
# FrontLine Installation Guide

Gatling Corp

Version 1.6.2, 2019-01-24

# Table of Contents

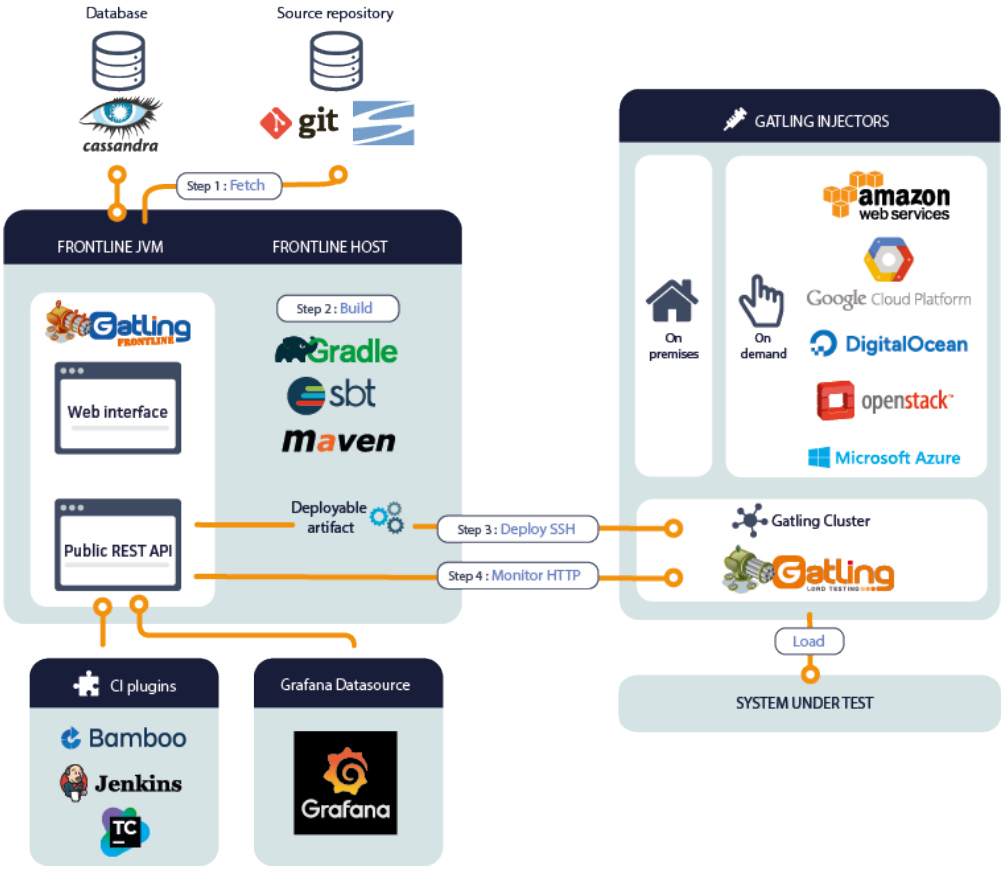
# Chapter 1. Introduction

## 1.1. Audience and Goals

This document is intended for operations people in charge of deploying the FrontLine components.

It describes FrontLine's architecture, components, how to install them and what the prerequisites are.

## 1.2. Architecture Overview




FrontLine consists of:

- A Cassandra database
- The FrontLine components:
  - The API:
    - A component which reads metrics data from Cassandra
    - A rich web client layer that talks to the API
- The FrontLine Extensions:
  - A Grafana Datasource to query FrontLine metrics
  - Continuous Integration plugins for Jenkins, Bamboo and TeamCity
- Gatling injectors to be used like standard Gatling OSS, with extra features so FrontLine can

collect data from them

## 1.3. Document Conventions

In this document, you'll find several mentions to some placeholders in capital letters.

