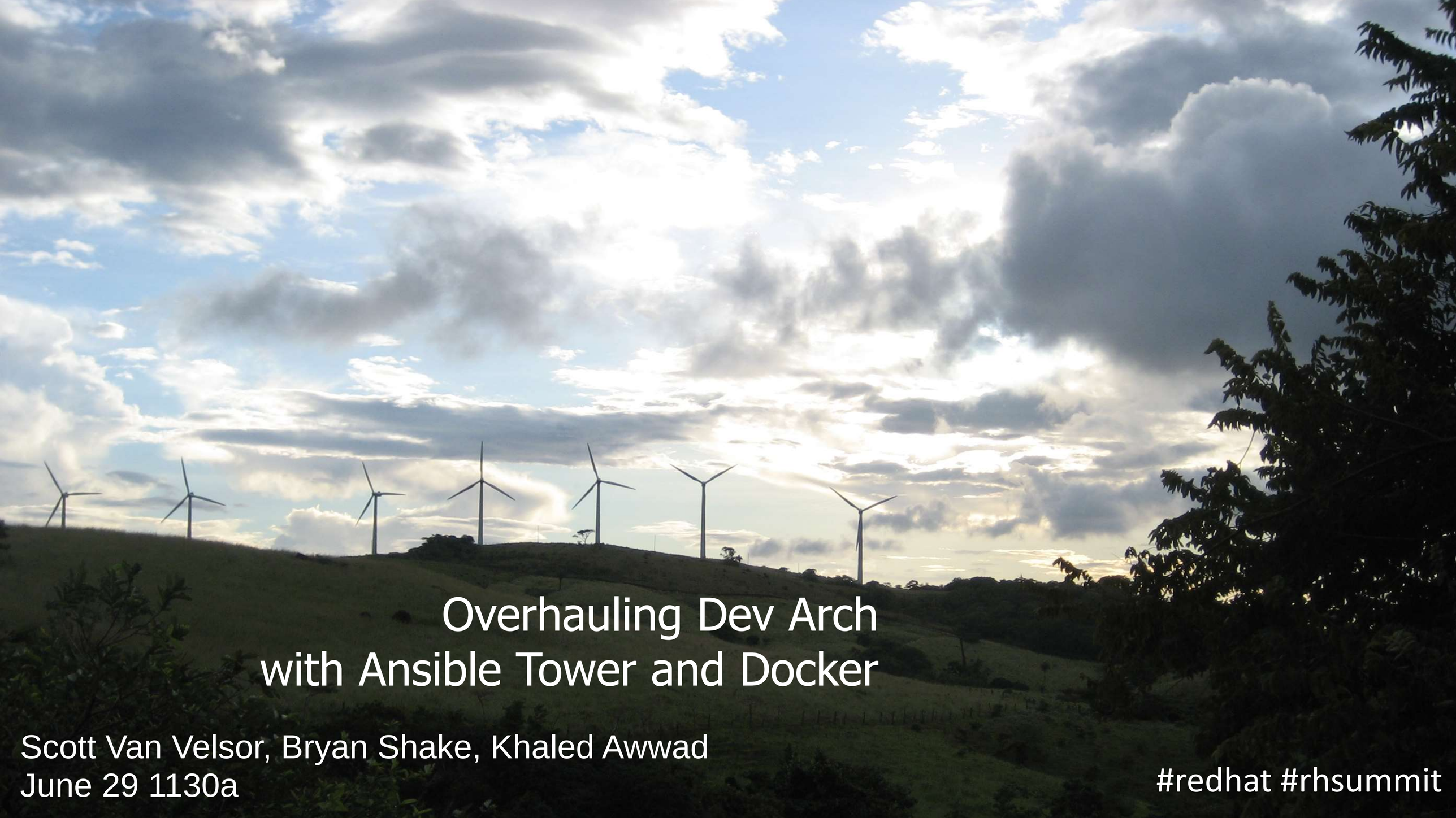


# Overhauling Dev Arch with Ansible Tower and Docker

Scott Van Velsor, Bryan Shake, Khaled Awwad

June 29 1130a

#redhat #rhsummit

# origins

# the landscape that came before

- × branch & path limits
- × no automation
- × confusion
- × heroics

endevor

the devs

subversion

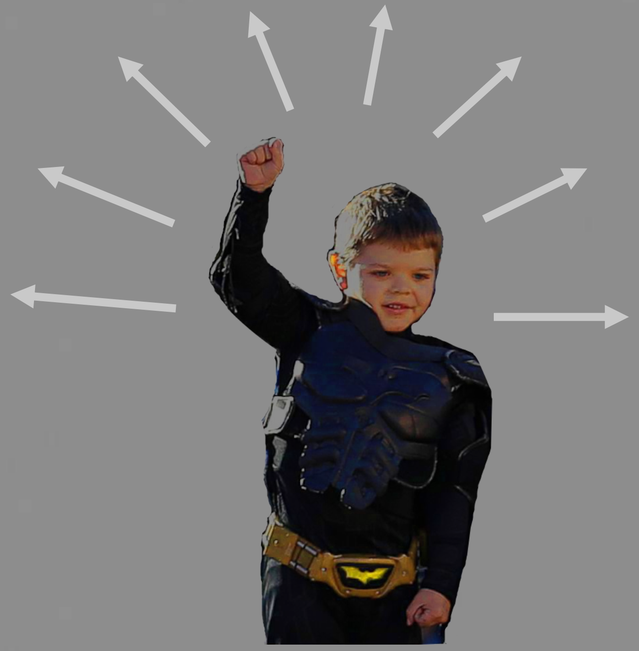
nexus

- ✓ build on check-In
- ✓ semi-automated deployments
- × environments built by hand
- × heroics

hudson

the frame

VB migration coordinator thick client



dev arch hero

cleanest 42u racks ever

# processes of the old world

dev
- check-in
- update tool
- send email
- wait

test
- send migrate email
- wait

pm
- send approval email
- wait

arch
- check tool
- migrate
- update tool

**deployed!!**

rinse & repeat

how does this scale to multiple releases  and more environments?
how does this scale to a 100+ person development team?
how is this even sane?

what about coordinating deployments on both platforms?
what about coordinating database changes?
what about coordinating config & script changes?
what about environment changes?

# it was obvious

**that approach (combined with heroics) supported**

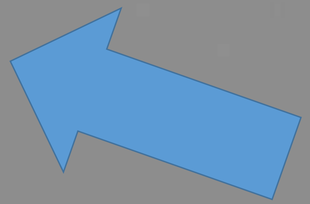
- 25+ applications (4 websites)
- 5 major releases/yr + service packs + emergency
- 8 environments
- *only* 2 release paths to production
- many, many late nights

*our enterprise is not small*

# the architecture challenge

**platform and architecture transition**

1. replace transactional mainframe CICS processing with an EAP tier
2. move all batch processing from COBOL batch on the mainframe to Spring Batch (Java)
3. move non-EAP websites from WebSphere App Server to EAP
4. create a distinct service tier on EAP for all application consumption
5. move from DB2 to Oracle



