Red Hat Summit

Connect

Enable Ubiquitous and Scalable Automation

Using the Validated AAP Config as Code Collections

Lee Shakespeare
Ansible Automation SSA
Red Hat

Tom Page
Services Architect
Red Hat

Lee Shakespeare

Ansible Automation SSA
Red Hat

Tom Page

Services Architect
Red Hat

Why Configuration as Code?

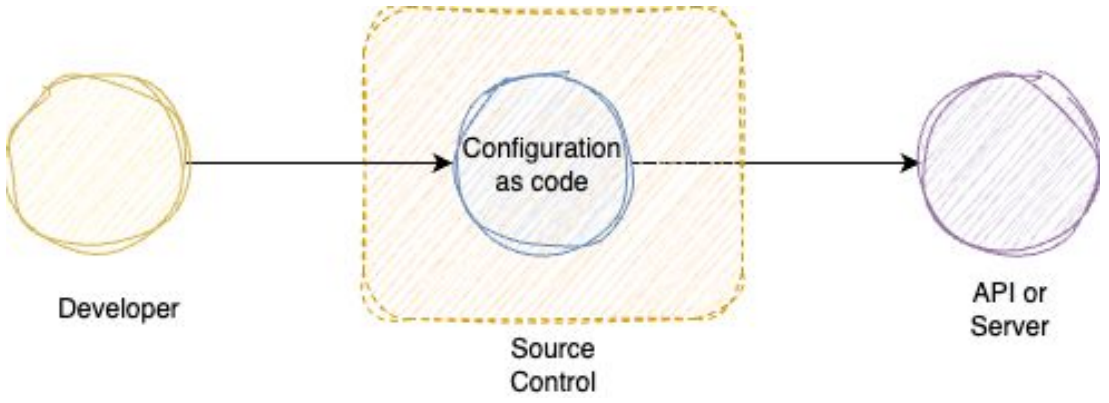
- Todays environments are complicated, dynamic and multi environment behemoths
  - Replicating configurations across environments can be error prone and time consuming
- Organisations require onerous change management and audit procedures
  - Technical controls embedded in the code prove compliance
- Dynamic environments evolve and unexpected events happen
  - Keeping a known good copy of your configuration allows rapid rebuilds and recovery
- Rapid, responsive, self service automation is expected for all components, AAP is no different
  - Integrations with ITSM event driven automation and GitOps tools enable organisation to move quickly and safely

Adoption

- Administrator actions such Platform configuration, Customer onboarding, Secret Rotation
- Export existing configurations as a starting point for organisations
- Use Git methodologies and practices with CI/CD to shift left and enforce policy
- Use configuration as data where the configure structures repeat
- Use the AAP RBAC to ensure privilege separation
- Set a direction, define a baseline, use customer tiering and migrate as needed

What is Config as Code

- Separate application (automation) code from the configuration
- A declarative description of the configuration you are trying to set
- Gives us the ability to source control and standardise the configuration of our application



What is AAP Configuration as Code?

"Config as Code (CaC) separates configuration from the application code"

What?

- Describing the state of AAP infrastructure configuration
- Structured format
- Automation tooling can process description and configure

Why?

- Increase speed of deployment
- Repeatability
- Error Reduction
- Consistency across environments
- Prevention of configuration drift

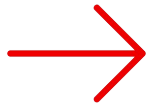
What is GitOps?

Declarative

A system managed by GitOps must have define its desired operating state separate from the procedures of how that state will be achieved

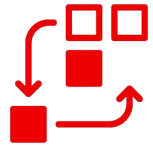
Versioned and Immutable

Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.



Pulled Automatically

Software agents automatically pull the desired state declarations from the source.



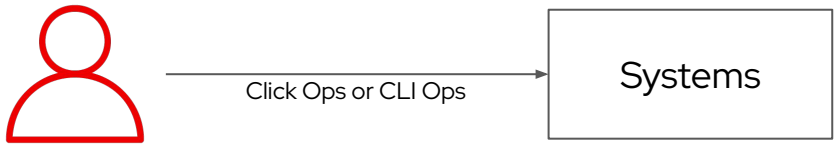
Continuously Reconciled

Software agents continuously observe actual system state and attempt to apply the desired state.

