Ansible on Windows Fundamentals

Introducing Ansible Automation for Windows

Goals of this Course

- Learn how to automate management of Microsoft Windows systems with Ansible
- Use Red Hat Ansible Tower (or AWX) and Git to centrally manage automation and automation code
- Write Ansible Playbooks to perform common tasks specific to Microsoft Windows systems
- Obtain, create, and run reusable automation code with Ansible Roles
- Leverage existing PowerShell Desired State Configuration resources with Ansible

Skills Needed for this Course

- Experience with system administration of Microsoft Windows Server
- A very basic understanding of Ansible (see “Getting Started with Ansible on Windows”)
- Limited Linux experience is sufficient
  (Linux will only be used briefly to install Red Hat Ansible Tower from an installation script)

Introducing
Red Hat Ansible Automation Platform

Red Hat

Objectives

- Explain the benefits of using Ansible for Microsoft Windows automation
- Review the basic concepts behind the operation of Ansible

Introduction

- Old approaches use manual tasks: checklists, documentation, memorized routine
  - Error prone, easy to skip steps, make mistakes
  - Maintenance is harder, IT environment is less stable
- Automation can help avoid problems caused by manual system administration
  - Ensure all systems are quickly and correctly deployed and configured
  - Frees you to do more productive work instead of repetitive tasks

Red Hat

What is Ansible?

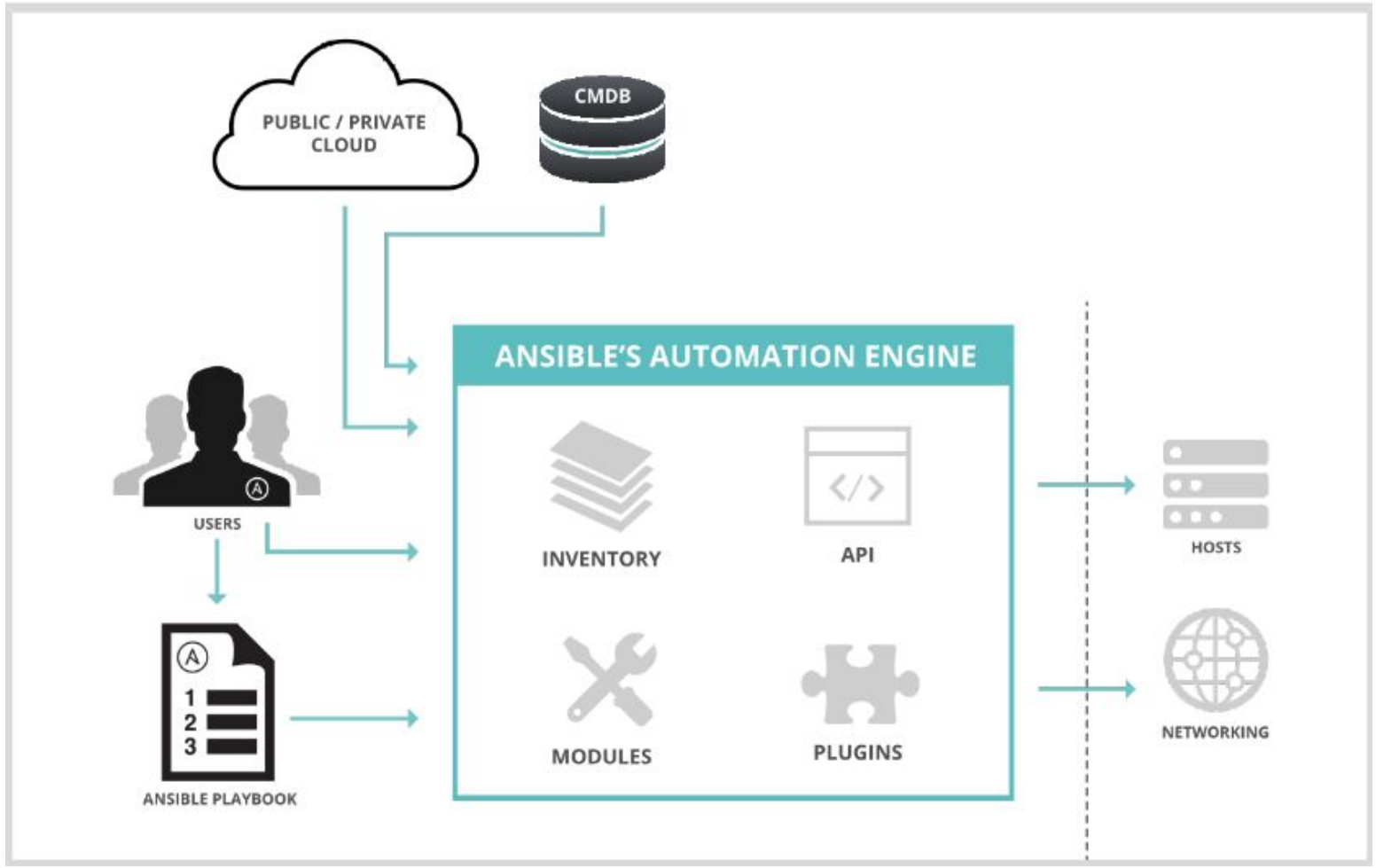
- An open source automation platform and a simple automation language.
- Designed to be simple for humans to read and write
- Automation code is written in text files that can be managed in version control
- Code written like a checklist that documents the state you want your infrastructure to have
- Thousands of “modules” allow you to automate tasks without deep programming skills
- Cross-platform: can manage diverse operating systems and network devices
- Agentless architecture requires no special code on managed hosts

