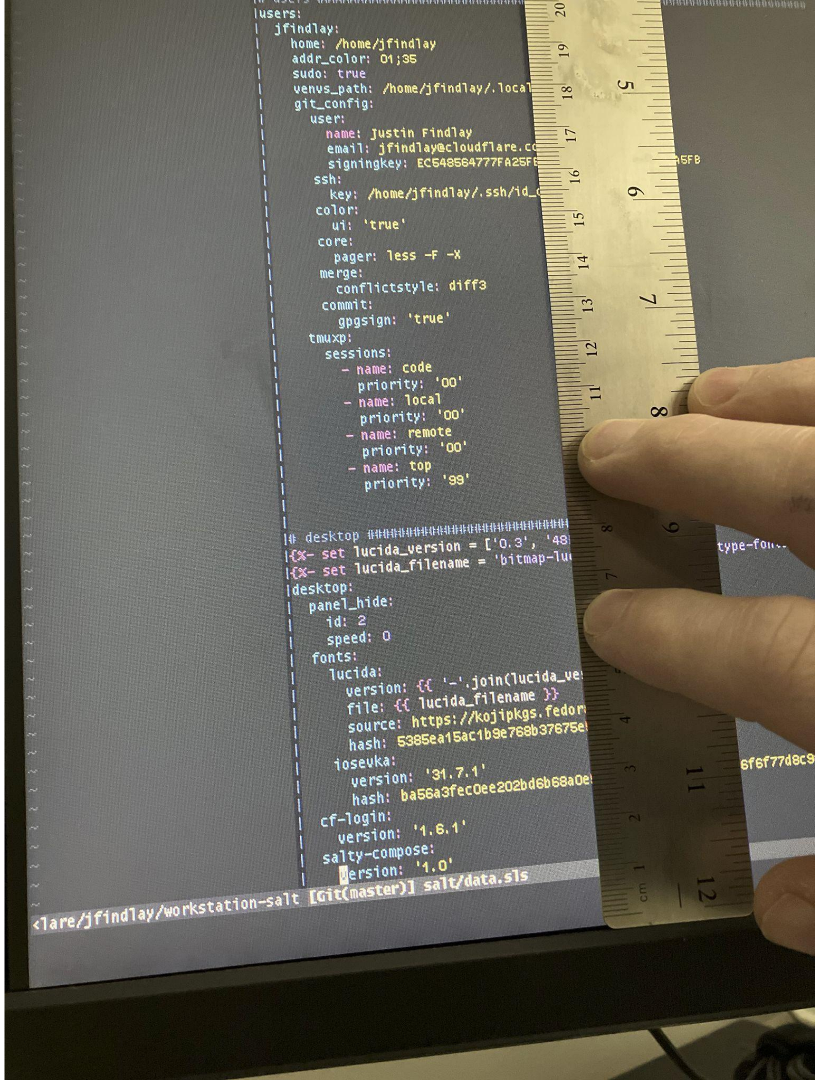
Justin Findlay
Systems Engineer

Against yaml+jinja

Really

- YamlCamp? JinjaCamp? (YAML users support group)
- Everyone has an opinion already, me too
- in 5 minutes I will try to explain **one of** the motivations to migrate from `jinja|yaml`



```
users:
  jfindlay:
    home: /home/jfindlay
    addr_color: 01;35
    sudo: true
    venvs_path: /home/jfindlay/.local
    git_config:
      user:
        name: Justin Findlay
        email: jfindlay@cloudflare.c
        signingkey: EC548564777FA25F
      ssh:
        key: /home/jfindlay/.ssh/id_
      color:
        ui: 'true'
      core:
        pager: less -F -X
      merge:
        conflictstyle: diff3
      commit:
        gpgsign: 'true'
    tmuxp:
      sessions:
        - name: code
          priority: '00'
        - name: local
          priority: '00'
        - name: remote
          priority: '00'
        - name: top
          priority: '99'


# desktop ########################################
{%- set lucida_version = ['0.3', '48
{%- set lucida_filename = 'bitmap-lu
desktop:
  panel_hide:
    id: 2
    speed: 0
  fonts:
    lucida:
      version: {{ '-'.join(lucida_ve
      file: {{ lucida_filename }}
      source: https://kojipkgs.fedor
      hash: 5385ea15ac1b9e768b37675e
    iosevka:
      version: '31.7.1'
      hash: ba56a3fec0ee202bd6b68a0e
  cf-login:
    version: '1.6.1'
  salty-compose:
    version: '1.0'
<lare/jfindlay/workstation-salt [Git(master)] salt/data.sls
```

Backstory

When I joined SaltStack in 2014

me: Whither salt DSL?

Salt OG devs: Salt doesn't have a DSL

Users can BYO DSL, https://docs.saltproject.io/en/latest/ref/renderers/all/

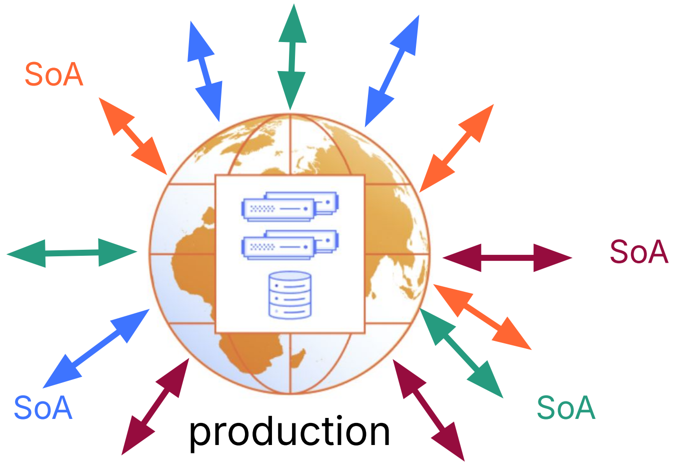
Salt users: `jinja`|`yaml` is the DSL

me: uh, can we fix DSL? Add a version field and create v2.0 fixing any problems we've found?

Salt OG devs: https://docs.saltproject.io/en/latest/ref/renderers/all/

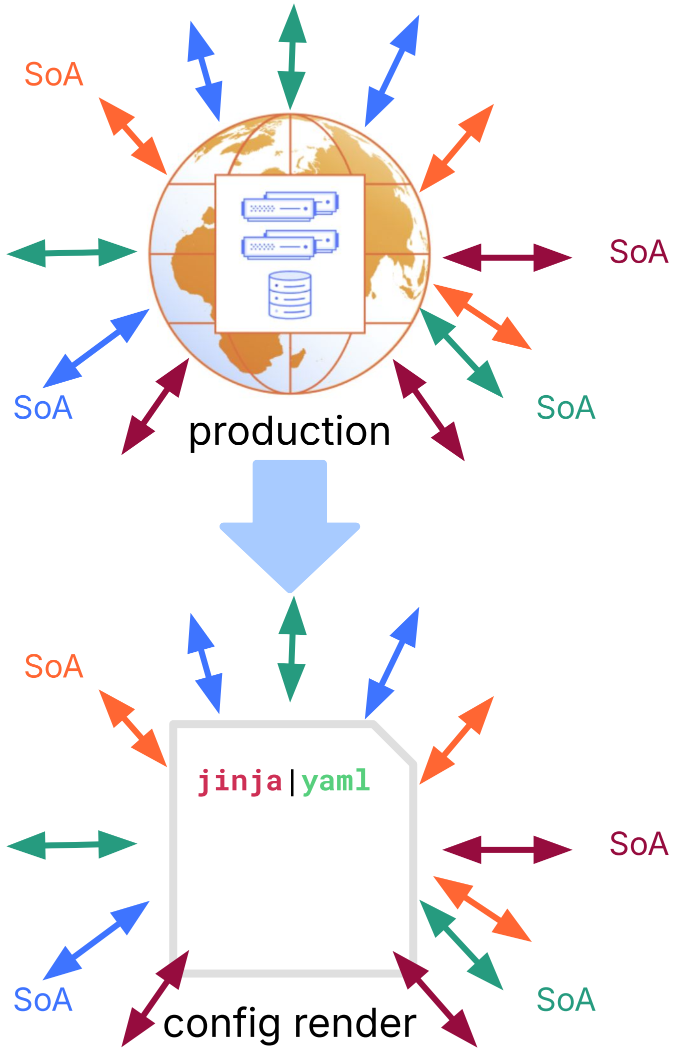
Config shift

- The evolution of production tends towards complexity with **many** sources of **authoritative information** (SoA) (Vault, NetBox, etc.) or other **essential systems**



