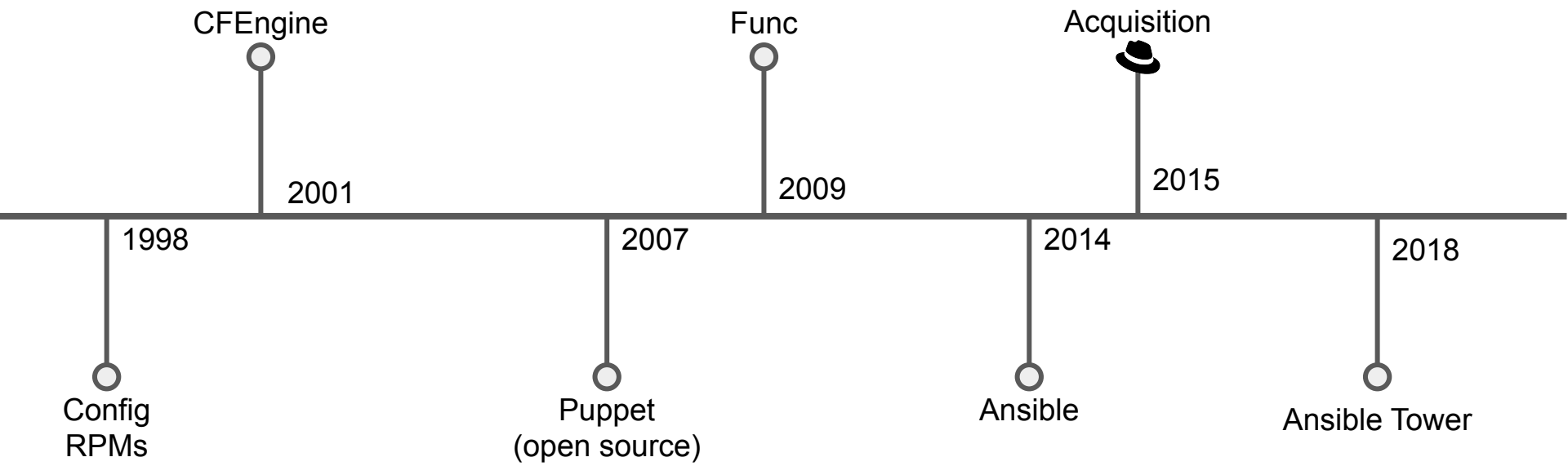
Pulling the puppet strings with Ansible

Brian J. Atkisson, RHCA
Infrastructure Domain Architect

Red Hat

A Story

A Brief History of RH IT Config Management

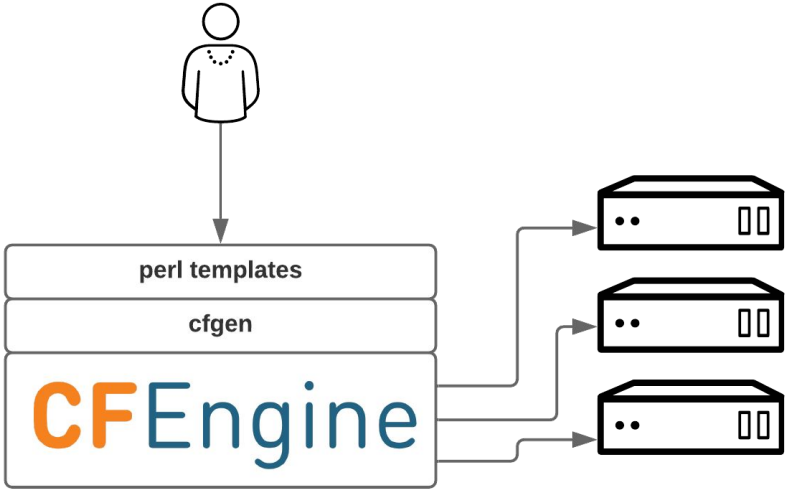


2005 - Initial release of Puppet
2012 - Initial release of Ansible

Red Hat

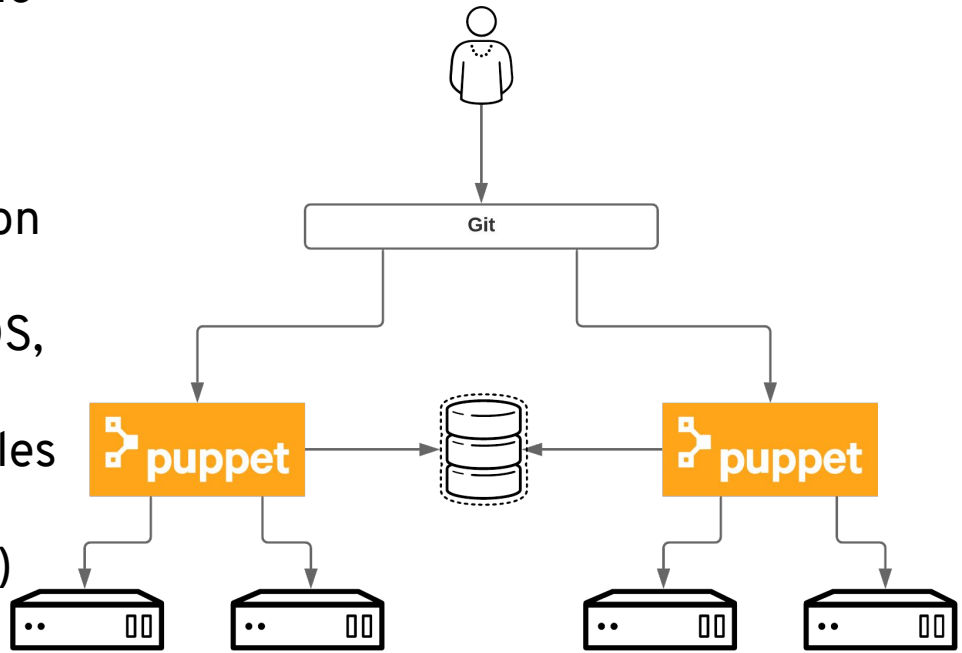
CFEngine Migration to Puppet (2007)

- Development teams doing own thing
- Applications codified into CFEngine by Operations
  - Thrown over the wall the night before a release
- Home-grown templating engine
  - cfgen
  - Perl “DSL”



Puppet Management Ideal

- Puppet modules developed along side applications
  - Stored in Git
  - Environment concept
