



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Automating Infrastructure

Network Programmability
(ACCB5029)

Applied Computing - WISA

Previously,

- ✓ Overview, Linux and Python Skills Review
- ✓ Devnet Developer Environment
- ✓ Software Development & Design
- ✓ Implement software version control using GIT
- ✓ Methods, Functions, Modules and Classes
- ✓ Data Formats
- ✓ APIs, Design and Architecture, REST APIs

This week,

- ✓ Understand Deployment Choices
- ✓ Creating and Deploying Application
- ✓ Continuous Integration/ Continuous Deployment

Previously, we discussed about Devnet skills, and course work briefing. We discussed Software Development and Design. I hope you had a clarity on Software Development life cycle and its development methodologies. Followed by discussion on version control systems, the importance of clean coding, overviewing object-oriented programming, Unit Testing. The last week's discussion on APIs, its designs, its use, architectural styles should have given clarity on how these technologies are used in real time environment providing better user experience. Today's lecture focuses on Application deployment.

- ✓ Previous week's Tutorial
 - Setup Lab environments, Review Linux and Python, PIP, and Python virtual environments, Version Control Systems, OOPS, Code Review and Testing, Integrate APIs
- ✓ This week's Tutorial and Independent learning
 - Continue to work with your Group.
 - **Lab 7: Create and Deploy a Sample Application**
 - **Lab 8: Building a CI/CD Pipeline using Jenkins**

Previously,

- ✓ Overview, Linux and Python Skills Review
- ✓ Devnet Developer Environment
- ✓ Software Development & Design
- ✓ Implement software version control using GIT
- ✓ Methods, Functions, Modules and Classes
- ✓ Data Formats
- ✓ APIs, Design and Architecture, REST APIs
- ✓ Understand deployment choices, creating & deploying Apps
- ✓ Continuous Integration/ Continuous Deployment
- ✓ Security

This week,

- ✓ Automating Infrastructure, Tools, Testing

We started discussion with Devnet skills, & coursework briefing. Continuing to explore Software Development and Design, understanding Software Development life cycle and its development methodologies. I hope the discussion on version control systems, the importance of clean coding, overviewing object-oriented programming, Unit Testing gave a clarity to understand the importance of Network programmability. We continued with APIs, its designs, its use, architectural styles and its deployment options and importance of CI/CD & Security. I hope this might give clarity on how these technologies are used in real time environment providing better user experience. Today's lecture focuses on Automation.

- ✓ Previous week's Tutorial
 - Setup Lab environments, Review Linux and Python, PIP, and Python virtual environments, Version Control Systems, OOPS, Code Review and Testing, Creating APIs, Deploy Application, Use automation tools (Jenkins).
- ✓ This week's Tutorial and Independent learning
 - Continue to work with your Group.
 - **Lab 9**

Automating Infrastructure

- Automation is **using code to configure, deploy, and manage applications together with the compute, storage, and network infrastructures and services** on which they run.
- For automation with Cisco infrastructure
 - the platforms can integrate with the common tools such as **Ansible, Puppet, Chef** and so on, or
 - provide direct API access to the programmable infrastructure.

Walk: Read only automation	Run: Activate policies & provide self-service across multiple domains	Fly: Deploy applications, network configurations, and more through CI/CD
✓ Using automation tools, information can be gathered about the network configuration. ✓ One can use a read scenario to audit configurations and do the next natural step, which is to put the configuration back into compliance. ✓ In the Automation Exchange, this shift is categorized as a walk-run-fly progression.	✓ With these Run stage automation scenarios, the users can safely provision their own network updates. ✓ On-boarding workflows can be automated, day-to-day network configurations can be managed, and daily scenarios can be pervaded.	✓ For more complex automation and programmable examples, the Fly stage of the DevNet Automation Exchange is used. ✓ Here the needs can be managed by monitoring and proactively managing the users and devices, while also gaining insights with telemetry data.

Automating Infrastructure

Why Automation?

- **Speed and agility** enable the business to explore, experiment with, and exploit opportunities ahead of their competition.
- **Dependency risks**
 - Today's software ecosystem is decentralized. Developers build individual components according to their needs and interests and mix and match components, infrastructure, and services needed to enable complete solutions and operate them efficiently at scale.
 - This ecosystem introduces new requirements and new risks:

Full Stack Automation: Automation is a key component of functional software-defined infrastructure and distributed and dynamic applications. The benefits of full-stack automation are:

- ✓ **Self-service**
- ✓ **Scale on demand**
- ✓ **Observability**
- ✓ **Automated problem mitigation**

Manual Operations add to costs, are time taking and are hard to scale. They are prone to human error, and documentation meant for humans is often incomplete and ambiguous, hard to test, and quickly outdated.

