

The Evolution of Ansible Tower at Gamesys



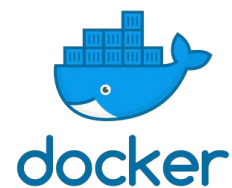
gamesys

Michael McCarthy - 03/06/2020



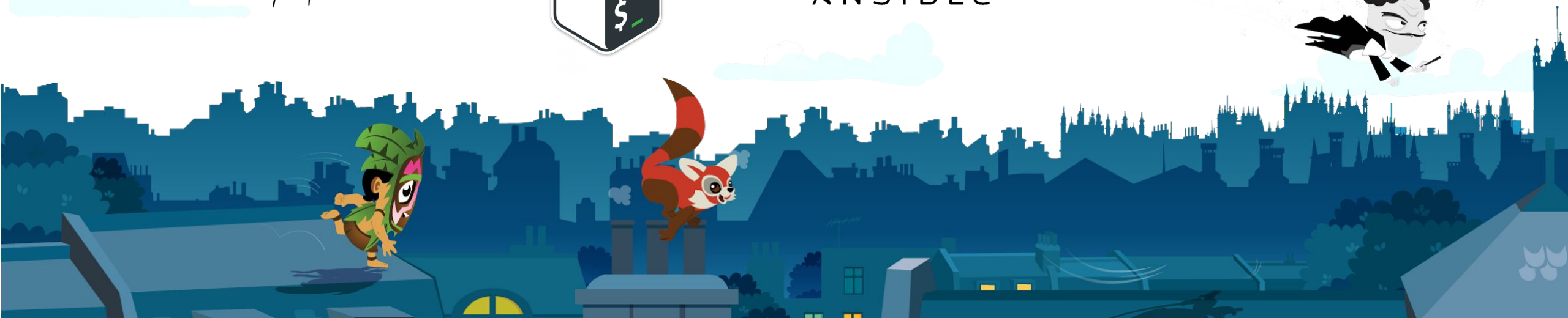
- Me - 12 years at Gamesys, 10 (dev), 2 (ops)
- Last ~4 years in 'platform' type roles (Platform Engineering, Delivery Engineering)
- I like platform roles - learn tech, help people, automate, make things faster, rinse and repeat!
- The theme of this is automation, self-service, reducing handoffs and generally Accelerating
- WARNING - may contain Puppet!

'Evolution' of Gamesys Deployment



What?

How?



Why Tower? What does Ansible not solve?



- Ansible is essentially running some commands against a target machine, typically remotely over SSH
- If those scripts are not in source control, or if you run with local changes and forget to commit, how is this any different to manually doing something on the host(s)? (Clue...it isn't)
- How do you have visibility over who did what (aside from 'history | grep')?
- If SSH credentials are involved in a playbook being run, how are these shared amongst your team?
- How do you provide an easy to use way to run your playbooks for less technical people if they involve multiple arguments?
- How do you let others run your playbooks without having to reveal sensitive data?
- Tower! At its most basic, Tower maps command line args into a web-based UI
- Tower adds auditing, authentication, a very rich authorization model, an easy to use interface, an API, notifications and much more
- Through Tower you can take the investment you've made in automation through Ansible to the next level via self-service

“Given a release that is entirely automated via a playbook, we could safely schedule the running of that playbook within Tower automatically from JIRA using the Tower API upon that release ticket being approved with no manual intervention necessary”





Learning from our mistakes



- Our big idea (2015):
 - development teams would write their playbooks for deployments (ownership)
 - playbooks would be ‘bootstrapped’ in Tower as templates (automation)
 - playbooks used for a release rather than manual steps in a ticket **for every environment** (consistency)
- Very different to what came before - Capistrano in Integration, a mix in Staging and Production
- Arguably...a partial success. So what happened?
- The per env Tower instances got popular. People would push more and more jobs through it, especially in earlier testing envs. This can cause severe contention, especially if any of those jobs take a while to run.
- The more teams started to use these Tower instances, the more teams become potentially locked-in to that version of Ansible, especially where modules were removed in later versions.
- Bootstrapping was running an Ansible playbook (to create job templates). We didn't do it regularly, ops didn't trust it. Lots of manual tweaking, faith was lost in the automation
- No sandboxing within Tower - entirely possible to break another team's templates
- Several other instances of Tower were provisioned to get around some of these problems -> higher total cost of ownership

Beginning of 2019...where were we?



- 
- More automation via more effort going in to playbooks for deployment...BUT
 - Shared Tower instances getting stuck on older versions and 'rotting' away
 - Contention, playbooks blocked waiting to run, one bad template can bring down Tower
 - No segregation within Tower
 - Still too many handoffs
 - Mindset difference between envs, e.g. inventory ownership in stg and prod
- 



- Some awesome (IMHO) features in Tower 3.2/3.3/3.4.
- Clustering - simply add more Tower nodes to distribute the load.
- Enhanced clustering via 'instance groups' - add nodes to an instance group and dedicate that instance group to an org/job template/inventory.
- Isolated instance groups - no more jump hosts
- Notifications - immediate feedback to teams when a template runs (no need to hassle your operator)
- Virtualenvs - create multiple custom Python virtualenvs and assign these at the job template/project/organization level. Each virtualenv has its own Ansible version and pip dependencies!
- More credential types including custom credential types
- Inventories from source - pull inventories from Git automagically!
- Dynamic inventories - pull groups from cloud provider

And hello Control Centre!



