SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2016

On-demand Authentication Infrastructure for Test and Development

Andrew Leonard
Dell EMC/Isilon

Agenda

- Static, shared authentication test infrastructure and its pitfalls
- Isilon's implementation of Authentication Provider as a Service (APaaS)
- Lessons learned, future directions

Authentication Test Infrastructure, Defined

Your product supports various authentication providers – e.g. Active Directory, LDAP, Kerberos, NIS, etc.

You want to test this support.

Authentication test infrastructure is the systems that provide endpoints to test against.

Authentication Test Infrastructure, in other words

If you're going to support Active Directory, you're going to need to test your software against a real AD deployment to make sure that it works correctly.

And you may need more than one environment to mimic different configurations (and scale) across your customer base.

Using Static Authentication Providers

Initial solution: Build and run a set of instances of each authentication provider that you need to test.

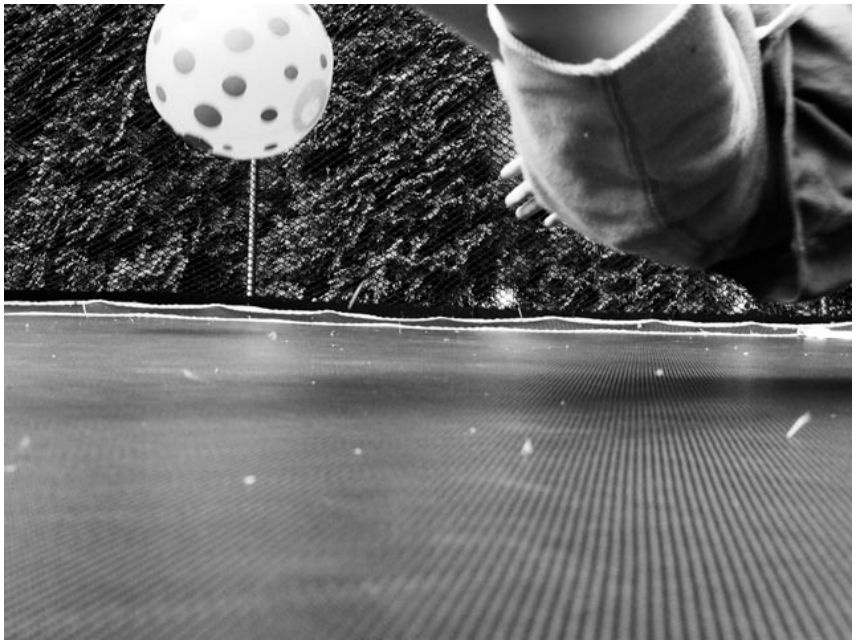
Your tests run, you find and fix bugs, everything is good.

This is how your customers run their authentication providers, most likely.

But then, problems

- The static authentication providers go down, or get corrupted: Maintaining large-scale infrastructure is a large-scale job.
- The underlying infrastructure you run on may not be as reliable as customers use for authentication providers.

More problems with static infrastructure



- Shared, static infrastructure means that tests can (will) contend for resources.
- Static infrastructure problems can make all your tests fail simultaneously – and take a long time to fix.

Even more problems

Development requires authentication providers to develop against... Sometimes in unusual configurations... and may need to (destructively) modify authentication providers.

What about testing how your product responds when an authentication provider becomes unreachable?

Idea: Authentication Provider as a Service (APaaS)

Dynamically deploy authentication providers and supporting services, on-demand, with flexible configuration.

Each test run and developer gets new, fresh infrastructure - no more contention.

Underlying infrastructure goes down? Once it's back up, flatten what you had and repave.

Concrete Example: Active Directory deployed with APaaS

Do exactly what you would do to provision AD normally…

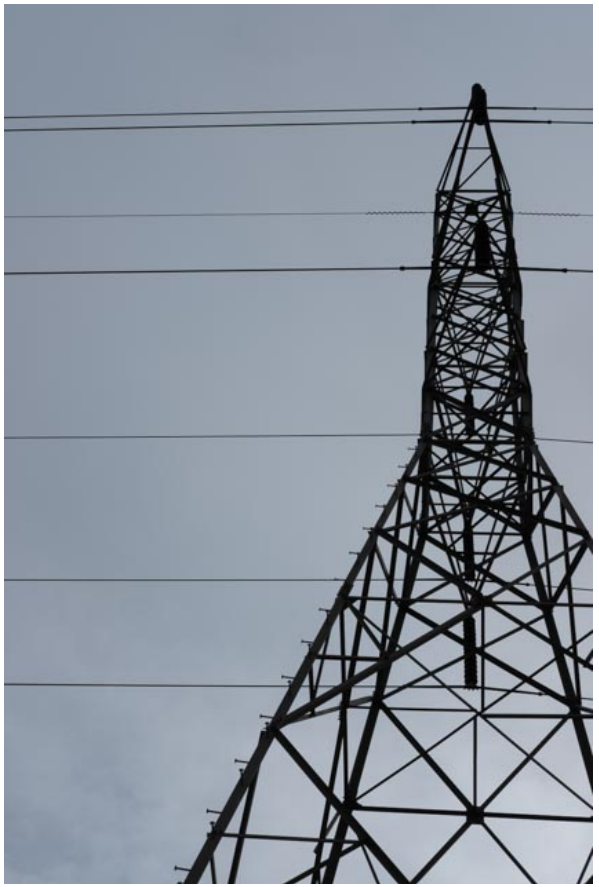
- Create VMs, assign/collect IP addresses
- Create DNS delegation
- Configure VMs - set machine names, configure time services, etc.
- Configure Active Directory - install ADDS, create and promote DCs
- Create users and groups

…just automated and on demand.