Automating Infrastructure

Software Defined Infrastructure: A case for Automation

- Software-defined infrastructure lets the developers and operators to use the software to requisition, configure, deploy and manage bare-metal and virtualized compute, storage and network resources.
- **Benefits:**
 - ✓ Self-service (platforms on demand)
 - ✓ Close specification, consistency, repeatability
 - ✓ Platform abstraction
- **Challenges:**
 - ✓ Developers must pay close attention to platform design, architecture, and security.
 - ✓ Access control is critical as cloud users with the wrong permissions can do a lot of damage to their organization's assets.
 - ✓ When cloud resources can be self-served quickly via manual operations, consumption can be hard to manage, and costs are difficult to calculate.

Automating Infrastructure

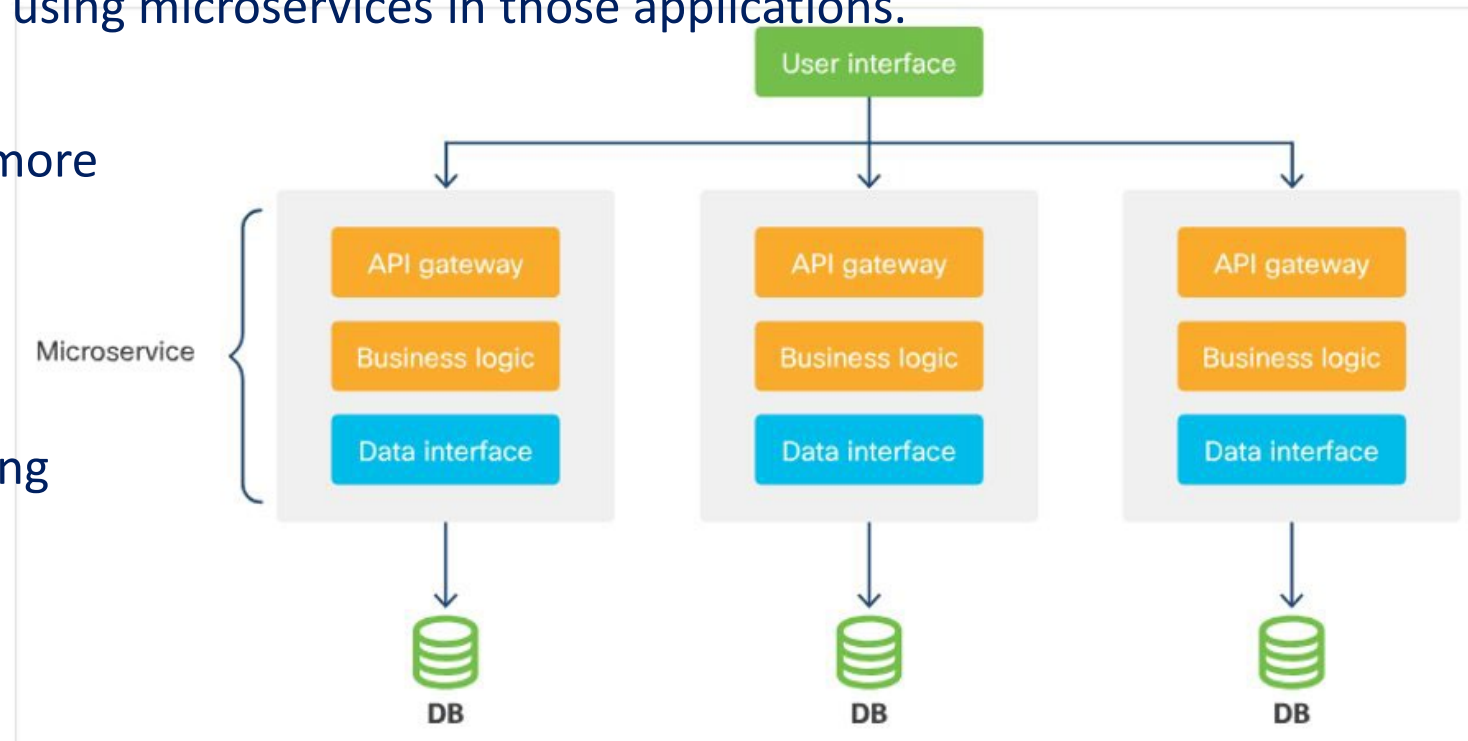
Our Applications: A case for Automation

- The applications which we were working in our labs: **Are they Dynamic, Distributed?**
- As a Group, discuss about the application we were working in our earlier labs.
- Could you (as a Team) identify any microservices in that application?

If Yes? Discuss on the benefits and challenges of using microservices in those applications.

If No?

- How can we improve our applications to be more scalable and flexible using microservices?
- Prepare a design (structure) to illustrate the use of microservices in those applications.
- Discuss on the benefits and challenges of using Microservices in those applications.