- Application teams manage application modules
- Infrastructure teams manage base OS, monitoring, IAM, etc modules
- Release engineering manages modules shared between application teams (JBoss, Java, Rails, Apache, PHP, etc)
- … and there was peace in our time



Realities of Puppet
a decade later...

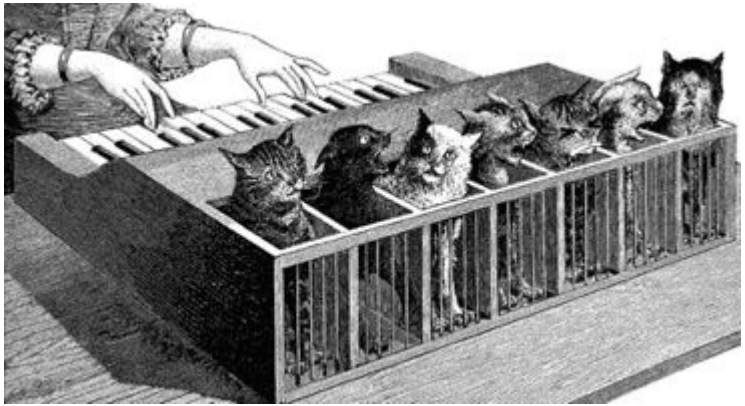
- Difficult balance between old and new operating systems
- Language is hard to learn
  - Manifests
  - ERB templates
  - Dependency ordering is difficult
    - Especially in large codebases
- Puppet was extremely slow, scaling challenges
  - Constant game of fix-the-bottleneck
  - Reliance on storeconfigs
  - Global scale

Realities of Puppet
a decade later...

