Chapter 5
Configuration Management with Ansible

Learning Topics

- Overview of Configuration Management
- Overview of Ansible
- Key Use Cases
- Ad-Hoc Commands
- Modules
- Playbooks
- Ansible Best Practices

Configuration Management

- Configuration is performed by executable files
  - Shell scripts or configuration files for IaC tools
- People mustn't log into servers to make adjustments
  - Except when developing the scripts
  - It leads to instability
- Code based updates are fast
  - Much faster than a human can type

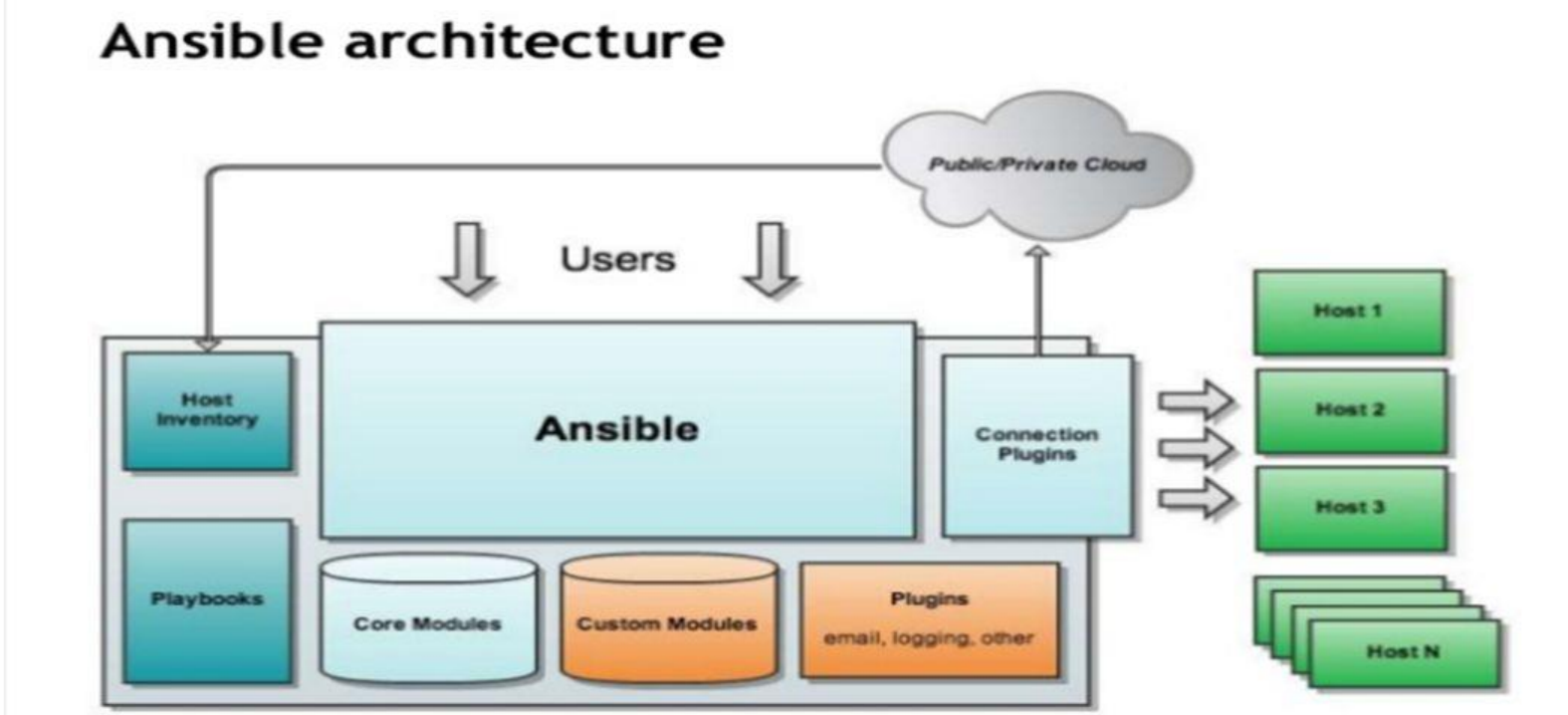
Overview of Ansible

- Open source (https://github.com/ansible/ansible)
- Written in Python
  - modules can be written in any language that can return JSON
- YAML configuration files
  - lots of small files but easy to maintain
- Uses an agentless, SSH-based, model
