PassFort

Architecture Guide

Metadata

Review

Accountable Director	CTO
Policy Author	CTO
Date Approved	Sep 2017
Date Last Reviewed	May 2020
Date Of Next Review	Yearly, Apr 2021

Document History

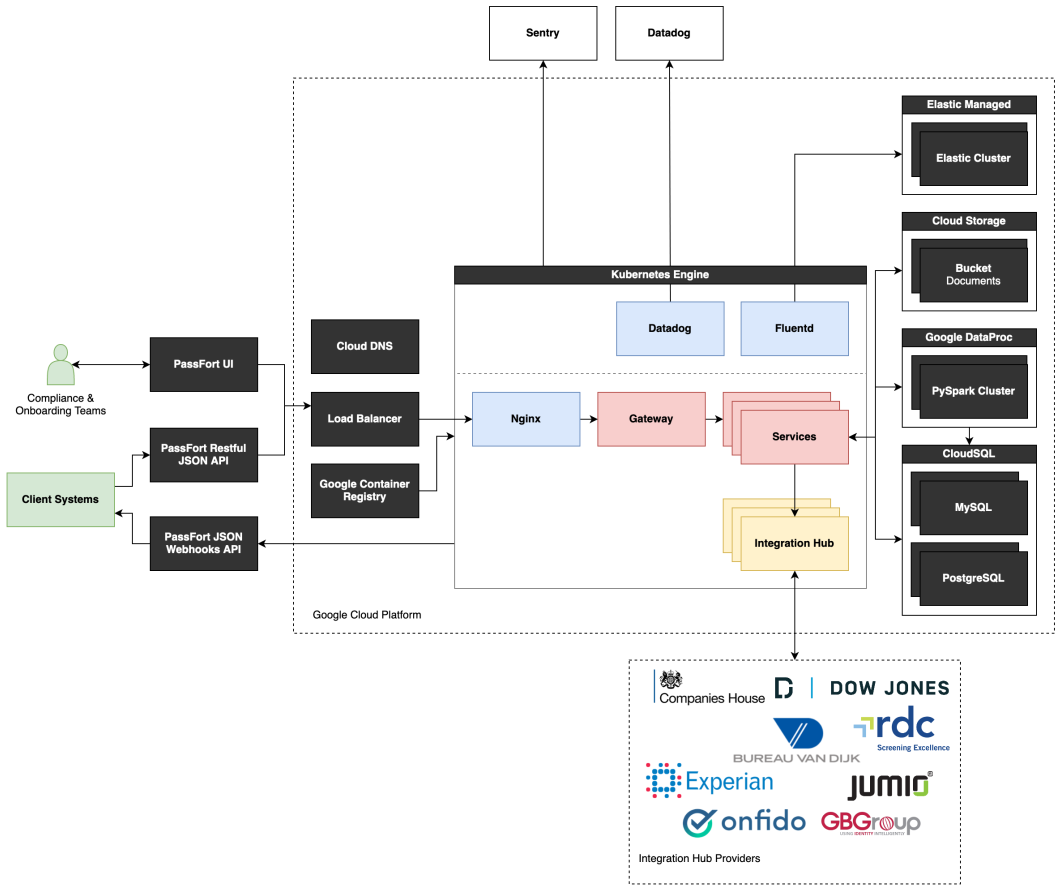
Version	**Date**	**Author**	**Description of Changes**
1	Sep 2017	CTO	Initial Version
2	Jun 2019	CTO	Document deprecated
3	May 2020	CTO	Reintroduce document Add version metadata Major updates

Table of Contents

1. Infrastructure

1.1 Overview

PassFort runs as a SAAS application. All clients therefore run on shared infrastructure. A high level diagram of the system architecture follows, along with written descriptions.



PassFort is entirely hosted in Google Cloud Platform (**GCP**), and runs as a micro-services application on top of Google Kubernetes Engine (**GKE**). It is backed by three types of data store:

- **Databases:** We handle both MySQL and PostgreSQL databases, hosted on Google CloudSQL (a managed database service). These are run in high availability mode, with a hot standby server.
- **File storage:** We store files in Google Cloud Storage, which stores flat files. These are versioned. We leverage both Google's own encryption layer, as well as our own application encryption layer.

- **Search:** We maintain a managed elastic search cluster to power the search features of the application.

We have a data processing pipeline built in PySpark, on top of Google DataProc.

We use Hashicorp Vault for key management, which we manage ourselves.

We utilise three products for monitoring PassFort:

- **Datadog** is used for metric monitoring & APM.
- **Elastic** is used for log management and search
- **Sentry** is used for error reporting and investigation.

We have CircleCI for continuous integration and storage. Our codebase is managed in GitHub.

All access to PassFort is over HTTPs. We enforce TLS 1.2 and above, in line with industry standards. We integrate with LetsEncrypt to generate certificates.

1.1.1 Geography

Currently, all PassFort infrastructure is hosted in GCP's Belgian region. Backups are distributed to GCP's other EU regions.

1.2 Google Cloud Security

We leverage GCP as our infrastructure as a service (IaaS) provider. They are responsible for much of the low level security, including **physical data-centre security** and **network security**. We also use some of their managed offerings, as described above, where GCP take responsibility for the ongoing running and maintenance of those tools.

You can find more information about GCP's approach to security at https://cloud.google.com/security.

1.3 Prisma Cloud

We have purchased a commercial cloud security solution called Prisma Cloud (https://www.paloaltonetworks.com/prisma/cloud) This includes:

- Container vulnerability scanning, both pre-runtime and at run-time.
- Web application firewall (WAF)

- Infrastructure configuration compliance & audit – ensuring that our GCP setup does not deviate from standard industry practices

1.4 Audit Logging

All events taken within our infrastructure are audit logged by our infrastructure providers. These audit logs are monitored by Prisma Cloud, and alerts are generated when configuration is changed, or falls outside of specified parameters.

2. Application

2.1 Overview

PassFort services are exclusively accessed over a single Restful JSON API, with JSON webhooks used to communicate to client systems. Our services are written in a combination of Python and Rust. Our services are configured via terraform and kubernetes.

We are migrating core services to Rust to leverage its strong type safety, and its ability to obviate certain classes of error.

Our front end web application (the PassFort **Portal**) is a single page application written in Javascript (on the React framework). This consumes the same API that is made available to clients. We use flow to improve type safety.

2.2 Access Management

2.2.1 Authentication

Clients can authenticate with PassFort in a number of ways:

- **Username and password:** Passwords are hashed using **SHA512**. After initial sign in we utilise sessions to maintain authenticated. This authentication method can be augmented with a second factor (2FA) – we currently support SMS.
- **Single Sign On (SSO):** PassFort is integrated with Okta which allows us to leverage most industry leading SSO providers. As above, we use sessions after the initial authentication. In many cases we can synchronise teams into PassFort from the upstream SSO provider.
- **API Key:** System-system communication requires a 192bit API Key. All accounts have a master API key (which can be regenerated), but secondary keys can be instantiated with scoped permissions. This mechanism requires the token to be passed as a HTTP header with every request.

2.2.2 Authorisation

PassFort supports a powerful permission system across its infrastructure.

These permissions can be configured in the UI, and typically allow for “read”, “write” or “no” access across the different areas of the PassFort product. Permissions can also be scoped around specific products.

Permissions are grouped into named roles, which can again be configured through our UI. These can finally be attached to either a “team” (which grants the role to all team members), or to individual users.

The permissions system is implemented internally via JWTs (JSON web tokens). Our gateway attaches a JWT describing the permission to the request. These JWTs are verified by a common library which is used across our micro-services.

This authorisation system is the primary way through which client data is segregated. It has been validated through a 3rd party security audit.

2.3 Data

2.3.1 Data Storage

PassFort stores personal data across two databases - MySQL and PostgreSQL. We utilise these two technologies while we migrate from MySQL to PostgreSQL. These databases are managed by GCP through their CloudSQL product. They run in high availability mode, with hot standbys.

We also collect files from some customers. These are encrypted at the application level (with per-institution keys) using AES256. These are then stored in GCP’s Storage product, where they are re-encrpted by GCP. These are then backed up in a mirrored storage bucket. We control access to these buckets using GCP IAM. This is in turn managed by our terraform configuration.

2.3.2 Data Backups

A detailed discussion of our backup strategy is included in our **Disaster Recovery Policy**.

2.3.3 Deletion

PassFort offers three forms of data deletion:

- **Archival:** this hides a profile from the UI, but maintains all data within the profile. This can be actioned through the API or UI (with a specific set of permissions).

- **Full profile deletion:** this deletes a profile & its data fully. We will track the deletion within the institution level audit log. This allows our clients to meet their obligations as data controllers under GDPR. This can be actioned through the API or UI (with a specific set of permissions).
- **Full institution deletion:** this deletes an institution and all associated data (including the institution audit log). This would typically be actioned after off-boarding a client, within 30 days. This cannot be instructed through the API, and should be actioned through your customer success manager.

2.4 Monitoring & Logging

2.4.1 Audit Logging

All actions within the platform are logged within our application-level audit service. This audit service logs actions taken by:

- Users
- Client applications (through API Keys)
- PassFort's smart policy automation layer

We track audit items at two levels:

- **Profile level:** all events which affect a specific client profile are tracked within that profile's audit trail.
- **Institution level:** events not tied to a specific profile (e.g. smart policy configuration changes; profile deletions) are tracked across an institution-specific audit log.

2.4.2 Application Logging

All services log data into an elastic cluster. We use a managed elastic cluster which lives within GCP. As policy, logs do not contain PII - we default to logging UUIDs.