Ansible's Agentless Architecture

- Ansible does not require you to install a custom agent on managed hosts
- Protocols and software included with the operating system are leveraged
  - Windows Remote Management (or PowerShell Remoting Protocol) and PowerShell
  - SSH and Python used on Linux systems
- Advantages of using common, well-tested and understood tools
  - Simpler to prepare systems
  - Reduces security risks

Ansible Architecture

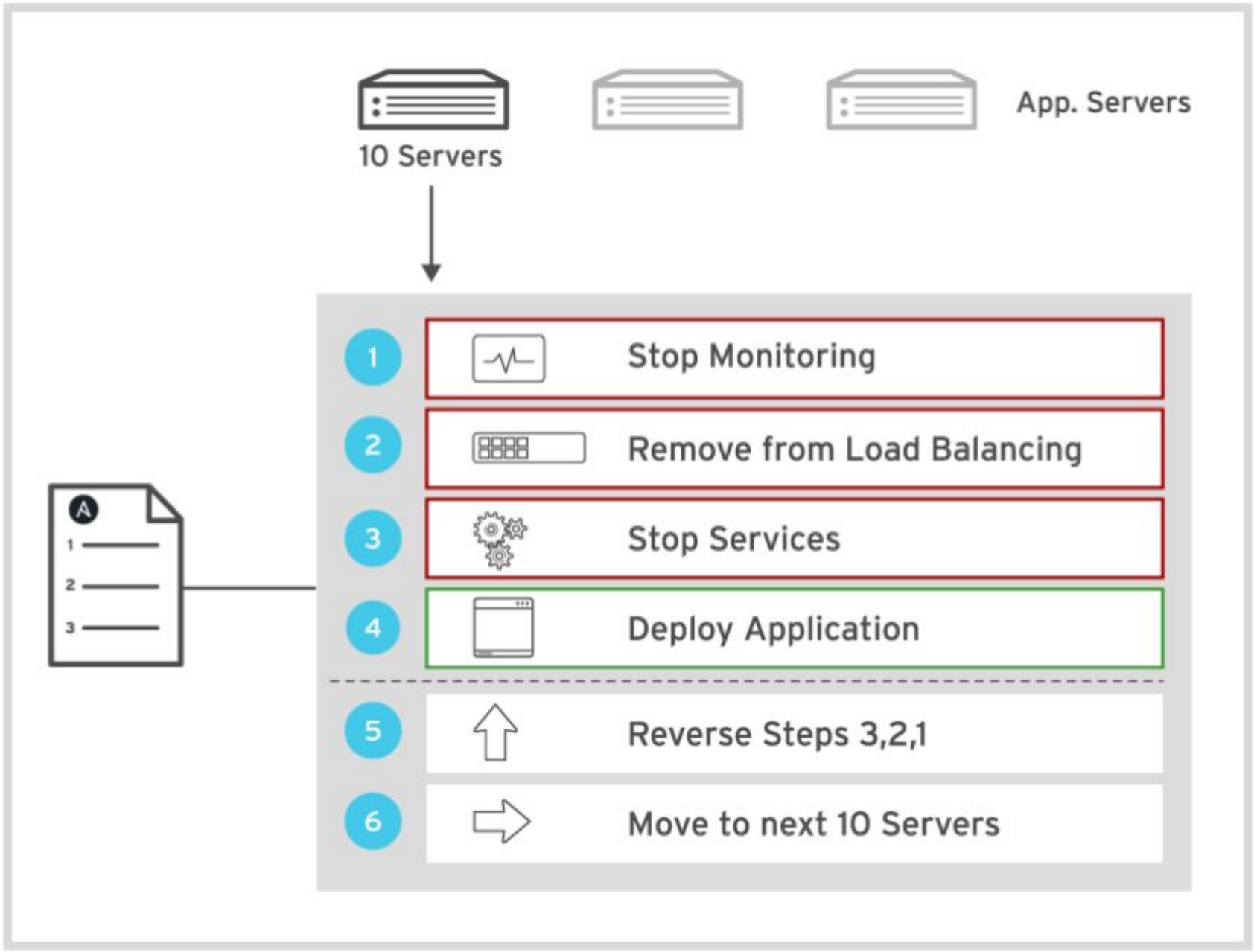
The next few slides will detail the various components of Ansible and how they work together.



