ANSIBLE

Ansible

Table of Contents

Course Description

Through hands-on labs, students will learn to automate system administration tasks on managed hosts with Ansible, learn how to write Ansible playbooks to standardize task execution, centrally manage playbooks and schedule recurring execution through a web interface with Ansible AWX/Tower. Students will also learn to manage encryption for Ansible with Ansible Vault, deploy Ansible Tower or AWX and use it to manage systems, and use Ansible in a DevOps environment with Vagrant.

Course Objectives

- After completing this course, students will be able to:
- Install and troubleshoot Ansible on central nodes and managed hosts
- Use Ansible to run ad-hoc commands and playbooks to automate tasks
- Write effective Ansible playbooks
- Protect encrypted data needed for tasks with Ansible Vault
- Use Ansible Tower or AWX to more easily manage enterprise Ansible deployments
- Work with Ansible in conjunction with Cloud in a DevOps environment

Audience

System and cloud administrators who need to automate cloud provisioning, configuration management, application deployment, intra-service orchestration, and other IT needs.

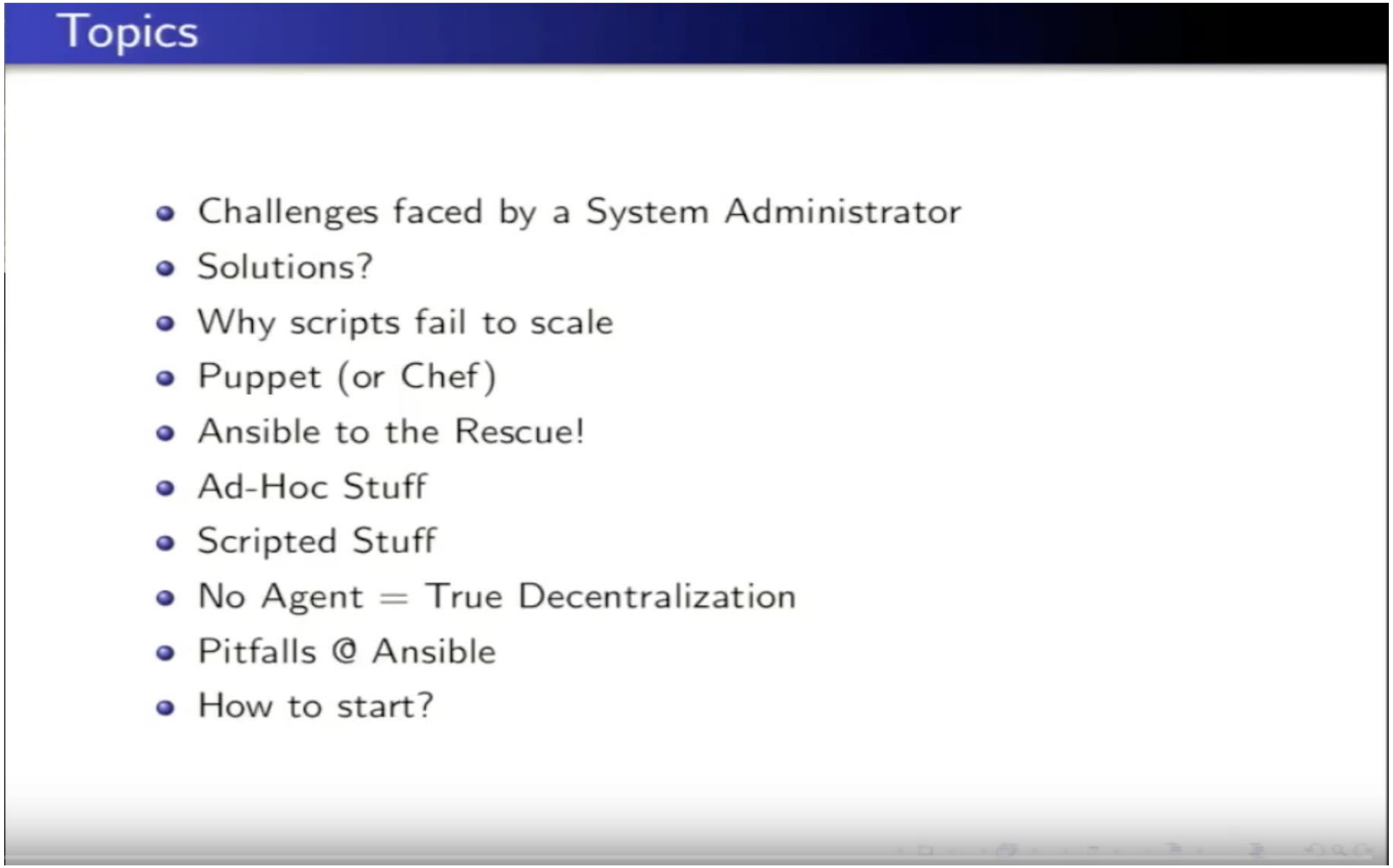
Day1

- Module 1: Course introduction
- Module 2: Introduction Ansible
- Module 3: Deploy Ansible
- Module 4: Implement playbooks