- REPLACE_WITH_YOUR_REPOSITORY_URL: the url of the private repository that you were given alongside with your license key

This placeholder only makes sense for on premise customers. AWS Marketplace customers spawn a pre-installed AMI and already have all the dependencies they need.

- REPLACE_WITH_LATEST_FRONTLINE_VERSION: 1.6.2 at the time this document was edited

# Chapter 2. Installation

## 2.1. JDK

FrontLine components runs on a JVM and requires a modern Hotspot-based JDK 8 or 11.

We recommend you use JDK builds from AdoptOpenJDK.

Other JVMs such as OpenJ9 are not supported.

## 2.2. Linux

FrontLine and injectors are intended to be running on Linux 64 bits.
Injectors are intended to run on Kernel >= 3.10. It's possible to use OSX as a development environment.
Windows and Unix platforms such Solaris or AIX are not supported.

As FrontLine is about duration measurement and logging events in time, we advice that:

- your system clock is properly synchronized from an NTP server
- you disable power saving Linux features, so clock source doesn't actually shift and stays monotonic

Make sure that the JVM processes run with a user with sound permissions.

## 2.3. Cassandra

If you upgrade from a previous FrontLine version, we recommend you to keep a backup of your data.

### 2.3.1. Download

Download and install Cassandra.

As of FrontLine 1.6.2, we require at least Cassandra 3.10. FrontLine has been tested against Cassandra 3.10 to 3.11.3. If possible, we advise you go with the latest stable version.

### 2.3.2. Deployment

Running a single node (without clustering) is a good start.

For an initial evaluation, you can host FrontLine and the Cassandra instance on the same host. Consider a 8 cores (4 cores with hyperthreading) host with 8Gb of RAM and 20Go of disk space.

### 2.3.3. Configuration

The default configuration is a good start.

Finally, remember the host you configured, as you will need it later to configure the contact points of FrontLine. Keyspace and replication configuration will be handled by FrontLine.

## 2.4. FrontLine Components

### 2.4.1. Download

FrontLine is packaged as a zip bundle that can be downloaded from our maven repository.

It's not currently possible to browse our repository. You must directly hit the proper urls. We will make browsing possible in a future version.

#### Manually

You can download each component manually (only for on-premise customers).

```
REPLACE_WITH_YOUR_REPOSITORY_URL/content/repositories/releases/io/gatling/frontline/fr
ontline-bundle/REPLACE_WITH_LATEST_FRONTLINE_VERSION/frontline-bundle-
REPLACE_WITH_LATEST_FRONTLINE_VERSION-bundle.zip
```

#### Scripted

A more convenient way is to use a script. Here's an example that downloads all the components in a directory of your choice, defaulting to `./frontline-latest`. This script is only useful for on-premise customers.

*example.sh:*

```
#!/bin/sh

# The two variables you must change
version=REPLACE_WITH_LATEST_FRONTLINE_VERSION

destination=./frontline-latest

repo_url=REPLACE_WITH_YOUR_REPOSITORY_URL

mkdir ${destination}
cd ${destination}

archive=frontline-bundle-${version}-bundle.zip
wget ${REPLACE_WITH_YOUR_REPOSITORY_URL
}/content/repositories/releases/io/gatling/frontline/frontline-bundle/${version}/
${archive}
```

### 2.4.2. API HTTP Server

The API is the backend of FrontLine's dashboard. You have to to launch it first in order to create or update the FrontLine schema in the Cassandra database.

### 2.4.3. Launch

You can launch the API in the background using the following command:

```
[... frontline-bundle ]$ ./bin/frontline
```

The dashboard will then be accessible by default on port `10542`. You need to connect to the dashboard to fill in your license key.

The API will log its PID and write it to a `pidfile` which names will also be echoed. You can provides you own path to a custom pidfile this way:

```
[... frontline-bundle ]$ ./bin/frontline -p pidfile
```

Using the foreground mode will cancel the handling of a pidfile.

#### Configuration

Check the `conf/frontline.conf` file for parameters you might want to edit.

```
licenseKey = REPLACE_WITH_YOUR_LICENSE_KEY ①
```

① Provided license key

```
http {
  port = 10542 ①
  ssl { ②
    #certificate = "/path/to/domain.crt" ③
    #privateKey = "/path/to/domain.key" ④
    generateSelfSignedCertificate = false ⑤
  }
}
```

① API HTTP bind port

② SSL configuration, activated if both `certificate` and `privateKey` are uncommented and points to valid files, or if `generateSelfSignedCertificate` is true.

③ Path to the certificate (or full chain) file. Must be an X.509 certificate chain file in PEM format.

④ Path to the private key file. Must be a PKCS#1 or PKCS#8 private key file in PEM format.

⑤ For testing purpose, you can make FrontLine produce a self signed certificate

```
injector {
  httpPort = 9999 ①
  startTimeout = 120
  enableLocalPool = false ②
}
```

① Injectors HTTP listening port, so FrontLine can connect and collect the stats

② Enable local injector pool (not for production use)

```
security {
  superAdminPassword = gatling ①
  secretKey = "FmXUsw[d2QDLVddD" ②
}
```

① MUST BE CHANGED! password for the FrontLine superAdmin account

② MUST BE CHANGED! key for encrypting cookies. Must be 128, 192 or 256 bit (not bytes) long.

```
cassandra { ①
  contactPoints = [{
    host = localhost
    port = 9042
  }]
  keyspace = gatling
  replication = "{'class':'SimpleStrategy', 'replication_factor': 1}"
  batchGroupingSize = 25
  credentials { ②
    #username = "hello"
    #password = "world"
  }
  runsCleanup { ③
    #timeOfDay = "15:10" ④
    #maxRunsBySimulation = 30 ⑤
    #maxRunAge = 100 ⑥
  }
}
```

① Can be adapted to your current Cassandra cluster configuration.

② The username/password credentials for connecting to Cassandra

③ You can configure daily cleanups for your runs in this part.

④ The hour of the daily cleanup, mandatory to activate the feature. The format is ISO 8601 (e.g.: 17:45).

⑤ The maximum number of runs by simulation. Can be combined with <6>.

⑥ The max age for the runs, in days. Can be combined with <5>.

```
ldap { ①
  #host = localhost ②
  #port = 389 ③
  #baseDn = "dc=example,dc=com" ④
  #distinguishedName = "cn=John Doe,ou=Users,dc=example,dc=com" ⑤
  #password = "secret" ⑥
  #usernameAttribute = uid⑦
  #firstNameAttribute = givenName
  #surnameAttribute = sn
  #mailAttribute = mail
  #connectTimeoutMs = 5000⑧
  #responseTimeoutMs = 10000⑨
  #personObjectClass = person⑩
}
```

① The LDAP configuration, use this part of the config only if you want to enable LDAP based user management

② Uncommenting this line enable LDAP based user management. Correspond to your LDAP server IP address / hostname

③ The port used to access your LDAP server.

④ The base DN where your users are stored in your LDAP

⑤ The distinguished name of a read-only technical account used to search on your LDAP

⑥ The password of the above technical account

⑦ You can override default attribute names in LDAP

⑧ The connect timeout to your LDAP

⑨ The response timeout when searching your LDAP

⑩ The objectClass of your users if they have one. Used to filter out search results

```
ldap {
  ssl { ①
    #format = "PEM | JKS" ②
    pem { ③
      #serverCertificate = "/path/to/domain.pem" ④
      #clientCertificate = "/path/to/domain.pem" ⑤
      #privateKey = "/path/to/domain.key" ⑥
    }
    jks { ⑦
      #trustStore = "path/to/truststore.jks" ⑧
      #trustStorePassword = "secret" ⑨
      #keystore = "path/to/keystore.jks" ⑩
      #keystorePassword = "secret" ⑪
    }
  }
}
```

