_members %}
            {{comma()}} "{{ hostvars[mem]['ansible_host'] }}"
          {% endfor %}
        ]
      }
    ]
  }
}
```

A close-up image of a ninja character wearing a black mask and a red and black headband, holding a red and black sword. The background is dark and moody.

JSON



- › JSON (JavaScript Object Notation) is a lightweight data-interchange format.
 - It is easy for humans to read and write.
 - It is easy for machines to parse and generate.
 - JSON is a text format that is completely language independent but uses conventions that are familiar to programmers.
 - JSON is ideal data-interchange language.
- › JSON is used by default in Ansible output unless otherwise specified



JSON - Continued



- › Because JSON is structured it is easy to search and parse
- › For some devices output is not structured and thus very difficult to parse. The usage of something like `parse_genie` or `ntc-ansible` will help structure unstructured data:
 - <https://github.com/networktocode/ntc-ansible>
 - NOTE: This will not be covered in this workshop

```
system_info": {
  "base_mac_address": "0a:0e:7f:e1:55:d2",
  "chassis_serial": "492f20ec-6fcc-bd43-fdd1d6cc5e64",
  "hardware_information": [
    {
      "model": "Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz",
      "name": "cpus",
      "type": "base-board",
      "versions": [
        {
          "name": "cache size",
          "version": "46080 KB"
        },
        {
          "name": "cores",
          "version": "2 (physical:2)"
        },
        {
          "name": "cpu MHz",
          "version": "2299.928"
        },
        {
          "name": "cpu sockets",
          "version": "1"
        },
        {
          "name": "cpu stepping",
          "version": "1"
        }
      ]
    }
  ]
}
```

F5 Modules



› bigip_*

- There are approximately 150 bigip modules to work with
 - bigip_facts, bigip_node, bigip_command, and bigip_config are a few examples
 - https://docs.ansible.com/ansible/latest/modules/partner_maintained.html#f5

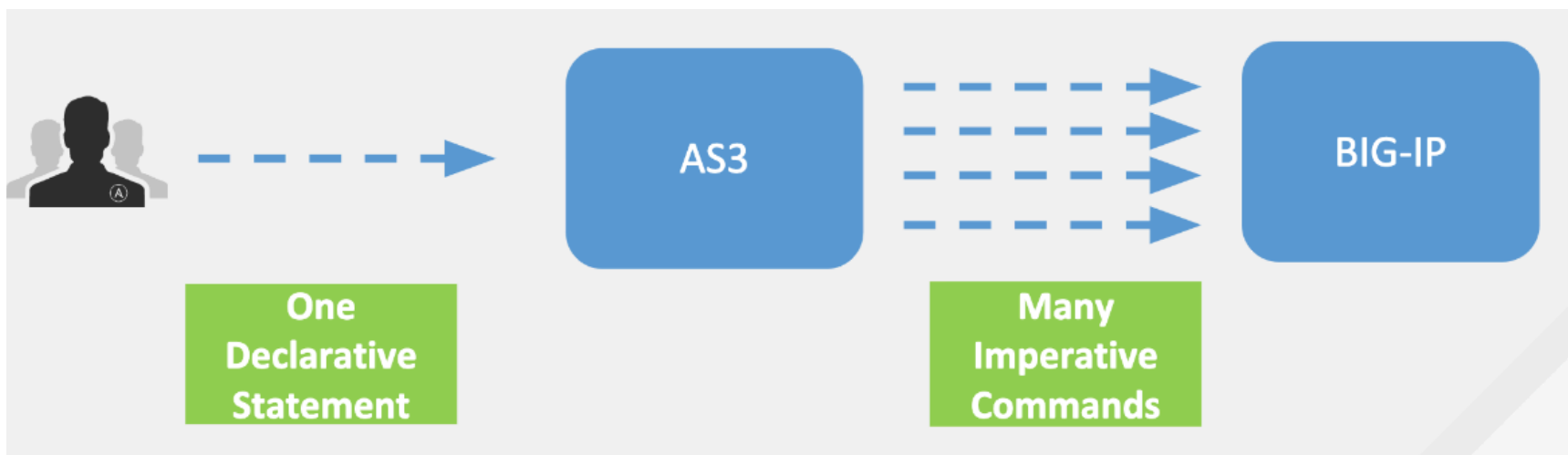
› bigiq_* (NOTE: bigiq will not be covered in this workshop)

- There are approximately 13 bigiq modules to work with
 - bigiq_device_info, bigiq_device_discovery, bigiq_application_http, and bigiq_utility_license are a few examples
 - https://docs.ansible.com/ansible/latest/modules/partner_maintained.html#f5

F5 AS3



- › App Services 3 Extension (AS3)
 - AS3 Uses a declarative model
 - You send a declaration file using a single Rest API call



F5 AS3 - Continued



- › Where is AS3 a good fit?
 - You require a declarative interface to abstract away the complexity of BIG-IP configuration
 - Organizations need to deploy BIG-IP configs as Infrastructure as Code (IaC) via integration with DevOps pipelines

- › Where is AS3 NOT a good fit?
 - Declarative interface is not desirable
 - Organization is unable to deploy iControl Extension RPM on BIG-IP
 - Organization wants to continue using imperative interfaces (GUI, TMSH, iControl REST APIs) to configure BIG-IP (not just monitor or troubleshoot)

F5 Cloud-Init

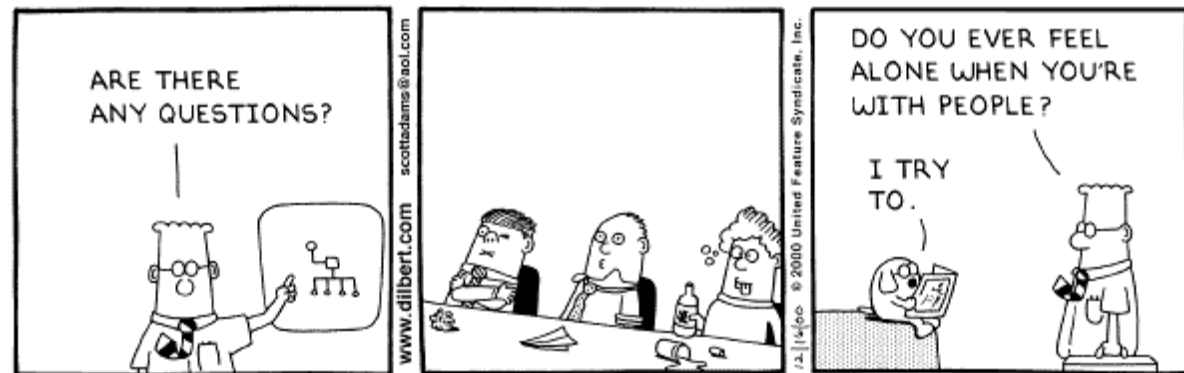


Cloud-Init is the industry standard start-up agent installed on virtual machines to facilitate cloud deployments. Beginning with BIG-IP VE 13.0.0, Cloud-Init is automatically installed on some platforms. You can speed up the initialization of your BIG-IP instance by passing user-data to perform tasks like onboarding and configuring BIG-IP. For example, to pass **user-data** you can use a bash startup script or cloud-init's built-in yaml-based configuration file, a cloud-config file.

<https://clouddocs.f5.com/cloud/public/v1/shared/cloudinit.html>

<https://github.com/f5devcentral/tmos-cloudinit#which-cloudinit-module-to-use>

Questions



Copyright © 2000 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited

Slide 397



Copyright © 2000 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited

Labs!!!

Username: Siduser<StudentID>

Password: <in chat>



THANK YOU