Module 1: Course introduction

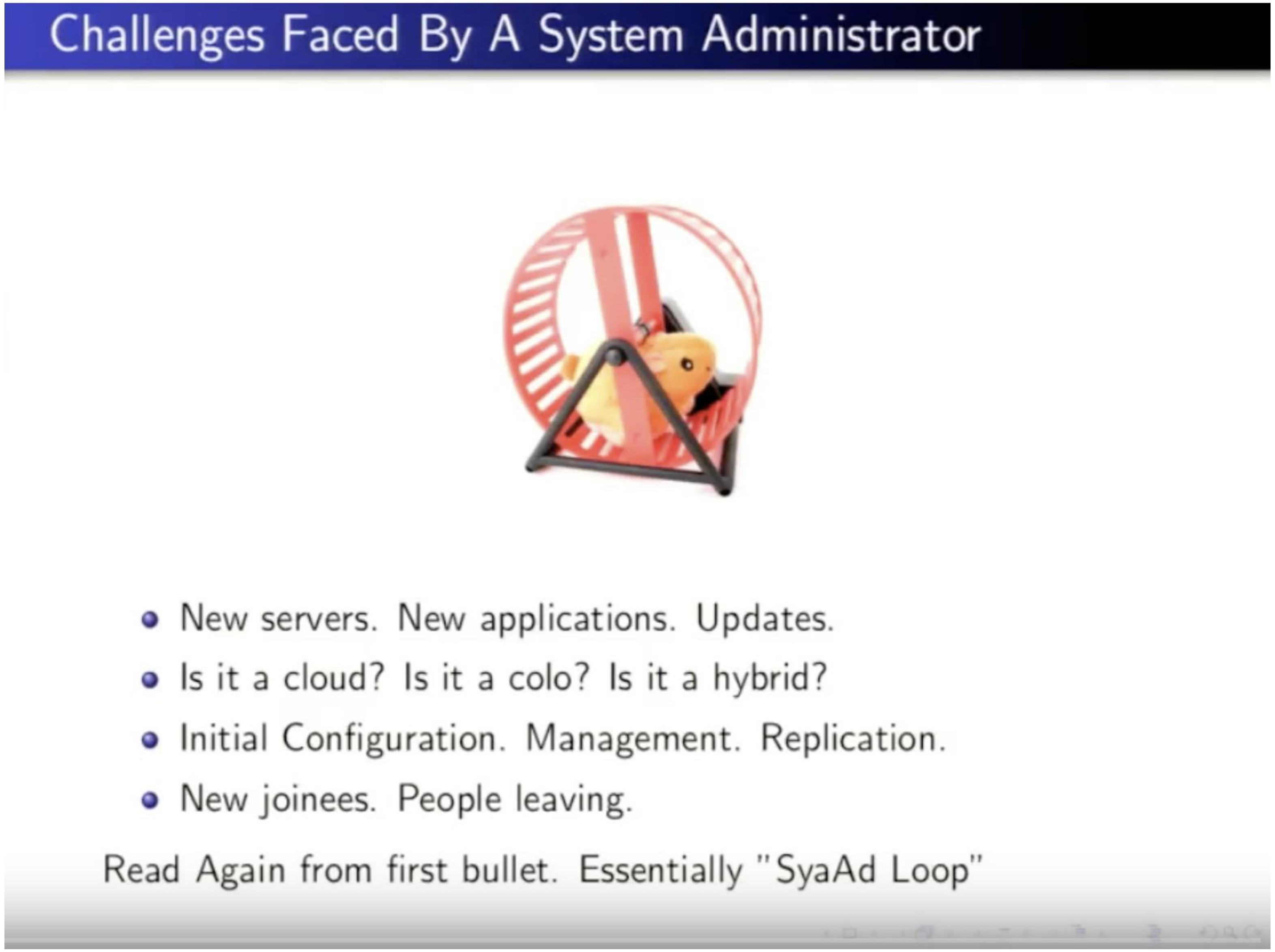
Introduce and review the course

Module 1: Course introduction



Topics

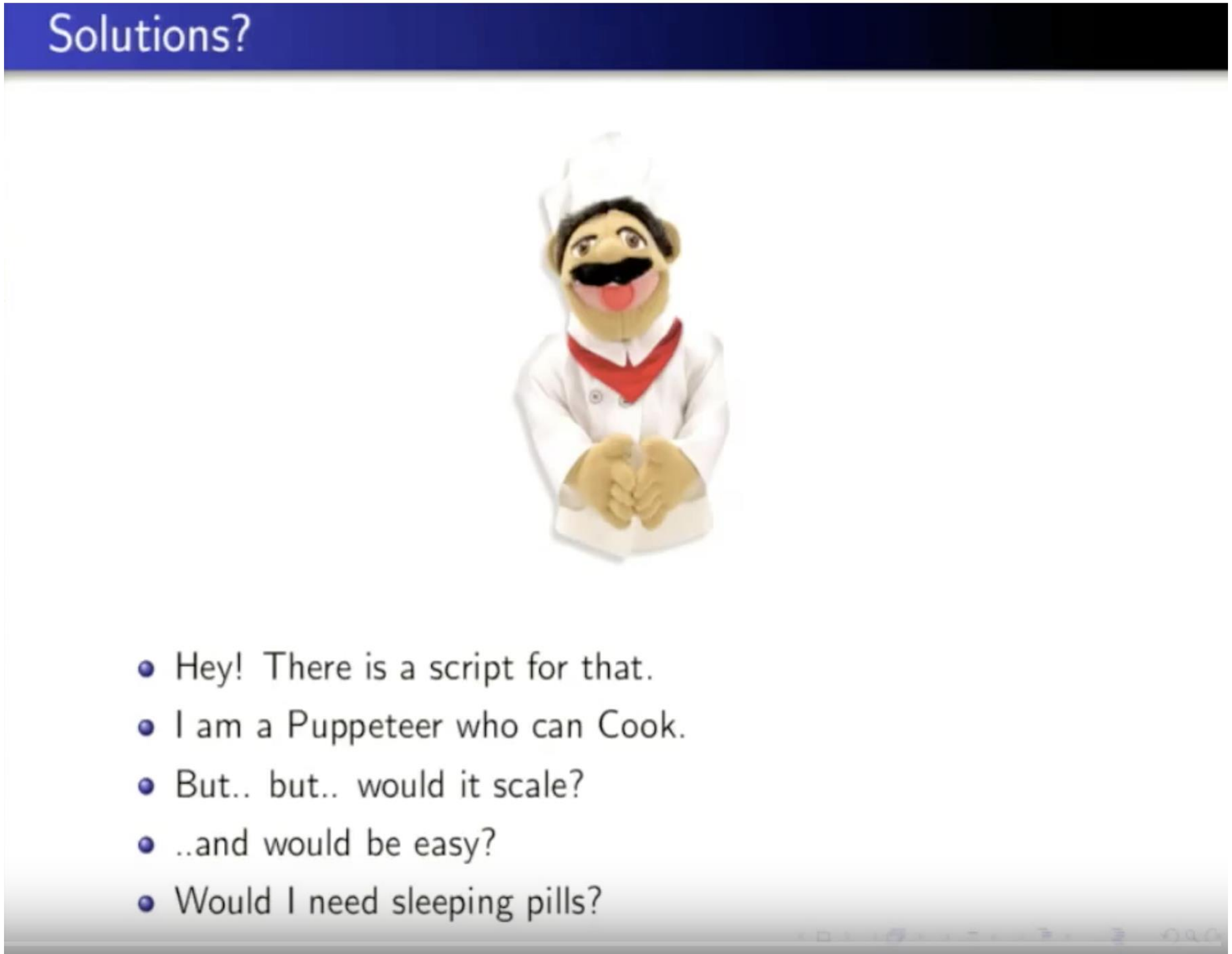
- Challenges faced by a System Administrator
- Solutions?
- Why scripts fail to scale
- Puppet (or Chef)
- Ansible to the Rescue!
- Ad-Hoc Stuff
- Scripted Stuff
- No Agent = True Decentralization
- Pitfalls @ Ansible
- How to start?



Challenges Faced By A System Administrator

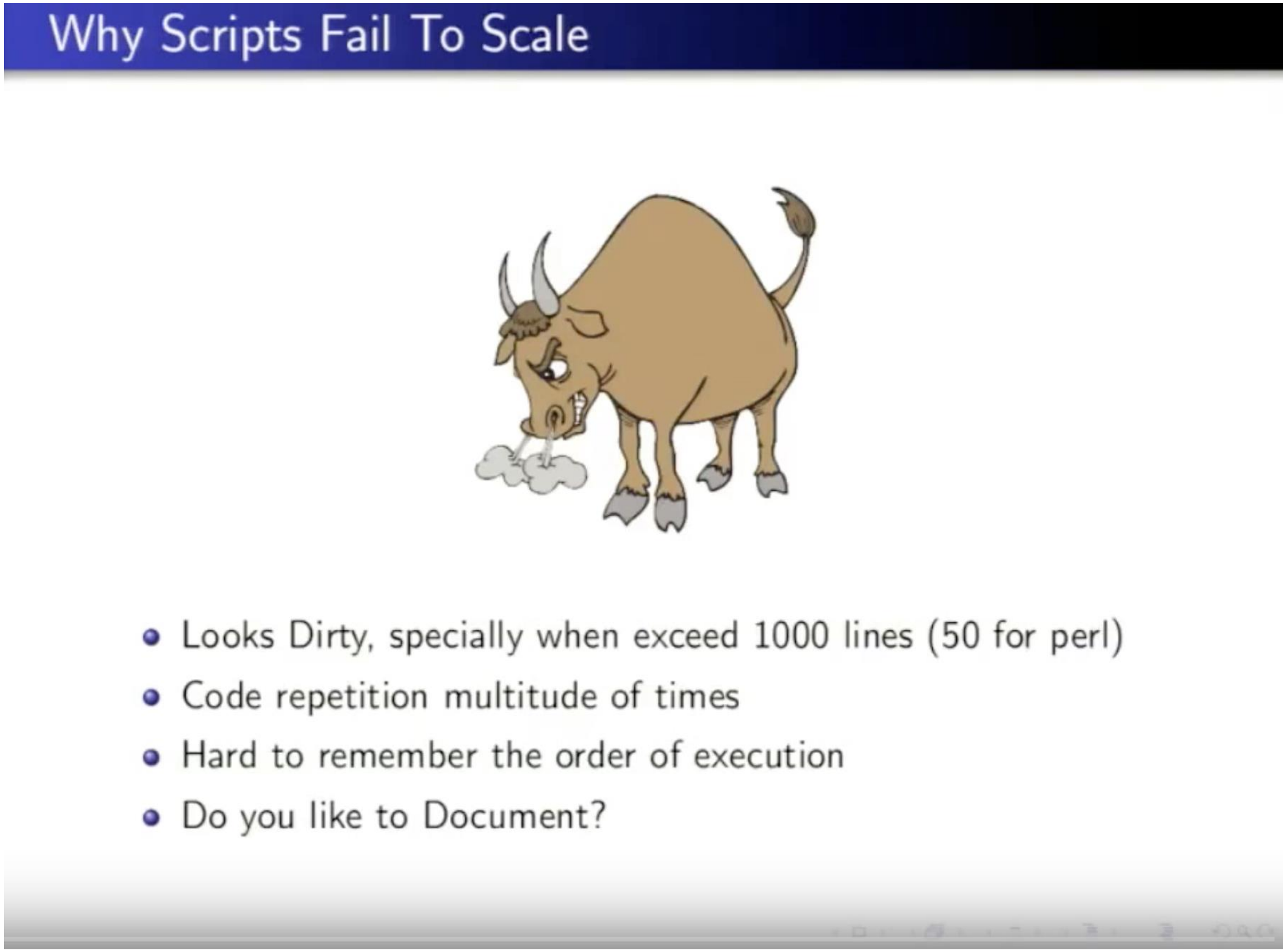
- New servers. New applications. Updates.
- Is it a cloud? Is it a colo? Is it a hybrid?
- Initial Configuration. Management. Replication.
- New joinees. People leaving.

Read Again from first bullet. Essentially "SyaAd Loop"



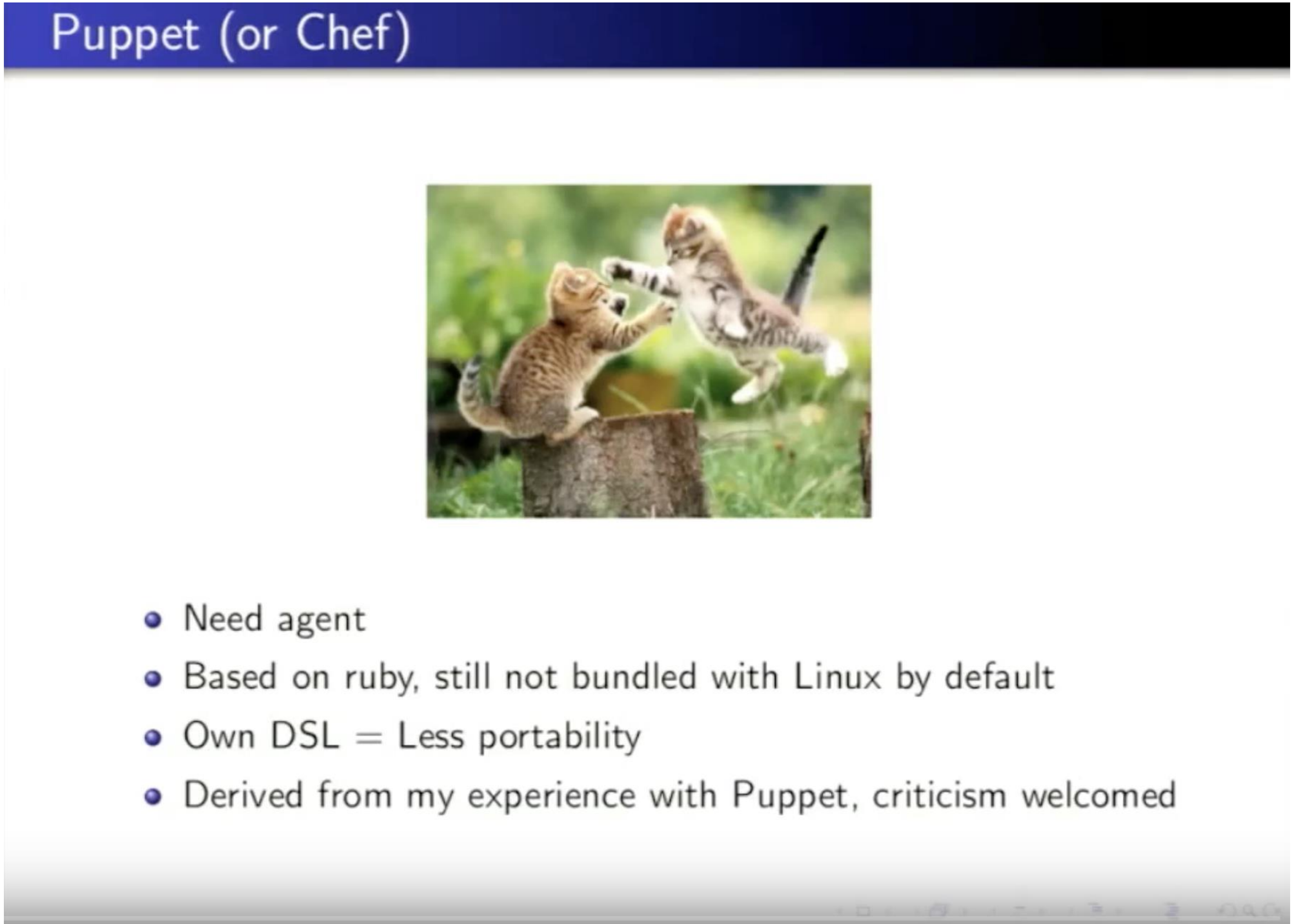
Solutions?

- Hey! There is a script for that.
- I am a Puppeteer who can Cook.
- But.. but.. would it scale?
- ..and would be easy?
- Would I need sleeping pills?