**we did this while building the new infrastructure and transitioning data & components between data centers**

# path to automation

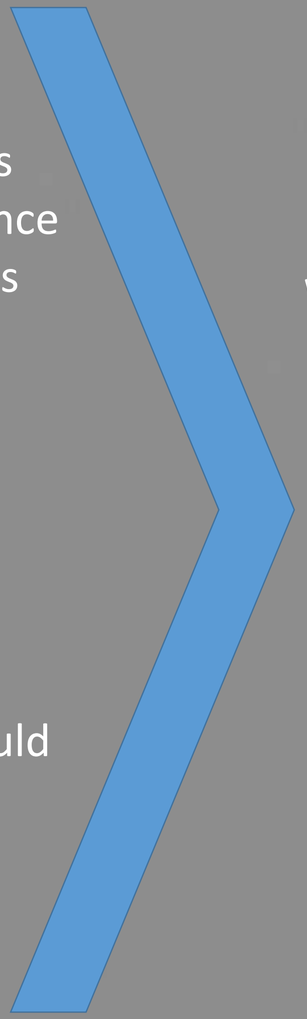
# figuring out the plumbing

**goals…**

+ accommodate more environments, paths, & releases
+ **100%** automated environment creation & maintenance
+ **100%** coordination of code, config, database changes
+ consolidate/integrate authentication
+ containers (control) for all except perf & prod
+ **eliminate the human factor**

**tenants…**

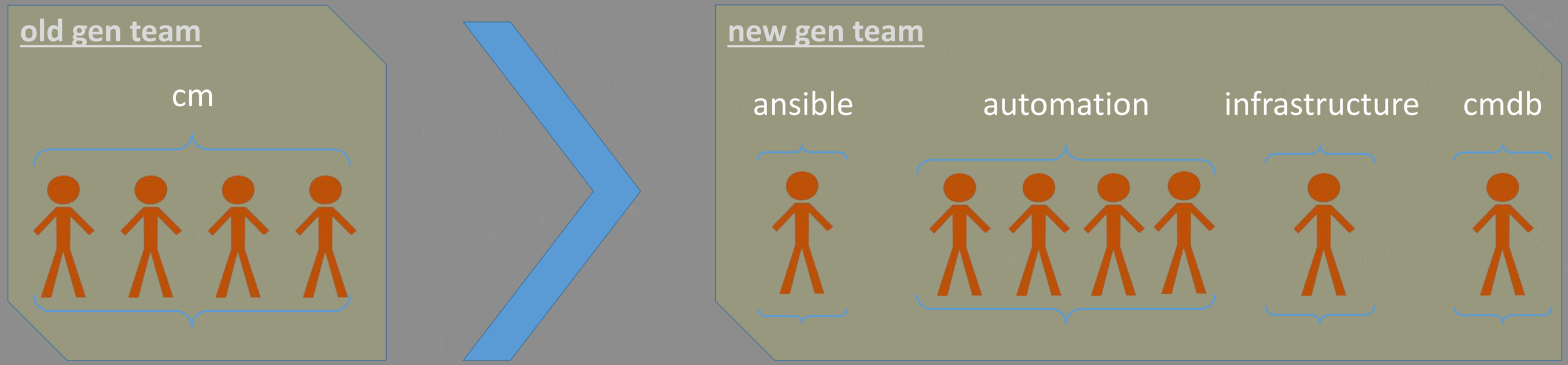
+ if it can't be automated, redesign it
+ application/environment/enterprise complexity should never be a factor (see previous)
+ 100% automation, not 99%
+ reduce vendor/tech lock-in
+ it has to be manageable



**what we need…**

+ configuration repo – CMDB, property files
+ an engine – Ansible/Chef/Puppet
+ a platform – Docker
+ many smart (*new*) people
+ **time**

