GitOps auf echtem Blech
oder: moderne Verwaltung von Desktop Systemen

Verwaltung von Ubuntu Desktops für Softwareentwickler mit GitOps, IaC und CI/CD
Patrick Banholzer, ITT/ME, Open Source Automation Days, 2021/10/05

Mercedes-Benz
The best or nothing.
Personal information

Patrick Banholzer

Current Position:
Mercedes-Benz AG
System Architect / DevOps Engineer
Since 2019
Responsible for ~2500 Ubuntu desktops at Mercedes-Benz R&D

Experience:
Deep knowledge in everything that is IT infrastructure
15+ years of professional Linux experience
Motto: automate everything!
Agenda

Challenges in a big (Windows) corporate network

Requirements of different dev teams

Journey from Old-IT to New-IT
  – What we did it in the past
  – How we evolved by using DevOps and FOSS

Where are we today?

Why Saltstack? What about Puppet / Ansible ...?

What’s Next?
Challenges: in a (big) Windows Network

Workplace
Email, Calendar, Collaboration, MS Office

Authentication
Kerberos, LDAP, SSSD, MS ActiveDirectory

Corporate Network
Proxy, Firewalls, DNS, VPN, ...

Processes
Asset Accounting, Asset Management

Security
x509, TPM2, LUKS, Permissions

Mercedes-Benz
Challenges:
we definitely need config management

As even systems that are not customized for special project setups apply more than 400 salt-states.
Requirements and wishes of sw developers

- Everyone wants `sudo` (they really need it in the most cases)
- keep users from fighting the config-management
- less experienced users tend to break their systems
  → quite hard job for operations team to keep up with

- Window-Managers (KDE / Gnome / XFCE / i3 / awesome?)
Requirements and wishes of sw developers

• **Nvidia Drivers and CUDA** – each project wants its own setup

• **Local vs. remote / VDI**
  • Access huge data needs system directly connected to clusters
  • VDI is not that comfortable (*blurry, laggy, ...*)

• Custom “in Car” systems for autonomous driving projects
  • **stability / traceability / conformity / security**
  • More than five different high performance systems in a trunk
Three factors for successful configuration management

- Transparency
- Flexibility
- Reliability
The Journey

OLD IT
- Intransparent
- Closed
- Static
- Slow

NEW IT
- Transparent
- Open
- Flexible
- Fast

GitOps auf echtem Blech | Patrick Banholzer ITT/ME | 2021/10/05
Old World of OS Configuration Management

Monolithic and rigid approach

- No Sharing
- No (granular) Access Control
- No Workflow
- Vendor Lock
- No Open Source
Problems with Old World Approach

- No tenants
- No API, no interfaces
- No automatic testing
- No quality checks / any admin can do anything
- No sharing of code & configs
- No continuous integration of multiple contributors
- No client introspection / no state definition

Only a bunch of scripts that do things.
Riscs and Weakness we had

POOR WORKING MODEL
- Ticket → manual execution
- Low level of automation
- New code brings back previously solved bugs
- Missing workflow enforcement
- Missing enforcement of knowledge sharing

RESULTED IN
- Nobody knows what the others do
- Quality varies with admin experience and knowledge exchange
- No audit trail
- No status information of clients
- No history about clients
The DevOps Approach:
Adapt Best Practices from agile Software Development

**IaC / GitOps**
everything lives in git

**Automation and Integration**
via APIs and open interfaces

**Continuous Integration and Delivery**
commit & test & deliver fast and reliable

**Tested and approved**
by process design
Gains of Infrastructure as Code

Open, cooperative and stateful

Co-operation: Share your code!

Workflow: Know what happens!

Time to Delivery: Be fast!

Quality: Apply Gates!
Lifecycle of a device

- Order and Deliver with internal shop system
- Provision and set parameters with Foreman
- Configure and control with Saltstack
- Operate and deploy with code and git
- Updated via Ubuntu repository snapshots (might be Katello & Pulp in future)
Setup of the Infrastructure

One Foreman that is connected to many Salt-masters running in containers on multiple servers

→ **One API** to talk to (Foreman)

→ **Servers + Containers** also managed through Salt + Foreman

→ **One Git** repo per environment
  → Backend Systems
  → Clients
  → Cars
  → IoT / Edge Devices

→ **One (or Multiple)** container per environment
The Foreman

- Technical CI management
- Bare Metal Deployment
- Parameters, Database, API
- Reporting and Auditing
Saltstack

- Configuration management
- Has parameters from foreman
- Configured in GIT
  - repeatable, stable, transparent, traceable
- Defines a target state and applies it!
## Comparison of Saltstack to competitors

<table>
<thead>
<tr>
<th>Software</th>
<th>Ready to use Solutions</th>
<th>Implemented in</th>
<th>Interaction with Foreman</th>
<th>Used at Daimler / Mercedes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saltstack</td>
<td>+ (Formulas)</td>
<td>Python</td>
<td>+ (plugin)</td>
<td>YES (default with SLES 15)</td>
</tr>
<tr>
<td>Ansible</td>
<td>+ (Galaxy)</td>
<td>Python</td>
<td>+ (plugin)</td>
<td>YES</td>
</tr>
<tr>
<td>Puppet</td>
<td>++ (PuppetForge)</td>
<td>Ruby</td>
<td>++ (native)</td>
<td>YES</td>
</tr>
</tbody>
</table>

- **Connection Type**
  - Saltstack: Minion → Master – Message Queue
  - Ansible: Master → Target – SSH
  - Puppet: Agent → Master – Request

- **Software needed on clients**
  - Saltstack: Salt-Minion
  - Ansible: Python
  - Puppet: Puppet Agent

- **Used by**
  - Daimler / Mercedes: YES
Workflow

Jira Task → Sprint planning / Emergency Change → Git Feature Branch → Implementation → Local Tests → Commit → Git Push → Jira Task

Issue
Bug Report
Change Request

Test Pipeline

Salt Lint
Saltcheck (Unit Tests)
Run each state in docker environment

Successful test is mandatory

Automatic deployment

Merge into Master → Successful test + 2 approvals → Pull Request → Manual tests on int. environment (including friendly-users) → Merge into int → Test run → Pull Request

Mercedes-Benz

GitOps auf echtem Blech | Patrick Banholzer ITT/ME | 2021/10/05
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes

This does not work
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes

What if this change still needs fixes?

This does not work
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes
- Create feature-branch from master
- Cherry-Pick commits from integration
- Pull-Request into master / production
Git Workflow

- Create feature-branch
- Commit changes
- Test changes
- Pull-Request into integration
- See if problems occur
  - eventually push additional fixes

- Create feature-branch from master
- Cherry-Pick commits from integration
- Pull-Request into master / production
Benefits and problems with this workflow

**PROS**
- Deploy when ready
- Parallel work
- No blocking changes
- Easy revert possible

**CONS**
- Commits can be missed
- Gap between int and prod
- PRs to master must be carefully planned
Test pipeline enables local testing

Docker / Docker-Compose help with setup

- **Setup & run** test-environment fast
- **Tear it down** even faster!
- **Run again**...
- Each admin pc can run tests
- Local tests compare to pipeline
What's next?

**DEVOPS CONFIG MANAGEMENT**
- Improve test coverage
- Tests in virtual environment
- Black-box tests on hardware
  - Tests of fresh installations
    - At least once a day
    - On each supported hardware
- Improve Update-Management

**DESKTOP FEATURES**
- Ship to Desk installations
- Zero Trust Client setup
- Support of E2E encrypted emails
- Self-Service platform
  - Get rid of proxies
Your Questions!?