- Manages full code lifecycle
- "Batteries included"
  - modules are included out of the box
- Only requires one CLI invocation
- No Windows (for control box)
- Encryption capabilities out the box: easy to manage
- Idempotent: $f(x) = f(f(x))$
  - get a box into a desired state

Overview of Ansible Cont.

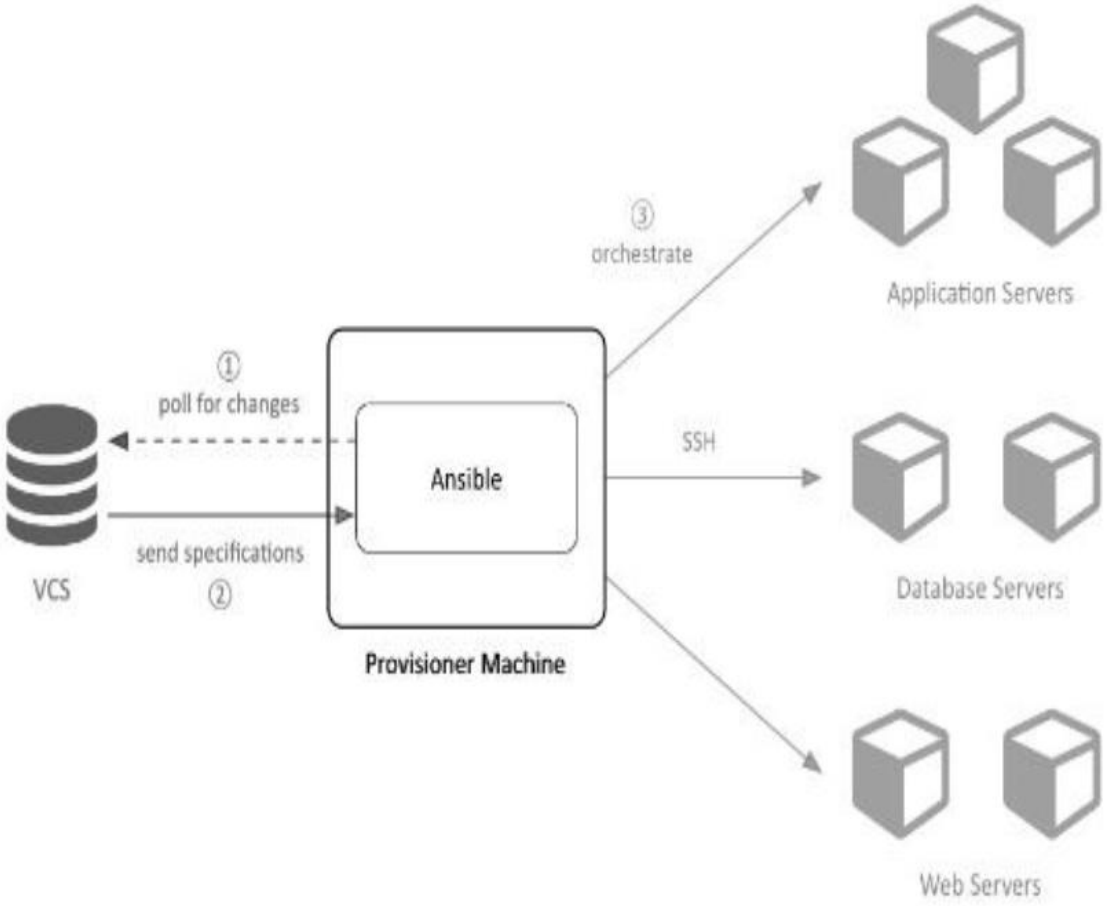
- The design goals of Ansible are:
  - Minimal in nature
  - Consistent
  - Secure
  - Highly reliable
  - Low learning curve
  - Only OpenSSH is required
  - Does not deploy agents to nodes
  - Ansible playbook can be unchanged to prevent unexpected side-effects on the managed systems
  - Playbooks use an easy and descriptive language based on YAML and Jinja templates
  - Management systems should not impose additional dependencies on the environment