# keep hands off by applying more hands



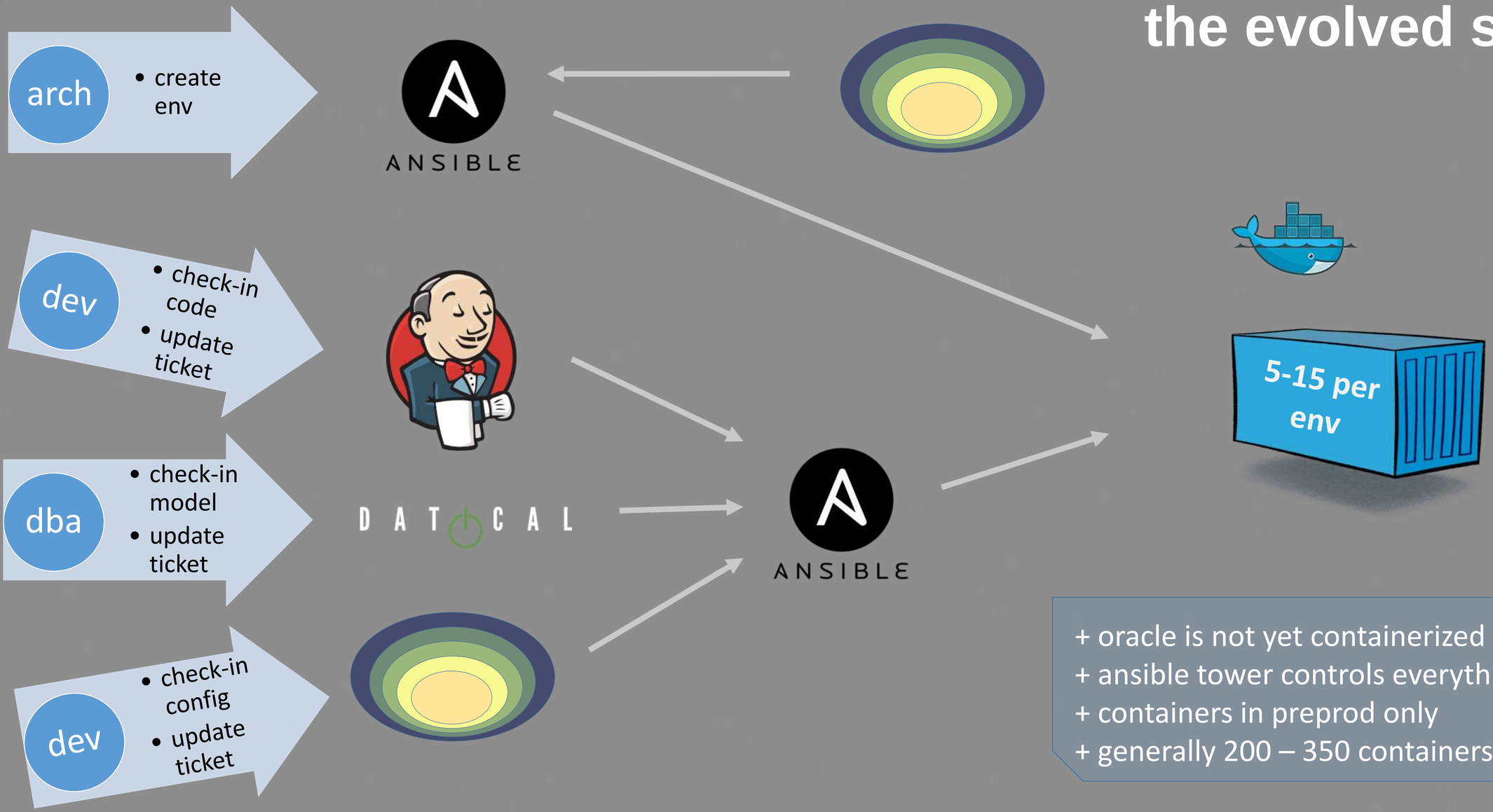

+ team size grew (for now)
+ other operations teams shrank
+ people now touch things through Ansible Tower