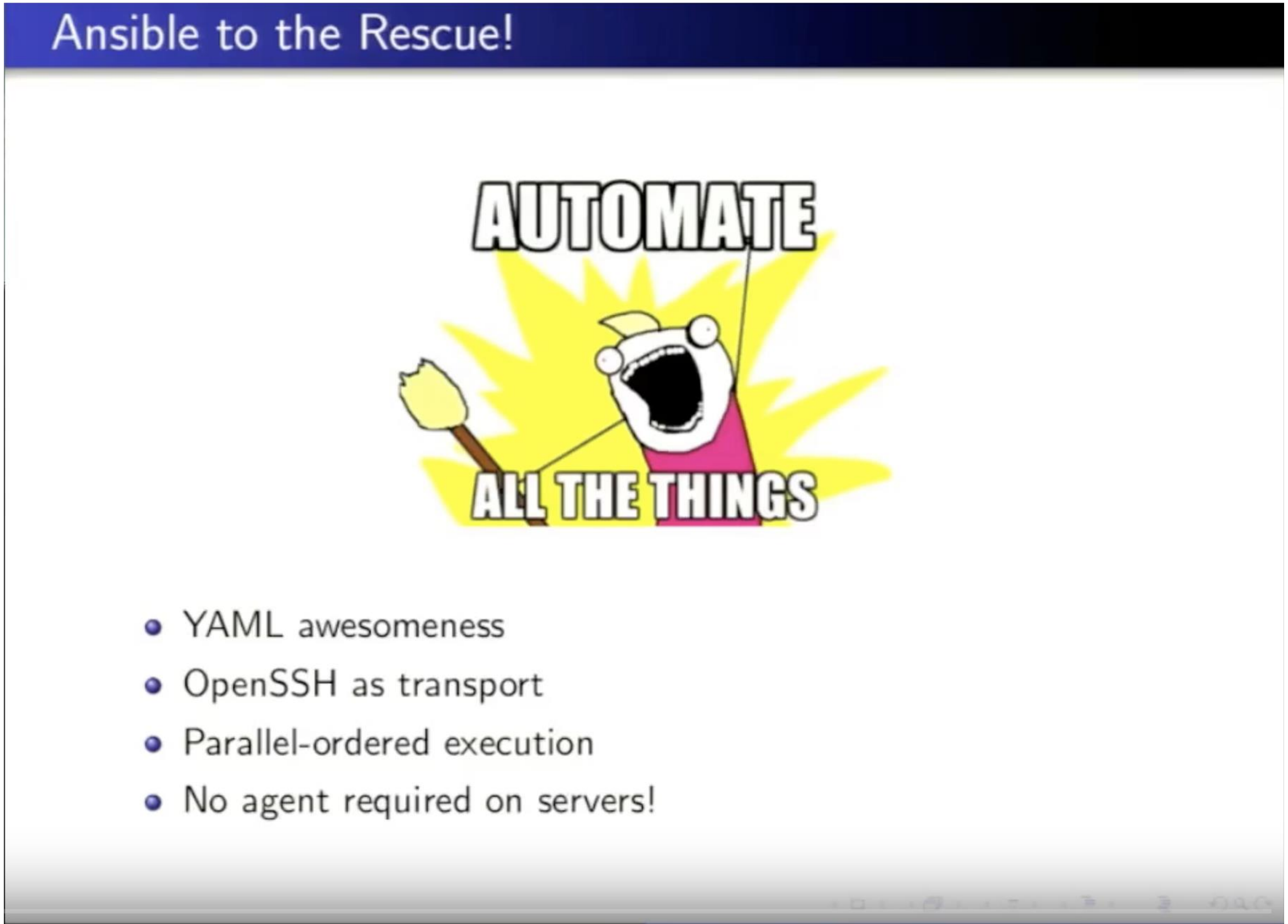
Why Scripts Fail To Scale

- Looks Dirty, specially when exceed 1000 lines (50 for perl)
- Code repetition multitude of times
- Hard to remember the order of execution
- Do you like to Document?

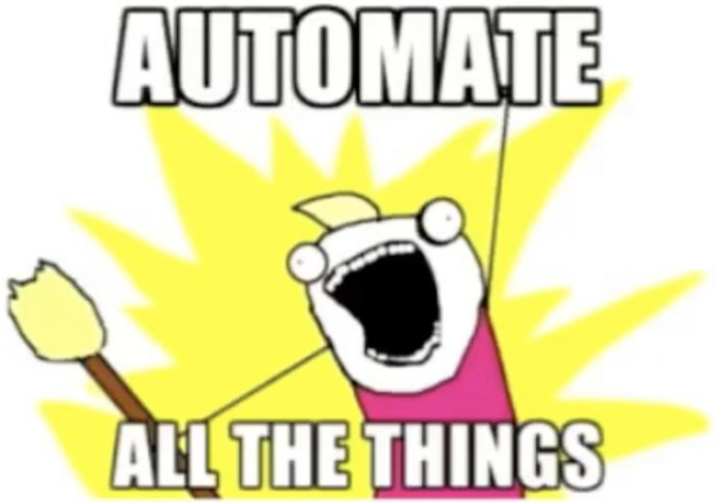


Puppet (or Chef)

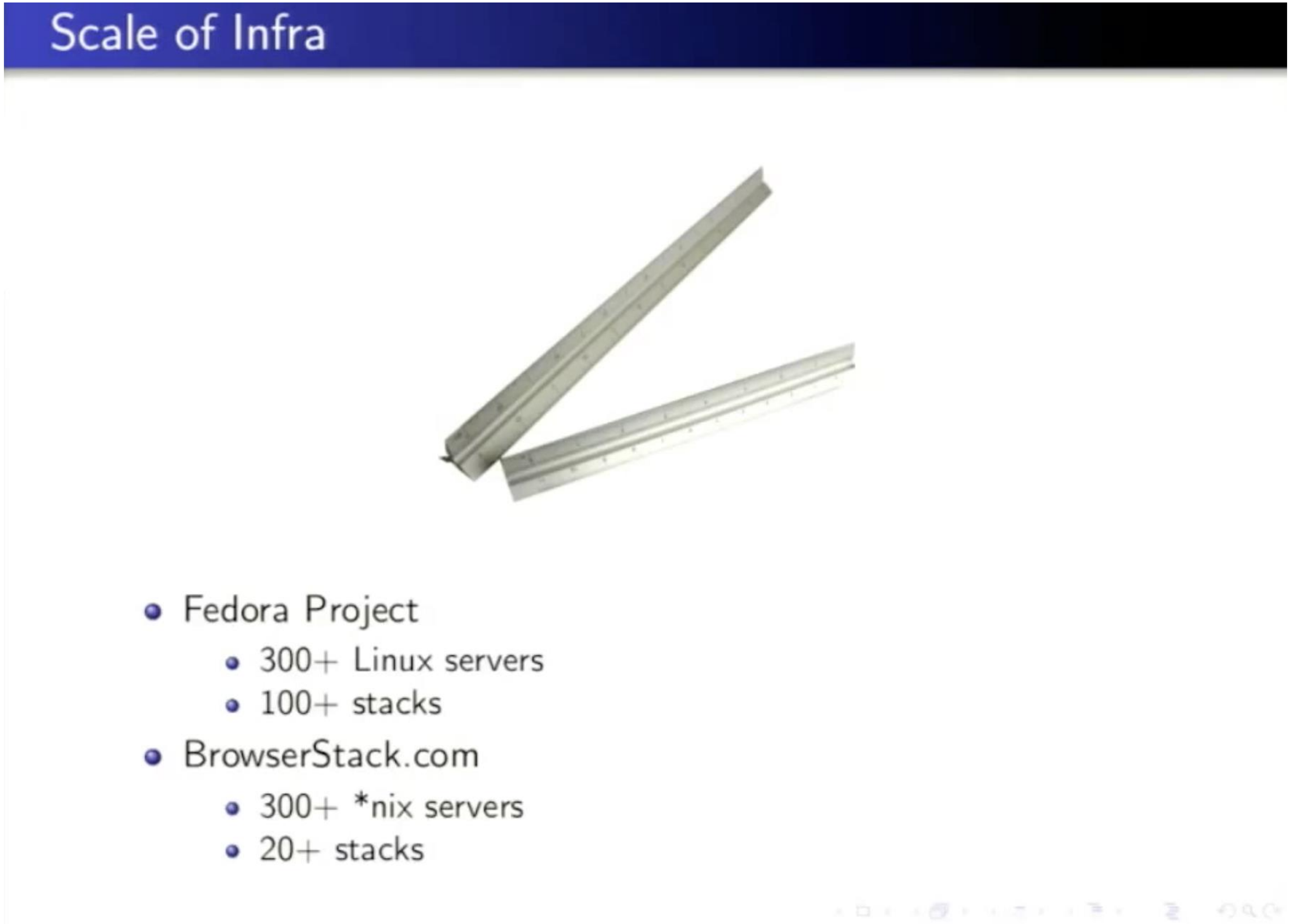
- Need agent
- Based on ruby, still not bundled with Linux by default
- Own DSL = Less portability
- Derived from my experience with Puppet, criticism welcomed



Ansible to the Rescue!



- YAML awesomeness
- OpenSSH as transport
- Parallel-ordered execution
- No agent required on servers!



Scale of Infra

- Fedora Project
  - 300+ Linux servers
  - 100+ stacks
- BrowserStack.com
  - 300+ *nix servers
  - 20+ stacks

Ansible Installation And Configuration

- yum, apt, pip
- If you can ssh, you can run Ansible
- No need to manage separate acls
- Run ad-hoc or scripted commands

Module 2: Introduction Ansible

- **Describes the terminology and architecture of Ansible**

What is Ansible?

Ansible is a radically simple IT automation engine for

- environment and infrastructure provisioning
- configuration management
- application deployment
- etc.

Why Yet Another Tool?

“I wrote Ansible because none of the existing tools fit my brain. I wanted a tool that I could not use for 6 months, come back later, and still remember how it worked.”

Michael DeHaan, Ansible Founder

“We need to do a rolling deployment of changes that have certain dependencies (including external services).
With Ansible this becomes trivial.
Puppet on the other hand feels like the Wild West.”