- Our implementation was even harder to learn
  - Pre-dates tooling
- Updates approaching impossible
  - Large code base
  - Updating modules from puppet 3.x to 5.x
- App developers disliked Puppet
  - A lot of overhead
  - (and Ruby just isn't cool anymore)
- We managed too many objects
  - including OS defaults
- Puppet is extremely weak at orchestration

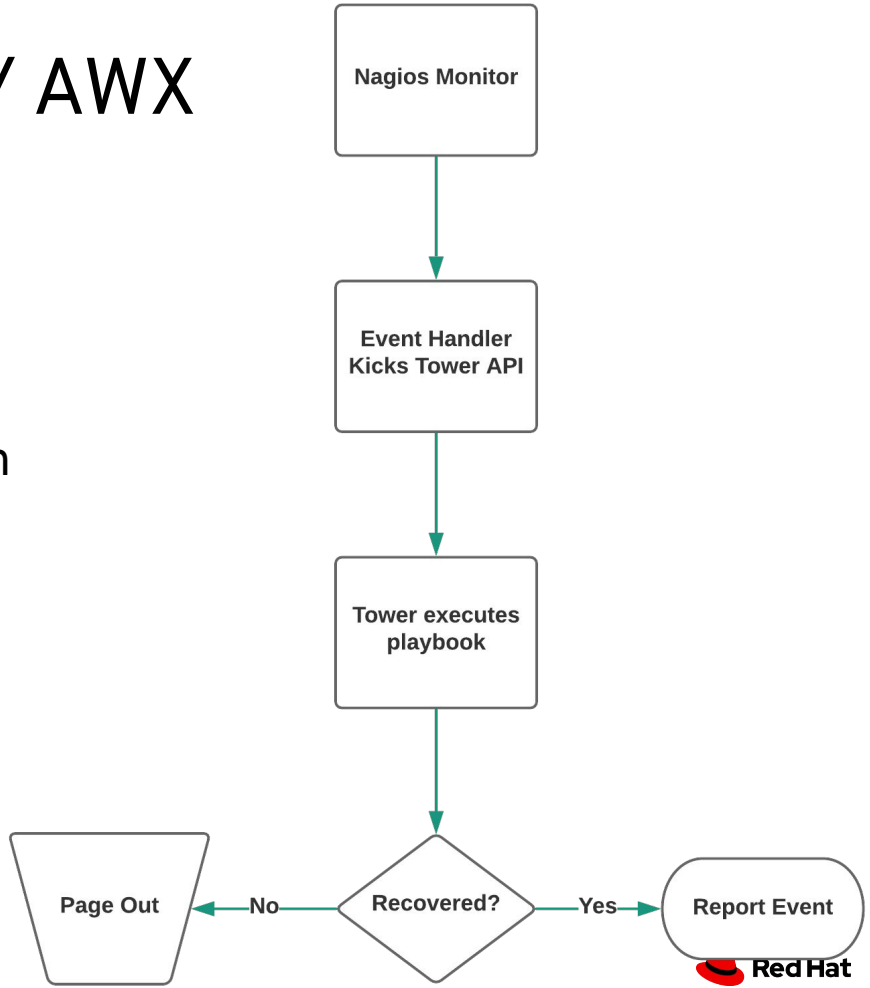
Hybrid Approach

- Ansible for orchestration
  - Puppet runs kicked off by Ansible
  - Native and/or trivial integration with cloud solutions, appliance APIs and just about everything else
- Dependency ordering
  - No more periodic race conditions
  - Just write the playbook in order
- Easy to learn and understand
- Agentless
  - OS Permission model



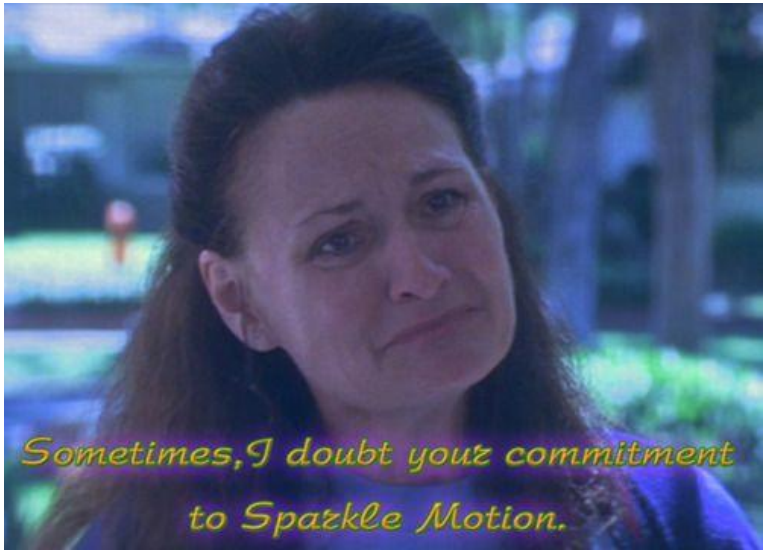
Tower / AWX

- Centralized playbook execution
- RBAC
  - Credential management
- Remote (API) playbook execution
- Autohealing systems
  - Nagios event handlers
- Metrics