# the new landscape


** without the meltdowns

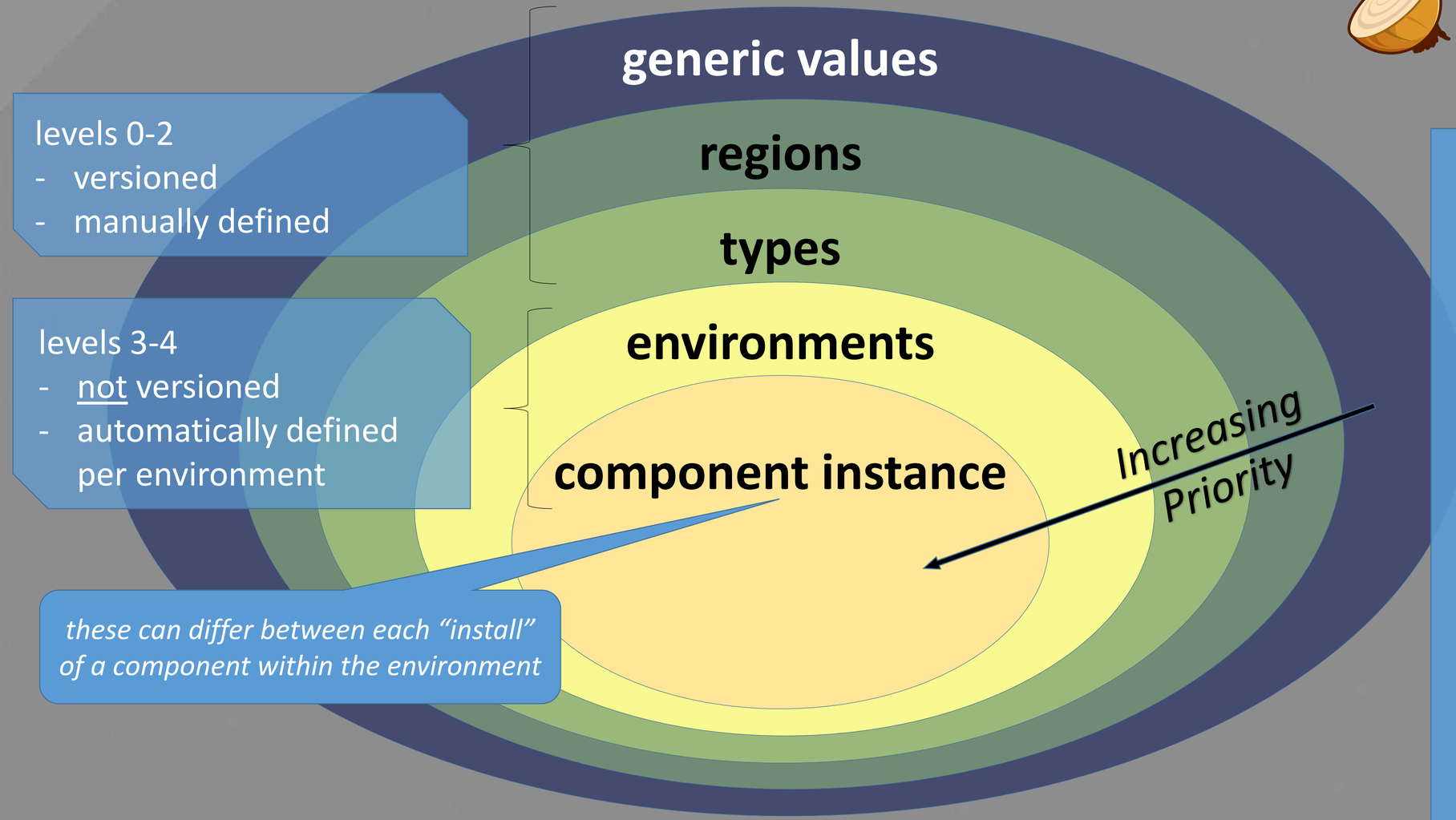
# well tuned

# the evolved state




+ oracle is not yet containerized
+ ansible tower controls everything
+ containers in preprod only
+ generally 200 – 350 containers in use