Concrete Example: Active Directory deployed with APaaS - continued

And when you're done testing:

- Delete the VMs; and
- Delete the DNS delegation.



Implementation of APaaS

APaaS is primarily built on top of two open source tools:

- Ansible provides scaffolding for configuration and orchestration.
- Consul monitors the health of the authentication providers, and serves as a data store.

Ansible and APaaS in five slides (1)

"Ansible is a free-software platform for configuring and managing computers which combines multi-node software deployment, ad hoc task execution, and configuration management." --Wikipedia

Ansible and APaaS in five slides (2)

Ansible runs from a “control machine” and connects to its clients over SSH or WinRM.

“Playbooks” describe a concrete ordering of provisioning or configuration tasks to carry out on specific hosts, making it easy to “orchestrate” deployments.

Ansible and APaaS in five slides (3)

Ansible has a large collection of batteries-included modules for configuration primitives (installing software, creating VMs, templating files).

When we needed to write our own modules – e.g. for DNS configuration or Active Directory – it was straightforward to do so in Python or PowerShell.

Ansible and APaaS in five slides (4)

There are other, similar tools to Ansible; we chose it for:

- Simplicity of orchestration;
- Ease of extension; and
- No client daemon to install and manage on the clients, and no additional PKI.

Ansible and APaaS in five slides (5)

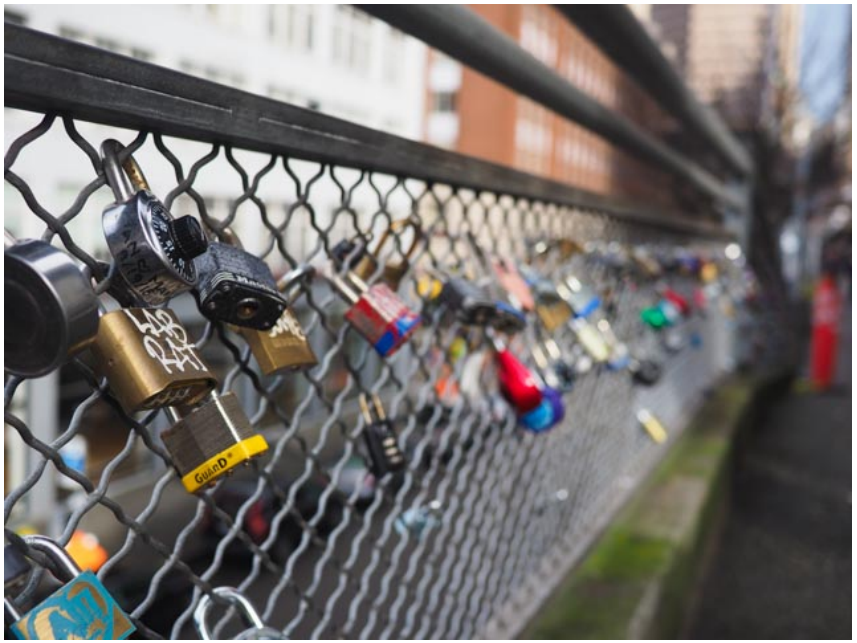
What Ansible does in APaaS:

- (Optionally) deploys VMs
- Configures DNS
- Deploys authentication providers

- And tears it all back down

Consul in APaaS

Consul is an open source product from Hashicorp: It provides service discovery, health checks and key/value storage.



Consul in APaaS (2)

1. Ansible needs an "inventory" - a list of hosts on which to act; APaaS uses Consul's key-value store to hold this information.
2. Nodes built by APaaS report their health back to Consul. Benefit: APaaS can be queried by tests or developers as to whether the environments it built are functioning or not

Tying APaaS Together

- RESTful HTTP front-end built using Python/Flask; simplifies building a UI for humans, and allows automated access for tests.
- Simple, independent backend workers: Fork an Ansible process and clean up the inventory when done.
- RabbitMQ used as an asynchronous queue between front-end and back-end processes.

Lessons Learned Implementing APaaS

Reducing test noise is important – and static infrastructure can create a lot of noise.

Lessons Learned Implementing APaaS

VM templates matter: Buggy or inconsistent templates can cause automated deployments to fail before they start.

Recommendation: Automate and test your template build process.

Lessons Learned Implementing APaaS

With dynamic infrastructure, health checks and monitoring are as important as they are with static infrastructure.

Future Directions for APaaS

- More complexity, more authentication providers
  - Active Directory trusts
  - Kerberos, NIS
- Other third-party non-authentication providers
  - Antivirus
  - Audit logging

Future Directions for APaaS (2)

Docker: Decrease deployment time from minutes to seconds, take advantage of existing container management tools.

Questions?

Thanks!