Red Hat

Ansible Concepts and Architecture

- The inventory is a set of hosts and groups of hosts managed by Ansible
  - Can be written by hand or dynamically generated from a central source of truth
- A playbook is a YAML text file that contains one or more plays
- Each play performs a list of tasks on a set of hosts, in order
- Each task runs a module to do a specific thing on or for the managed host
  - If the host is already in a correct state, the task does nothing and continues
  - If the host is not in a correct state, the task changes it to be correct
  - If the task cannot correct the host's state, it reports failure and stops for that host
- Ansible makes it easy to write playbooks that can be rerun safely



Red Hat

Ansible Concepts and Architecture

- A control node is installed with Ansible and is used to run playbooks
  - Contains the Ansible Engine software and the playbook and its supporting materials
  - Red Hat Ansible Tower provides a central web interface, authentication, and API for Ansible
- A managed host is a machine that is managed by Ansible automation
  - Does not have Ansible installed
  - Does need to be configured to allow Ansible to connect to the host
  - Must be listed in the inventory (or generated by a dynamic inventory script or plugin)

Red Hat Ansible Tower

- Red Hat Ansible Tower helps you control, secure and centrally manage Ansible automation.
- Can be your authoritative control node to run playbooks.
- Has a web-based user interface and a RESTful API.
- Users with no Linux experience can use the web-based UI to easily run Ansible.
- Different users have different levels of access to playbooks, hosts, and authentication credentials.
- Allows central tracking and logging of automation jobs.
- Makes it easy to manage and use different versions of playbooks under a version control system

Red Hat

Some Use Cases for Ansible

- Configuration Management
  - Centralizing configuration file management and deployment.
- Application Deployment
  - Effectively manage the entire application life cycle from deployment to production.
- Provisioning
  - Streamline the provisioning process.
- Continuous Delivery
  - Create a CI/CD pipeline using Ansible playbooks.
- Security and Compliance
  - Define security policy using Ansible playbooks.
- Orchestration
  - Define how multiple configurations interact.