- ‘Control Centre’ is a new Tower instance, per environment type (Integration, Staging, Production)
- It is essentially some branding to distinguish this new Tower instance from all the other Tower instances that have popped up over time. So what’s different now?
- We use Puppet to provision Ansible Tower on our environments!
- Puppet installs the software, creates the virtualenvs, speaks to the Tower API to apply settings such as licenses, job isolation, external log aggregation etc.
- Arguably some of this could be done by Ansible, but questions over who triggers this the first time (maybe one day this could be a Terraform provisioner step?)
- Apart from the provisioning by Puppet, none of this is Gamesys ‘invented here’. This is a GOOD thing! Control Centre is arguably nothing more than a version of Ansible Tower, with some organizations, some virtualenvs, some custom Credentials, and a hook...

The Hook



```
"Player Services - Incentive":
  provisioning_user_password: ENC[PKCS7,MIIBiQYJKoZIhvcNAQcDoIIBe]
  provisioning_project:
    url: git@github.gamesys.co.uk:Incentive/incentive-tower-bootstrap.git
    git_branch: master
    playbook: site.yml
    custom_virtualenv: /var/lib/awx/venv/incentive-ansible-2.9.6/
  scm_credential_inputs:
    ssh_key_data: ENC[PKCS7,MIIH/QYJKo]
  virtualenvs:
    - name: incentive-ansible-2.9.6
      ansible_version: 2.9.6
      pip_packages:
        - name: openshift
          version: 0.8.8
        - name: PyYAML
          version: '5.1'
        - name: ansible-tower-cli
          version: 3.3.8
  other_credentials:
    - name: Player Services - Incentive - VPX Credential
      kind: cloud
      type: VPX
      inputs:
        username: ENC[PKCS7,MIIBeQYJKoZIhvcNAQcDoIIBajCCAWYCAQA]
        password: ENC[PKCS7,MIIBeQYJKoZIhvcNAQcDoIIBajCCAWYCAQA]
        host: https://ns.int07.integration.pgt.gaia
```

- Puppet creates the project as a Tower project and the job template according to the spec in hiera
- Puppet makes certain variables available through the inventory of that provisioning template which can be used in future templates (generic env stuff to avoid duplication)
- Puppet creates certain ‘credentials’ which can be used by the provisioner and/or passed on to subsequent templates (machine creds, custom creds). The most important is the Tower ‘cred’
- Teams create all their Projects, Job Templates, Notifications etc from within Ansible playbooks, typically using the tower_ modules (but also REST API/CLI)

Clarification of responsibilities



TEAMS

- Create a project in Git where their provisioning code will live.
- Provide a GitHub private key to Operations for encryption to check out said project.
- Write code in said project to provision all job templates and other projects.
- PR against hiera with details of the team, the project and any Virtualenv requirements.

PUPPET

- Creates the team/area as an organization in Tower
- Create the provisioning project under that team
- Create the provisioning template (but not schedule a run..yet)
- Create any extra credentials to provide to your project (Artifactory, Vault etc)

THE PROVISIONING PROJECT

- Runs a playbook that creates Tower projects/job templates/notification templates etc

Eating our own dog food



- We (ops) have an organisation in Control Centre just like dev teams do
- We try wherever possible to use Control Centre to run/schedule our playbooks
- We make these available for self-service in some cases
- We use our bootstrapping playbook as a 'reference' (while also learning from others)
- It's also in our own best interests to make it good!

Mid 2020...where are we?



- All teams within our platform have some presence - 13 orgs, 100s of templates
- Mostly consistent runs between environments - network connectivity still trips us up
- Learning curve - ops + dev
- Innovation from teams in using notifications, workflows etc etc. Just point them at the docs
- Authorization model - We've not yet tackled how authorization works in a safe automated way when teams own their own templates
- Some bumpy upgrades - hard to effectively test all use cases pre the first environment upgrade
- Good retro feedback
- Accelerate metrics improving, things generally getting faster



Next Steps / the future



- 3.7 upgrade - equal parts scared but excited
- Migration to Ansible Collections, in particular the AWX collections
- Move fully from ansible-tower-cli to AWX CLI
- Look at how we can run Tower in containers - currently the way we use virtualenvs does NOT make this easy
- (Isolated) instance groups in the cloud - deploying into GCP etc
- SSO integration (rather than service accounts)
- Once we crack the auth model we can do much more self-service
- External log aggregation to Splunk - CyberSecurity catching any leaked sensitive data
- Hashicorp Vault integration?
- Ensure operations are comfortable with the 'new' way (developer ownership of playbooks and inventories)





Questions?