User **IUseRhetoric** on reddit.com

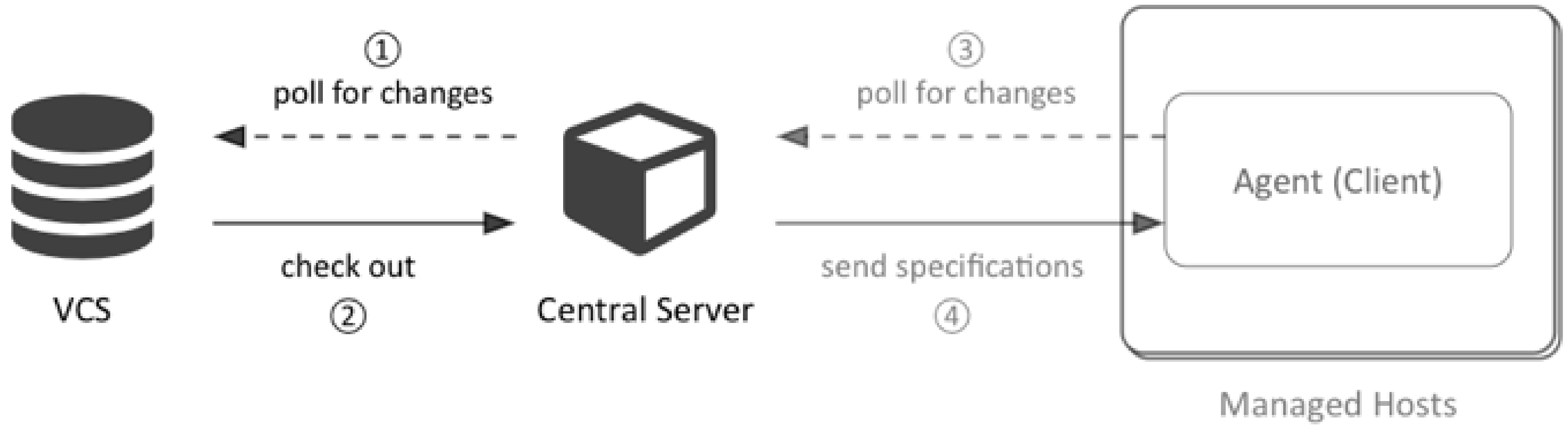
Ansible Design Principles

» No Agents ✓

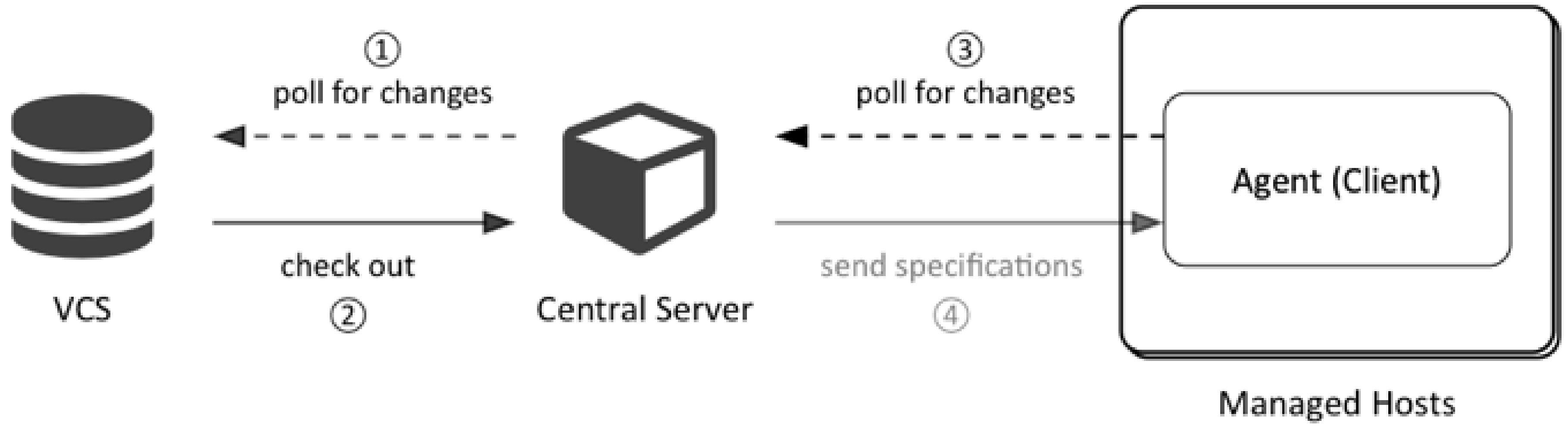
» No Scripting ✓

» Simple and Powerful ✓

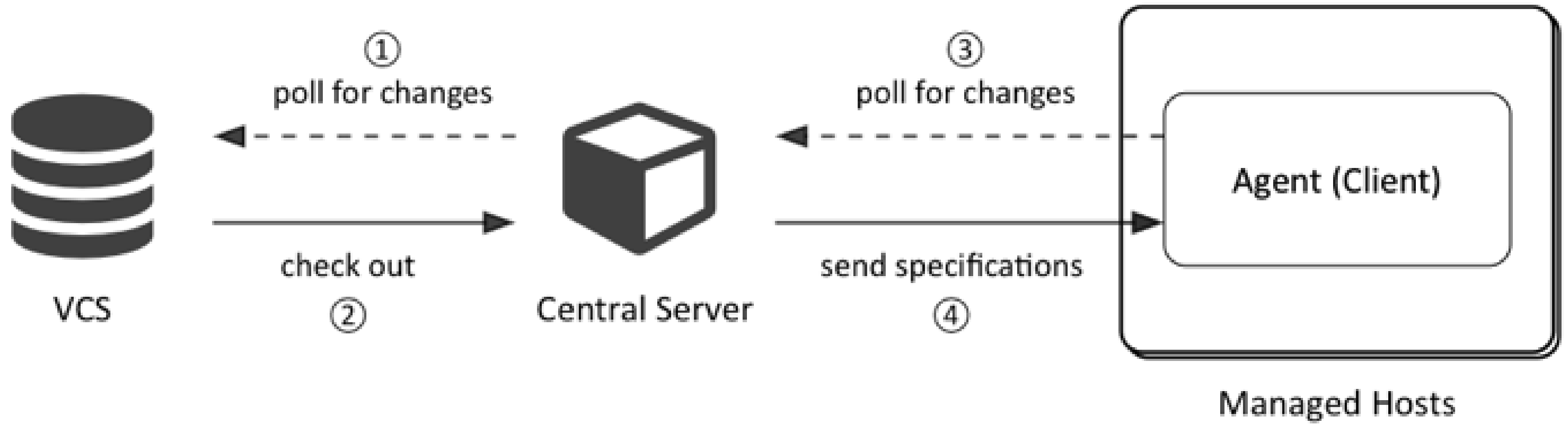
Agent-Based Architecture