Full Adoption

- AWS and OpenStack infrastructure
  - Infrastructure, OS configuration and orchestration
  - Long-lived and ephemeral VMs
- Traditional Data Center
  - Provisioning
  - Laying down Puppet-generated OS templates
  - Orchestration and Releases
- OpenShift and container management
- CI/CD
  - Jenkins -> Ansible -> all the things

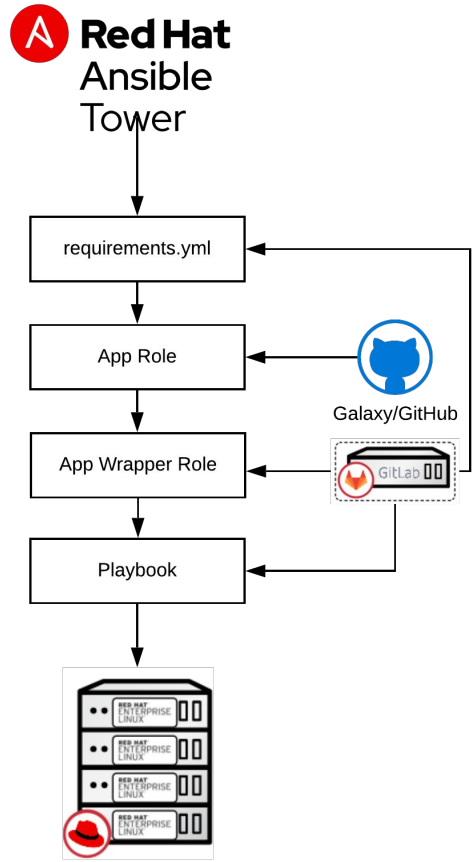


Federated Management

- App teams run their VMs top-to-bottom
- Infrastructure teams provides ‘suggested’ Ansible roles (easy path)
  - Tower integration
- Option to BYOCM
  - Mandatory and optional configuration
    - Compliance auditing with OpenSCAP
- Centralized role distribution (“Galaxy”)
  - Promotes open source development (app role)
  - Inner sourcing for RH-only bits (profile role)

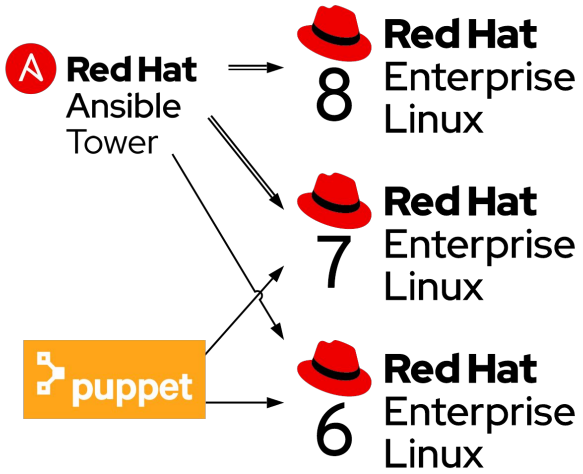
Playbook Management

- Roles contain smallest functional component
- Encourages Reuse
- All Apps have at least two roles
  - Main functionality (often multiple roles)
    - Anyone outside Red Hat should be able to use
    - Publish on Galaxy
    - Re-use what is on Galaxy
  - Wrapper role
    - Our environment-specific logic