Ansible Architecture



Ansible Orchestration

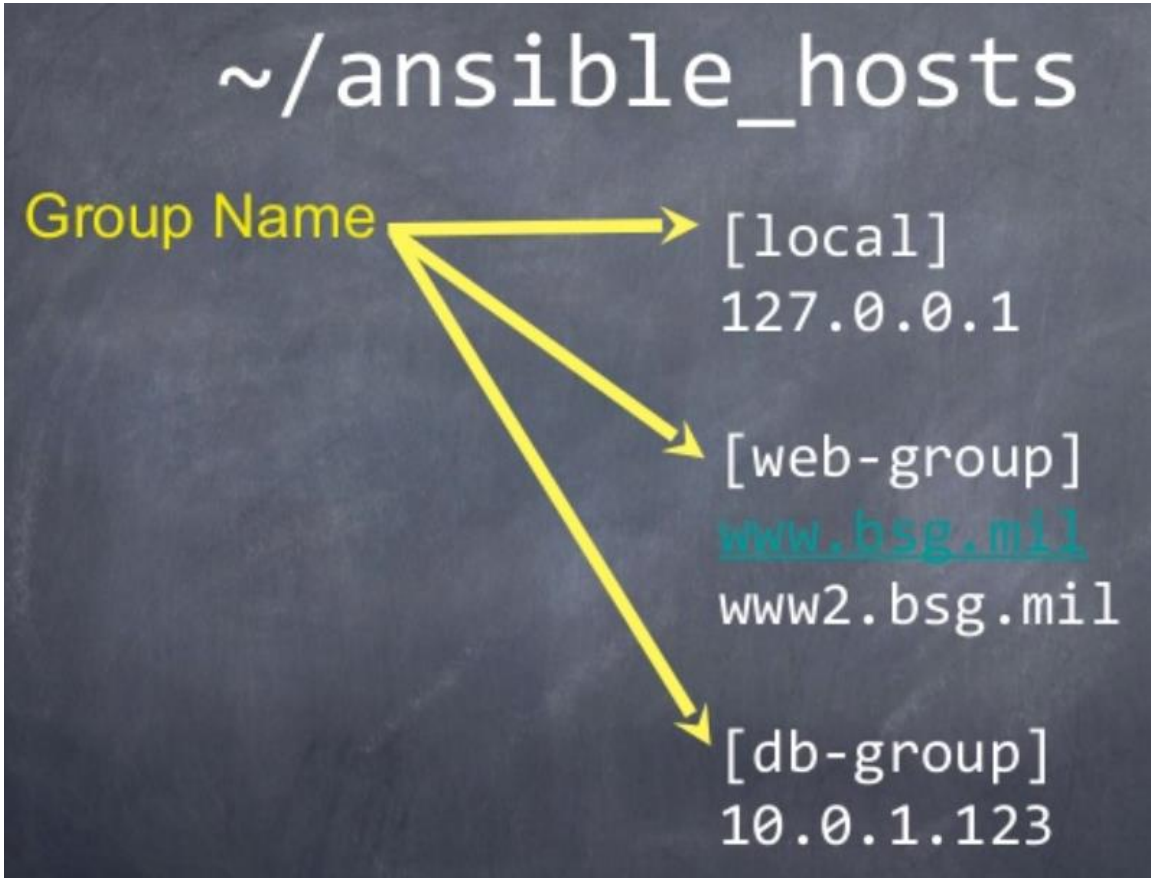
- Orchestration begins with single controlling machine
- Nodes are managed by a controlling machine over SSH
- To orchestrate nodes, Ansible deploys modules to nodes over SSH
- Modules are stored in the nodes and communicate with controlling machine through JSON protocol