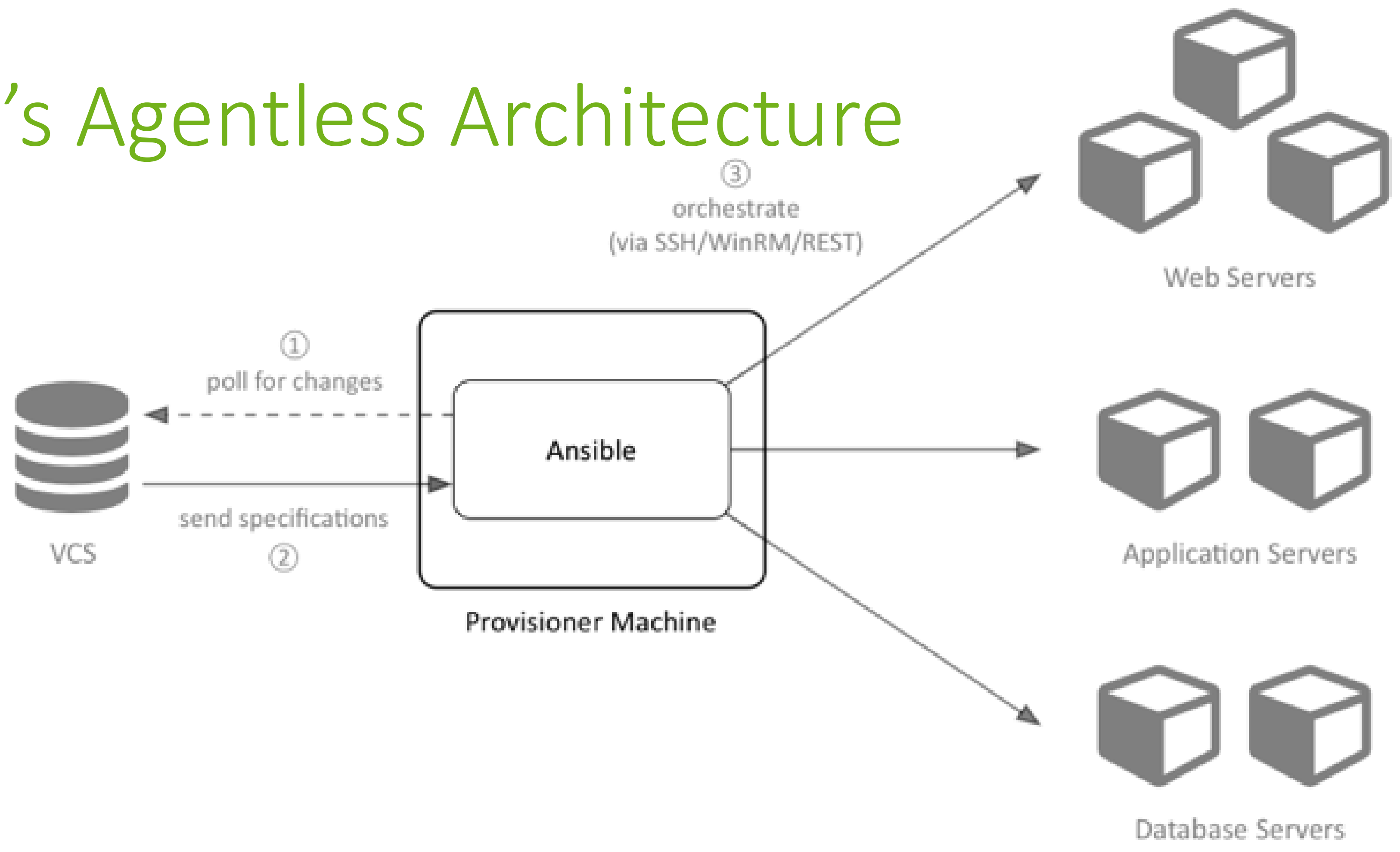
Agent-Based Architecture



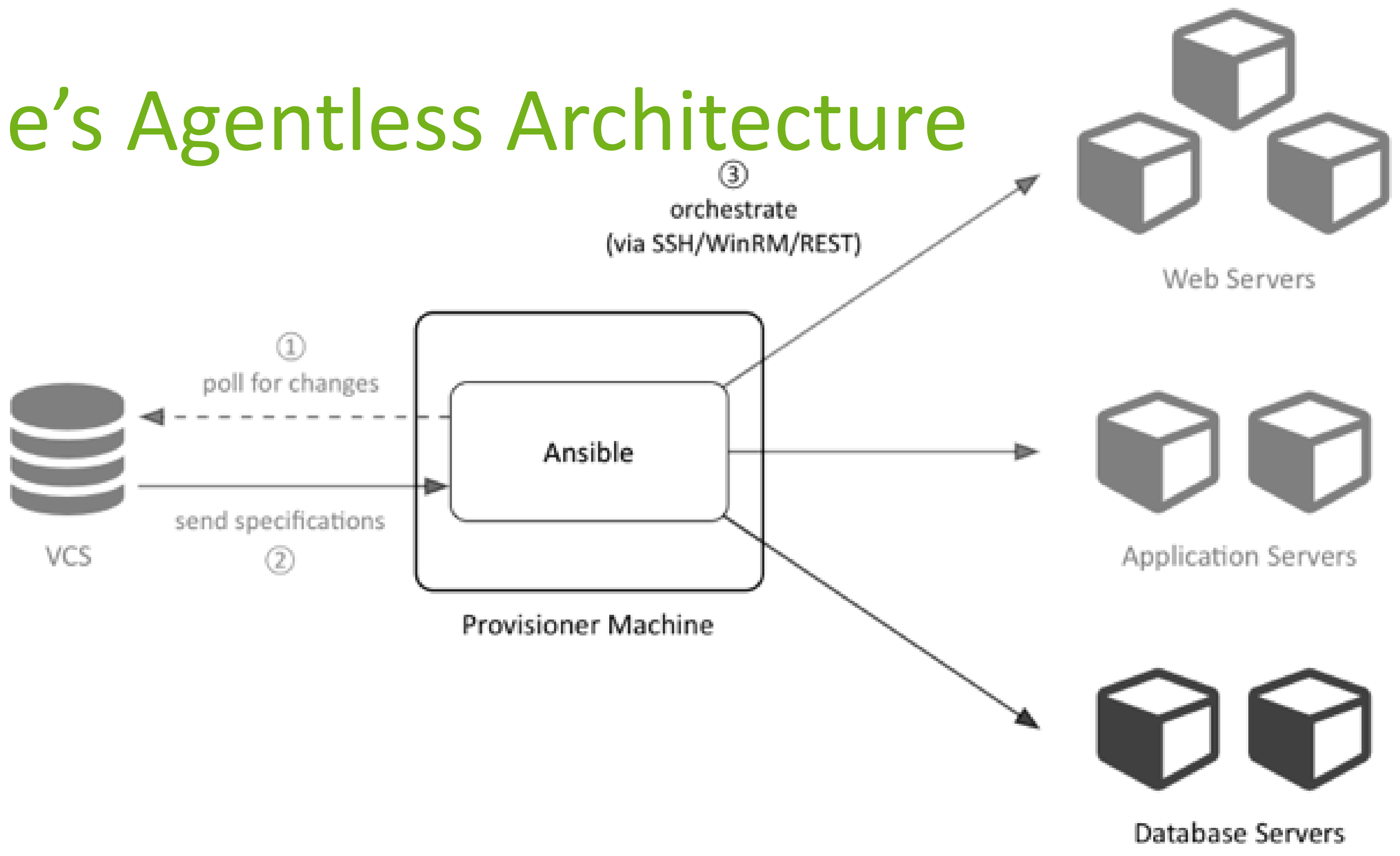
Agent-Based Architecture



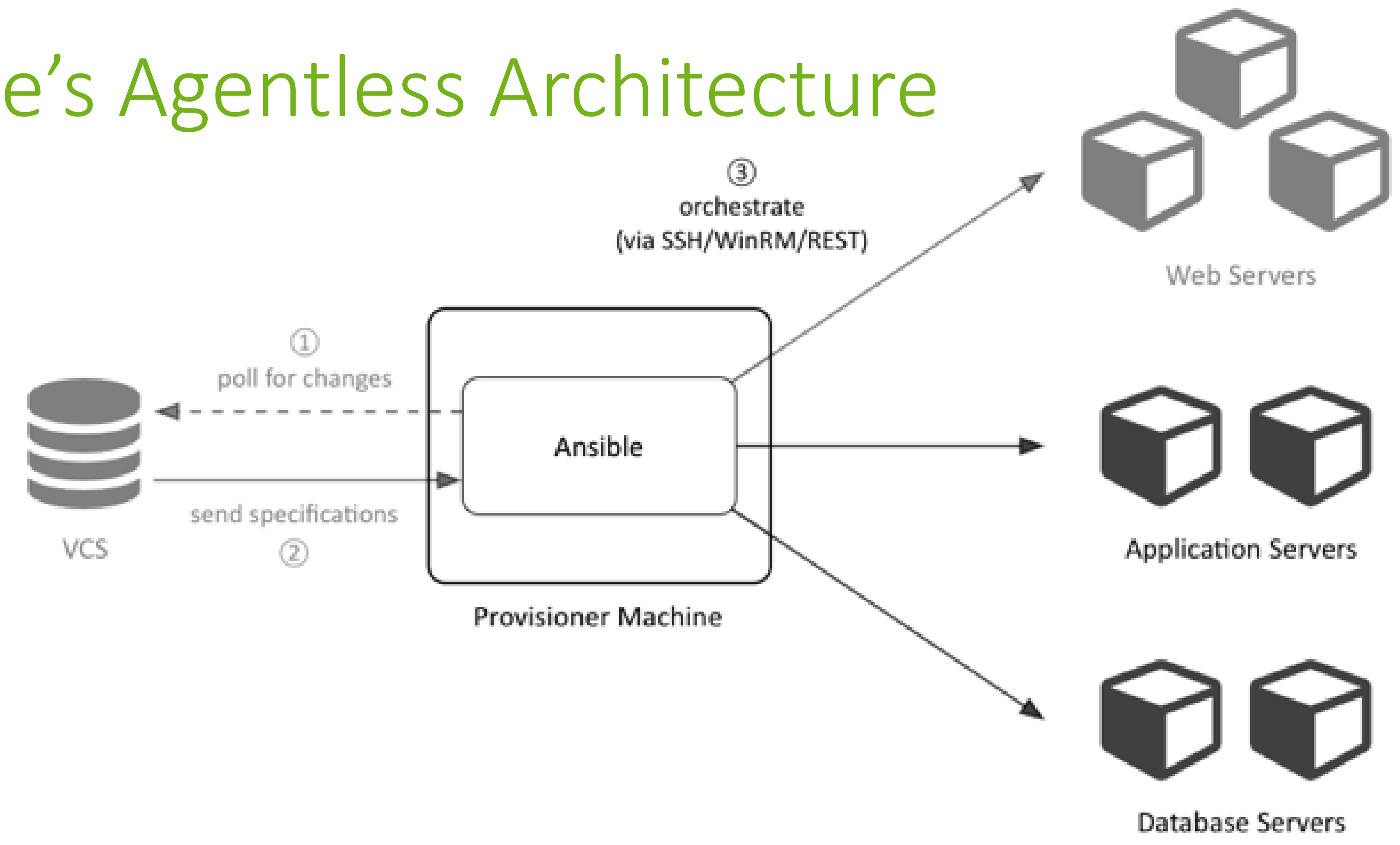
Ansible’s Agentless Architecture



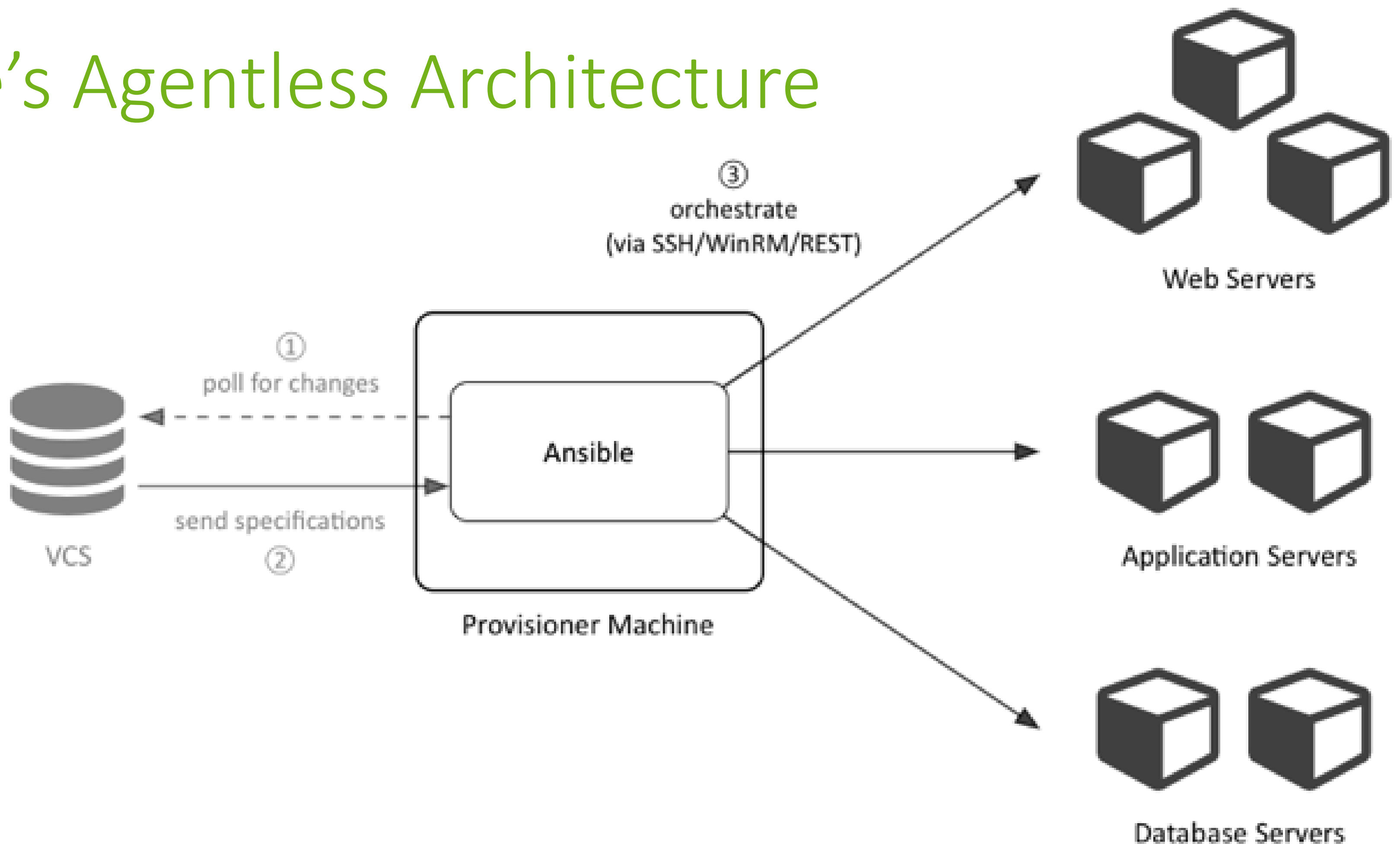