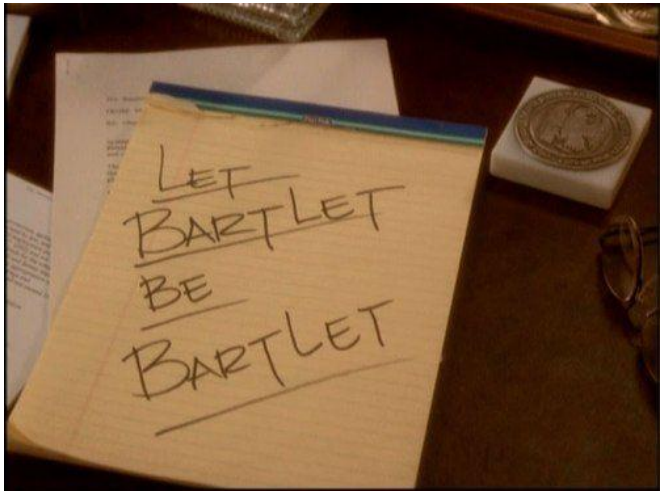
RHIT Operating System Support

- RHEL6 requires Puppet (in our environment)
- RHEL7 is last Puppet-managed OS
- Our aging Puppet deployment will not support RHEL8 in the current form, Ansible only
  - Upgrade your stuff. <- note the period
- Most RHIT run Fedora, RHEL or Mac
  - Ad hoc playbooks
    - Apple's unfrozen caveman python version problematic
  - Tower-executed playbooks



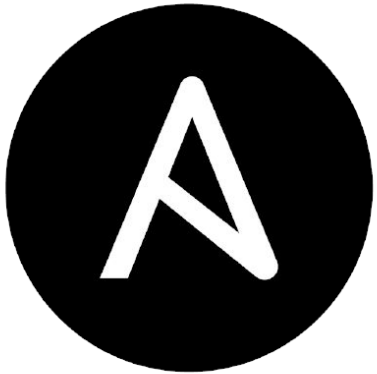
Let Ansible be Ansible

- Use the best tool for the job
  - Not always Ansible
  - Sometimes that means just writing a script
- Don’t be afraid to write an Ansible module
  - Contribute it upstream, they love it



Let Ansible be Ansible

- Data elements removed from configuration management
  - Service auto-discovery
    - Istio, DNS, etc
  - User permissions and authorizations
    - FreeIPA/IdM/LDAP/AD
  - Secrets
    - Evaluating new secret storage



Lessons Learned

- Only manage what you need to manage
  - Config management != audit system
  - Don’t manage the defaults
- Use authoritative data sources where you can
  - LDAP, DNS SRV records, etc
  - Service mesh
  - Dynamic inventories are awesome, use them
    - Satellite/FreeIPA/$cloud
- Don’t be clever

Lessons Learned

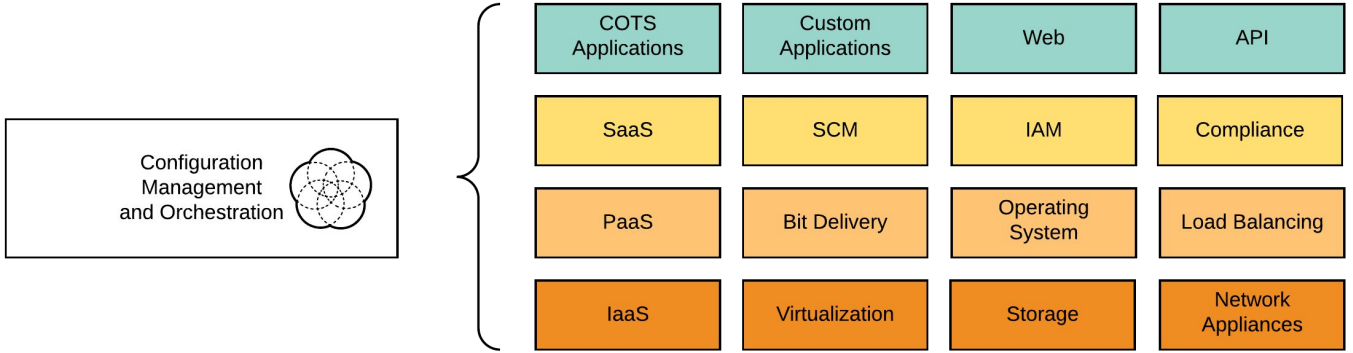
- Tools don’t fix people problems
- Config Management Standards Body
  - Work with service owners
  - Set standards and tools
  - Consumable, supportable code
  - Encourage reuse
  - Stop the crazy
  - Coordinate version updates

Lessons Learned

- Unmanaged things are unmanaged
  - Don’t make manual changes
- Use Tower or AWX
  - Centralized, auditable playbook execution
  - Credential management
  - Dynamic Inventory
  - Playbook chaining and advanced workflows
  - Scheduled jobs
    - More useful than you’d think
    - Replaced many cronjob’d perl scripts
  - Notifications

Next Steps

- Open hybrid data center using this model
  - OpenStack
  - Red Hat Virtualization
  - Ceph
  - OpenShift



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat