Location-independent management of CarPCs in the cloud

Patrick Banholzer (Mercedes-Benz AG)
Stefan Bogner (B1 Systems GmbH)

October 4th, 2022

Mercedes-Benz
The best or nothing.

Personal information

Patrick Banholzer

Current Position:

Mercedes-Benz AG
Cloud & DevOps Engineer

Building CICD & Testing for ADAS

Experience:

Deep knowledge in IT infra & automation
Motto: automate everything!

Stefan Bogner

Current Position:

B1 Systems GmbH
Linux Consultant und Trainer

System- and Configmanagement & Deployment
Linux Client Management

Experience:

All Linux / Open Source

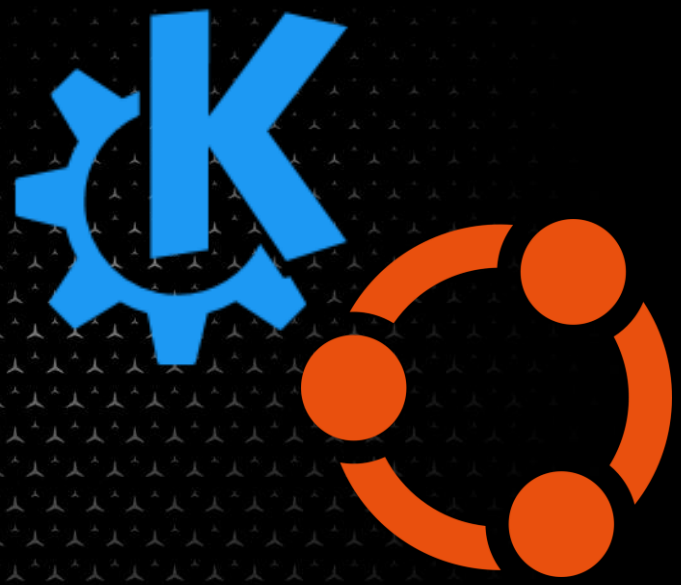
Agenda

- What is a "CarPC"?
- Purpose and enabled use-cases of CarPCs
- Legacy operations setup
- External factors for architecture design
- Pro/Con Legacy vs. Cloud
- New approach, architecture
- Tools & Services

- Deployment process
- Technical challenges

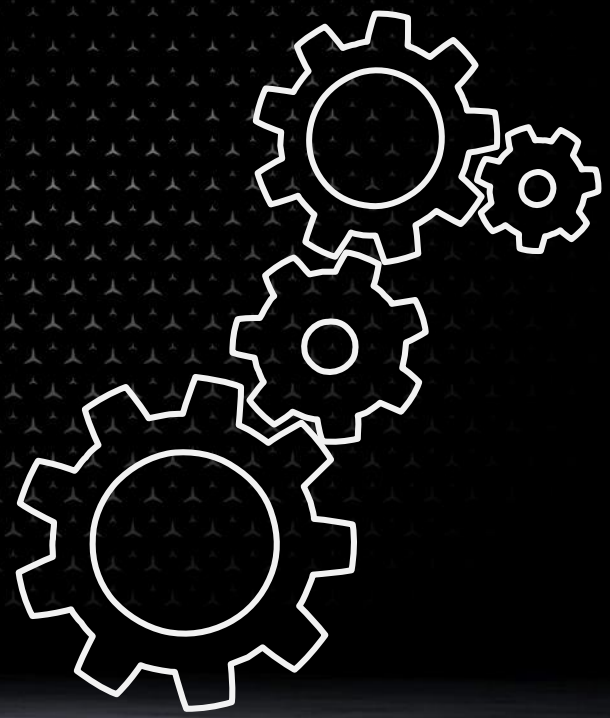
What is a CarPC?

- Standard PC in industrial chassis
- Located in the trunk of a dev-car
- OS: Ubuntu LTS
- UI: KDE Plasma with custom software
- Inputs / observation done by front-seat passenger via head unit and additional displays
- Input devices touch display & keyboard + touchpad



Purpose of a CarPC

- Used in automated driving projects
- Control unit for measurement system in the car
- Start recording, trigger and tag situations, check sensor functionality
- Visualization of measurement data
- SW-Updates of measurement systems and ECU’s



Enabled use-cases

- Remote access to system
- Update and configuration
- Health check and debugging
- Meta info transfer to backend