Config shift

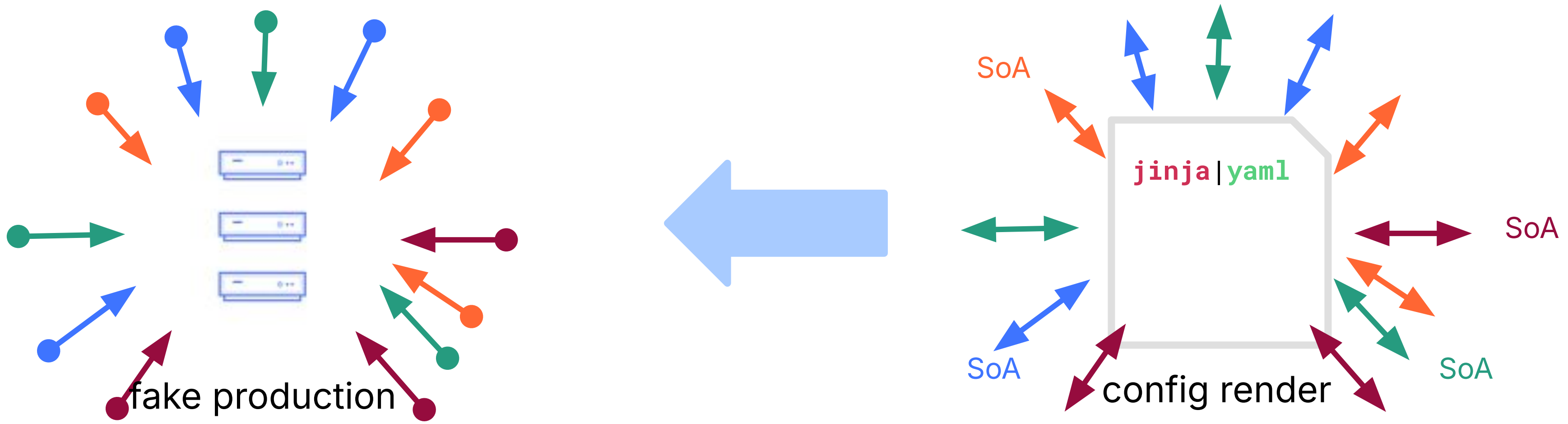
- The evolution of production tends towards complexity with **many** sources of **authoritative information** (SoA) (Vault, NetBox, etc.) or other **essential systems**
- That complexity will naturally **manifest** in `jinja|yaml` JIT-style config management



Config shift

Production configs can become impossible to replicate leftwards in integration (CI) or development

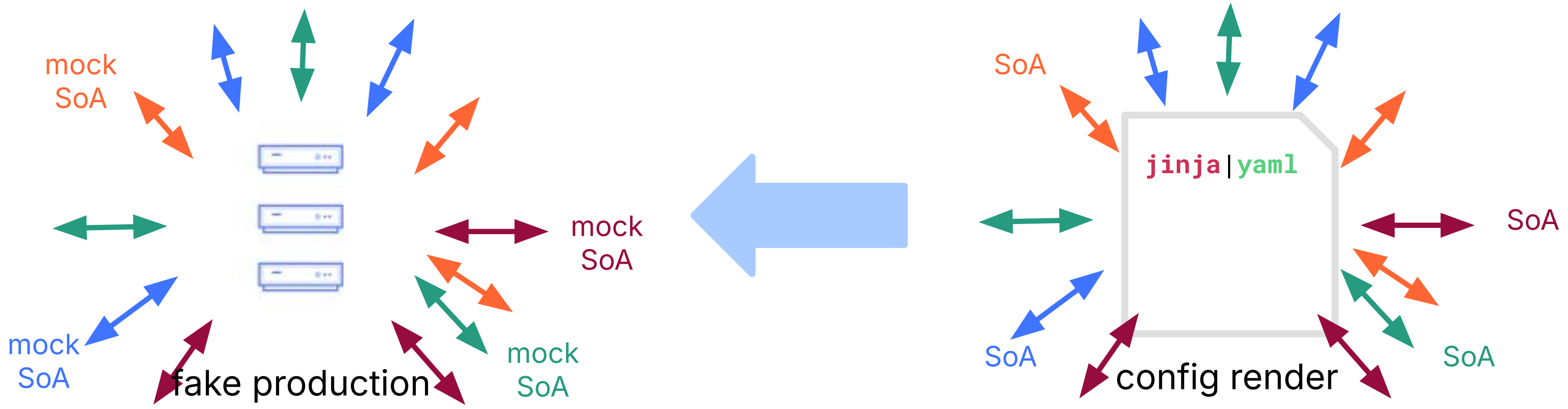
- Users have to add conditional environment logic to stub dev/CI values