Automating Infrastructure

- These business and technical needs, trends, and dynamics, encourage developers and operators to use automation everywhere for the following tasks:
 - ✓ **Manage all phases of app building, configuration, deployment and lifecycle management.** This includes coding, testing, staging, and production.
 - ✓ **Manage software-defined infrastructures on behalf of the applications you build.**
 - ✓ **Alongside the applications, to preserve, update, and continually improve the automation code.** This code helps to develop, test, stage, monitor, and operate the apps at production scales, and in various environments. Increasingly, all this code can be treated as one work-product.

DevOps

- In the past, Development and Operations are two different aspects.
 - ✓ **Dev (Development):** creating applications was the job of software developers
 - ✓ **Ops (Operations):** ensuring that apps work for users and the business

Characteristics	Dev	Ops
Cares about	Tailor-made applications and how they work	Applications and how they run, and its Infrastructure, OS, network, etc
Business as	Profit center: demands resources	Cost center: provides and accounts for resources
on-call rotation	Occasionally (only when issues are escalated)	Regularly (point of spear)
Performance	Abstractly (including bad metrics)	Concretely (SLA compliance, issues resolved)
Skills required	Deeper than broader: Languages, APIs, architecture, tools, process and so on	Broader than deeper: Configuration, administration, OS, automation, and so on
Agility required	Move fast, innovate, break things, fix later	Investments must be extensively justified, expectations managed

DevOps

- Discussions in 2008 moving forward emphasized the importance of merging Development and Operations.
- Introduction of “Infrastructure as a Code”
- Importance of Version Control System
- The need for collaboration and communication between Dev and Ops Teams
- Importance of automation both in development and operations (*Areas like Testing, deployment and Infrastructure management need to be automated*)

DevOps

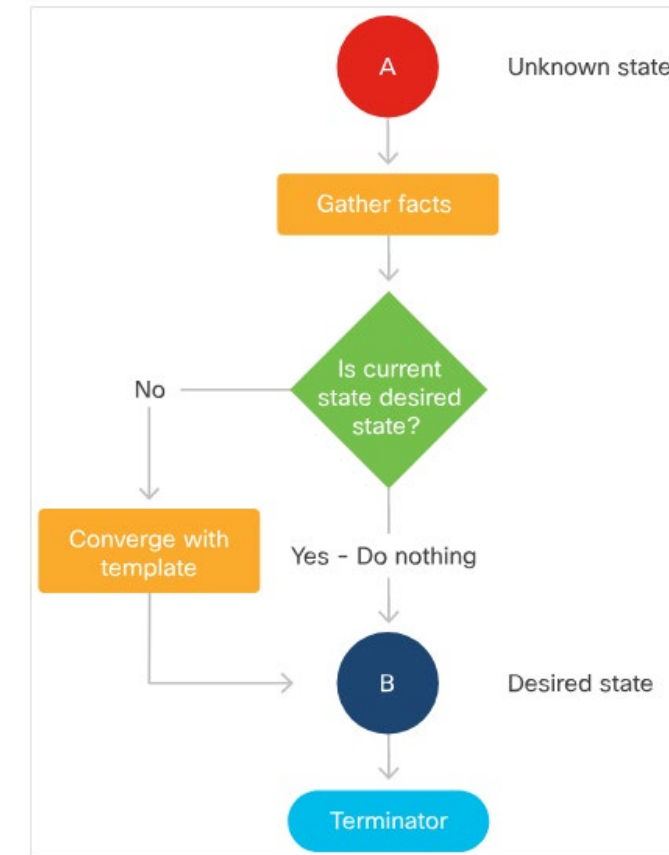
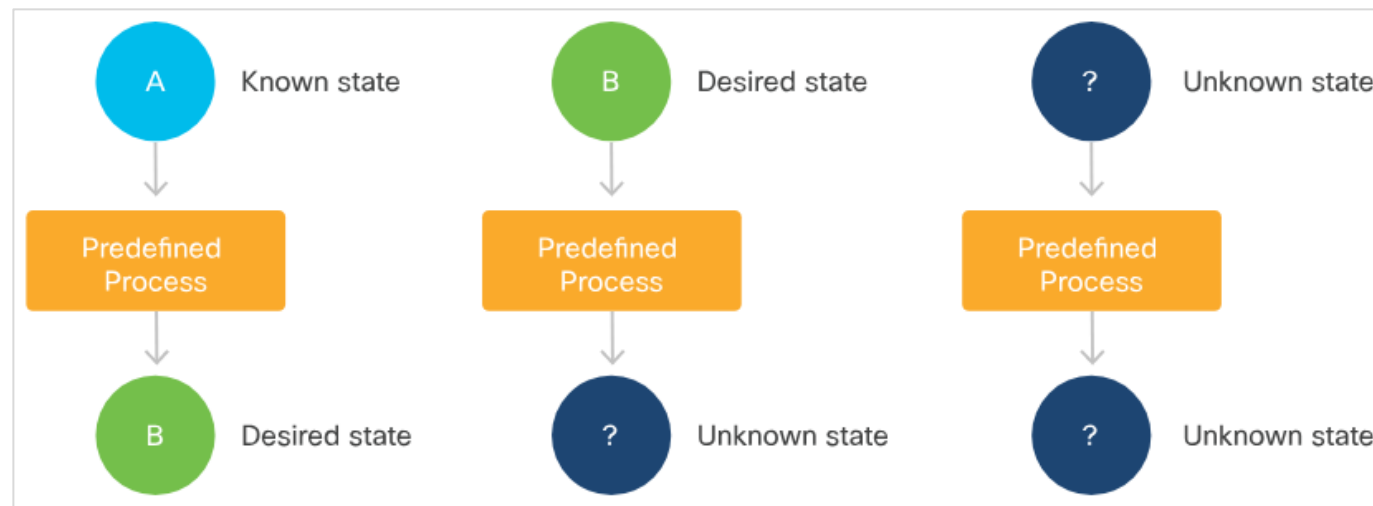
- DevOps/SRE have many core principles and best practices:
 - ✓ A focus on automation | The idea that "failure is normal" | A reframing of "availability" in terms of what a business can tolerate
 - ✓ **SLOs and SLIs**
 - ✓ DevOps must deliver measurable, agreed-upon business value and the statistical reality of doing so perfectly is impossible.
 - ✓ These ideas are codified in a Service Level Objective (SLO) that is defined in terms of real metrics called Service Level Indicators (SLIs).
 - ✓ SLIs map to the practical reality of delivering a service to customers.
 - ✓ SLO/SLI methodology **permits cheaper, more rapid delivery of business value** by removing the obligation to seek perfection.
 - ✓ SLO/SLI methodology also influences the pace, scope, and other aspects of development to ensure and improve adequacy.
 - ✓ DevOps is **co-evolving with technologies like virtualization and containerization**, enabling a unified approach and unified tool set to **support coordinated application and infrastructure engineering**.