# the cmdb onion




levels 0-2
- versioned
- manually defined

levels 3-4
- not versioned
- automatically defined per environment

*these can differ between each "install" of a component within the environment*

**capabilities**

**+ version controlled and auditable**
**+ region/type overrides (versioned)**
**+ environment/component overrides**

**+ new version branching/insertion**
**+ version auto-inheritance (COW)**

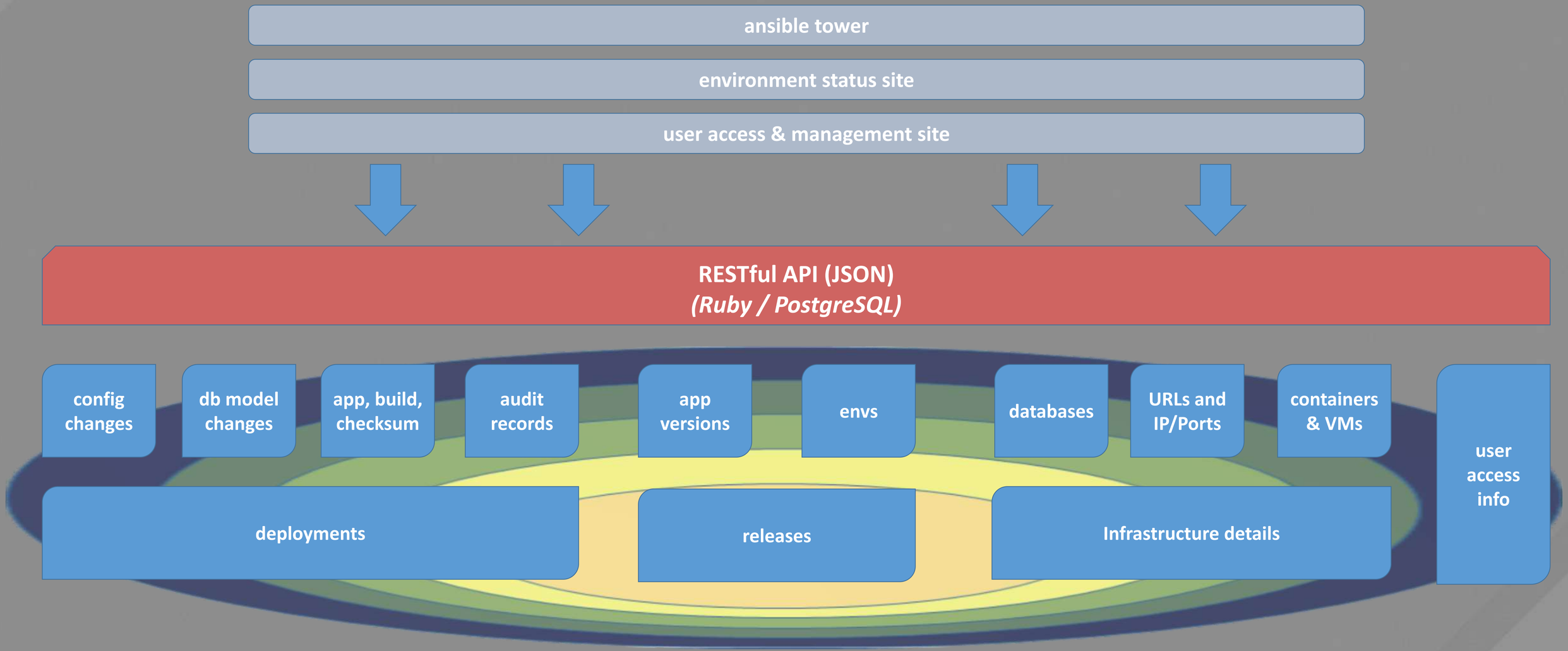
**+ separate version for each area (prior slide) to allow independent mapping to each environment**