Inventory

- Defines the Infrastructure
- Lists Resources to Manage
- Static Inventory – Sourced from a Text File
- Dynamic Inventory – Generated from a Script
- Individual Resources and/or Group of Resources

Inventory


~/ansible_hosts
Group Name
[local]
127.0.0.1
[web-group]
www.bsg.mil
www2.bsg.mil
[db-group]
10.0.1.123

Lab 1 : Ad-hoc Commands

- Run Miscellaneous Ad-hoc Commands

Lab 2 – Shell Module

Lab 3 – User Module

Lab 4 – APT Module

Lab 5 – File Module

Ansible Modules for:

EC2, Rackspace, Linode, OpenStack, Digital Ocean

Route53, S3, RDS

MySQL, Postgres, Riak, Mongo

Airbrake, Monit, Nagios, NewRelic, Pingdom

Netscaler, BigIP, Arista

FlowDock, HipChat, IRC, Jaber, Email

Playbook

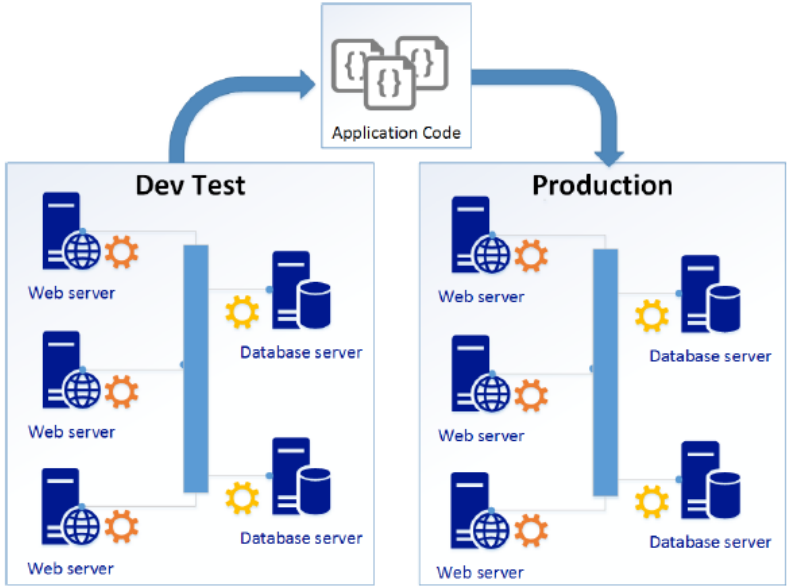
- It is a YAML Document
- Instructions to Configure Nodes
- Defines Set of Plays
- A Play is a Set of Task, Run on Hosts
- Plays bring together Inventory & Tasks

Infrastructure Management Issues

- Releasing a new service used to be complex
- Need to find or purchase hardware
- Expensive and often time consuming
- Hardware needs to be configured to support the applications
- Software installation and configuration is time consuming
- Difficult to automate

Infrastructure as a Code

- ‘Infrastructure as a code’ is a modern approach to manage infrastructure
- Infrastructure as Code (IaC) is essential to DevOps
- Computing and network infrastructure are defined in code
- This can be stored in source control systems



Lab 6 – Working with Playbooks

Ansible Best Practices

- Use Playbooks Instead of Ad-Hoc Commands
- Use Separate Inventory Files for Various Environments
- Always Mention the “State”
- Use Version Control for Your Playbooks

This concludes Chapter 5 – Configuration Management with Ansible

Let us move to Chapter 6 – Container Orchestration with Kubernetes