External factors on design decisions

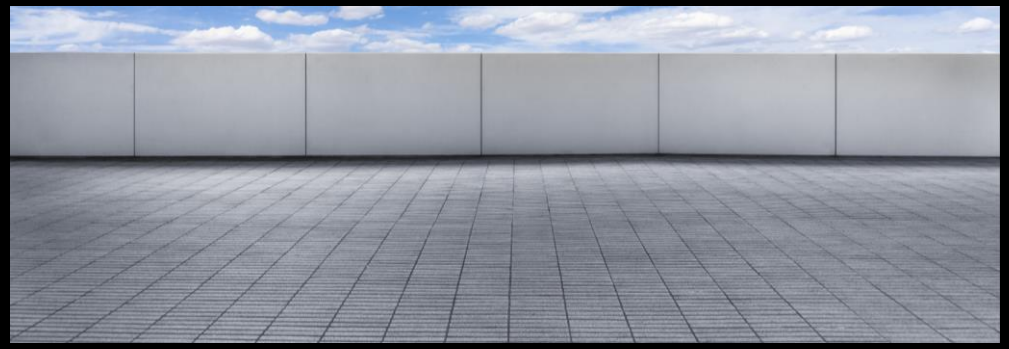
- Worldwide fleet operations
  - Collecting data (on street)
  - Support developers in test /development vehicles
  - Continuous QM within test-benches

- "Always On" remote connectivity
  - Not only garage and workshops
  - >> Aswell on the road and far-off test campaigns

- Enable third parties who are involved in development

Legacy approach of operations & setup

Maintaining and managing systems within Mercedes-Benz Corporate Network by relying on proven concepts and technologies



Advantages

- Reuse of existing services and processes
- quicker "time to market"

Disadvantages

- Setup and reinstallation only possible in Mercedes-Benz locations
- Limited by VPN connections / bandwidth
- All contributing employees of partners need access to Mercedes-Benz network
- Time consuming processes for onboarding users of 3rd party employees

State of the art cloud approach

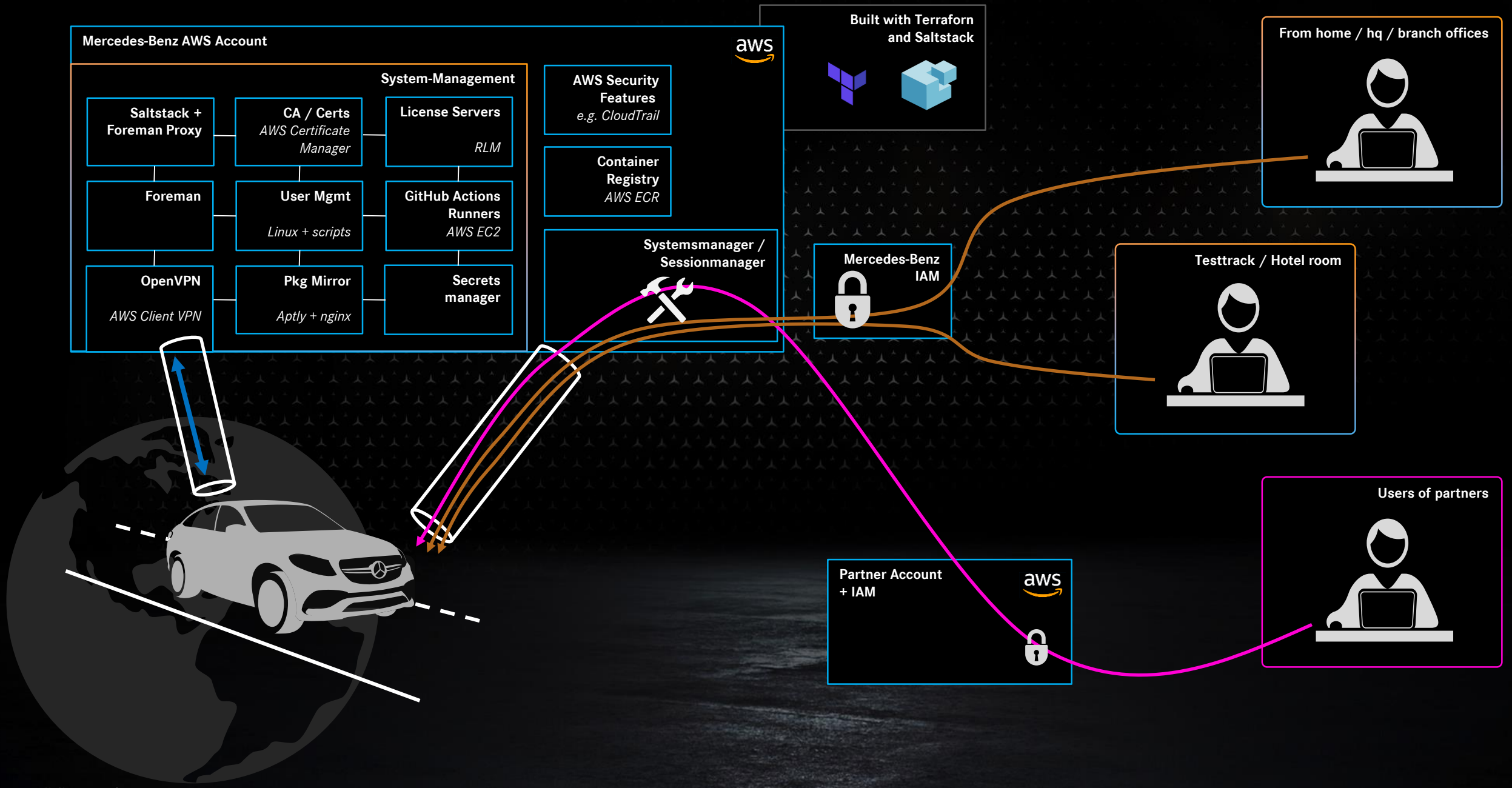
Maintaining and managing systems decoupled of corporate network by building everything inside cloud infrastructure but keep parts of IaC client management technology

