Anthony Ruhier

INFRASTRUCTURE SOFTWARE ENGINEER

8 048 Zürich, Switzerland

✉ aruhier@mailbox.org

🌐 [linkedin.com/in/aruhier/](https://www.linkedin.com/in/aruhier/)

<https://github.com/aruhier>

Né en 1993, Français

Permis de séjour C

Expériences professionnelles

- 2022 – Now **Software Engineer, Site Reliability Engineer on Borg Node**, *Google*, Zürich (Suisse)
SRE-SWE travaillant dans l'équipe Borg Node SRE pour gérer et maintenir la stabilité, disponibilité, l'isolation et les performances du coeur de la machinerie Google à grande échelle (~5 millions de nodes). L'équipe Borg Node SRE est spécialisée dans l'architecture et le support d'outils, services, process et règles nécessaires pour la gestion de la complexité inhérente au support du parc croissant de Google.
- 2019 – 2022 **Network Engineer, Software and Automation on CorpNet**, *Google*, Zürich (Suisse)
Ingénieur réseau focalisé sur le développement et l'automatisation, travaillant dans l'équipe Corporate and Entreprise Infrastructure pour délivrer la nouvelle génération du réseau et Data Centers Corp de Google. Développe des logiciels au sein de services distribués, abstractions et composants de systèmes étant au cœur de Google.
- 2018 – 2019 **Network DevOps Engineer**, *Online/Scaleway*, Paris (75116)
Travaille avec les SRE backbone sur le maintien et l'amélioration du parc, en leur concevant et développant des outils — principalement en Python — qui leur permettent ensuite de paralléliser et automatiser leur actions visant plusieurs milliers d'utilisateurs.
- 2014 – 2017 **Apprenti Ingénieur Développeur pour Opérateur Télécom**, *Trinaps*, Belfort (90000)
Lead développeur, concevant des micro-services centrés sur l'automatisation de processus redondants, liés à la conception d'un Système d'Informations permettant à l'entreprise de gérer efficacement ses données.

Diplômes

- 2014 – 2017 **Ingénieur en Informatique**, *UTBM*, Ingénieur par Apprentissage, Belfort (90000)

Projets

- 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) **Automation of operations on the Corp Backbone**
Go
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 **Rate limiting de l'entièreté du parc de serveurs**
Python
Celery
Mise en place d'un rate limiting dynamique sur plus de 3K switches maison d'Online, variant suivant l'offre du client, avec une taille de burst calculée suffisamment large pour être le moins contraignant possible. Impacte plus de 50K serveurs, a permis d'économiser plusieurs 100gbps de bande passante.
- Projet personnel **Framework Python de QoS sous Linux** (<https://github.com/aruhier/pyqos>)
Python
Traffic Control
Framework pour construire des règles de QoS complexes pour Linux, de façon hiérarchiques tout en embarquant des optimisations prédéfinies. Les règles peuvent prendre avantage du système de POO de Python pour être le plus clair possible. Importé et utilisé à TRINAPS sur les installations événementielles, pour gérer la QoS de plus de 5K utilisateurs simultanés, sur une bande passante WAN limitée (200mbps).

Projet personnel

Infrastructure personnelle visant l'auto-hébergement

Web : Nginx comme ingress controller pour Kubernetes.

SGBD : PostgreSQL comme principal SGBD, MariaDB pour les applications qui le requièrent.

Hyperviseur : 3 nœud Kubernetes sur KVM (Libvirt en tant qu'hyperviseur) sous Debian, avec des sauvegardes automatiques vers des NAS iSCSI+NFS ZFS via virt-backup et zrepl.

Orchestration : Plus de 30 pods et serveurs sous ArchLinux, gérés uniformément par Ansible, de la configuration du réseau aux bases de données, Nginx et certificats X.509.

DNS : Serveurs d'autorités Knot avec signature DNSSEC automatique, résolveurs Unbound.

Réseau : Les serveurs sont répartis sur 6 VLAN, via OpenVSwitch et un switch Mikrotik SFP+. Le trafic est soumis à des règles de QoS, via PyQoS. Le trafic extérieur IPv4/IPv6 est dynamiquement routé par un serveur dédié, via BGP et Wireguard. Résolveurs DNS en anycast.

Programmation

Impérative Go (*Readability from Google*), Python (*TDD*), Rust

Web Django, Flask, Javascript

Autres Ansible, Helm, Shell, SQL

Compétences linguistiques

Anglais **Niveau C1**, BULATS – 85/100 (2016)