CONFIDENTIAL Red Hat associate and NDA partner use only,
No further distribution

Imperative State Architecture

Manual



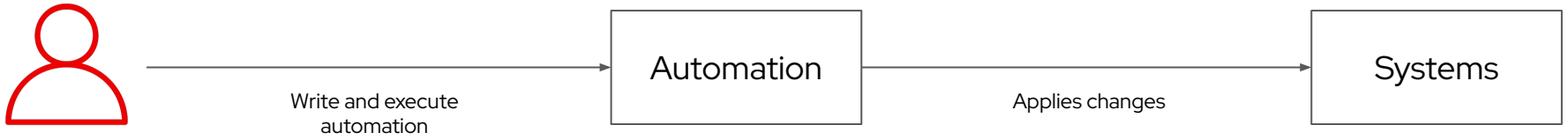
Cons:

- do we really have to list them?

CONFIDENTIAL Red Hat associate and NDA partner use only,
No further distribution

Imperative State Architecture

Automated



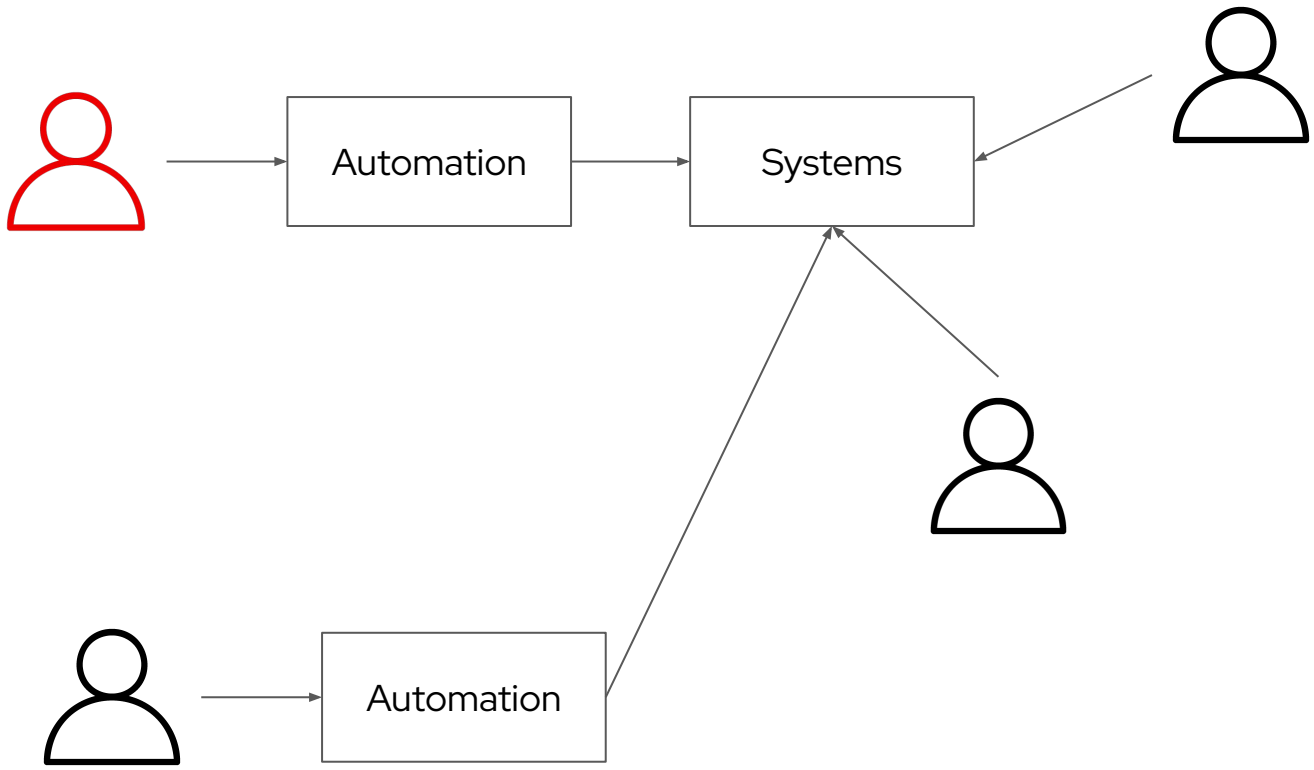
Cons:

- you have to maintain the automation
- automation and state get conflated together
- drift is not automatically brought back in line
- and...