**versioned areas**

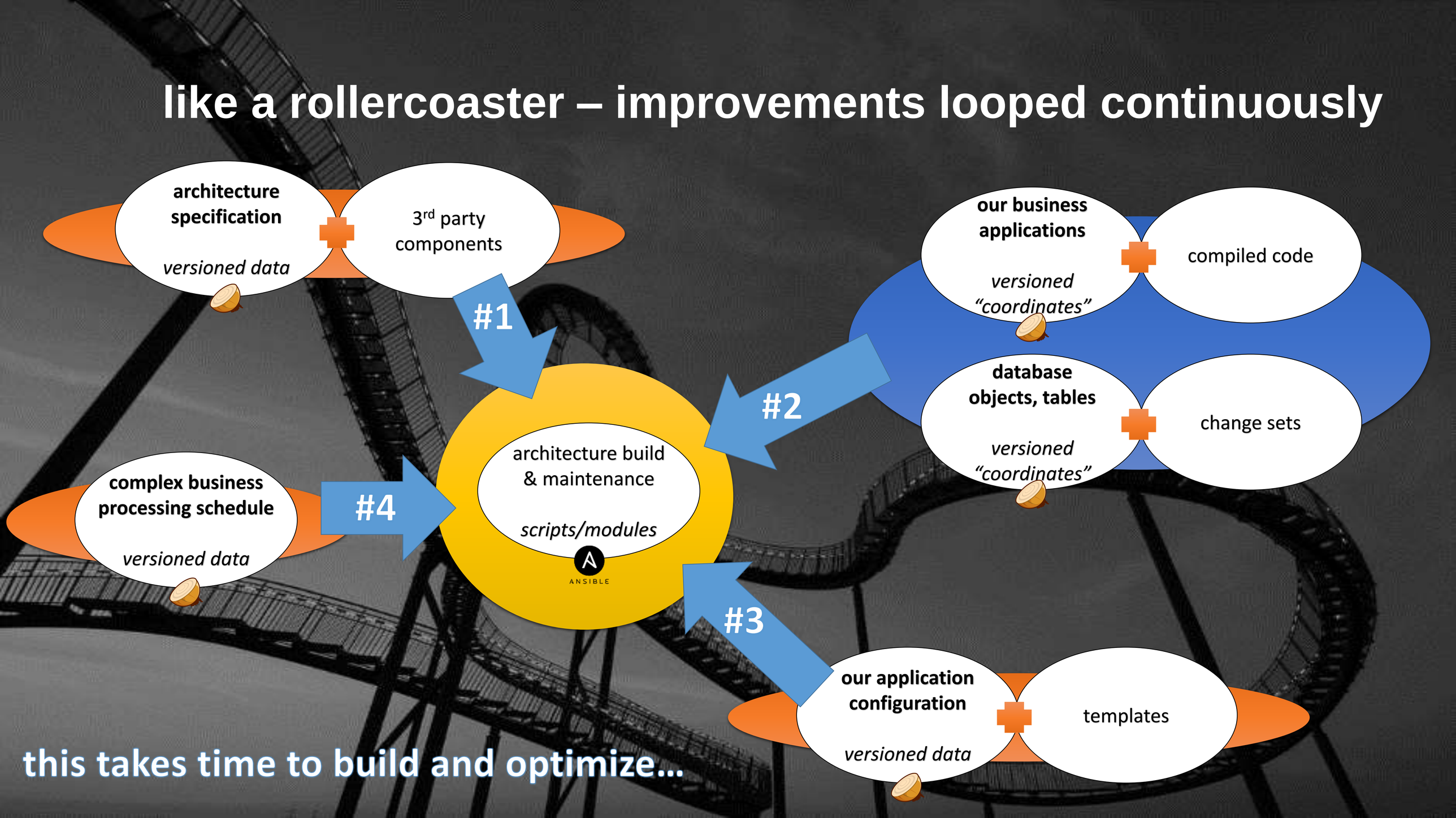
**+ architecture**
**+ application (code/database)**
**+ application (configuration)**
**+ job schedule**

| | |
|---|---|
| **regions:** | prod, preprod, aws-east |
| **types:** | prd, cct, pdt, uat, pft, ops, srt, … |
| **environments:** | uat004, cct007, pft009, prd001 |
| **components:** | eap, apache, clamav |

