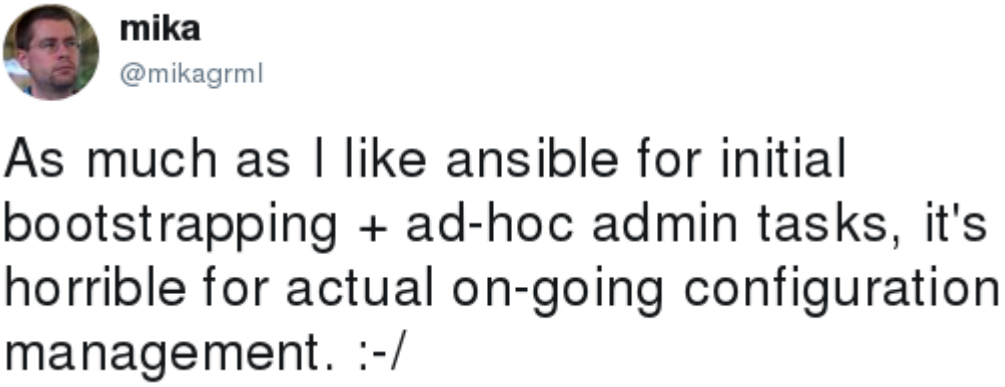
ansible, puppet, continuous, configuration, management, vaults, hiera, jerakia, mcollective, choria

Usage of Ansible for Continuous Configuration Management

Local backup of:

- https://michael-prokop.at/blog/2017/12/16/usage-of-ansible-for-continuous-configuration-management/

It all started with a tweet of mine:



I received quite some feedback since then and I'd like to iterate on this.

I'm a puppet user since ~2008 and since ~2015 also ansible is part of my sysadmin toolbox. Recently certain ansible setups I'm involved in grew faster than I'd like to see, both in terms of managed hosts/services as well as the size of the ansible playbooks. I like ansible for ad hoc tasks, like `*ansible -i ansible_hosts all -m shell -a 'lsb_release -rs*'` to get an overview what distribution release systems are running, requiring only a working SSH connection and python on the client systems. ansible-cmdb provides a nice and simple to use ad hoc host overview without much effort and overhead. I even have puppetdb_to_ansible scripts to query a puppetdb via its API and generate host lists for usage with ansible on-the-fly. Ansible certainly has its use case for e.g. bootstrapping systems, orchestration and handling deployments.

Ansible has an easier learning curve than e.g. puppet and this might seem to be the underlying reason for its usage for tasks it's not really good at. To be more precise: IMO ansible is a bad choice for continuous configuration management. Some observations, though YMMV:

- ansible's vaults are no real replacement for something like puppet's hiera (though Jerakia might mitigate at least the pain regarding data lookups)
- ansible runs are slow, and get slower with every single task you add
- having a push model with ansible instead of pull (like puppet's agent mode) implies you don't get/force regular runs all the time, and your ansible playbooks might just not work anymore once you (have to) touch them again
- the lack of a DSL results in e.g. each single package management having its own module (apt, dnf, yum,....), having too many ways how to do something, resulting more often than not in something I'd tend to call spaghetti code
- the lack of community modules comparable to Puppet's Forge
- the lack of a central DB (like puppetdb) means you can't do something like with puppet's exported resources, which is useful e.g. for central ssh hostkey handling, monitoring checks,...
- the lack of a resources DAG (Direct acyclic graph) in ansible might look like a welcome simplification in the beginning, but its absence is becoming a problem when complexity and requirements grow (example: delete all unmanaged files from a directory)
- it's not easy at all to have ansible run automated and remotely on a couple of hundred hosts without stumbling over anything — Rudolph Bott
- as complexity grows, the limitations of Ansible's (lack of a) language become more maddening — Felix Frank

Let me be clear: I'm in no way saying that puppet doesn't have its problems (side-rant: it took way too long until Debian/stretch was properly supported by puppets' AIO (All In One) packages). I had and still have all my ups and downs with it, though in 2017 and especially since puppet v5 it works fine enough for all my use cases at a diverse set of customers. Whenever I can choose between puppet and ansible for continuous configuration management (without having any host specific restrictions like unsupported architectures, memory limitations,... that puppet wouldn't properly support) I prefer puppet. Ansible can and does exist as a nice addition next to puppet for me, even if MCollective/Choria is available. Ansible has its use cases, just not for continuous configuration management for me.

The hardest part is to leave some tool behind once you reached the end of its scale. Once you feel like a tool takes more effort than it is worth you should take a step back and re-evaluate your choices. And quoting Felix Frank:

```
OTOH, if you bend either tool towards a common goal, you're not playing to its respective strengths.
```

Thanks: Michael Renner and Christian Hofstaedtler for initial proof reading and feedback