The Ansible Way

- Complexity Kills Productivity
  - Simpler is better.
- Optimize for Readability
  - Ansible Playbooks can clearly document your workflow automation.
- Think Declaratively
  - Ansible is a desired-state engine.
  - Ansible is not a scripting language and should not be treated as one.

Design of a Windows Automation Infrastructure

Red Hat

Objectives

- Discuss recommended practices for Microsoft Windows automation with Ansible
- Explain the proof-of-concept architecture that will be used throughout the remainder of the course
- Explain how to install Red Hat Ansible Tower

Recommended Ansible Architecture for Windows

- Use Red Hat Ansible Tower as the control node
- Store Ansible Playbooks in Git (or another version control system)
- Edit Ansible Playbooks in a text editor, like Visual Studio Code, that can directly work with Git

Lab Architecture Used in this Course

- A Red Hat Enterprise Linux or CentOS system to be installed with Red Hat Ansible Tower
- A network-accessible Git repository (may use GitHub, GitLab, or other service)
- A workstation installed with Visual Studio Code (on Microsoft Windows or some other supported OS)
- One or more Windows Server computers to manage with Ansible
- All systems must be able to communicate with each other over the network

Why Red Hat Ansible Tower?

- Web-based user interface simplifies operation of Ansible for users with no Linux experience
- It can automatically grab the latest playbook from Git (or be pinned to use a specific version)
- Authentication and permissions system helps manage who can change which hosts, use specific host credentials, run certain playbooks, and so on
- The central service enables logging of playbook runs and auditing of events
- It provides a REST API that can be integrated with automated DevOps workflows