CONFIDENTIAL Red Hat associate and NDA partner use only,
No further distribution

Imperative State Architecture

Automation...Inevitably



Cons:

- can't prevent people from still do manual, or detect it, or auto correct
- multiple people can and will write their own automation with their own state which will compete

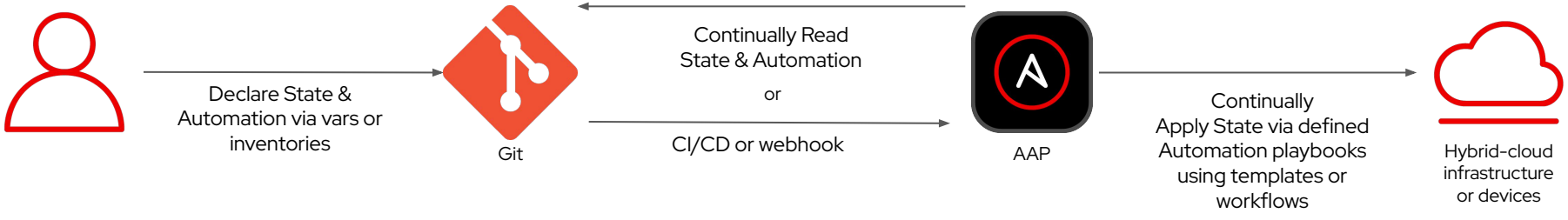
CONFIDENTIAL Red Hat associate and NDA partner use only,
No further distribution

Declarative State Architecture

Cons:

- it's a more complicated infrastructure to set up and maintain
- you can’t “cheat” and just manually change something through Click-Ops, it will be overwritten
  - (this is really a pro, but some see it as a con)
- it can be less intuitive to declare state/intent then writing imperative automation mixed with state
- when operations are dependent on each other, things can get tricky

AAP Declarative State Architecture



Tools to make the Ansible Platform Declarative

Certified

- ansible.controller
- ansible.hub
- ansible.eda
- ansible.platform

Validated

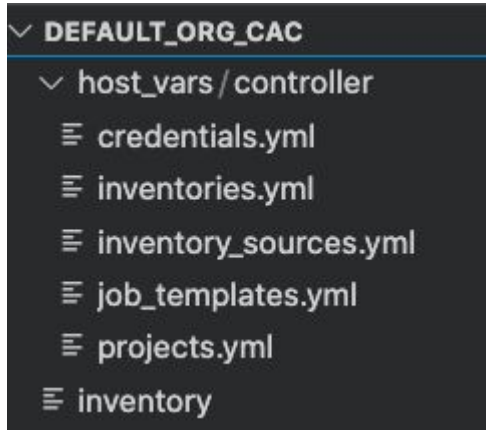
- infra.aap_configuraiton
- infra.aap_utilities
- infra.ee_utilities
- infra.aap_configuration_extended

13

infra.aap_configuration

- Enhances the modules from the certified platform collections
- Adds structure to how to define objects at scale
- Offers roles for each object type
- Additional ‘helper’ roles as opinionated overall procedure

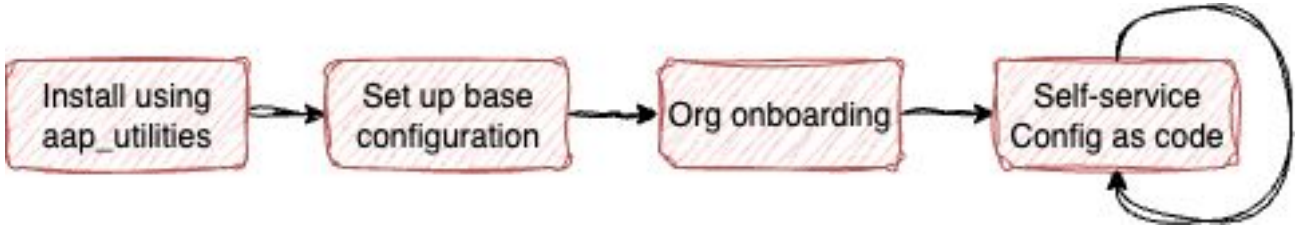
```
---
controller_projects:
  - name: Demo project
    organization: Default
    scm_branch: master
    scm_type: git
    scm_update_on_launch: true
    scm_url:
https://github.com/ansible/tower-example.git
```