Automation

- There are different aspects of automation. Referring to our earlier discussions and lab activities, that focused on automation in software development.
 - + Jenkins which primarily a continuous integration and continuous delivery (CI/CD) tool that is used to automate various parts of the software development process
 - + Building, Testing, and Deploying applications.
 - + Automation of software development workflows and integrates with a wide range of tools like version control systems, testing frameworks, and deployment tools.
- Other aspect of automation:
 - + setup and configuration of infrastructure and applications.
 - + Ansible is used to manage the configuration of multiple servers or instances, using playbooks.
 - + It defines the desired state of the infrastructure and the tasks needed to achieve that state.
 - + Ansible is focused on the automation of infrastructure, rather than software development workflows.

Procedural Automation

- An ordered sequence of commands aimed at achieving a goal.
- The sequence may include flow-control, conditions, functional structure, classes, and more.
- Carefully-written procedural scripts and declarative configuration tools examine targets before performing tasks on them, and only perform the tasks needed to achieve the desired state.
- This quality of software is called idempotency.



Automation Tools

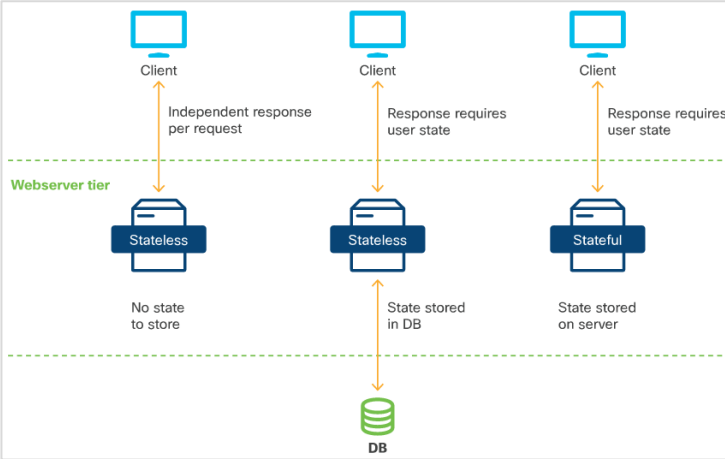
- Automation tools offer powerful capabilities compared to ad-hoc automation strategies using BASH, Python, or other programming languages. These tools enable developers to:
 - + Simplify and standardize
 - + Accelerate development with out-of-the-box features
 - + Facilitate reusability, segregate concerns, promote security
 - + Perform discovery and manage inventory
 - + Handle scale
 - + Engage community

