Anthony Ruhier

Infrastructure Software Engineer

8 048 Zürich, Switzerland
+33 688 162 558
aruhier@mailbox.org
aruhier.fr
https://github.com/aruhier
Borned in 1993, French

Work experience

2022 – Now **Software Engineer, Site Reliability Engineer on Borg Node**, *Google*, Zürich (Switzerland)
As an SRE-SWE, I work inside the Borg Node SRE team to manage and defends the stability, availability, isolation, and performance of Google's core machinery at scale (~5 millions nodes). Borg Node SRE team specializes in architecting and supporting the tools, services, processes and policies requisite to manage the full complexity inherent in supporting Google's growing fleet.

2019 – 2022 **Network Engineer, Software and Automation on CorpNet**, *Google*, Zürich (Switzerland)
As a Network Engineer focused on software and automation, I work on the Corporate and Enterprise infrastructure team to deliver Google's next generation network and corporate data centers. I build software for distributed services, abstractions and the components of the system that operates and powers Google.

2018 – 2019 **Network DevOps Engineer**, *Online/Scaleway*, Paris (France)
Working with the Backbone SREs on maintaining and improving the entire network, by designing and coding tools — mostly in Python — to allow them to parallelize and automate their actions on a larger scale, to target a hundred of thousands of users.

2014 – 2017 **Apprentice Software Engineer for a local Telecom Operator**, *Trinaps*, Belfort (France)
Lead developer, designing micro-services focused on the automation of redundant processes, linked to the conception of an Information System allowing the company to efficiently store and use its data.

Qualifications

2014 – 2017 **Master's Degree in C.S**, *UTBM*, Apprenticeship Program in 3 years, Belfort (France)

Projects

Google (Borg SRE)
Lead of the System Overhead Review program
Owner of the System Container and its resources management, present on every machine in the fleet of Google, isolating the minimal set of OS daemons required for a machine to receive some workload. This area has a usage of a *single digit* PB of RAM and *single digit* millions of CPU cores. I managed and reviewed the resources requests initiated by the daemons developer teams, and improved the process, visibility and monitoring to keep this area under control and budget.

Google (Network)
Go
Automation of operations on the Corp Backbone
With the upcoming deprecation of several internal tools, certain common operations on the Corp Backbone were risking to lose their automation. I worked on designing several central services, working in synergy with other services designed by my team members, to re-implement the entire automation of the Corp Backbone from scratch in order to modernize it and align it with the Prod Network. The services I designed and implemented are used in every turn-up of a backbone device, specifically for the interconnection with the rest of the network. This scope of this area went from the intent automation to the configuration pushed on the network devices.

Online/Scaleway
Python
Celery
Rate limiting the entire server farm uplinks
Dynamically set rate limiting on 3K Online's custom switches, depending on the customer offer, with a computed burst size large enough to be the least restrictive possible. Impacted more than 50K servers, saved several 100gbps of bandwidth.

Personal project
Python
Traffic Control
Python framework to setup a QoS on Linux (https://github.com/aruhier/pyqos)
Framework to build complex QoS rules for Linux, in a hierarchical way with built-in tweaks. Rules can make use of the Python OOP system to be as much clear as possible. Imported and used at TRINAPS for events setups, to manage the QoS of more than 5K simultaneous users, with a really tight WAN bandwidth (200mbps).

Personal project

Personal infrastructure, to satisfy a will of self-hosting

Web: Nginx as ingress controller in Kubernetes.

Databases: PostgreSQL as main DBMS, MariaDB for applications requiring it.

Hypervisor: 3 Kubernetes nodes over KVM (Libvirt as hypervisor) on Debian, with automated backups to 2 ZFS iSCSI+NFS NAS, using virt-backup and zrepl.

Management: More than 30 pods and hosts running on ArchLinux, uniformly managed by Ansible from the network configuration to the databases, X.509 certificates and Nginx setup.

DNS: Authoritative servers use Knot with automatic DNSSEC signing, resolvers use Unbound.

Network: Hosts are spread into 6 VLAN, via OpenVSwitch and a Mikrotik SFP+ switch. Traffic is subject to QoS policies, using PyQoS. IPv4/IPv6 external traffic is dynamically routed using BGP and Wireguard. DNS resolvers in anycast.

Programming

Imperative Programing	Go *(Readability from Google)*, Python *(TDD)*, Rust
Web development	Django, Flask, Javascript
Others	Ansible, Bash, SQL

Language skills

English	**C1 Level**, BULATS – 85/100 (2016)
French	**Native speaker**
German	**B1 Level**