Red Hat

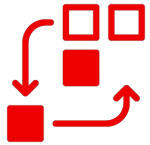
Setting up AAP

- Use infra.aap_utilities to simplify and automate the installation
- Bootstrap the Platform with initial configuration
  - Automation for org onboarding
  - Auth configuration
  - Settings
  - Global shared credentials
- Creates a base for the Platform



Self-service Config as Code

Self-service Config as Code Aims



Scalability

Increase scalability by allowing self-service creation

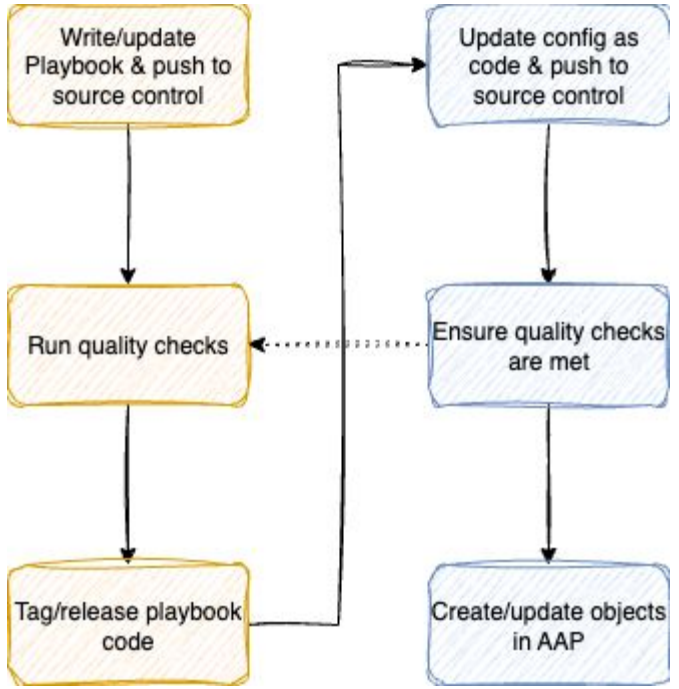
Meeting Standards

Ensuring security & best practice standards met

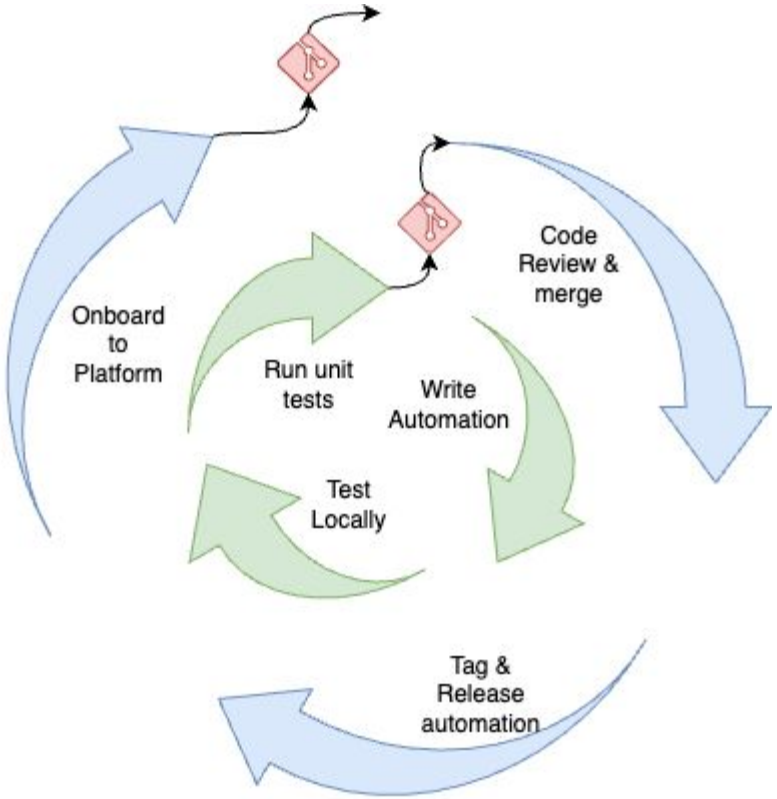
User Journey

Create a positive user journey

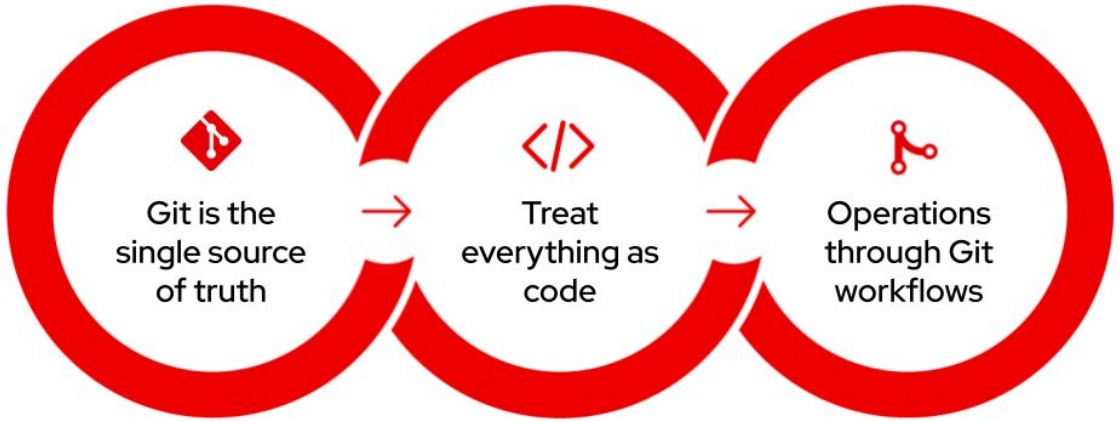