2.4.3 Metric Monitoring & Alerting

We use DataDog (https://datadoghq.com) for listening to metrics and alerting on those metrics. Alerts are filtered into several channels:

- **Out of hours, high priority, alerts:** alerts which indicate a significant event and must be responded to out of hours by our on duty out-of-hours support team.

- **In hours, high priority, alerts:** alerts which indicate a significant event but do not have substantial time pressure. These alerts are reviewed by our in hours support team on the next working day.
- **Pre-promotion alerts:** alerts which are being evaluated for noise before promotion into one of the above channels.
- **Demoted alerts:** alerts which are have been too noisy to be effective and are demoted pending improvement.

2.4.4 Vulnerabilities

We have a 0-critical and 0-high vulnerability policy, and review all others. We have a 14 day SLA-to-resolution from the moment a resolution is published.

We monitor for vulnerabilities using:

- **GitHub monitoring -** this monitors code dependencies for vulnerabilities.
- **Prisma -** this monitors our containers (both in the registry and those running in our cluster).

2.5 Scalability

2.5.1 Application Bottlenecks

Our application relies on transaction locking at the profile level to maintain complete data consistency. This means that our application-level performance limitations are around the number of concurrent operations per profile, rather than the number of concurrent operations per institution or across the entire application.

2.5.2 Infrastructure Scaling

All application core services are stateless, and rely on our database for state management. As they are run on a kubernetes cluster as containers, they can therefore all be scaled horizontally, for as long as the database can handle it.

Our current architecture has been load-tested to 700k profiles per day, which was achieved without scaling any hardware. We currently run millions of profiles per year with a single database master with 2 vCPUs and 7.5GB memory. There is therefore plenty of room to scale the database vertically.

Thereafter, there is the option of horizontally scaling the database via sharding. Given our applications locking behaviour (referenced above), we expect to be able to shard easily at the profile level.

PassFort currently runs as a single cluster, and we have no short term plans to run multiple clusters in an active-active configuration.

3. Operations

3.1 Access

Access to production systems is restricted.

Three users are designated administrators have complete access to production systems. This access is only used in exceptional circumstances.

The vast majority of changes are made via configuration-as-code. These changes go through our standard SDLC, where changes are reviewed and approved by peers and senior engineering management. This is then deployed into our staging and production environments by our CI/CD systems.

Our infrastructure is accessed via a VPN, which requires 2 factor access (hardware token). We use GCP's IAM system (managed through terraform) to lock down access to the infrastructure.

3.2 Environments

PassFort maintains a production and a staging environment. These are functionally identical, with similar access restrictions. The staging environment runs on a copy of production data (made at least monthly).

3.3 SDLC

3.3.1 Design

The first phase of our SDLC involves doing design work on a feature. As part of this we do a risk assessment of the feature. This informs our testing plan and deployment plan. When a feature is high risk, it requires review and sign off from senior engineering stakeholders at the design phase.

3.3.2 Testing

PassFort has an extensive automated test suite. This includes:

- **Unit testing:** testing specific functions within a service.
- **Feature testing:** testing flows within a specific microservice, with other services mocked out.

- **Integration testing:** testing our microservices in aggregate. We exercise this via API calls, headless browser UI tests, along with visual regression tests.

These tests are run via CircleCI as part of development and as part of our deployment process.

3.3.3 Review & Approval

All features are peer reviewed, with at least one review required before a feature is merged into our codebase. We run a secondary risk assessment as part of this review process, identifying any material changes between the design and the eventual implementation.

For features which are identified as high risk, reviews are required from senior engineering stakeholders.

Where possible, we also run static analysis tooling to capture common errors.

3.3.4 Deployment

PassFort services are continuously deployed to – we deploy to them multiple times a day.

All PassFort services are have automated CI/CD.

We release features on Monday to Thursday. We only release hot-fixes out of these hours. If PassFort has to perform maintenance which will result in downtime, we target a maintenance window of 8-9am on Sundays. This will be published ahead of time on https://status.passfort.com.

3.4 Maintenance

PassFort, or our the providers of our managed services, occasionally have to perform maintenance that causes downtime. This typically includes upgrading the versions of our core infrastructure (e.g. our databases). Although we try to avoid these events, they usually happen 1-2 times per year. Recent downtimes have only been for a few minutes.

We have a consistent maintenance window which has been judged across our entire (global) client base's peak operating times. This is **Sunday 8-9am UTC**. We always provide advanced notice, through https://status.passfort.com.

3.5 Support

PassFort support can be accessed via support@passfort.com. For those on our premium support package, there is also a 24/7 phone number for P1 issues.

For more information about SLAs, please refer to your contract.

PassFort has on duty engineering 24/7. We split these into an “in hours” (**IH**) rotation and an “out of hours” (**OOH**) rotation:

- **IH engineers** act as a second line of support, for both client issues and internally generated issues (e.g. alerts from our infrastructure monitoring).
- **OOH engineers** are available to fix P1 issues identified by our clients and by our infrastructure monitoring.