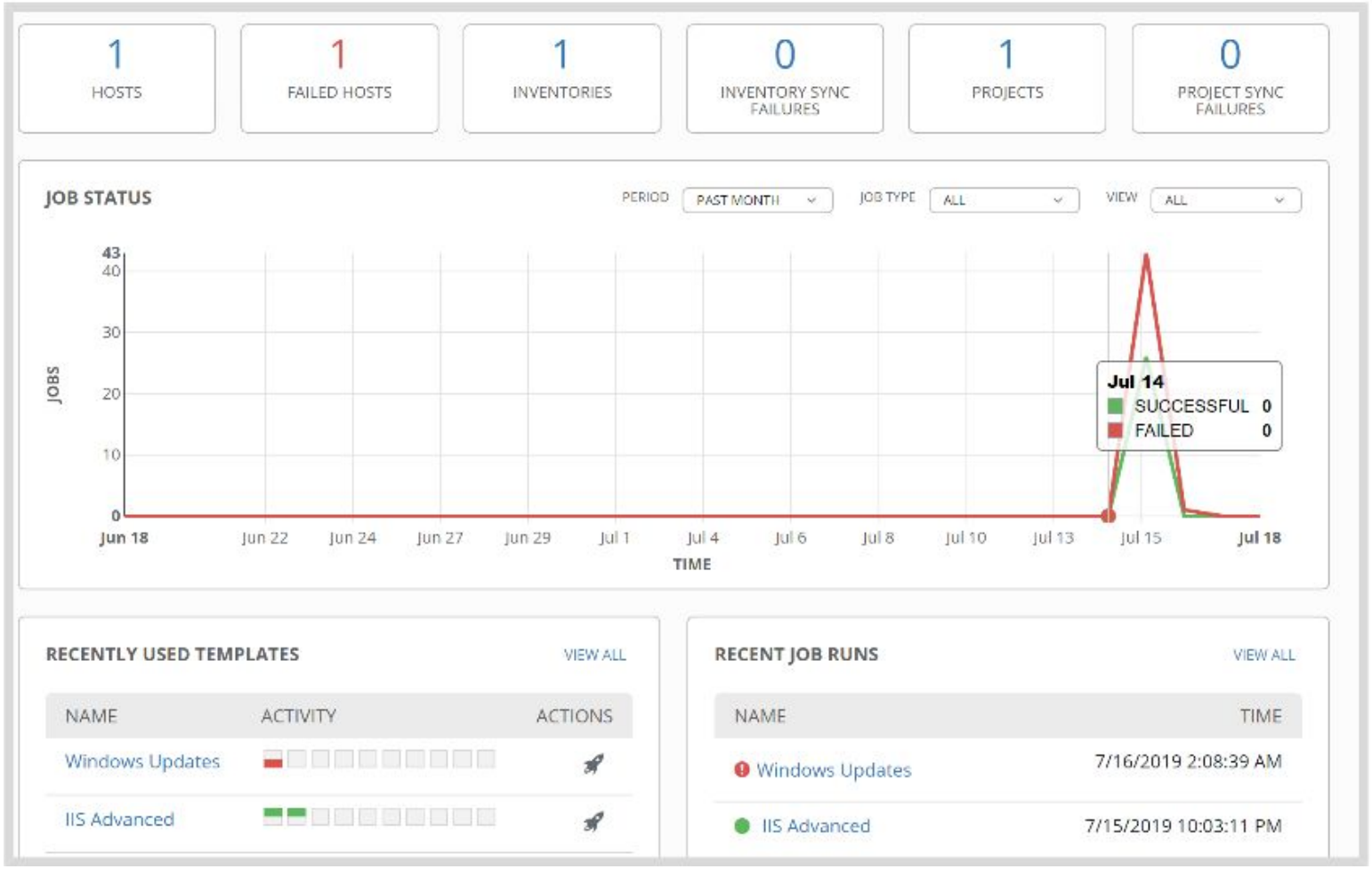
Installing Red Hat Ansible Tower

- This class uses a single-node installation
- Needs a system installed with Linux ("bare metal", virtual machine, or cloud instance)
  - Red Hat Enterprise Linux or CentOS 7.4 or later
  - 2 CPU / 4 GB RAM / at least 40 GB storage
  - On Amazon AWS EC2, can use m4.large instances
- You can get an evaluation copy of Red Hat Ansible Tower from Red Hat
  - https://www.ansible.com/tower-trial
  - Re-download software only: https://releases.ansible.com/ansible-tower/

