A post-CM infrastructure delivery pipeline

... or why I'm here to learn more

@beddari at @cfgmgmtcamp

Problem statement

We were using CM tooling but NOT WINNING

What we had built with love -

√ automated tests
√ JEOS + baseline + role
√ monitoring across envs

– was a total failure!

√ non-manageable rebuild times
√ envs were starting to “leak”
√ upgrades were high risk

“Our systems are …
eventually repeatable”

Darn it, test that small change in prod

People told us
“CLEARLY
you are doing something wrong”


docker docker docker docker docker docker
docker docker docker docker docker docker
docker docker docker docker

Solution:

We stopped doing configuration management

input | change | output

input

Inputs are typically managed artifacts

Repos, packages, images, containers

change

Feed input to Packer which in turn runs a builder that applies *change*, producing output

output

A *versioned artifact*, suitable for consumption or further processing

Repos, packages, images, containers

Abstraction

is key

A input-change-output chain is a *project*

•

A project is *versioned in git*

•

Artifacts are *testable*

Your new job is ...

describing state to produce artifacts
and keeping that state from drifting

We didn't throw it all out

A *system* consists of 1+N roles,
a *role* consists of 1+N profiles.

First lesson learned

Think about managing state,
where do you want it?

https://github.com/Nubisproject/nubis-docs/blob/master/MANIFESTO.md

Abstraction is future proofing

http://jerakia.io

"Decoupled from any particular configuration management system"

Does it work?

YES

Packer with masterless Puppet
Terraform and Ansible to deploy and replace
Masterless Puppet to audit and correct drift

Single YAML configuration data store

yum upgrade considered harmful