Ansible’s Agentless Architecture



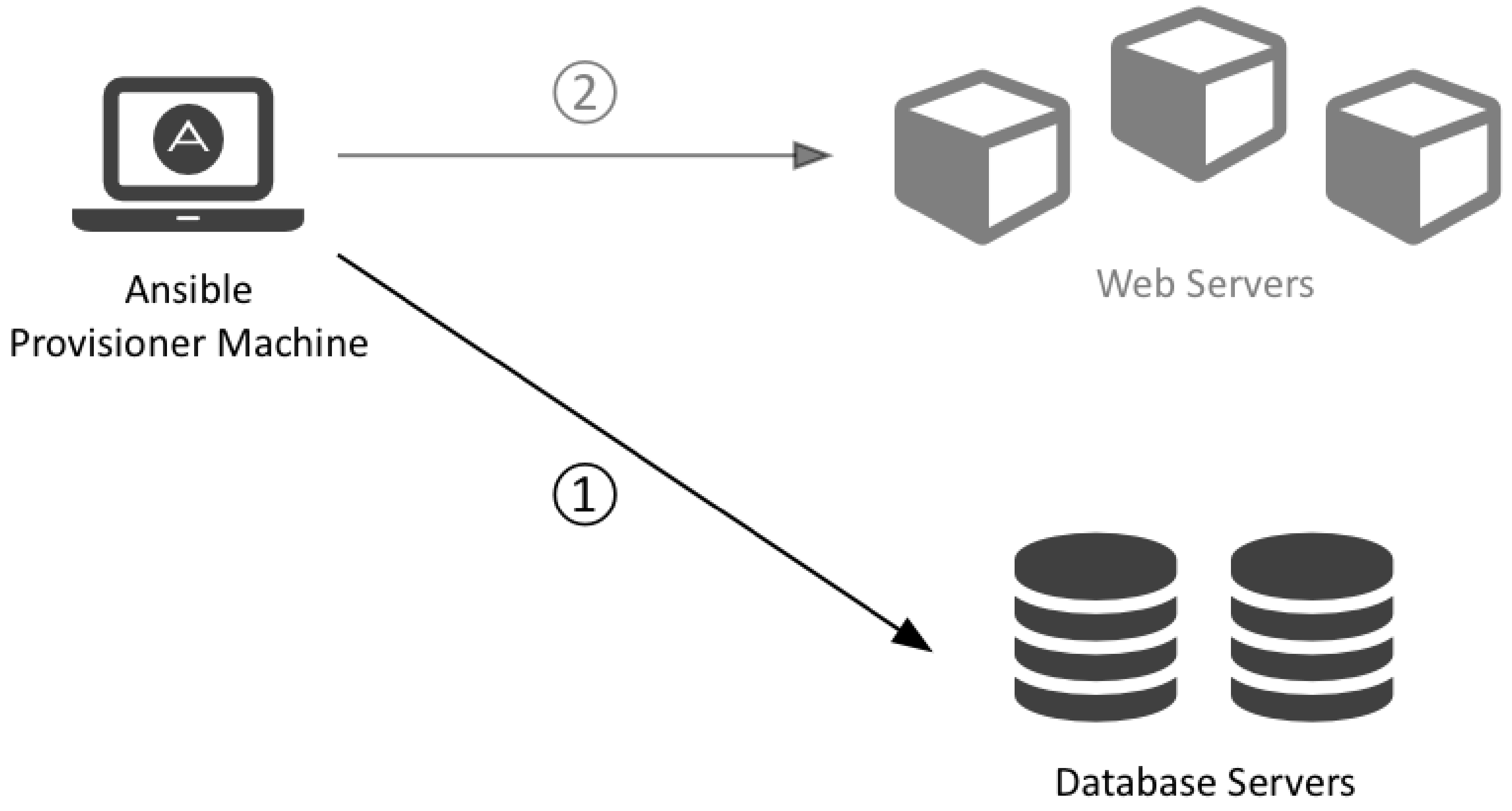
Ansible's Agentless Architecture



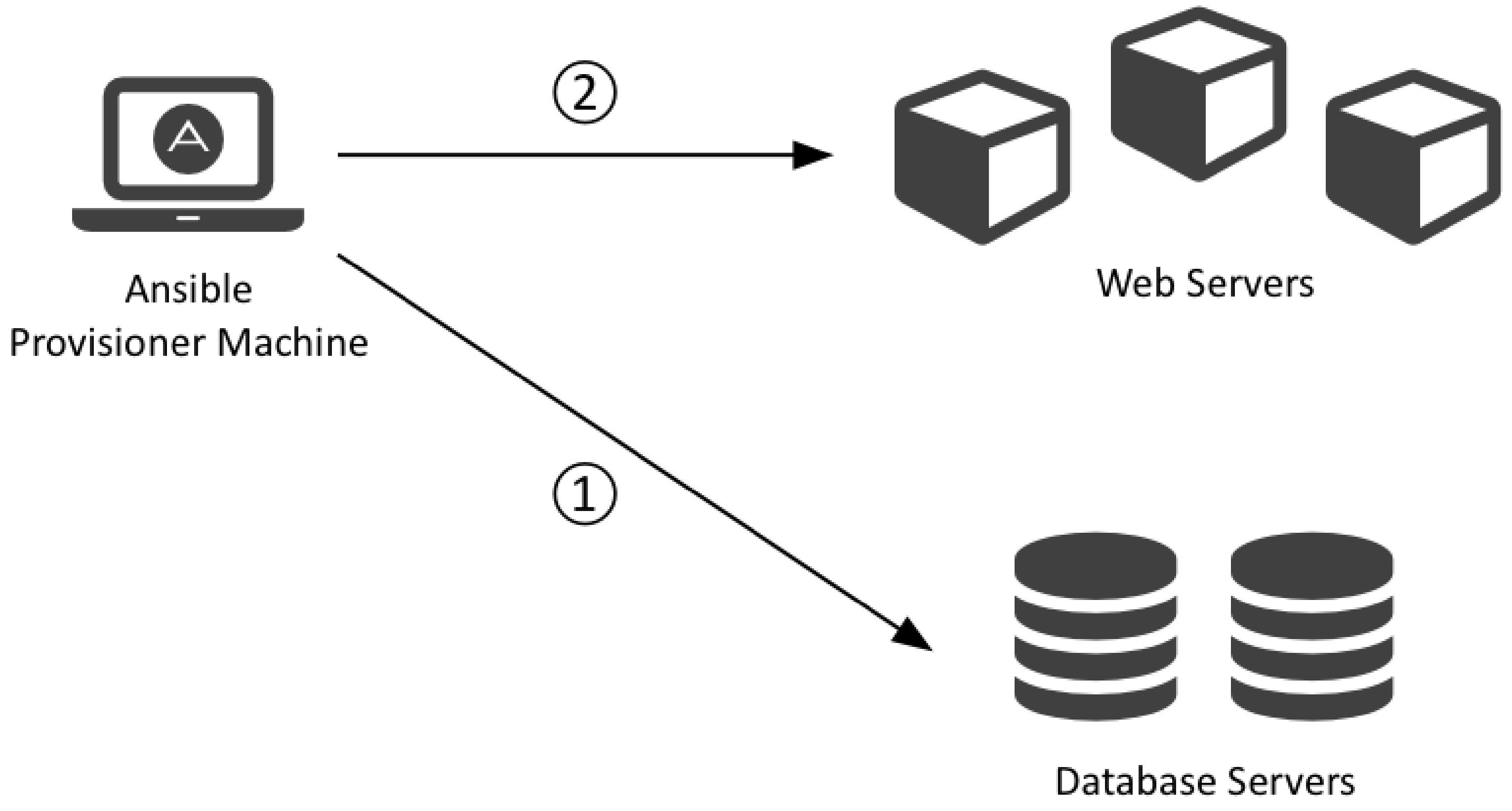
Ansible’s Agentless Architecture



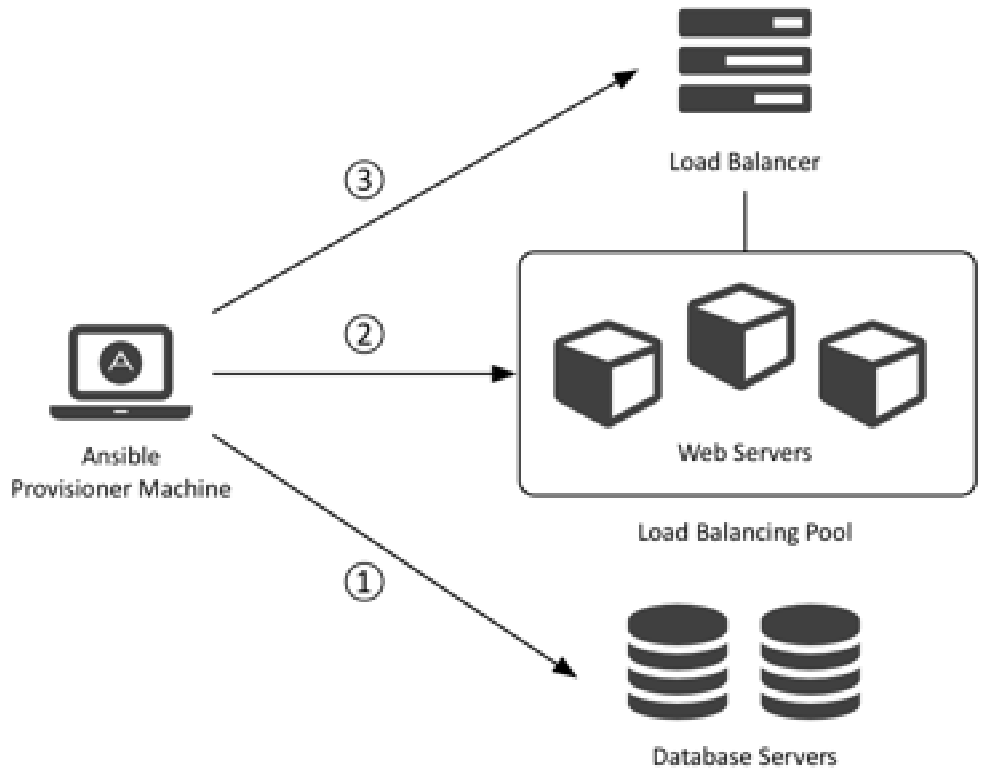
Ansible is an Orchestration Engine. So What?