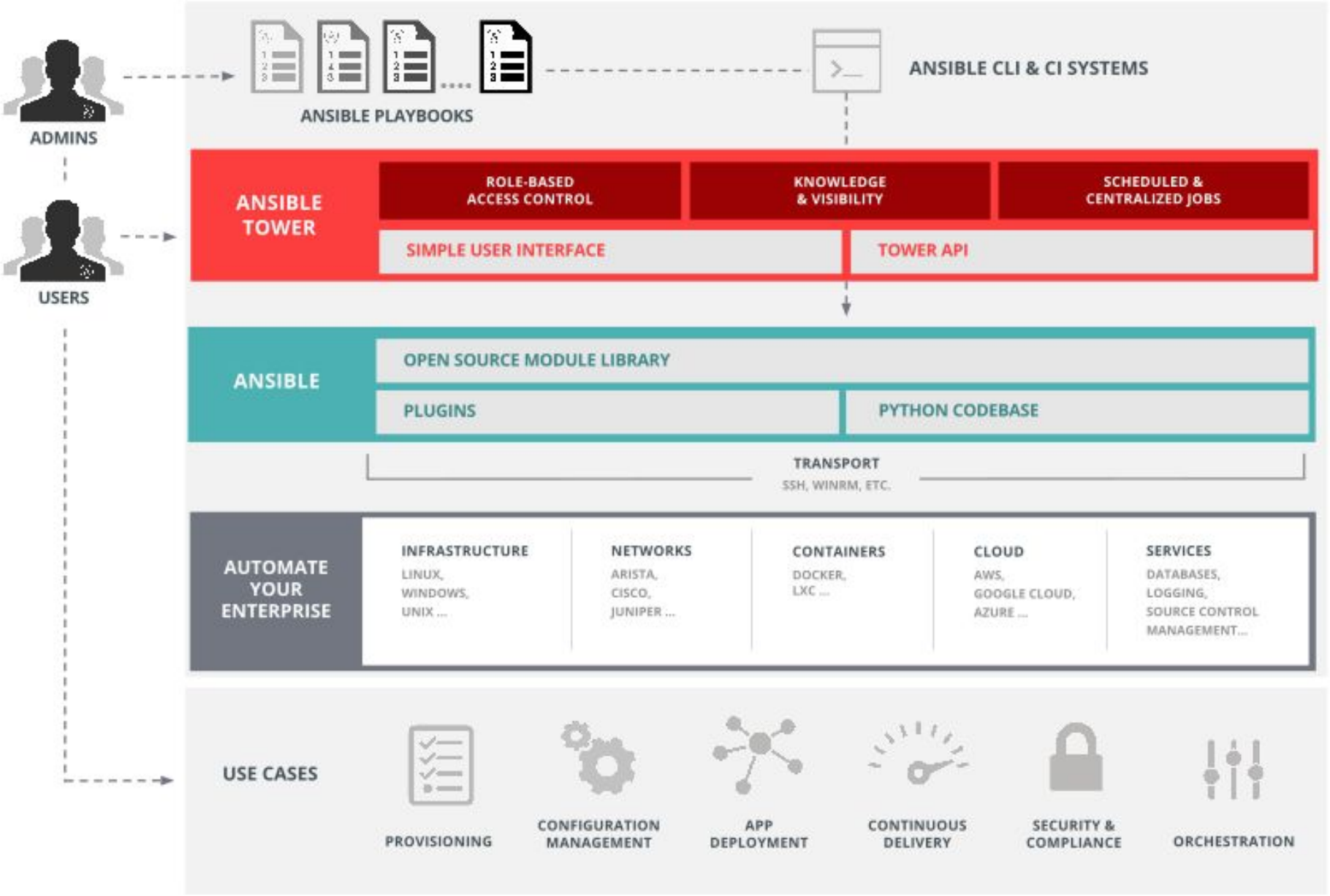
Installing Red Hat Ansible Tower

- Downloads a compressed archive file named something like:
  - **ansible-tower-setup-bundle-3.6.2-1.tar.gz**
- Extract the archive file on your Linux system:
  - **tar zxf ansible-tower-setup-bundle-3.6.2-1.tar.gz**
- In the unpacked directory, edit the **inventory** file
  - Set your admin_password (will be used for the “admin” user to log in)
  - Set the pg_password and rabbitmq_password
- Run the install script in the unpacked directory
  - **./setup.sh**

Red Hat Ansible Tower Dashboard



Red Hat



Red Hat

Upcoming Steps

- Setting up the Git repository and an editor
- Preparing the Windows hosts for automation with Ansible
- Configuring Red Hat Ansible Tower with a list of hosts to manage (the inventory)
- Configuring Red Hat Ansible Tower with a playbook to run

Managing Files in Git with Visual Studio Code

Objectives

- Explain why Ansible Playbooks should be stored under version control
- Explain the basic concepts behind the Git version control system
- Install and use Visual Studio Code to store files in a Git repository

Introducing Git

- Git is a distributed version control system.
- It allows users to manage changes to files in a collaborative manner.
- Each version of a file is committed to the system.
- Old versions of files can be restored, and a log of who made the changes is maintained.

Benefits of Git for Ansible

- Review and restore older revisions of Ansible Playbooks
- Compare two revisions of the same playbook to identify what changed
- Each change is logged: includes who did it, when, and message about why
- Editors can modify playbooks and other files, resolve conflicting changes, merge changes together
- Can make it easier to manage production and testing versions of files
  - Can use the latest version of a playbook in a particular branch
  - Can use a specific revision of a playbook

Preparing Your Git Repository

- We will assume you have a network-accessible Git repository
- For this course you can use a public repository (in practice, consider security needs)
  - https://github.com
  - https://gitlab.com
  - https://bitbucket.org
  - or any other convenient Git service
- You will need read-write access to the repository and the repository's URL

Introducing Visual Studio Code

- You can use a number of methods to work with files in Git
- In this course, we will use Visual Studio Code
- https://code.visualstudio.com/
- Visual Studio Code is a multiplatform, lightweight Integrated Development Environment (IDE) for editing and debugging code.
- The project itself is developed as (mostly) open source under the MIT license, at https://github.com/microsoft/vscode