Automation Tools

Provisioning	Configuration	Deployment	Orchestration
This refers to obtaining compute, storage, and network infrastructure (real or virtual) enabling communications, putting it into service, and making it ready for use by operators and developers.	This means installing base applications and services, and performing the operations, tasks, and tests required to prepare a low-level platform to deploy applications or a higher-level platform.	This involves building, arranging, integrating, and preparing multi-component applications or higher-level platforms, often across multiple nodes	This may refer to several things: • User-built or platform-inherent automation aimed at managing workload lifecycles and reacting dynamically to changing conditions. • Processes or workflows that link automation tasks to deliver business benefits, like self-service.

Automation works best when **applications can be made stateless**. This means that redeploying them in place does not destroy or lose track of the data that users or operators need.

The two states of an application are: **Not Stateless & Stateless**

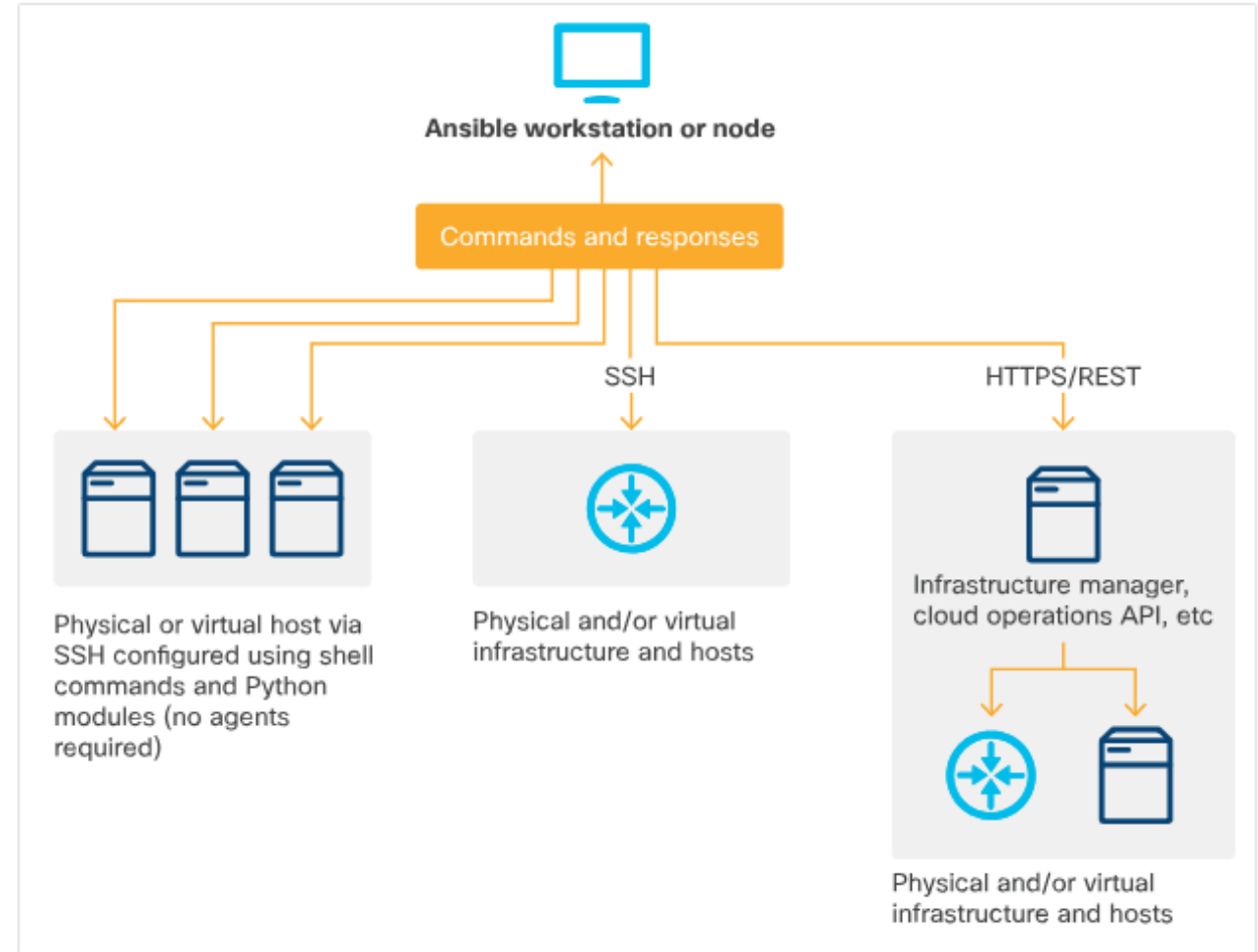


Automation Tools

- The first modern automation tool was **Puppet which was introduced in 2005** as open source, and then commercialized as Puppet Enterprise by Puppet Labs in 2011.
- The most popular automation tools are Ansible, Puppet, Chef.
- They share the following characteristics:
 - + Relatively easy to learn
 - + Available in open-source versions
 - + Plugins and adapters enable them to directly or indirectly control many types of resources
- **Ansible is available as open source**, and in a version with added features, from IBM/Red Hat, called Ansible Tower.
- Ansible is substantially managed from the Bash command line, with automation code developed and maintained using any standard text editor.

