Lessons learned from maintaining parts of Ansible

Felix Fontein

May 19th, 2026

Who am I?

Felix Fontein ('feːlɪks 'fɔntaɪ̯n)

✉ felix@fontein.de
felixfontein:matrix.org (Matrix)
@felixfontein
Software Engineer @ Plexim GmbH

Ansible user and contributor

- user since around 2015
- active contributor since 2018
- (co-)maintainer of several community collections, among them community.general
- member of Ansible Community Steering Committee

Roadmap

What is Ansible?

An orchestration tool

- goes through lists of tasks
- tasks are mostly idempotent actions
  - → template files, install packages, set up DNS records, configure web services, ...

What is Ansible?

An orchestration tool

- goes through lists of tasks
- tasks are mostly idempotent actions
  - → template files, install packages, set up DNS records, configure web services, ...

What you need to know

- Ansible is written in Python
- actions are Python scripts called **modules**
- Ansible has "**batteries included**"

History, and maintaining all of it (1/n)

- Started in 2012 by Michael DeHaan as an open source tool for provisioning servers

History, and maintaining all of it (1/n)

- Started in 2012 by Michael DeHaan as an open source tool for provisioning servers
- As it became more popular, many more modules were added
  - → most of them contributed by users

History, and maintaining all of it (1/n)

- Started in 2012 by Michael DeHaan as an open source tool for provisioning servers
- As it became more popular, many more modules were added
  - → most of them contributed by users
- Quickly led to problems:
  - → everything in one repository (`github.com/ansible/ansible`)
  - → Ansible is also a commercial product, but not everything included is (commercially) Supported

History, and maintaining all of it (2/n)

Multiple approaches to tackle these problems

- Use **Git submodules** for supported and unsupported stuff
  $\rightarrow$ got merged back into one repo later

History, and maintaining all of it (2/n)

Multiple approaches to tackle these problems

- Use **Git submodules** for supported and unsupported stuff
  → got merged back into one repo later
- Bot giving **module (group) maintainers** permissions (merge, close issues, ...)
  → less burden on folks with commit rights

History, and maintaining all of it (2/n)

Multiple approaches to tackle these problems

- Use **Git submodules** for supported and unsupported stuff
  → got merged back into one repo later
- Bot giving **module (group) maintainers** permissions (merge, close issues, ...)
  → less burden on folks with commit rights
- **Ansible Galaxy** for publishing roles (with extra modules/plugins)

History, and maintaining all of it (2/n)

Multiple approaches to tackle these problems

- Use **Git submodules** for supported and unsupported stuff
  → got merged back into one repo later
- Bot giving **module (group) maintainers** permissions (merge, close issues, ...)
  → less burden on folks with commit rights
- **Ansible Galaxy** for publishing roles (with extra modules/plugins)
- **Ansible collections**: name-spaced bundle of Ansible content

History, and maintaining all of it (3/n)

2020: The Big Collection Split (March 9th, 2020)

- Split up big `ansible` package into `ansible-base` (later `ansible-core`) and collections
- Most modules and plugins (around 4400) were moved to collections

History, and maintaining all of it (3/n)

2020: The Big Collection Split (March 9th, 2020)

- Split up big `ansible` package into `ansible-base` (later `ansible-core`) and collections
- Most modules and plugins (around 4400) were moved to collections
  - content with active maintainer groups (both Supported + community) got their own collections
  - **everything else** → `community.general`: 1325 modules, hundreds of plugins

History, and maintaining all of it (4/n)

community.general

- “dumping ground”
- basic idea: bot allows community to maintain things
- not clear who does other (general) work (maybe Ansible community team?)

History, and maintaining all of it (4/n)

community.general

- “dumping ground”
- basic idea: bot allows community to maintain things
- not clear who does other (general) work (maybe Ansible community team?)
- some modules I was helping with ended up in community.general (c.g), so I got involved...

History, and maintaining all of it (5/n)

First steps for community.general

- get CI working
- figure out how to do changelogs
- figure out how to do versioning and releasing
- split off more community collections
  - community.network (that was a large chunk of content)
  - several smaller ones
  - some of them died quickly, others are still around today

History, and maintaining all of it (6/n)

community.general 1.0.0 (July 31st, 2020)

- Roughly half of content that got initially merged is gone
- There is a fixed release schedule
  - major releases every 6 months
  - minor releases every 2 months
- Few human collection maintainers (like me), lots of module maintainers (through bot), lots of unmaintained stuff

History, and maintaining all of it (7/n)

How does the bot work

- BOTMETA.yml assigns maintainers to every file in collection
- PR needs two **`shipit`**s from maintainers or three **`shipit`**s from community members
- If PR author is maintainer, PR has implicitly already one **`shipit`**
- Users with commit rights count as maintainers for everything

History, and maintaining all of it (8/n)

Does the bot work well?