① Your TLS configuration for LDAP (you don't need this part if you use plain LDAP)

② Choose what will be the format of your trust store/key store. Can be either PEM or JKS

③ The configuration that will be used if you chose "PEM" in the format

④ Path to the server certificate if your LDAP certificate is not signed by a JDK trusted CA

⑤ Path to the client certificate if you need mutual authentication

⑥ Path to the client private key if you need mutual authentication. The key format must be PKCS8

⑦ The configuration that will be used if you chose "JKS" in the format

⑧ Path to the trust store containing the server certificate if your LDAP certificate is not signed by a JDK trusted CA

⑨ Password for the trust store

⑩ Path to the key store containing client certificate and private key if you need mutual authentication

⑪ Password for the key store

```
grafana {
  #url = "http://localhost:3008/dashboard/db/frontline-requests" ①
}
```

① Url to your Grafana dashboard using the Frontline datasource (create a link in Frontline dashboard to the Grafana dashboard)

If you want to modify a value, don't forget to uncomment the line, by deleting the # sign. Any changes to the frontline.conf file needs a FrontLine restart to take effect.

See HOCON documentation for more information on this format.

#### Source Control System Client

If you intend to have FrontLine build tests from sources, it needs to be able to fetch the test sources from your remote source repository, ie:

- a client for your Source Control System (ex: git, svn, perforce, etc) to be installed on the API host
- this client to be available for the user running the API JVM process

#### Build Tool Client

If you intend to have FrontLine build tests from sources, it FrontLine needs to be able to build the fetched resources, ie:

- a client for your build tool (ex: sbt, maven, gradle, etc) to be installed on the API host
- this client to be available for the user running the API JVM process

### Network Access

The FrontLine host needs network access to:

- your Cassandra cluster
- your source repository (if building from sources)
- your binary repositories (if building from sources), typically:
  - Maven central repository: https://repo1.maven.org/maven2
  - GatlingCorp maven repository: http://repository.gatling.io
  - JCenter repository (sbt and gradle users only): https://jcenter.bintray.com/
- the hosts where it will try to deploy Gatling injectors
- your cloud provider API (if deploying on-demand instances on public cloud providers)

Don't forget to open the `22` (for SSH) and `9999` (for HTTP) ports on the injectors. If you don't, your runs will appear as `Broken`.

### Injector Deployment Credentials

Check section 3 of this document.

### Permissions

- Execute permission to JDK path
- Execute permission to source control system client
- Execute permission to build tool client
- Read permission to unzipped FrontLine bundle
- Read/write permission to the logs directory
- Read/write/exec permission on tmp directory If exec permission is not possible because `/tmp` is mounted with `noexec`, you'll have to configure a different directory without `noexec`. Edit the FrontLine launch script and pass an additional System properties `-Djna.tmpdir=PATH_TO_DIR_WITHOUT_NOEXEC`. If you don't you'll run into an issue such as `java.lang.UnsatisfiedLinkError: /tmp/jna-3506402/jna4812891826558064540.tmp: /tmp/jna-3506402/jna4812891826558064540.tmp: failed to map segment from shared object: Operation not permitted`.

### Logging

The API server uses the Logback library for logging. By default, it will log on the filesystem, check logback.xml file. Feel free to tune the default behavior if needed.

### LDAP

FrontLine is able to use LDAP to manage its users. The LDAP mode has been tested with OpenLDAP, and Active Directory servers, but it should work with all regular LDAP implementations.

**Run Cleanup**

FrontLine can be configured to automatically delete runs based on max-age and/or max number of runs by simulation.

# Chapter 3. Injectors Deployment

FrontLine enable users to configure either on demand or on-premises pools. In FrontLine, pools are instances cluster where you deploy Gatling instances and your simulations.

Valid characters for a pool name are letters, digits, space, dash and underscore.