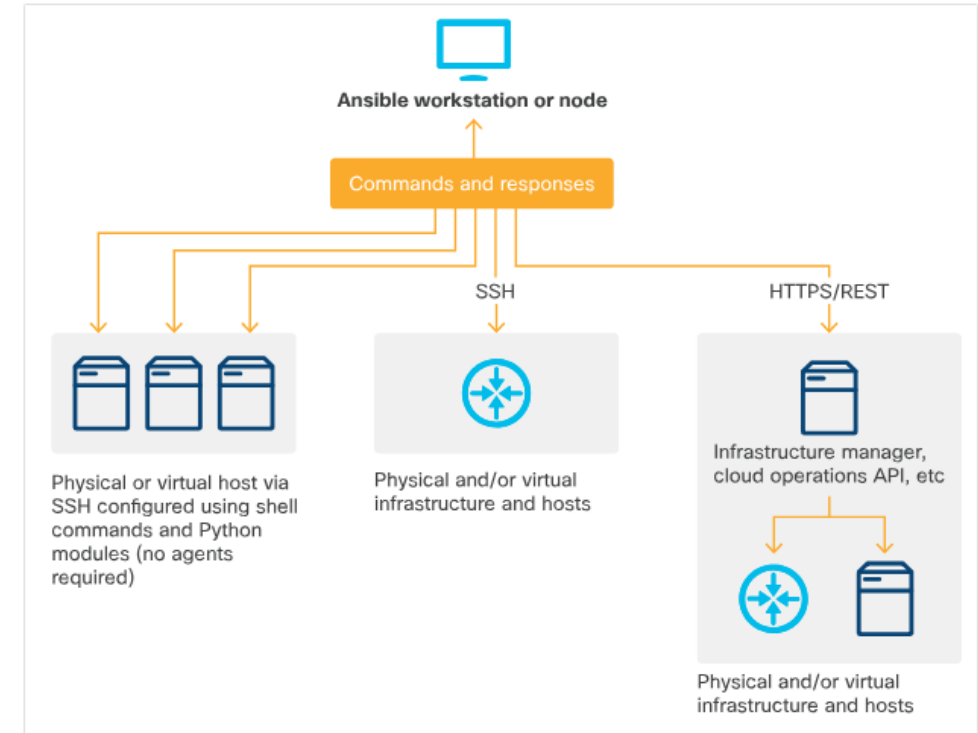
Automation Tool: Ansible

- Ansible Architecture: **Simple and Lightweight**
- Control node runs on any Linux machine running Python 3. All system updates are performed on control node.
- Control node connects to managed resources over SSH and enable Ansible to:
 - + Run shell commands on a remote server, or transact with a remote router, or other network entity, via its REST interface.
 - + Inject Python scripts into targets and remove them after they run.
 - + Install Python on target machines if required.
- Plugins enable Ansible to gather facts from and perform operations on infrastructure that can't run Python locally.



Automation Tool: Ansible

- Installing Ansible: The Ansible control node application is installed on a Linux machine from its public package repository.
- Ansible code structure: In the Ansible code structure,
 - + work is separated into YAML (.yml) files that contain a sequence of tasks,
 - + executed in top-down order.
 - + Ansible has hundreds of pre-built Python modules that wrap operating-system-level functions and meta-functions.
- Playbooks and roles: An Ansible playbook can be written as a monolithic document with a series of modular, named tasks.
 - + Developers build a model of a complex DevOps task out of low-level playbook task sequences called roles and then reference these in higher-level playbooks, sometimes adding additional tasks at the playbook level.
 - + This segregation of concerns ensures clarity, reusability and shareability of roles.