- **No.**
- In the beginning more technical problems (like files preventing merges)

History, and maintaining all of it (8/n)

Does the bot work well?

- **No.**
- In the beginning more technical problems (like files preventing merges)
- There are too few active module maintainers
- For many modules, no active maintainers at all

History, and maintaining all of it (8/n)

Does the bot work well?

- **No.**
- In the beginning more technical problems (like files preventing merges)
- There are too few active module maintainers
- For many modules, no active maintainers at all
- So most reviewing and all merging is done by collection maintainers

History, and maintaining all of it (8/n)

Does the bot work well?

- **No.**
- In the beginning more technical problems (like files preventing merges)
- There are too few active module maintainers
- For many modules, no active maintainers at all
- So most reviewing and all merging is done by collection maintainers
- Haven't seen a bot-merged PR for years now...

History, and maintaining all of it (9/n)

Further problem: test coverage

- Many older modules have no tests at all
- Since a few years, new modules must come with tests

History, and maintaining all of it (9/n)

Further problem: test coverage

- Many older modules have no tests at all
- Since a few years, new modules must come with tests

Further problem: still working?

- Sometimes it is unclear whether modules actually still work
- We sometimes notice when hard-coded DNS names no longer resolve
- There is no telemetry for Ansible content

History, and maintaining all of it (10/n)

Couldn't Red Hat...?

- Red Hat sells support for Ansible
- This covers Supported collections
- All `community.*` collections are **not** Supported
- (Doesn't mean that Red Hat employees aren't sometimes working on them)

History, and maintaining all of it (10/n)

Couldn't Red Hat...?

- Red Hat sells support for Ansible
- This covers Supported collections
- All `community.*` collections are **not** Supported
- (Doesn't mean that Red Hat employees aren't sometimes working on them)

Community collections are community supported

Community = everyone using/interested in Ansible

(that includes **you** and **me**!)

What do users expect? (1/n)

Some expect too much, obviously ;-)

Basic expectations:

- Bugs are fixed.
- Backwards compatibility.
- New features.
- Releases happen when needed.
- Help (support) is provided.

What do users expect? (2/n)

When creating issues

- Help questions: **someone** reads them, and responds
- Bug reports: **someone** reads them, triages them, and hopefully fixes them
- Feature requests: **someone** reads them, triages them, and hopefully implements them

What do users expect? (3/n)

When creating PRs

- There are **tests** that tell if they break anything
- **Someone** reviews them (do they make sense, are they correct)
- **Someone** eventually merges them.

When asking for help

- There is a **place** where users can ask for help
- **Someone** reads them, and tries to answer

What can we/I/you deliver? (1/n)

This is one of the **most fundamental questions** for all open source projects!

(Next to choosing license, actually writing the code, ... :-))

Generally this depends on:

- How many resources do you have?
- How many users do you have?
- How demanding are your users?
- How broad is the project?

What can we/I/you deliver? (2/n)

Let's try to answer this for community.general:

- How many resources do you have?
  → **way too few**

What can we/I/you deliver? (2/n)

Let's try to answer this for community.general:

- How many resources do you have?
  → **way too few**
- How many users do you have?
  → **a lot**

What can we/I/you deliver? (2/n)

Let's try to answer this for community.general:

- How many resources do you have?
  → **way too few**
- How many users do you have?
  → **a lot**
- How demanding are your users?
  → **mixed**, most only contact when encountering bugs or missing features

What can we/I/you deliver? (2/n)

Let's try to answer this for community.general:

- How many resources do you have?
  → **way too few**
- How many users do you have?
  → **a lot**
- How demanding are your users?
  → **mixed**, most only contact when encountering bugs or missing features
- How broad is the project?
  → **extremely**

What can we/I/you deliver? (3/n)

How can we try to maximize usefulness without investing too many resources?

- I have limited resources
  (I don't want to burn out, I also have a family, a job, friends, and other hobbies)
- I cannot otherwise increase the number of resources

What can we/I/you deliver? (3/n)

How can we try to maximize usefulness without investing too many resources?

- I have limited resources
  (I don't want to burn out, I also have a family, a job, friends, and other hobbies)
- I cannot otherwise increase the number of resources
- **Help people to help themselves**
  (Hilfe zur Selbsthilfe)

What can we/I/you deliver? (4/n)

Enable other people to help

What can we/I/you deliver? (4/n)

Enable other people to help

Remember, when you try to contribute...