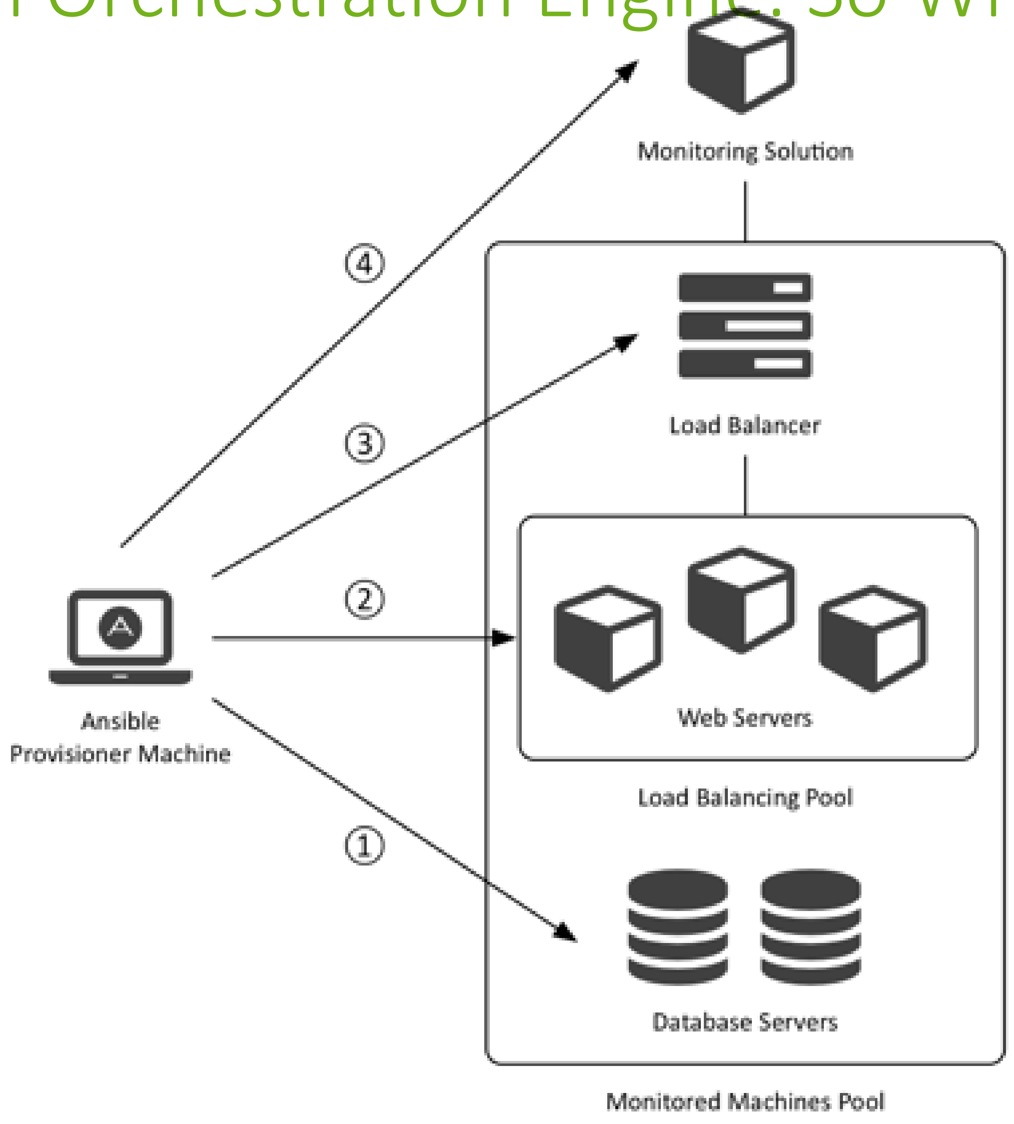
Ansible is an Orchestration Engine. So What?



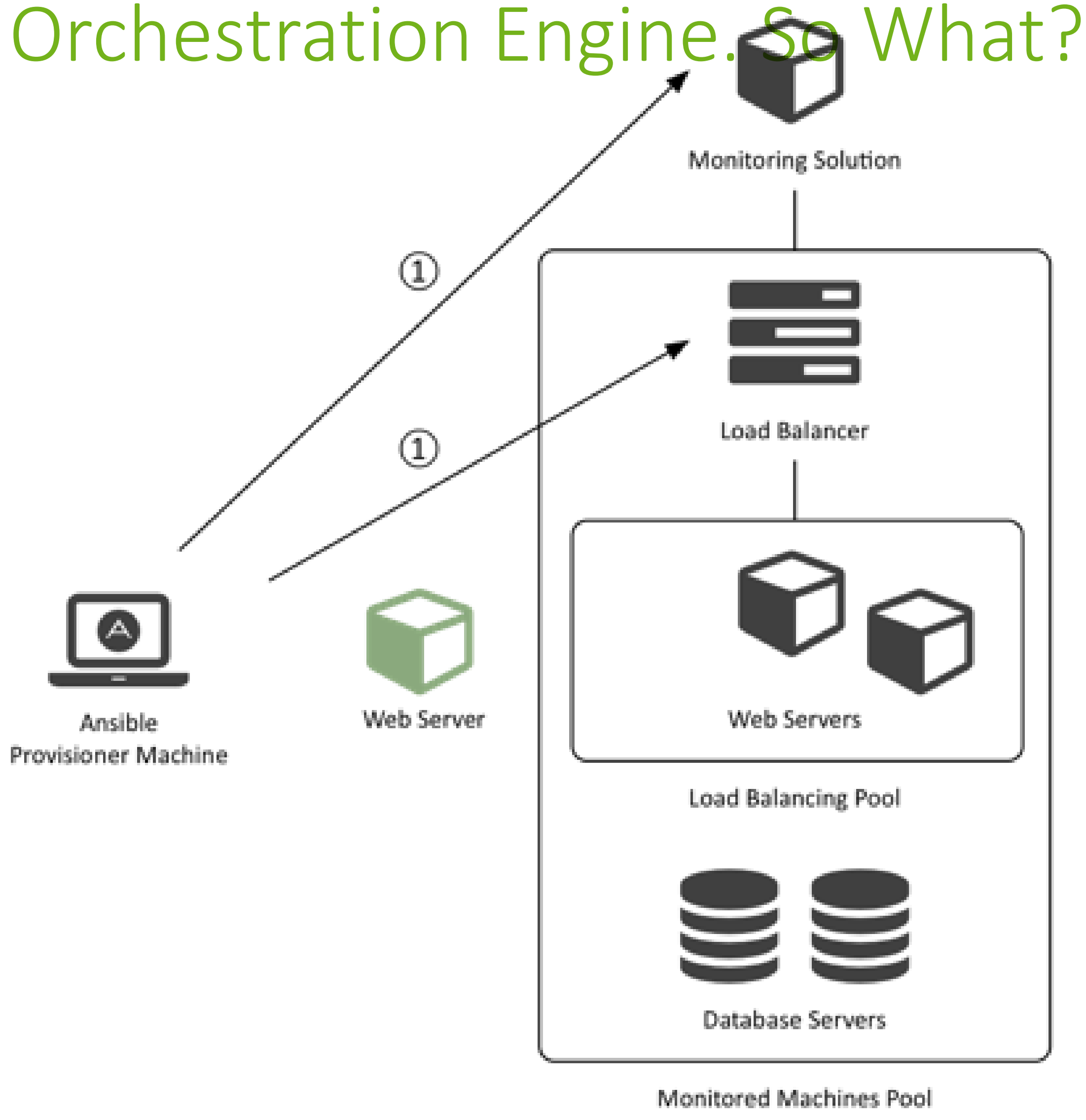
Ansible is an Orchestration Engine. So What?



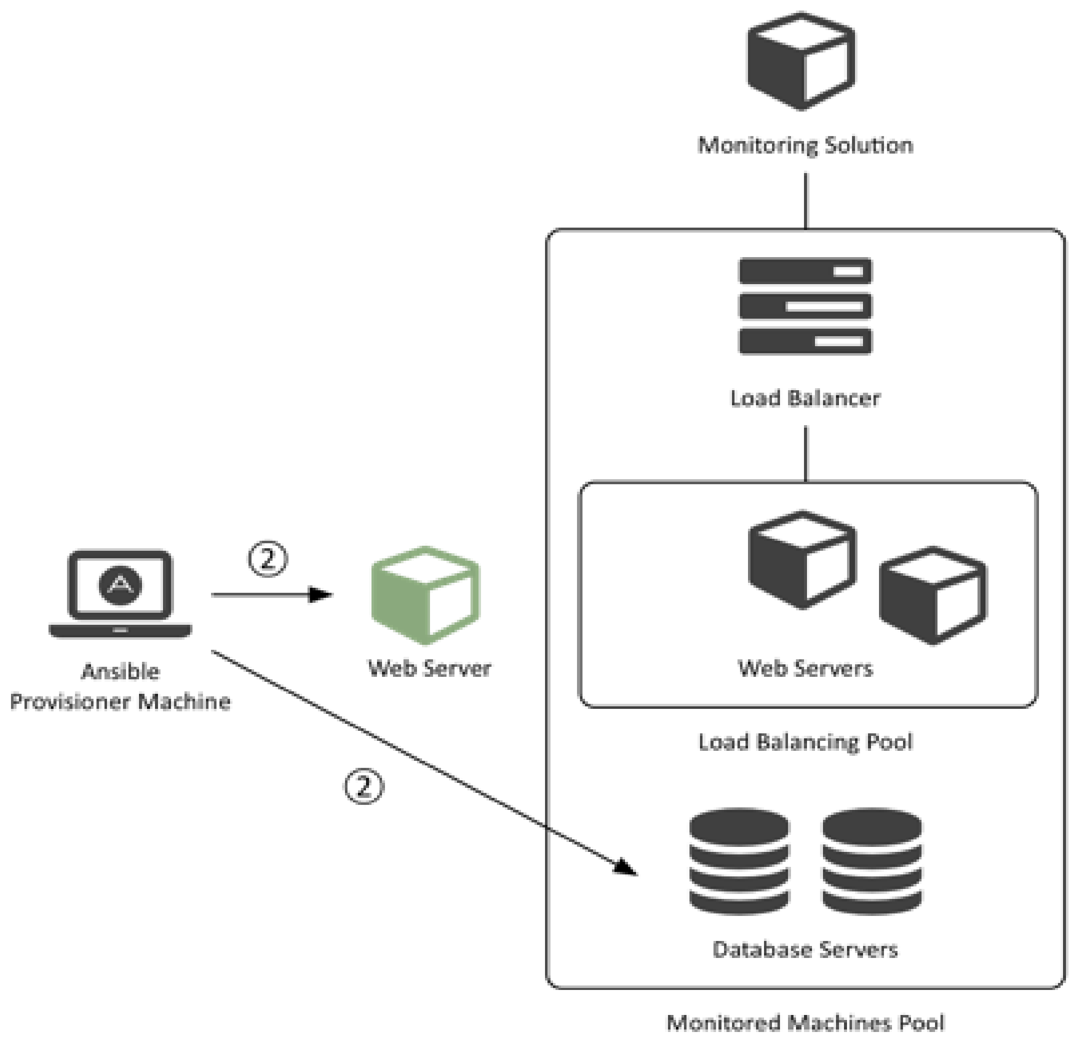
Ansible is an Orchestration Engine. So What?