## 3.1. Requirements

The hosts running the Gatling injectors must:

- run on Linux 64 bits, with Kernel >= 3.10
- have a JDK 8 or 11 installed
- be reachable from the FrontLine host over SSH (port 22) and HTTP (port 9999 by default)
- have a ssh key with no password configured

## 3.2. Local

It's possible to have FrontLine use a "Local" pool to deploy a single injector on the same host. This option is turned off by default and has to be enabled:

*frontline.conf:*

```
frontline {
  injector {
    enableLocalPool = true
  }
}
```

This option is only intended to be used for demos and as a quick start when evaluating FrontLine.

It should definitively be disabled once your FrontLine installation will go live, or you'd risk ending up with FrontLine lacking resources (CPU, network) because a load test is eating all of them.

## 3.3. On-premises

It's very easy to configure on-premises pools from FrontLine:

- Create a pool
- Create a host by providing hostname, username, credentials and optional custom working directory (default is `/tmp`). The working directory should be executable.
- Assign the created pool to this host

## 3.4. On Demand

FrontLine is currently managing five different cloud providers: AWS, GCE, OpenStack, DigitalOcean and Microsoft Azure.

### 3.4.1. AWS

There are requirements before creating an AWS pool:

- Create an AMI from your AWS console, or ask GatlingCorp to share one with you.
- Configure AWS API access keys on the FrontLine host using one of these methods:
  - 1. If you've installed FrontLine on AWS EC2, you can directly set a IAM Role to the instance.
  - 2. Environment Variables – AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
  - 3. Java System Properties – aws.accessKeyId and aws.secretKey
  - 4. The default credential profiles file – typically located at `~/.aws/credentials`. See AWS Credentials File Format

FrontLine requires the following permissions (or grant `AmazonEC2FullAccess` if you don't care about fine-grained permissions):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:Describe*",
        "ec2:CreateTags",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "ec2:AllocateAddress",
        "ec2:AssociateAddress",
        "iam:PassRole" ①
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

① ONLY REQUIRED WHEN SETTING INSTANCE PROFILE ON INJECTORS

The AMI needs to have a JDK from 8 to 11 installed, a key pair without password configured and the port 22 and 9999 opened.

### 3.4.2. GCE (On-premises license only)

There are requirements before creating a GCE pool:

- Create a project from Google console
- Enable `Google Compute Engine API` from Google API Manager console
- Create a Service Account key from Google console: API & Services ⇒ Credentials ⇒ Create credentials ⇒ Service account key. We support JSON, P12 & PEM keys.
- Create a template from GCE console

The GCE Account used must have the `instanceAdmin` role.

The template must have the port 22 and 9999 opened.
The template needs to have a JDK from 8 to 11 installed and a key pair without password configured.

### 3.4.3. OpenStack (On-premises license only)

There are requirements before creating a OpenStack pool:

- Get credentials information from `Access & Security` tab.
- Create an image (snapshot) from an existing instance.

The OpenStack User might need some special permissions to launch instances.

The image needs to have a JDK from 8 to 11 installed, a SSH key pair without password configured and the port 22 and 9999 opened.

### 3.4.4. DigitalOcean (On-premises license only)

There are requirements before creating a DigitalOcean pool:

- Create an API Token from `API` tab, with the write scope.
- Create an image (snapshot) from an existing droplet.

The image needs to have a JDK from 8 to 11 installed, a SSH key pair without password installed and the port 22 and 9999 opened.

### 3.4.5. Microsoft Azure (On-premises license only)

There are requirements before creating an Azure pool:

- Get the credentials to Microsoft Azure: follow the Azure documentation and save the key, client, subscription and tenant id.
- Create a virtual network.

- Create an image by following the Azure documentation
- Create and save a SSH key pair without password.

The Azure User used must have the `Virtual Machine Contributor`, `Network Contributor`, and `Storage Account Contributor` permissions.

The image needs to have a JDK from 8 to 11 installed and the port 22 and 9999 opened.