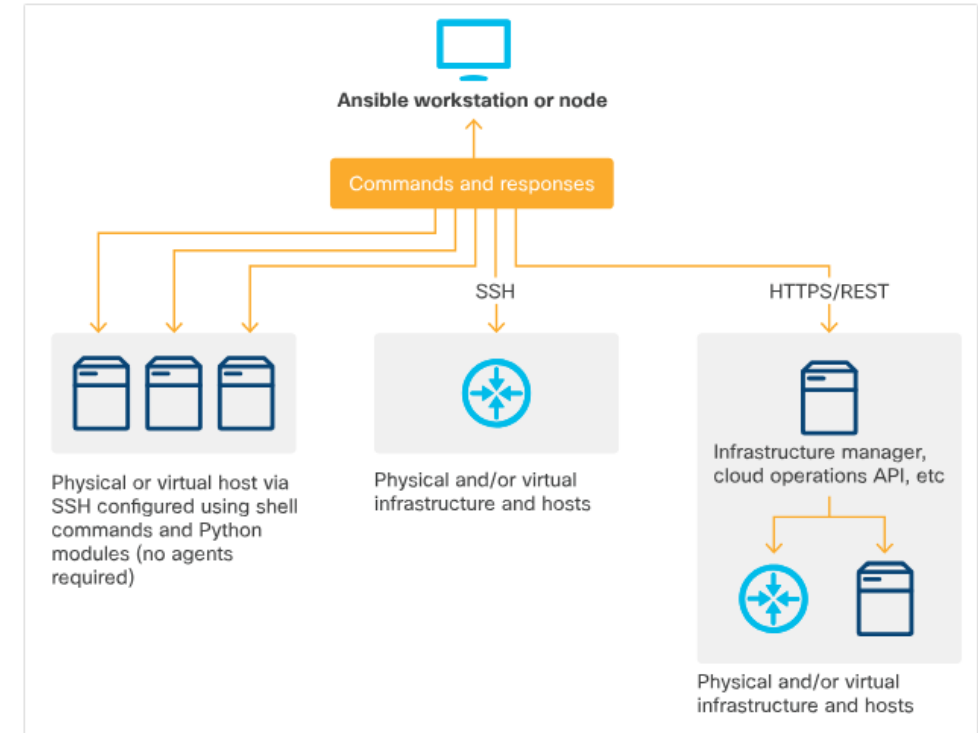
Automation Tool: Ansible

Ansible project organization

- Ansible projects are organized in a nested directory structure. The hierarchy is easily placed under version control and used for GitOps-style infrastructure as code.
- Ansible folder hierarchy elements include Inventory files, Variable files, Library and utility files, Main playbook files.

Ansible at scale

- There are scaling challenges for large organizations, such as managing and controlling access to many Ansible nodes flexibly and securely. This also includes putting remote controllers seamlessly and safely under control of centralized enterprise automation.
- For this, there are two control-plane solutions: Red Hat Ansible Tower and AWX project.
- Larger-scale Ansible implementations also benefit from Ansible Vault, a built-in feature that enables encryption of passwords and other sensitive information.



Automation Tools

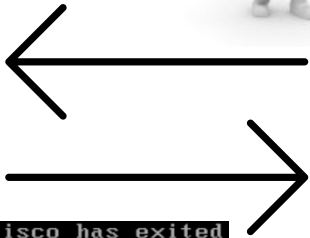


CSR1Kv

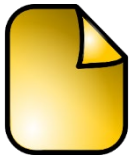
```
*Mar 30 04:38:30.544: %SYS-6-LOGOUT: User cisco has exited
.56.102)

***                               ***
***                               ***
*** Cisco Networking Academy       ***
***                               ***
*** This software is provided for   ***
*** Educational Purposes           ***
*** Only in Networking Academies   ***
***                               ***
***                               ***

CSR1kv>en
CSR1kv#sho ip int br
Interface      IP-Address      OK? Method Status
GigabitEthernet1 192.168.56.101 YES DHCP up
```

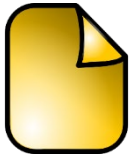


Ansible



inventory file (hosts)

```
csr1000v > hosts
# Enter the hosts or devices for Ansible playbooks
CSR1kv ansible_user=cisco ansible_passwprd=cisco123! ansible_host=192.168.56.101
```



Ansible.cfg (default files)

```
devasc@labvm:~/labs/devnet-src/ansible$ ansible --version
ansible 2.9.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/devasc/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/devasc/.local/lib/python3.8/site-packages/ansible
  executable location = /home/devasc/.local/bin/ansible
  python version = 3.8.2 (default, Apr 27 2020, 15:53:34) [GCC 9.3.0]
devasc@labvm:~/labs/devnet-src/ansible$ cat /etc/ansible/ansible.cfg | more
# config file for ansible -- https://ansible.com/
#
# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, and ansible.cfg in
# the following locations:
#
# - /etc/ansible
# - ~/.ansible
# - ~/.config/ansible

[defaults]
# config file for ansible-csr1000v
# Use local hosts file in this folder
inventory=./hosts
host_key_checking = False # Don't worry about RSA Fingerprints
retry_files_enabled = False # Do not create them
deprecation_warnings = False # Do not show warnings
```

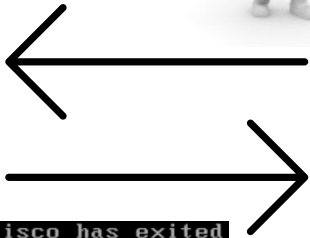
The ansible.cfg file tells Ansible where to find the inventory file and sets certain default parameters.

