Managing cloudy dCache storage pools with Ansible.

Ulf Tigerstedt – NeIC/NDGF-T1/CSC
Hepix fall 2017, KEK, Japan
19.10.2017
tigerstedt@iki.fi
NDGF-T1 disk and tape pools

• 7 sites.
• CentOS (6+7), Gentoo, Ubuntu
• Systems managed by local sysadmins
• Large differences between the sites
• Create once, update config seldom, upgrade often
Local admins

- Updates to dCache or OS requires coordination between dCache admins and local admins, with one or the other initiating the update.
- Updating dCache now takes a day, generally giving each site an hour.
- Mistakes happen.
Solution

GO AWAY

OR I WILL REPLACE YOU WITH A VERY SMALL SHELL SCRIPT
Cloudifying dCache

- Separate out local admin tasks and dCache tasks
- Having root is complicated.
- Systems are wildly different
- dCache runs as nonroot anyway
dCache requirements

- Java, preferably by Oracle
- bash
- logrotate
- tar
Solution: Ansible

- Common configuration and process
- Easy to group by location and hardware
- Fast updates.
- Deploy dCache and java as tar.gz files as
Solution: 3 Ansible roles

- Config: Creates the config files and the basic environment
- Upgrade: Updates or installs dCache and java.
- Createpool: creates the actual pool(s)
Example: Production playbook@NDGF-T1

```yaml
# dcache tar pools hanging on production system
- hosts: dcache-tar-pools-production
  remote_user: "{{ pool_remote_user }}"
  become: false
  roles:
    - { role: dcache-tar-pool-config, tags: ['config'] }
    - { role: dcache-tar-pool-upgrade, tags: ['upgrade'] }
    - { role: dcache-tar-pool-createpool, tags: ['createpool'] }
```
Limits/downsides

- Only for pools, no other services
- Requires ssh access to be open
- Logs might now go to /home instead of /var as previous
Code

- Available from https://github.com/tiggi/ansible-dcache-tarpools
- Documentation and examples will shortly show up.