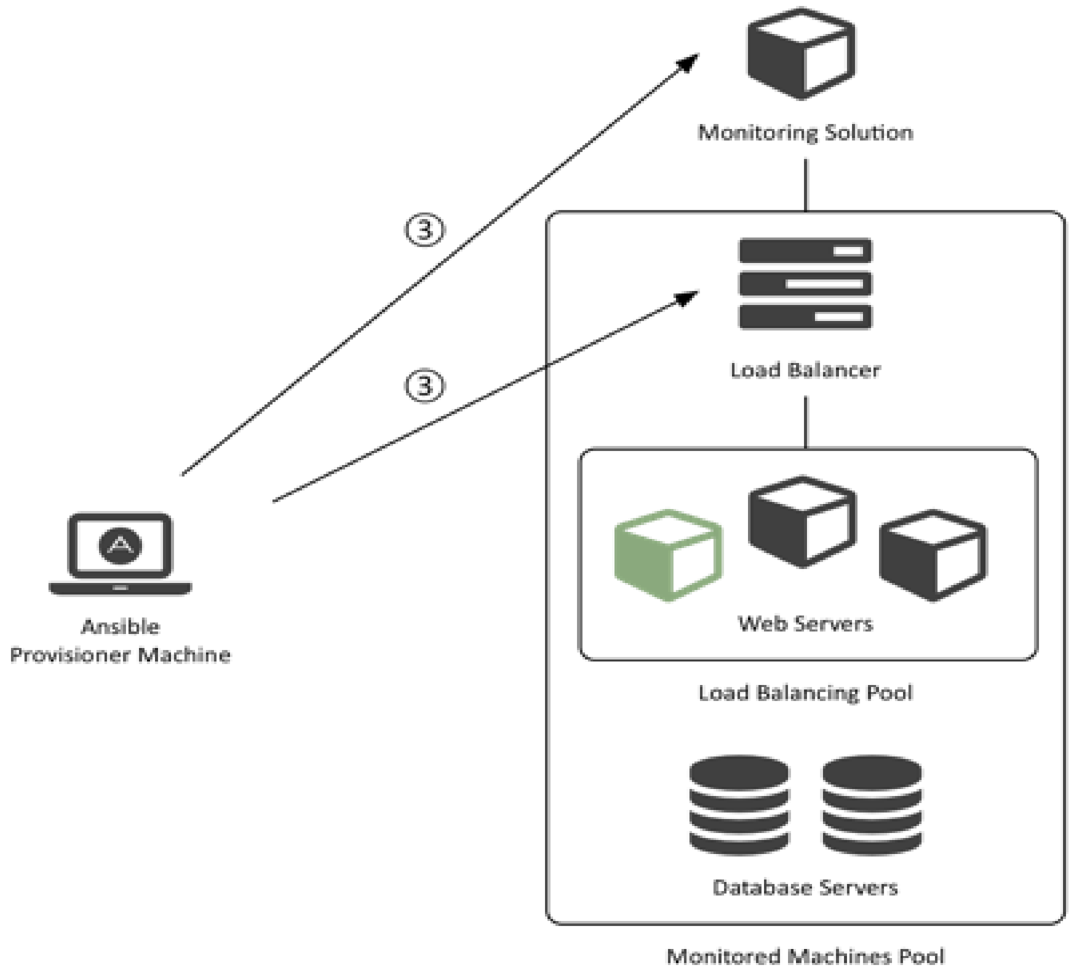
Ansible is an Orchestration Engine. So What?



Ansible is an Orchestration Engine. So What?



Ansible is an Orchestration Engine. So What?



Module 3: Deploy Ansible

- **Install Ansible and run ad hoc commands**

Module 3: Deploy Ansible

Getting started with Ansible

•Install Ansible

•RHEL/CentOS/Fedora: yum install ansible

•OS X: brew install ansible

•Windows: doesn't install directly — use a VM or cygwin

•Most platforms: pip install ansible

Set up target hosts

Ansible is most easily used if you can ssh as a normal user to a host without a password, and then sudo to root without a password.

Module 3: Deploy Ansible

Set up ssh keys

```
ssh-keygen -f ansible
ssh-add ~/.ssh/ansible
ssh-copy-id -f ~/.ssh/ansible $targethost
```

Demo: Test ping

Test with the ping module:

```
ansible -m ping target
```

Setting up sudo

```
echo "ansible_user (ALL) NOPASSWD: ALL" > \
  /etc/sudoers.d/ansible
```

In some cases you will want to lock this down further, perhaps with a password (you can enter a sudo password if you add -K to your command line).

Module 3: Deploy Ansible

Demo: Test sudo setup

```
ansible -a whoami target
ansible -a whoami -b target
```

(Note that ansible runs the command module by default so no -m is needed)

Module 3: Deploy Ansible

Basic Concepts

- Modules
- Playbooks
- Tasks
- Templates
- Handlers
- Variables
- Inventory
- Roles

Modules

•A single module allows the execution of a self-contained task.
•Modules are designed to provide an abstraction around simple and complex tasks to allow them to be repeatable and handle error conditions nicely
•We've already briefly seen the `ping` and `command` modules.

Built-in modules

•There are modules for an awful lot of things — e.g.:

- configuring services in AWS, Google, Azure, Openstack etc.
- installing OS packages
- writing to files
- updating network devices
- configuring databases
- and many others...

See the Ansible Module Index for a full list of categories.

Module 3: Deploy Ansible

Demo: ansible

Using the ansible command line utility, it's easy to run a simple module to get all of the facts from a repo

```
ansible -m setup target
```

or run an ad-hoc task

```
ansible -m file -a "state=directory path=~/throwaway" target
```

Module 3: Deploy Ansible

Demo: ansible-doc
ansible-doc is very useful for finding the syntax of a module without having to look it up online
e.g. ansible-doc mysql_user

Module 4: Implement playbooks

- **Write Ansible plays and execute a playbook**

Ansible Playbooks

$> ansible-playbook [–i <inventory>] <playbook.yml>

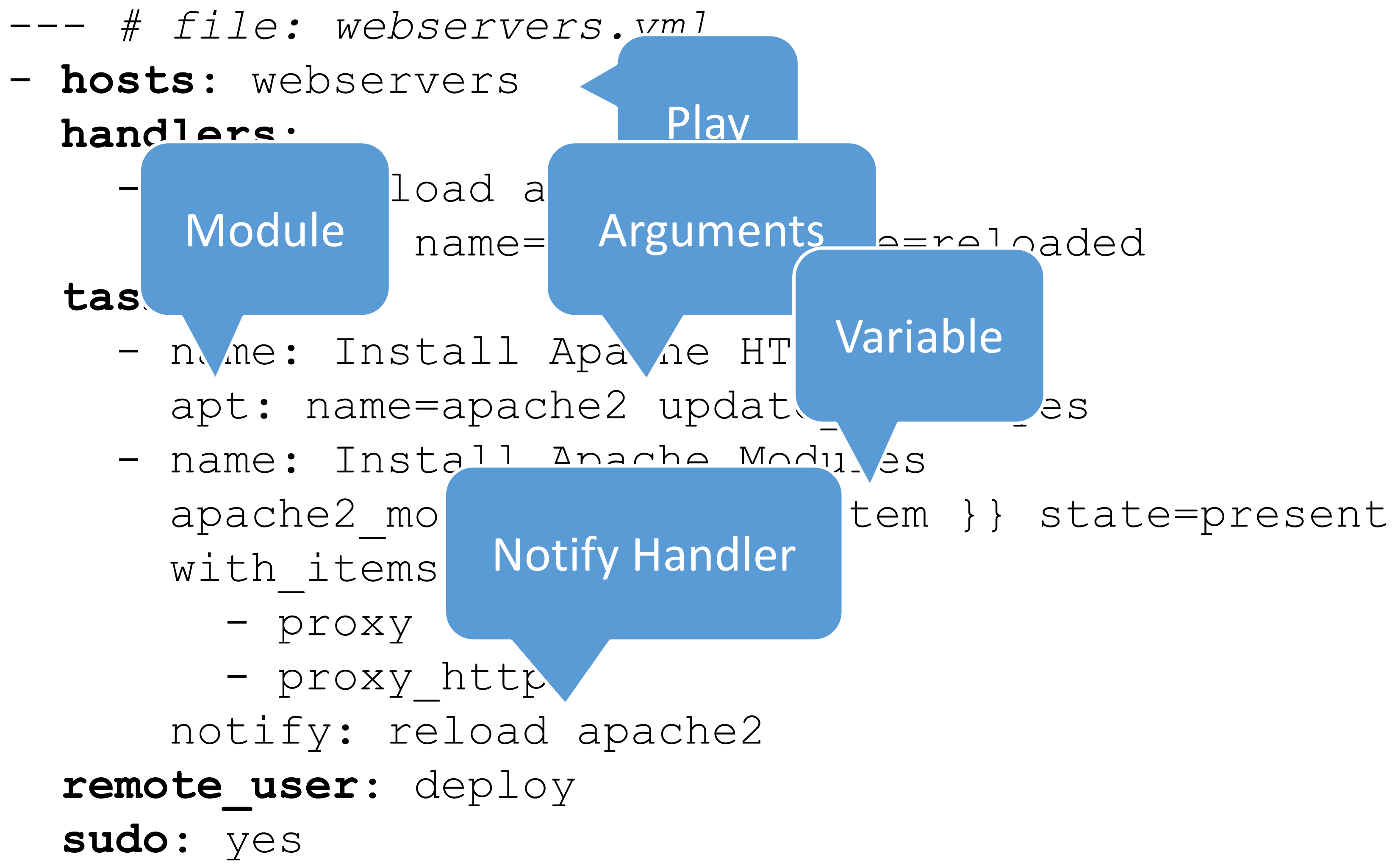
What is a Playbook?

» Describes policies your managed machines shall enforce

» Consist of **vars**, **tasks**, **handlers**, **files, templates** and **roles**

» Expressed in the **YAML** format (dictionaries, lists and scalars)

Example: Ansible Playbook



```
--- # file: webservers.yml
- hosts: webservers
  handlers:
    -         load a
            name=               e=reloaded
  tas
    - n  me: Install Apa  he HT
      apt: name=apache2 updat              es
    - name: Install Apache Modu  es
      apache2_mo             tem }} state=present
      with_items
        - proxy
        - proxy_http
      notify: reload apache2
  remote_user: deploy
  sudo: yes
```

Example: Ansible Playbook

```
--- # file: playbook.yml
- include: balancers.yml
- include: webservers.yml
- include: dbservers.yml
- include: monitoring.yml
```

Ansible Inventories

- Ansible provisions groups of servers at once
- Groups and hosts are defined in inventories
- Use inventories for staging, production, etc.

Static vs. Dynamic Inventories

- Static: text files expressed in an INI-like format
- Dynamic: Python scripts for dynamic environments (cloud)
- Static + Dynamic: combine multiple inventories (hybrid cloud)

Example: Static Inventory

```
# file: production

[balancers]
www.example.com

[webservers]
www[0-9].example.com

[dbservers]
db[a:f].example.com

[monitoring]
dynatrace.example.com
```

Group

Host

Numeric Range

Alphabetic Range

Dynamic Inventories

Python scripts that get data from dynamic sources such as:

- *Cloud*: Amazon, DigitalOcean, Google, OpenShift, OpenStack, etc.
- *Distributed Information Services*: LDAP, etcd, etc.

Example: Ansible Playbook

```
$> ansible-playbook -i production webservers.yml

PLAY [webservers]
*******************************************************
TASK: [Install Apache HTTP Server]
*******************************************************
changed: [www0.example.com]
changed: [www1.example.com]
...

PLAY RECAP
*******************************************************
web0.example.com: ok=3 changed=3 unreachable=0 failed=0
web1.example.com: ok=3 changed=3 unreachable=0 failed=0
...
```

questions?

Thank you