Automation Tools



CSR1Kv

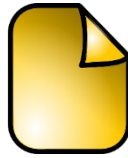
```
*Mar 30 04:38:30.544: %SYS-6-LOGOUT: User cisco has exited
.56.102)
**
**
*** Cisco Networking Academy ***
*** This software is provided for ***
*** Educational Purposes ***
*** Only in Networking Academies ***
**
**
CSR1kv>en
CSR1kv#sho ip int br
Interface          IP-Address      OK? Method Status
GigabitEthernet1   192.168.56.101 YES DHCP    up
```



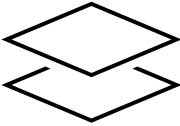
Ansible.cfg (default files)



Ansible



inventory file (hosts)



Playbook

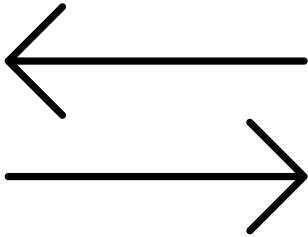
```
ansible-csr1000v > ! backup_cisco_router_playbook.yaml
1 ---
2 - name: AUTOMATIC BACKUP OF RUNNING-CONFIG
3   hosts: CSR1kv
4   gather_facts: false
5   connection: local
6
7   tasks:
8     - name: DISPLAYING THE RUNNING-CONFIG
9       ios_command:
10         commands:
11           - show running-config
12       register: config
13
14     - name: SAVE OUTPUT TO ./backups/
15       copy:
16         content: "{{ config.stdout[0] }}"
17         dest: "backups/show_run_{{ inventory_hostname }}.txt"
18
```

An Ansible playbook is a YAML file containing one or more plays. Each play is a collection of tasks

Automation Tools



CSR1Kv



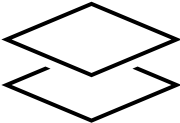
Ansible.cfg (default files)



Ansible



inventory file (hosts)



Playbook

```
devasc@labvm:~$ cd labs/devnet-src/ansible/ansible-csr1000v/
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ mkdir backups
devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook backup_cisco_router_playbook.yaml

PLAY [AUTOMATIC BACKUP OF RUNNING-CONFIG] *****
TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [CSR1kv]

TASK [SAVE OUTPUT TO ./backups/] *****
changed: [CSR1kv]

PLAY RECAP *****
CSR1kv : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$ ansible-playbook backup_cisco_router_playbook.yaml -i hosts

PLAY [AUTOMATIC BACKUP OF RUNNING-CONFIG] *****
TASK [DISPLAYING THE RUNNING-CONFIG] *****
ok: [CSR1kv]

TASK [SAVE OUTPUT TO ./backups/] *****
ok: [CSR1kv]

PLAY RECAP *****
CSR1kv : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

devasc@labvm:~/labs/devnet-src/ansible/ansible-csr1000v$
```

```
EXPLORER  hosts  ansible.cfg  backup_cisco_router_playbook.yaml  show_...
OPEN EDIT...  ansible-csr1000v > backups > show_run_CSR1kv.txt
  hosts ansible-csr1000v 1 Building configuration...
  ansible.cfg ansible-csr1000v 2
  ! backup_cisco_router_playbook.yaml 3 Current configuration : 3916 bytes
  show_run_CSR1kv.txt 4 !
  5 ! Last configuration change at 03:25:28 UTC Thu Mar 30 2023
  6 !
  > ansible-apache 7 version 16.9
  ansible-csr1000v 8 service timestamps debug datetime msec
  backups 9 service timestamps log datetime msec
  show_run_CSR1kv.txt 10 platform qfp utilization monitor load 80
  ansible.cfg 11 no platform punt-keepalive disable-kernel-core
  ! backup_cisco_router_playbook.yaml 12 platform console virtual
  ! checklocalinfo.yaml 13 !
  hosts 14 hostname CSR1kv
  servers 15 !
  ! backup_cisco_router_playbook.yaml 16 boot-start-marker
  17 boot-end-marker
  18 !
```



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Trawsnewid Addysg, Trawsnewid Bywydau
Transforming Education, Transforming Lives

Feedback/ Any Questions?

Module Expectations and Daily Feedback - ACCB5029

Tue, Oct 07, 2025

<https://vevox.app/#/m/136436531>

Session ID: 136-436-531