- Provide an offering to allow creation of own projects, job templates, credentials, inventories, etc.
- Reduce Platform team's need to perform checks
- Shifts ultimate responsibility to automation creators, but Platform team create pipelines to ensure standards
  - Making use of provided tooling such as ansible-lint and Molecule
- Creation of a versioning policy to further enhance best practices



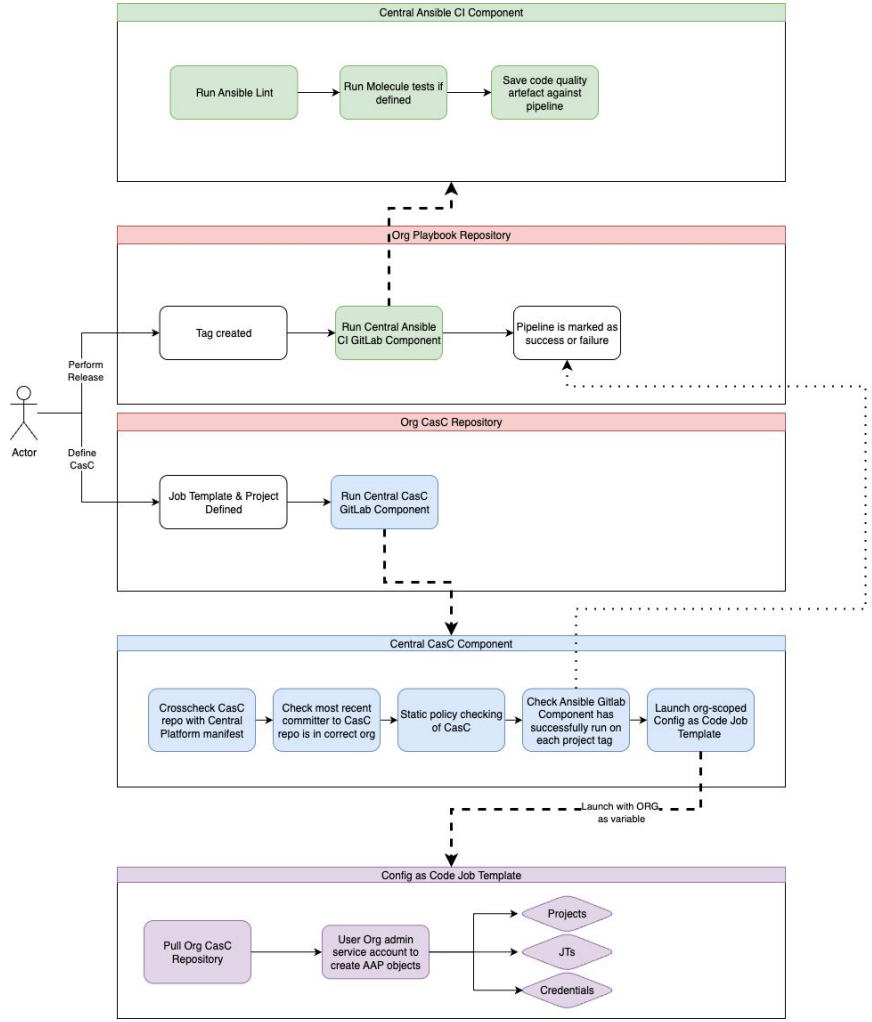
- Creation of a smooth user journey
- Easy to create the objects without needless impediments
- Enhancing inner/outer loop development of automation
  - Dev Automation Platform
  - ‘ClickOps’ export offering to higher environments



