Salt To Ansible Migration Guide

Copyright

First edition, 2026

Introduction

I'm writing this in the second half of 2025 soon after retiring with 35 years in software development. I used Salt extensively for the last 7 years, and before that I had been using Puppet for 7 years. I was often involved in tasks that eventually became known as DevOps, mostly focusing on configuration management at smaller companies.

I became disillusioned with Salt after the company was sold to VMware which itself was later acquired by Broadcom. Now in late 2025, Salt isn't getting much attention or support from its current owner. Many of the core Salt developers have left the company, and the cadence of bug fixes and releases seem to be slowing down Also before retiring, I looked at the job market and found almost nobody is looking for people with Salt experience while at least some people were still looking for Ansible experience

With that in mind, I decided to learn Ansible by porting the Salt states for my home lab over to Ansible. I kept notes along the way, and thought other developers might benefit if I published what I learned.

Please let me know ~~if~~ when you find errors in the text, so I can fix them in future editions. Any errors are of course my own fault.

If you want to support me more than paid for this book (if you paid for it at all), you can 'buy me some Ceylon tea' with the QR code below. If you're a software developer, working on a migration project covered in this book, and you have some extra unused training allowance dollars going to waste, maybe send some of them my way. This book should save you at least a day of work, so maybe send me half a day's pay per developer on you team my way. Those tea tariffs won't pay themselves.

Enjoy!

Audience

This book is geared towards three groups of readers.

The first group is Salt users who want to quickly get up to speed with Ansible because they are porting over their existing Salt states to Ansible tasks for whatever reason. Maybe they just don't want to be pigeon-holed into knowing one provisioning system. Maybe their company is forcing a migration to Ansible as one of its number one priorities this quarter. Maybe Salt's current owner is doing something new and innovative to drive away users.

The second group is users who don't know Salt and have inherited some Salt states that they need to port over to Ansible. If you aren't already familiar with Salt, check out Appendix A.

The third group has a size of one - myself. I'm writing this to prove to myself that I understand how I migrated my states well enough to document it as per the Feynman technique.

I assume all the groups have a good understanding of Salt, YAML, the Jinja2 templating system, and at least enough programming experience to understand Jinja2 variables, conditionals, and looping constructs. I'm also assuming readers have access to the Salt and Ansible documentation sites so they can look up more info on the many Salt and Ansible basics I'll be glossing over.

The purpose of the book is to show you a migration path, not to teach you Salt or Ansible from scratch.

I assume you're using git for version control, and you have some system in place for tracking issues and bugs.

Acknowledgements

The cover photo is from https://pxhere.com/en/photo/777522. According to the site it has a CC0 Public Domain license, but didn't include the name of the photographer. Thanks to whoever took the picture, and thanks for putting it in the public domain.

I used Obsidian 1.10.6 for editing the book's markup and generating the pdf, and Dia 0.97.3 for the diagrams.

Disclaimers

Software Versions

During my migration work, Salt was at version 3007.10 and Ansible was at version 2.18.6. The content below may be obsolete if there are newer versions of these tools available by the time you're reading this. *Sic transit gloria mundi*[1].

Limitations

There a number of features of the Salt ecosystem this book currently doesn't cover such as

- No coverage of Salt reactors
- No coverage of Salt mines

- No coverage of custom Salt facts
- No coverage of custom Salt states
- Only salt.pillar.file_tree is covered for ext_pillar migration
- No coverage of Salt or Ansible for Windows (all my hosts are using Debian or Ubuntu)
- SELinux-specific issues (my hosts are all in permissive mode)
- No coverage of porting Salt-specific unit tests over to Ansible.
- No coverage of scaling Ansible to large networks. I'm assuming the number of hosts in your network is probably less than a hundred.

Owners of this book should receive free updates whenever there are new releases that cover these topics.

AI Usage

I didn't 'vibe code' my Ansible migration project, but I did use chatGPT a fair amount, especially for finding initial Salt->Ansible equivalents, and asking for explanations for various ansible-playbook and ansible-lint error messages.

As for this book, it was written all by me, a baseline human in late 2025, with no neural implants. I do have a dumb, artificial lens in my left eye after cataract surgery. If I cover my right eye everything is clear, but if I cover my left eye, everything looks like an old, sepia tone photo. That's a little freaky, but I don't think this has impacted my writing much.

Release History

Release	Date	Notes	
0.5	2026-01-15	(limited release)	
1.0	2026-01-19	Initial public release, mostly to see if LeanPub will display the PDF's TOC. (It doesn't)	

Migration Guiding Principles

Here are the principles I tried to adhere to while working on the migration. These hold for pretty much any migration project, and aren't really specific to Salt or Ansible.

Minimize external dependencies

Early on I decided to try to migrate as many of the Salt states as possible using vanilla Ansible `ansible.builtin.*` roles as much as possible. I avoided picking community-built roles from the `galaxy.ansible.com` site since I didn't know Ansible well enough to know how to choose well-written and well-maintained roles.

I also didn't want to introduce unneeded dependencies on my Ansible roles if I didn't really want to have to maintain. I also wasn't sure how much of a hassle it would be to find out when new versions of the community-built roles were available.

In general, it's better to avoid dependencies as much as possible.

Validate data as much as possible.

One of the big pain points in Salt was configuring the pillar data files correctly for new hosts. Salt had no native means of validating pillar data other than Jinja2 checks. It was pretty common for folks writing the pillar files to introduce indentation or spelling errors in the pilllar files that Salt would then ignore.

I was excited to see Ansible's support for JSON Schema for data validating and the argument_specs.yml support for validating role arguments. I incorporated these checks into the migration process below, and they were useful in pointing out some typos during development.

I was also please to see that Ansible had a pretty nice linter called ansible-lint, which I incorporated into my workflow. I managed to avoid having to disable any of the ansible-lint checks.

Avoid refactoring while migrating

Your current Salt states have probably been in use for some time and have been tested in production.

With that in mind, I tried to minimize refactoring the provisioning code as it is being migrated, sticking with a one-for-one migration of Salt state to Ansible task as much as possible, so I didn't end up losing any of the hard-earned provisioning steps that may only be captured in the Salt states.

That said, if I saw the current Salt states aren't "battening down the hatches" as tightly as they could be (say forgetting to set a mode on a file), I went ahead and added the missing