Advantages

- Setup and reinstallation from anywhere in the world
- Higher connection throughput
- More agility compared to corporate network
- User management can be shifted to the partners admins

Disadvantages

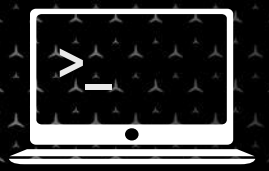
- Initial effort for solution design & infrastructure setup in cloud
- Time & effort for Mercedes-Benz internal data protection processes
- Building acceptance on developers that direct ssh to vehicles within the garage is not available


Mercedes-Benz AWS Account
aws
System-Management
Saltstack + Foreman Proxy
CA / Certs
AWS Certificate Manager
License Servers
RLM
Foreman
User Mgmt
Linux + scripts
GitHub Actions Runners
AWS EC2
OpenVPN
AWS Client VPN
Pkg Mirror
Aptly + nginx
Secrets manager
AWS Security Features
e.g. CloudTrail
Container Registry
AWS ECR
Systemsmanager / Sessionmanager
Built with Terraform and Saltstack
From home / hq / branch offices
Mercedes-Benz IAM
Testtrack / Hotel room
Users of partners
Partner Account + IAM
aws

Deployment process

- Prepare system in backend with CLI tool
  - Custom Shell interface providing templates and guided creation of host representation
  - Uses Foreman API to configure and set build-mode

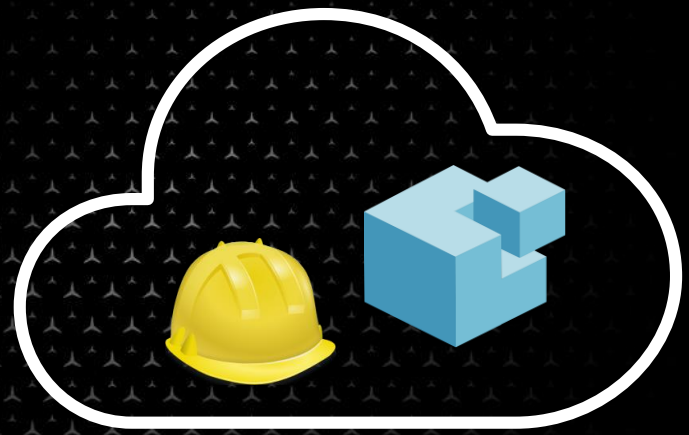
  >> no need for user privileges within Foreman



- Boot system with customized Kubuntu install-image
  - Make sure Internet connection is available (Ethernet / Wifi / 5G)
  - Provide access credentials for initial VPN connection

Deployment process

- Full automated install with foreman provided templates
  - Retrieve ubiquity templates from Foreman for automated installation
  - Create device specific VPN certificate using TPM2
  - Bootstrap system for Saltstack



- 1st Reboot
  - Connect to VPN and Saltstack
  - Run initial highstate to apply system configuration

Deployment process

- 2nd Reboot
  - Boot into fully configured system with KDE and smartcard logon
  - Connect to VPN, Amazon SSM and Saltstack

Takes less than 35min in total (45min from US West area)

Deployment process

- Continuous Config deployment using Saltstack
  - Minimal set of configuration states are deployed regularly
  - GitOps Workflow for configuration changes
    - Saltstack gitfs
    - Git-Tags as releases
    - Foreman and Pillarstack/Gitstack as data sources