Installing Visual Studio Code

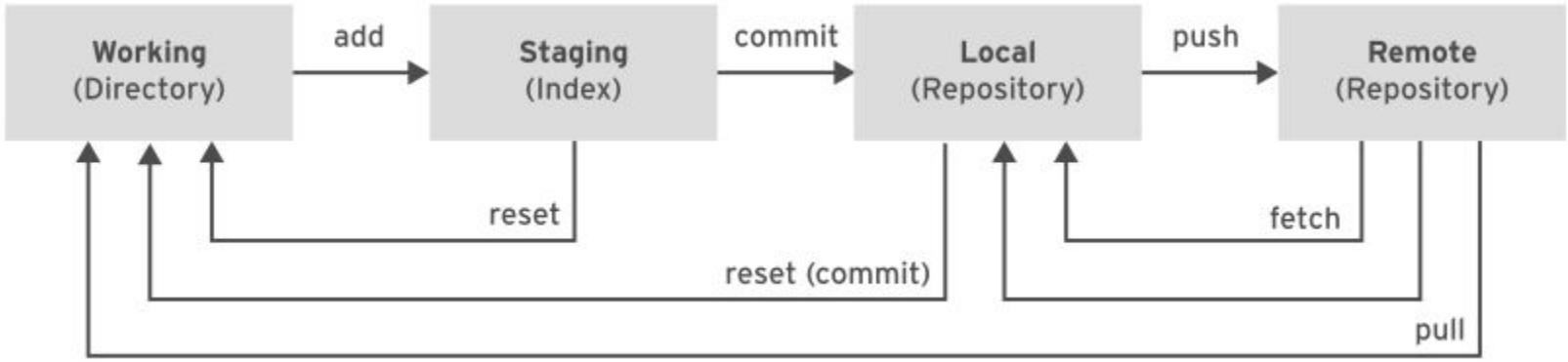
- Download and install Visual Studio Code: https://code.visualstudio.com/download
- Download and install Git for Windows: https://git-scm.com/download/win
  - Select “Use Visual Studio Code as Git’s default editor”
- After installation, run the **Git GUI** program.
  - Go to **Edit -> Options**
  - Configure the Global **User Name** and **Email Address** settings
- Run Visual Studio Code
  - Go to **View -> Extensions** and install the YAML extension from Red Hat

Getting Your Git Repository in Visual Studio Code

- Open the Command Palette with **View -> Command Palette** or by typing **Ctrl+Shift+P**
- Type **Git: Clone** in the Command Palette
- Enter the Repository URL of your Git repository
- Select the folder in which you want the Git repository to be saved locally
- Visual Studio Code's Explorer view on the left will show a list of all files in the repository

Introducing Git

- To use Git effectively, a user must be aware of the three possible states of a file in the working tree:
  - Modified - the copy of the file in the working tree has been edited and is different from the latest version in the repository.
  - Staged - the modified file has been added to a list of changed files to commit as a set, but has not yet been committed.
  - Committed - the modified file has been committed to the local repository.



Red Hat

Visual Studio Code Common Codes for Changed Files

Files which have been changed relative to the local copy of the repository will be marked with an indicator:

Indicator	Description
U (Untracked)	This file is not under version control yet.
M (Modified)	This file is tracked by git and has been modified.
A (Added)	This file has been added to the index and will be included in the next commit.
D (Deleted)	This file has been deleted. The file will no longer appear in the Explorer view, only in the Source Control view. The deletion of the file will be included in the next commit.

Storing Changes in Git

- Click on the Source Control icon at the left or press **Ctrl+Shift+G**
- Click on **+** to Stage Changes to the local repository
- Enter a commit message above STAGED CHANGES and press **Ctrl+Enter** to commit the changes to the local repository
- Push changes to the remote repository by clicking the status icons ("Synchronize Changes") next to the Git branch name at the bottom lower-left corner of the window