- GitOps practices trigger changes to AAP new config pushed
- Git becomes the source of truth for AAP configuration
- Optionally include logic to detect removed items from config

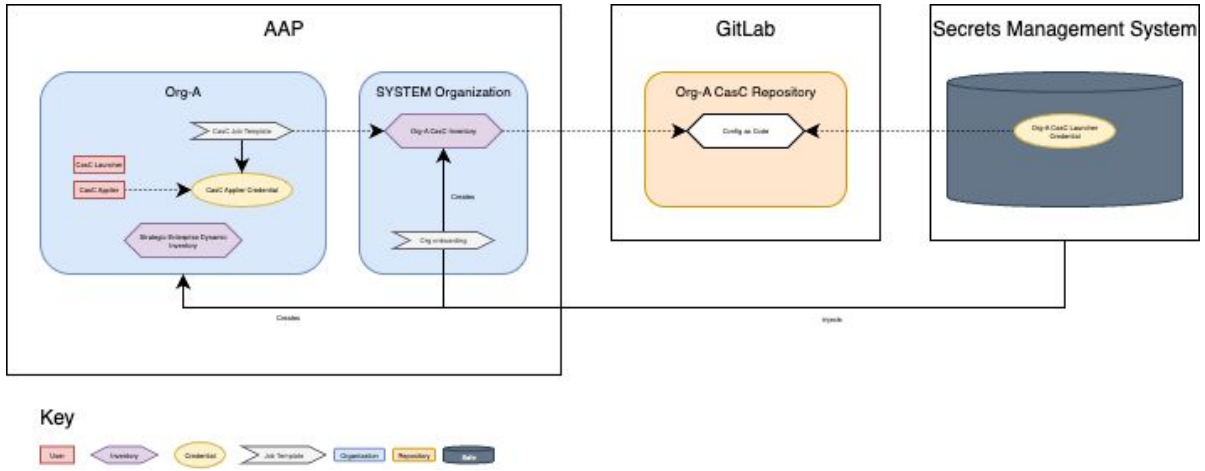


1. User creates/updates their playbook
2. User tags their repo
3. Pipeline runs to perform quality checks on playbook repo
4. User updates CasC with new tagged version
5. Pipeline runs to verify quality checks passed
6. Pipeline creates or updates objects in AAP



Organization-as-a-Service

- Part of a Platform as a product model
- Deliver single interface for customers to request an organization
- Facilitate easy onboarding to Platform customers by offering a self-service onboarding model
- Create template organization as code
  - Team structures (admin, execute, read)
  - CasC structures (applier jobs, projects, creds)
  - Strategic enterprise inventories (lookup CMDB based on org info)
  - Secret Management lookups
- Create reusable pipelines (EDA, Gitlab component, GitHub action etc) for enacting the config as code when it changes