Config shift

Production configs can become impossible to replicate leftwards in integration (CI) or development

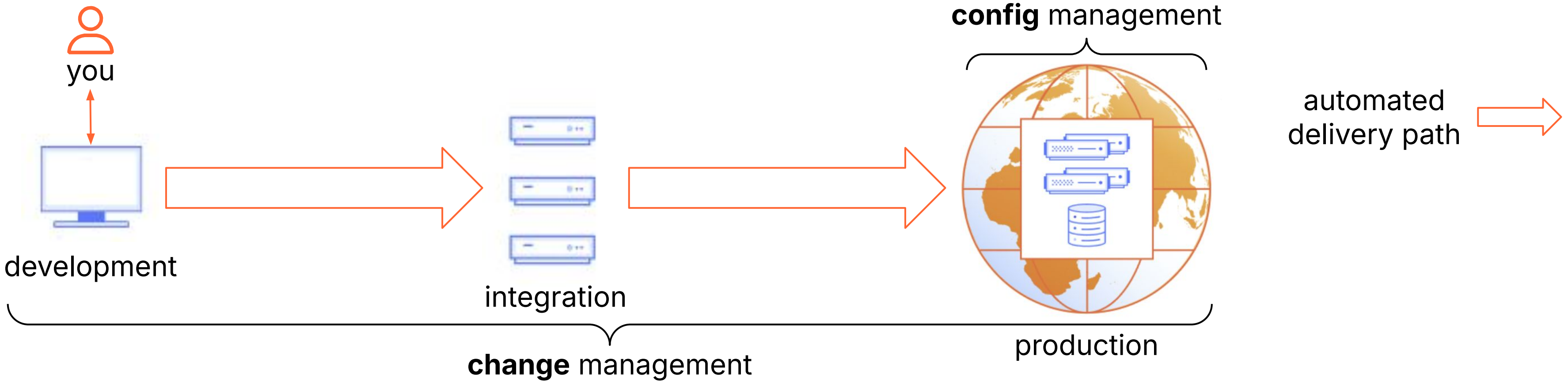
- Or mock each production SoA (good luck with that)



Config shift

Claim: **Config** management is not **change** management; it is actually **state enforcement**

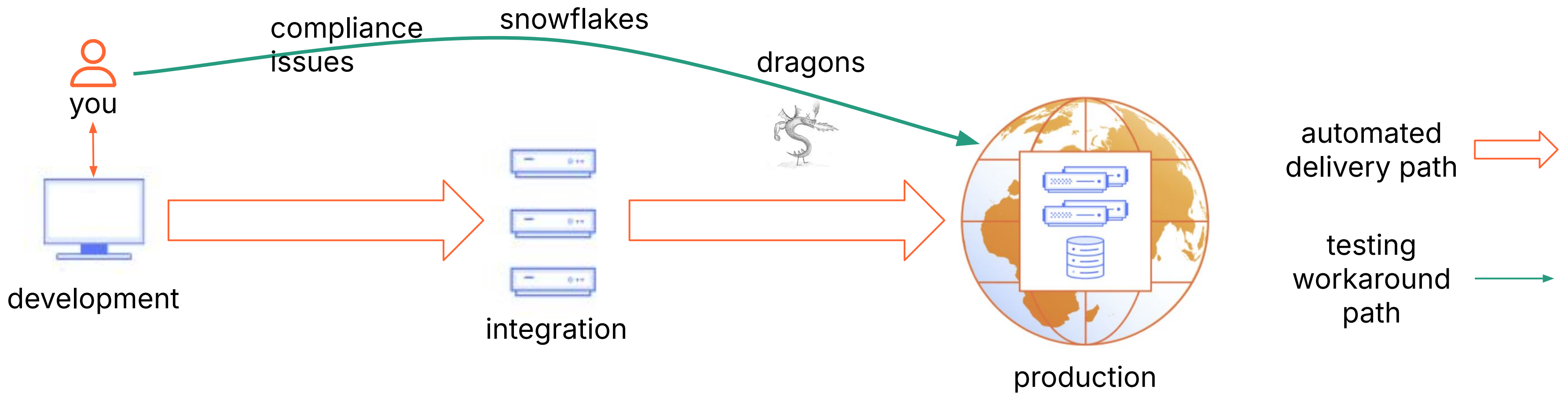
- It is the **right**most **shift**ed part of the delivery (CD) pipeline



