Clushible: Tidal Wave-like Configuration with Ansible

HPCSYSPROS SC23

Jared D. Baker
HPC Systems Engineer,
NCAR

Nov XX, 2023

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Our Targeted Environment

Derecho:

~2,500 Compute Nodes
~1300 Node Controllers
200 Slingshot Switches

8x Login Nodes
3x PBS / GPFS Quorum Nodes
3x Auxillary Nodes

3x Management Hypervisors
1x Management VM
9x SU-leader Nodes

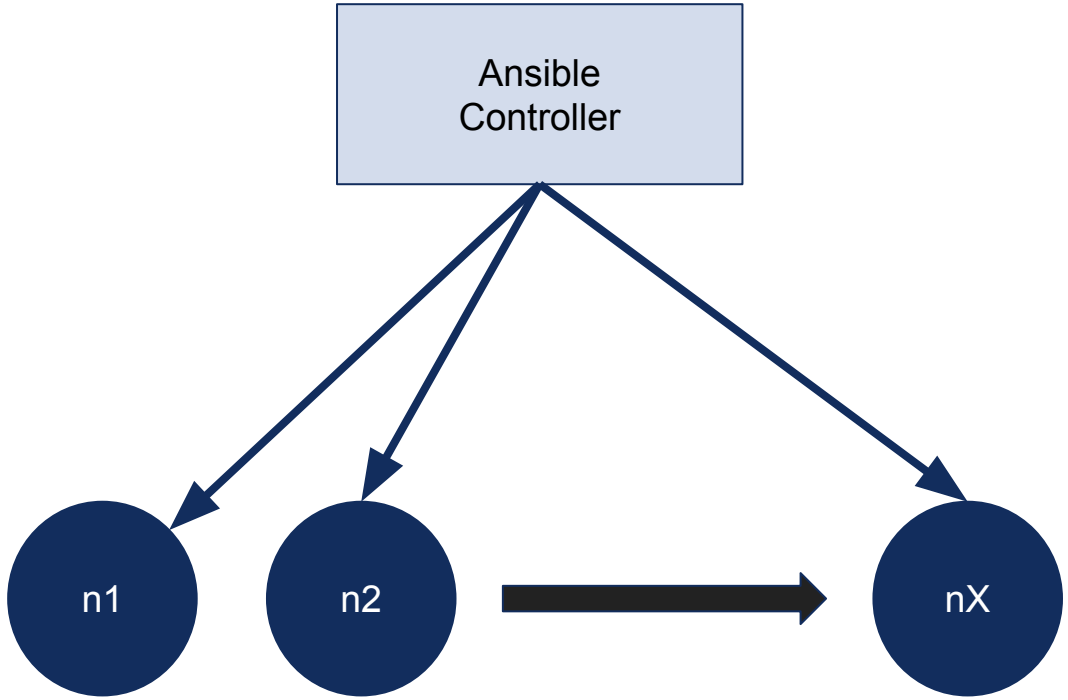
Traditional Ansible Playbook Flow

Traditionally, you have a single Ansible control host and that uses SSH to connect to various hosts and run the plays. Each tasks is an individual SSH connection.

Iterate over nodes until complete using max F forks.

First order optimizations:

- Profile plays / tasks and improve runtime of local commands / hotspots
- Improve connection usage via pipelining