# integration API



# like a rollercoaster – improvements looped continuously

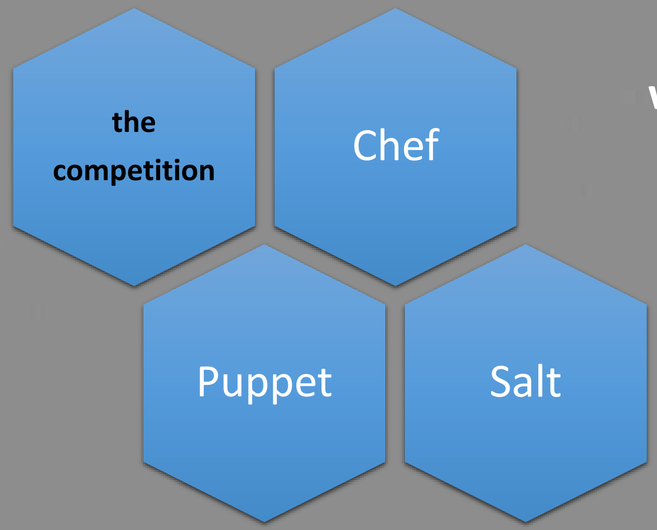



this takes time to build and optimize…

# why ansible?

**what we liked…**

+ geared toward system admins & archs vs developers
+ YAML – easy to learn & write playbooks
+ agent-less architecture, therefore nothing installed on target machines
+ ssh-based integration approach leverages solid security foundation
+ variable registration & reuse
+ excellent documentation




**what we had to live with…**

+ newer contender in the market place against Chef & Puppet which had more maturity
+ community still growing & issue reporting more challenging

# why containerize with docker?

+ prod runs on vms but that does not mean everything else needs to be on vms
+ our container granularity maps to our production vm granularity allowing us to run the same automation in all environments in the same way
+ the automation goes hand in hand with the containers – “treat them like cattle, not pets”
+ control control control – since we did not have control
+ environment/application isolation
+ infrastructure agnostic
+ more flexibility (see control x3)
+ docker is mature, right??
+ there’s nothing better than adding yet another thing to think about with security

# bumps & triumphs

# growing pains

**This did not happen without its major hurdles…  they were worth it.**

**some of the not so small issues…**

+ environment instability - too many environments on an unsupported os led to many sleepless nights
+ config management – keeping code and config in sync across all the different environments
+ chaos with deployments – deploy everything everywhere at all times brought deployment infrastructure to a crawl
+ limited infrastructure & growing demands – everything's critical and we could not support all the environment requests coming in
+ bottlenecks – error handling, hung jobs, slow jobs

# would we do it again?

**Yes, we would do it again without hesitation.  It's not even a conversation.**

**things to know before you start…**

+ select a strong, engaged team (they can be junior)
+ design, design, design, design, design review, plan, design, plan, plan, plan, design, design, design review, consider building
+ focus on security from the offset, don't try to layer in later (always true)
+ don't start with Docker on RHEL6
+ look to openshift to simplify many things now that it leverages Docker
+ don't completely re-platform your system when doing this
+ get the devs and testers comfortable and excited at the onset so they understand there will be bumps along the way

**cost of not automating**

**$$$ + sanity**

**cost of automation**

**it doesn't matter, just go do it**

# it’s never done

**tasks that remain**

+ implement a workflow process engine for release automation
+ further integrate jira ticketing (see previous)
+ automate and containerize oracle
+ cmdb more more more
+ full test suite automation (ops, perf, security, regression)
+ stop ssh’ing into everything
+ containers for prod
+ eap 7 + standalone
+ ansible 2.x – optimizing/rewriting scripts
+ automate what remains – target a self service, on-demand model
+ high availability (scaling & replication) with openshift

# find out more

scott van velsor
fscott.van.velsor.jr at accenturefederal.com

bryan shake
bryan.shake at accenturefederal.com

khaled awwad
khaled.t.awwad at accenturefederal.com