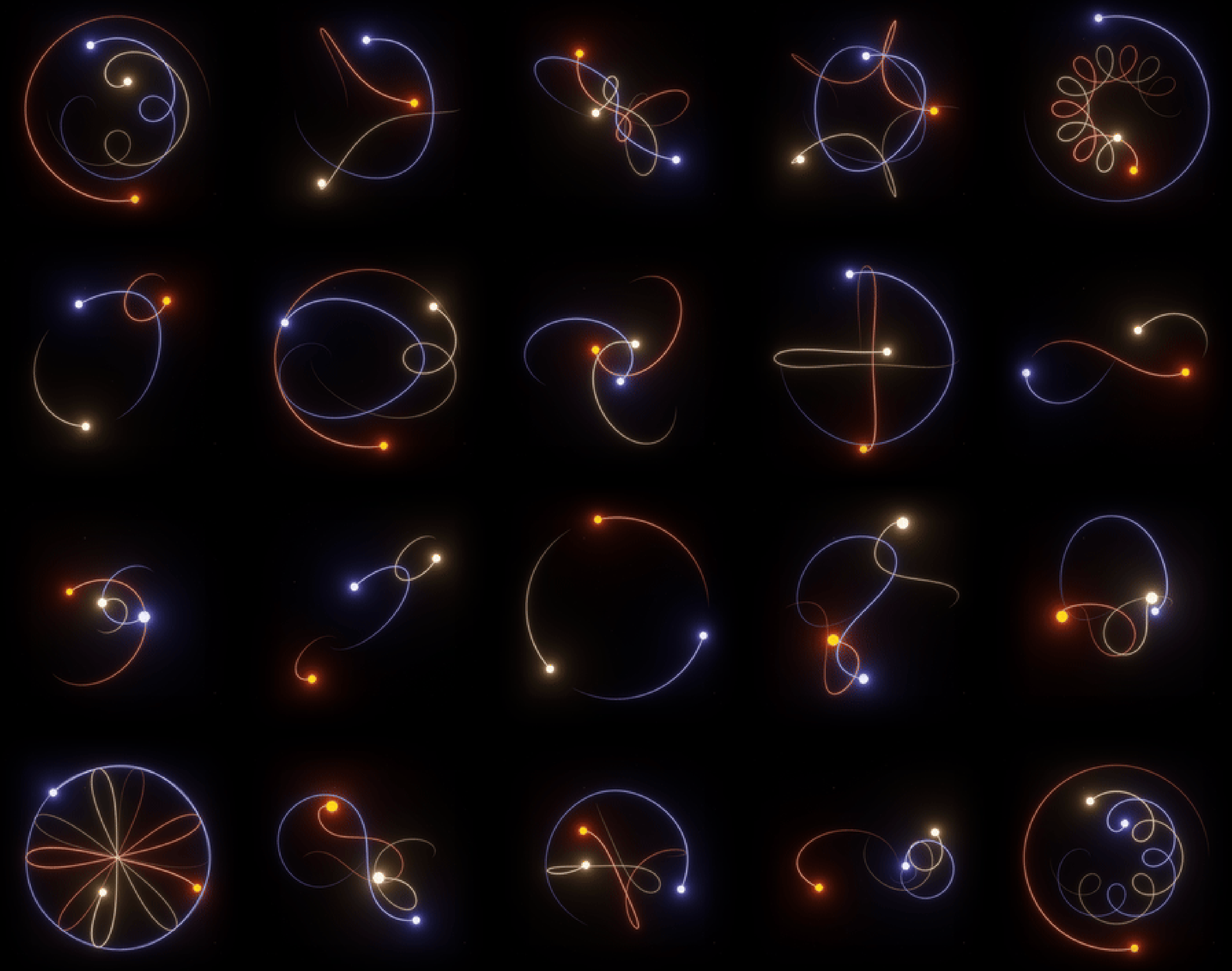
The 3 AVD Containers Problem

Three Containers - Endless Possibilities
Petr Ankudinov
Jan 2024



Origins

- Arista AVD collection can be installed manually
    - Very feasible in many cases and not going anywhere
    - Do this at your own risk. You may encounter weird problems, especially if environment has some history
- AVD all-in-one container
    - Was never officially documented or advertised, but quite actively used anyway. Around 60K total downloads so far.
    - Can be used as dev container (with some modifications) or standalone
    - It was never integrated with AVD CI and must be manually updated on every release
    - A lot of complexity to maintain
    - No plans to support it long term
- Ansible Automation Platform
    - Out of scope. For customers heavily relying on RedHat support and internal Ansible ecosystem

Motivation

- Better integration with AVD with automated image build on every release
- Must be documented and known to AVD users
- Reuse work done by Microsoft. It's not perfect for every use case, but quite a few developers are working on dev container features. Their contribution is appreciated and must not be wasted
- Better VSCode integration

> **WARNING**: AVD dev containers are in the preview phase. They are working well, but breaking changes can happen and they must not be advertised to customers as fully supported solution.

AVD Dev Containers

```
python:3.{9-11}-slim-bullseye
│
│               <--- trigger CI on every AVD release, pre-release, merge
│
└> base         :python3.{9-11}, latest
   │
   ├> dev       :python{9-11}, latest
   └> universal :python3.{9-11}-avd-devel, python3.{9-11}-avd-v4.5.0, latest
```

- Common use cases:
  - `base` - not to be used directly, base for all other images
  - `dev` - AVD contributors and testing new features/branches. AVD collection is not pre-installed
  - `universal` - AVD collection is pre-installed, ready to use
- All containers are multi-platform - linux/arm64, linux/amd64