- ... and you don't know your way around
- ... CI is broken (unrelated to your PR)
- ... your PR gets ignored
- ... your PR never gets merged
- ... there are no new releases

What can we/I/you deliver? (4/n)

Enable other people to help

How to do that?

- Point users/contributors to resources
- Help maintaining CI
- Help keeping up minimum standards in PRs
- Help getting PRs merged
- Help getting releases out

What can we/I/you deliver? (5/n)

So what do I do for community.general?

Issues

- I subscribe to all notifications so I know about new issues
- I quickly look at every new issue
- Check classification (bug / feature request)
- Point people looking for help to Ansible Forum
- Bot usually links to source of module/plugin
- If I have an idea, I write a (short) comment

What can we/I/you deliver? (6/n)

So what do I do for community.general?

PRs

- Thank people for their contribution
- I mainly look at formal things
  - changelog fragment
  - documentation and code standards
  - backwards compatibility
  - does change look sensible without knowing the tool
- "If nobody objects, I'll merge this in ∼a week"
- Close inactive PRs (or let bot do it)

What can we/I/you deliver? (7/n)

So what do I do for community.general?

Maintain CI

- Update CI matrix for new ansible-core versions
- Update CI matrix when ansible-core adds or removes support for specific Python versions
- Update CI matrix when ansible-core adds or removes support for specific CI target platforms
- Look at failing nightly builds

What can we/I/you deliver? (8/n)

So what do I do for community.general?

Releases

- I decide which PR gets backported to which stable branch
- Patchback bot does the backporting
  - If there are conflicts, it's up to someone else to create a backport
  - I only create manual backports if I consider something important enough
- I do regular releases (every four weeks)

What can we/I/you deliver? (8/n)

So what do I do for community.general?

Resources

- In **average** maybe 1-2 hours per day?
- That's 15-30% of a full-time job (42h/week)

What can we/I/you deliver? (8/n)

So what do I do for community.general?

Resources

- In **average** maybe 1-2 hours per day?
- That's 15-30% of a full-time job (42h/week)
- Disclaimer: **I do not have kids**
- Also I'm not the only one doing this

What can we/I/you deliver? (9/n)

So what do I do for community.general?

Is that enough?

- Number of open PRs roughly bounded
- Most PRs get merged
  (∼5600 PRs merged, ∼650 PRs closed)
- Most (PR) contributors seem happy
- Number of open issues is increasing

What can we/I/you deliver? (10/n)

How does this look like in times of AI?

- Alexei Znamensky (co-maintainer) is currently trying to **bring down the number of open issues** with the help of Claude Code

What can we/I/you deliver? (10/n)

How does this look like in times of AI?

- Alexei Znamensky (co-maintainer) is currently trying to **bring down the number of open issues** with the help of Claude Code
- We can try to provide help to contributor's AIs to polish contributions before we first see them

What can we/I/you deliver? (10/n)

How does this look like in times of AI?

- Alexei Znamensky (co-maintainer) is currently trying to **bring down the number of open issues** with the help of Claude Code
- We can try to provide help to contributor's AIs to polish contributions before we first see them
- AIs could help with **reviewing PRs** (especially looking at formal things and common mistakes)

What can we/I/you deliver? (10/n)

How does this look like in times of AI?

- Alexei Znamensky (co-maintainer) is currently trying to **bring down the number of open issues** with the help of Claude Code
- We can try to provide help to contributor's AIs to polish contributions before we first see them
- AIs could help with **reviewing PRs** (especially looking at formal things and common mistakes)
- AIs could help **maintaining CI**

What can we/I/you deliver? (11/n)

What could be highly problematic?

- AIs could try to help user support questions
- AIs could improve/extend documentation
- AIs could try to fix bugs automatically
- AIs could extend tests or produce missing tests

Someone with domain knowledge needed to review!
(And we don't have such folks most of the time!)

Conclusions

- Help people to help themselves
- Figuring out minimum work necessary to do that is important
- Bots and AIs can help
- Remember: this isn't a job, and don't burn out!

Thanks to all active community.general contributors!

In particular:

- Amin Vakil (ex-maintainer)
- Andrew Pantuso (also helped for some time)
- Alexei Znamensky (active for almost five years now!)
- John Barker, Andrew Klychkov, and Abhijeet Kasurde from Red Hat
- Nejc Habjan (`gitlab_*` modules)
- ... and many more :-)

Thank You for your attention!

Questions? Comments?