The GitOps Approach

IaC / GitOps

everything lives in git

Tested and approved

by process design

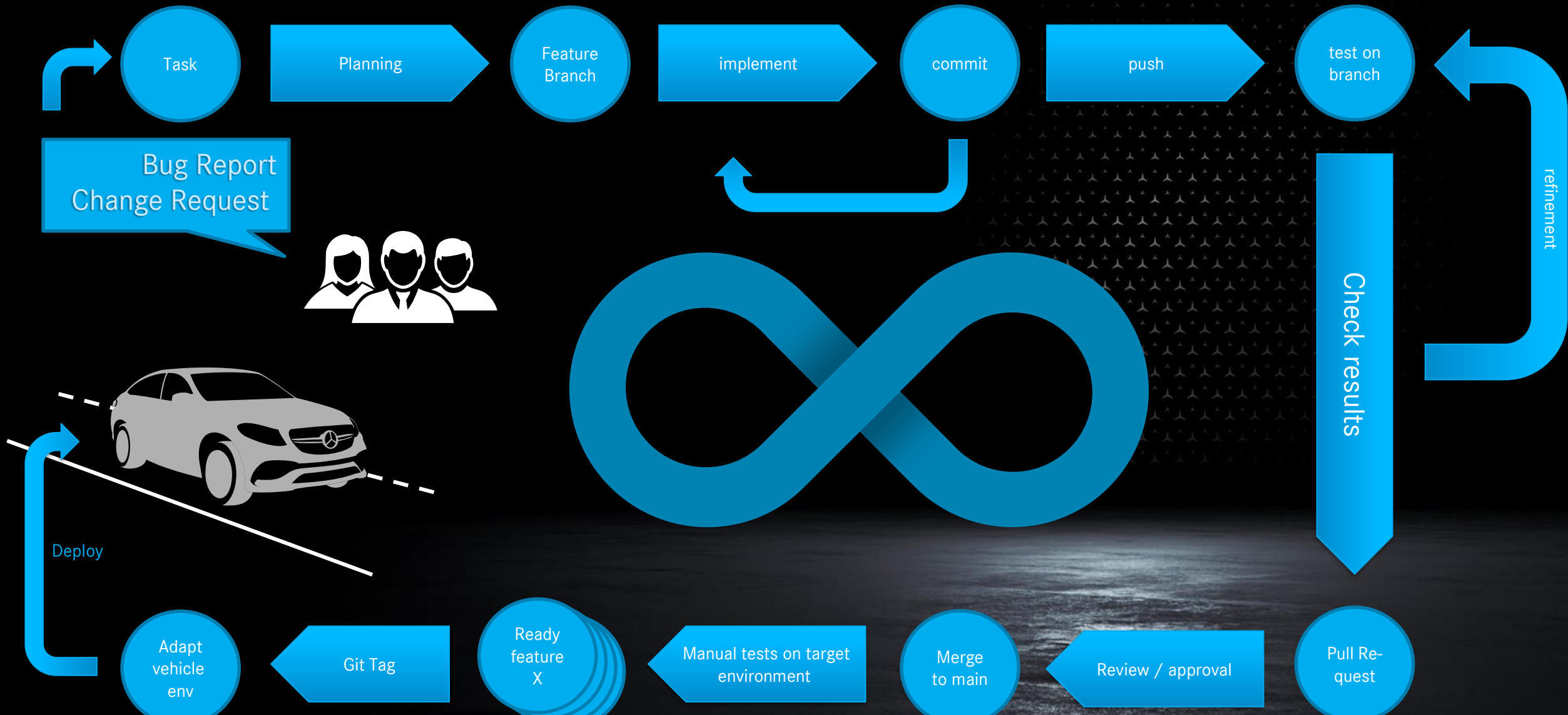
Automation and Integration

via APIs and open interfaces

Continuous Test and Delivery

commit & test & deliver fast and reliable

Workflow


Task
Planning
Feature Branch
implement
commit
push
test on branch
refinement
Check results
Bug Report
Change Request
Deploy
Adapt vehicle env
Git Tag
Ready feature X
Manual tests on target environment
Merge to main
Review / approval
Pull Re-quest

Whats next?

- Ubuntu 22.04 and next generation preparations

- Work with overlayfs to ensure clean environment on each boot.

Alternatively

- Automate redeployment even more
  - e.g. redeploy/rescue partition
    >> complete redeployment for each config change

Questions?

Patrick Banholzer

Stefan Bogner

Want to be part of it?