APIFORTRESS

# API Fortress On-Premises Options

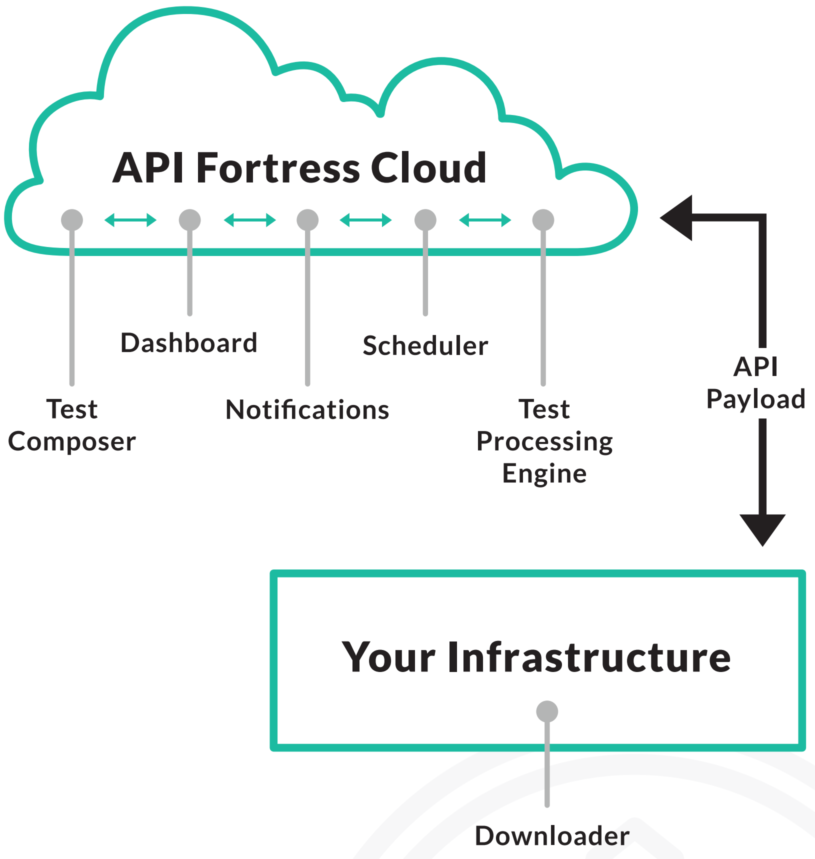
From the beginning, the API Fortress platform was built to work with the most stringent security restrictions. Thanks to a very modular platform, it can adapt to satisfy any corporate requirements. Below we have listed a few of the most common ways companies have deployed the platform.

## SIMPLE: Downloader

*"We have private APIs that are not available publicly."*

An API Fortress downloader resides in your infrastructure. The downloader is an agent that fetches payloads that are to be tested, and sends them to the API Fortress platform.

Solves for customers with APIs that are not publicly accessible. All operations are sent using a secured connection.

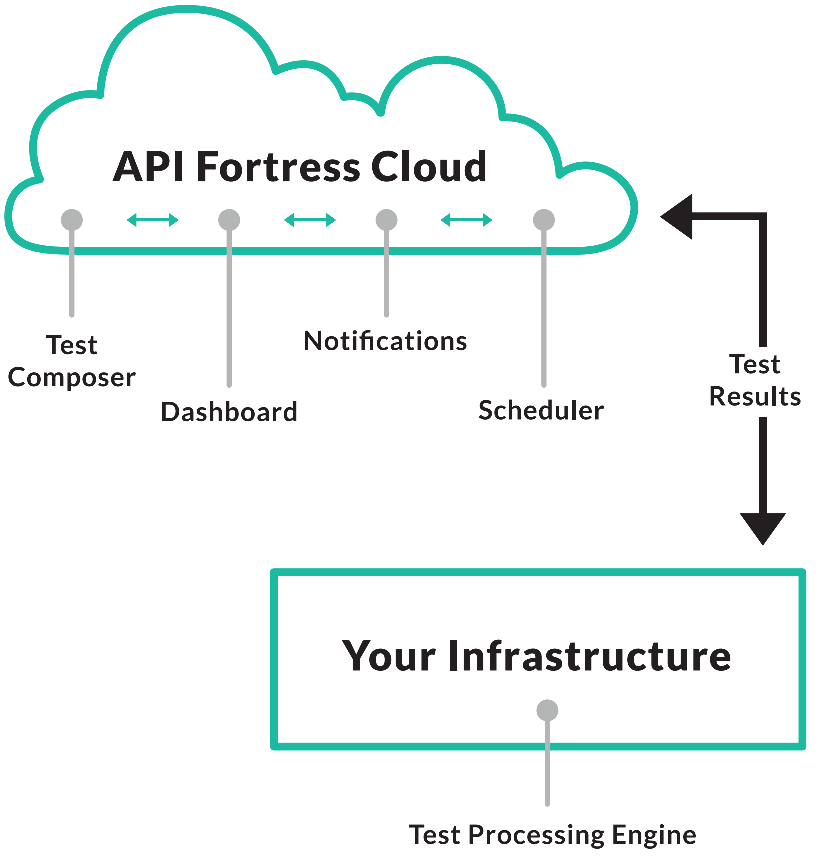


APIFORTRESS

## SIMPLE: Test Processing Engine

*"We have private APIs that are not available publicly, and can't be sent outside."*

The test processing engine, which executes the tests, resides within your infrastructure. The content of the API payloads remain within your infrastructure, and only the results are sent to the cloud. The rest of the API Fortress platform remains in the cloud.



## ADVANCED: Entire Platform

The entire platform is deployed in your environment.

**THIS INCLUDES:**

- Dashboard (includes test processing engine)
- Java/Tomcat based application that will provide the engine and the front end.
- Downloaders
- Java workers that will retrieve the resources for the tests.
- Scheduler
- Notifications
- Java application to organize and trigger scheduled events.
- Java application that takes handles the notification of events using various methods.
- PostgreSQL Database
- MongoDB Database
- RabbitMQ Message Queue

## MODERATE: Customizable Test Processing Engine

The test processing engine is a fully modular application where each component can be adjusted to meet security needs. Some of the customizable options include:

- The results can be sent to an internal database.
- The tests can come from GitHub.
- Scheduling can come from your own workflows.

### HERE ARE SOME EXAMPLES:

*"We want to keep the reports and metrics in our own databases."*

When a company is interested in handling the results of the tests internally with inspection instruments such as Splunk or Kibana

*"We want to keep the tests on-premises, to allow for versioning along with the rest of the code."*

Generally appreciated by teams that want to keep a detailed versioning of tests, together with the rest of the code.

*"We need to keep absolutely everything internal."*

This is the preferred choice when security policies prohibit any data from being sent to a 3rd party. It is also good highly automated environments where multiple instances are used to crunch a huge amount of data. With this option the dashboard, test composer, and notifications need to be handled by you.