Recommendations, Tips & Tricks

- We don’t want to add secrets to Git
- We can use ansible-vault to add encrypted secrets
  - Not always ideal or acceptable in enterprise environments
- AAP Credential lookups
  - AWS Secrets Manager Lookup
  - Centrify Vault Credential Provider Lookup
  - *CyberArk Central Credential Provider* Lookup (CCP)
  - CyberArk Conjur Secrets Manager Lookup
  - HashiCorp Vault *Key-Value* Store (KV)
  - HashiCorp Vault SSH Secrets Engine
  - Microsoft Azure *Key Management System* (KMS)
  - Thycotic DevOps Secrets Vault
  - Thycotic Secret Server
  - GitHub app token lookup
- infra.aap_configuration.credential_input_source

```yaml
controller_credentials:
 - name: gitlab
   description: Credentials for GitLab
   organization: Default
   credential_type: Source Control
   inputs:
     username: person
     # Note that password field left blank

controller_credential_input_sources:
 - source_credential: hashivault
   target_credential: gitlab
   input_field_name: password
   metadata:
     secret_backend: mykv
     secret_path: vault/path/to/gitlab/secret
     secret_key: GITLAB_PASSWORD
```

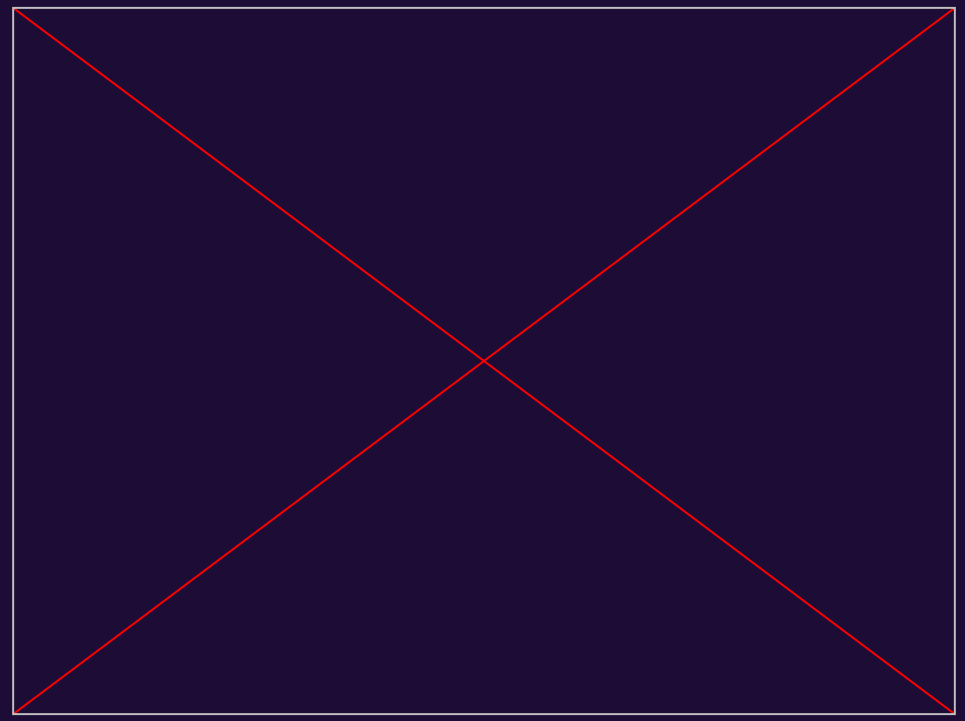

- Often trying to create many objects with similar values but only differ by a couple of values
- Ansible supports YAML anchors to help scale code
- Can define a ‘template’ item and reuse it multiple times

```yaml
_: &jttemplate
  job_type: run
  inventory: "{{ organization }} Inventory"
  execution_environment: "{{ organization }}_exec_env"
  webhook_credential: global_webhook
  webhook_service: gitlab
  notification_templates_error:
    - Slack_for_testing

# This is the main list of job templates to be created or updated.
controller_job_templates:
  - <<: *jttemplate
    name: helloworld_survey
    description: "A job template with a survey enabled."
    survey_enabled: true
    survey: "{{ lookup('template', 'template_surveys/basic_survey.json') | regex_replace('\\n', '') }}"
    project: controller Config
    playbook: helloworld.yml
    credentials:
      - Demo Credential

  - <<: *jttemplate
    name: helloworld_fixed
    description: "A job template with a fixed extra variable."
    project: controller Config
    playbook: helloworld.yml
    credentials:
      - Demo Credential
    extra_vars:
      input_value: "hello world"
```


Demo



Red Hat Summit

Connect

Thank you


linkedin.com/company/red-hat


facebook.com/redhatinc


youtube.com/user/RedHatVideos


twitter.com/RedHat