Config shift

Claim: **Config** management is not **change** management; it is actually **state enforcement**

- Users having to dry run in (a prescribed subset of) production to get config feedback is problematic



Config shift

If you don't know the configs you are supposed to be managing until they materialize in production

- Your changes are **not actually mediated by your CD**
- Users **can't test or iterate outside of a production context**
- Thus changes are either **slow** or **dangerous**

Have your waffle and eat it too

- Salt's bias for **flexibility** and **difficult learning curve** means users inevitably write **unsure copypasta** or **inscrutable abstractions**
- Can **config shift left** into dev/CI while **CI** workloads **remain constant** and **production** workloads **continue to scale**?

Have your waffle and eat it too

Design goals:

1. Give users authentic dev/CI config
2. Maintain `O(config)` CD complexity

Have your waffle and eat it too

2 | Maintain `O(config)` CD complexity

What does that even mean?

- `config` is a measure of the **size** of **pillar** or the **preprocessing** needed for **highstate**
- **Shifting config** rendering **left** in CD takes the workload off from salt minions (**very many**) and salt masters (**many**) to CI (**few**)
- Isn't that as **antiscalable** as you can get?

Have your waffle and eat it too

2 | Maintain `O(config)` CD complexity

- We're **reimplementing** the **central compute model** that salt was created as response to?

- We only need to **shift common configs** to CI
- Minions still autonomously **compile** and **apply** their **own** idempotent **highstate**

Have your waffle and eat it too

2 | Maintain `O(config)` CD complexity

We found almost **no variation in realized configs**

- **Users** actually **don't need** the full `systems ⊗ configs` permutation space
- So, we already had `O(config)` complexity and didn't know it because our **tooling obscures this fact**
- Bonus: **config change usecases** are **more legible** in the change management model than config management model anyway

Have your waffle and eat it too

2 | Maintain `O(config)` CD complexity

We still need to **connect to production SoAs** (Vault, NetBox, etc.)

- Change management **should also be in charge** of these config pipelines too
- So **integrate the services** with temporal/airflow/etc. **instead of salt** to deliver configs per environment

Have your waffle and eat it too

1 | Give users authentic dev/CI config

- If we don't need `jinja|yaml` (salt) to render config, **where do we put it**?
- How about a **modern config language** (CUE/Pkl) that has, you know
  - **Full typing**
  - **Rigorous grammar**
  - **Learned from** the **good** and **bad** of primitive yolo hacks like `jinja|yaml`

Have your waffle and eat it too

1 | Give users authentic dev/CI config

Derivative benefits from using a modern config language (CUE/Pkl)

- **Safe** for human consumption
- Amenable to **automated**
  - **Auditability**
  - **Compliance**

Open problems

- **Integrating production SoAs** (Vault, NetBox, etc.) into **change management** (temporal/airflow) is not well defined
- **Integrating** (the idea of) **production SoAs** into the **dev** environment
- How to define **OpenSLO** health mediated deployment for **config changes**
- Rendered config **format** and delivery **pipeline**
- Large **legacy codebase** written in `jinja|yaml`
- CUE/Pkl **salt renderer**

Justin Findlay
Systems Engineer

Thank you

Cloudflare Platform Configuration team

- Cian Leow
- Walter Clark
- Menno Bezema
- Marek Schwann

- Joe Groocock
- Vasilii Alferov