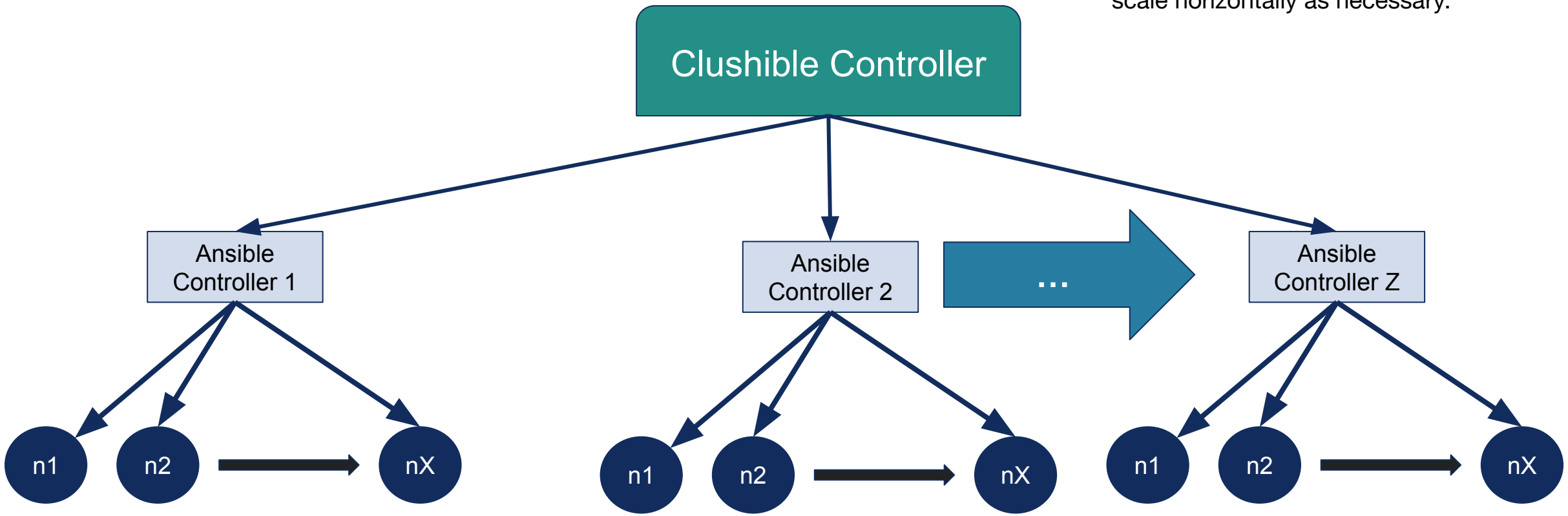


Problem: A single Ansible controller, especially one that resides on our limited VM, only scales so far with forks, pipelining, and using a push method. Thus, either change the way configuration is done (use a pull method) or scale out the number of Ansible controllers. We chose to keep our method of push as to keep secrets out of the images as much as possible and knowing we might have to make changes without forcing reboots.

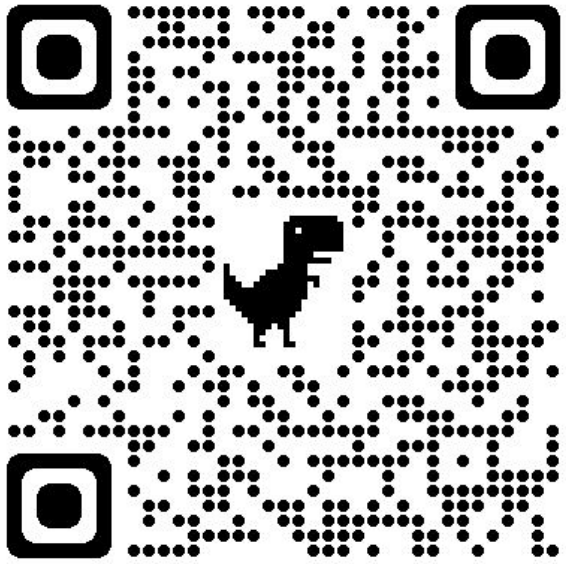
ClusterShell + Ansible = Clushible

Core idea, scale out the number of controller hosts that are used and orchestrate them instead of a single controller… A controller of controllers.

This provides us with an adequate tree concept that can be tuned to fit different sized runners (Ansible Controllers) and scale horizontally as necessary.



Clushible - Tidal Wave Configuration w/ Ansible



Implementation:

ClusterShell: NodeSet, NodeRange, and Remote Exec capability
Ansible: Configuration Management Tool

- Leveraged ClusterShell's ability to group and split node sets into individual host targets for sub-Ansible controllers (runners) and providing the necessary `--limit` expansion required since Ansible lacks any intuitive node ranges.
- Threadpool threads based on number of runners.
- Target number of forks based on runner hardware, but targeted node sets are all roughly the same size.
- Same Ansible playbooks and inventory files used everywhere.

Ansible Runs

Ansible Alone

Individual compute node run takes ~90 seconds. Extrapolating this means ~3855 minutes (~64 hours) in serialized fashions (i.e., absolute worst case scenario)

Perfect world, everything at once, and zero interference: ~90s

However, adding more nodes and depending on the number of forks allowed, a modest collection of hosts eventually increased per host timing went up.

Rough timing during a few emergency outages were roughly 12 hours of configuration time.

Previous speed up actions:

1. Increasing number of forks. Only scales so far with controller hosts
2. Pipelining with Mitogen to minimize target reconnects
3. Changing out callbacks for different output options
4. Attempted previous multiple Ansible runs from same controller.

Clushible: Experience

Clushible with 9 runner nodes

Takes approximately 1 hour

Where is the biggest time spent?

We used 9 runners, each with 192 forks, but that leaves us with a near approximate of 14 (13 full, 1 partial) individual node sets to operate on. Thus, each node set takes approximately 30 min.

Thus, about an hour of configuration time to bring up time.

Summary

Scaling horizontally has been a decent way to speed up our configuration leveraging hardware that was **already** in place.

Doesn't require an overly fancy solution to get the benefits, but certainly needs polish.

I believe this can be sped up more by reducing other parts of Ansible that don't provide value.

There is some additional fine tuning of the math left that would be beneficial

Clushible: Basic Timings

Basic Timings

Number of Nodes	Number of Runners	Time
255	1	1066
256	2	561s
256	9	242
1024	9	977

Next limits have been attributed to FS operations (GPFS cluster rejoin, CrayPE squashfs mounts, etc.)

Clushible Questions

Find Me:

Email: jbaker@ucar.edu

HPC SysPros Slack: @Jared

Github: @jbaksta