FrontLine Installation Guide

Gatling Corp

Version 1.7.1, 2019-05-14

Table of Contents

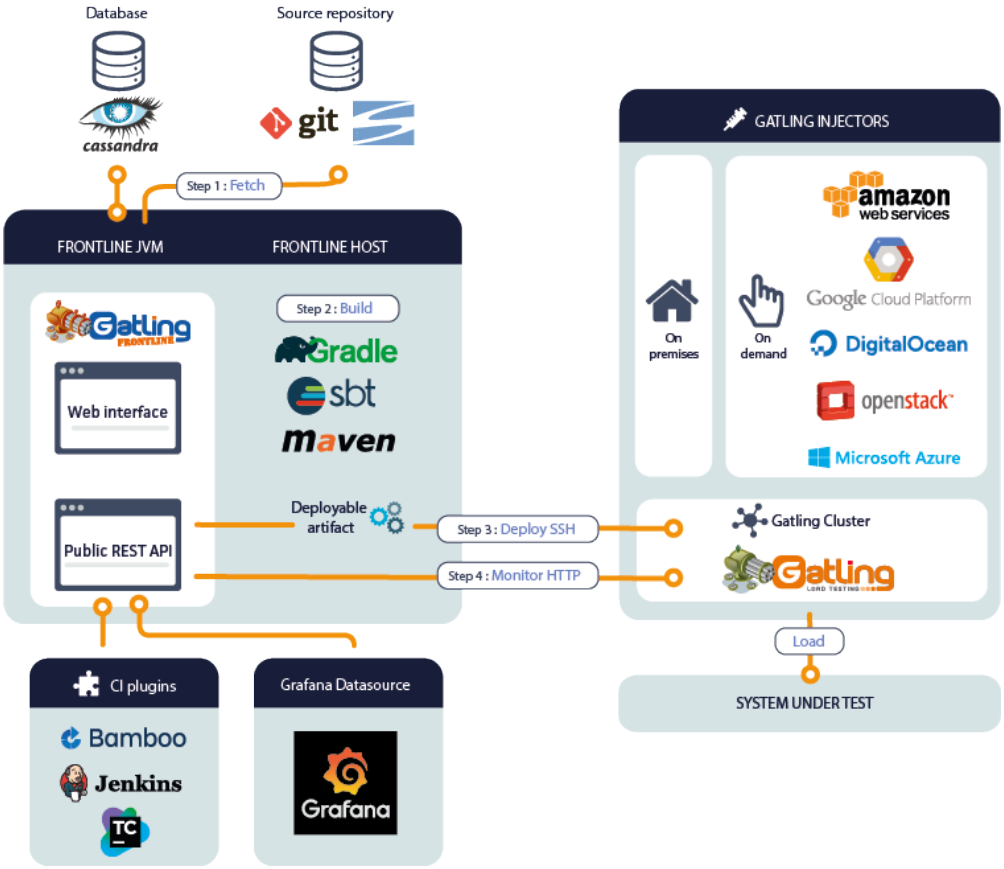
Chapter 1. Introduction

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: 1.7.1 at the time this document was edited

Chapter 2. Manual 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.

As of Cassandra 3, JDK 11 support is flagged as experimental. As a consequence, we only support running Cassandra with JDK 8. JDK 11 support is limited to FrontLine server and Gatling injectors only.

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 1.7.1, we require at least Cassandra 3.10. FrontLine has been tested against Cassandra 3.10 to 3.11.4. 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 hyper-threading) 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 creation will be handled by FrontLine.

For most use cases, one single Cassandra node suffice! Only run a Cassandra cluster if you need it and know how to operate it. In this case, make sure to configure replication properly in `frontline.conf`.

Source Control System Client (typically git)

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 FrontLine host
- this client to be in the PATH and executable for the user running the FrontLine JVM process

Build Tool Client (typically maven, gradle or sbt)

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 FrontLine host
- this client to be in the PATH and executable for the user running the FrontLine JVM process

Make sure that the build tool will be configured so that it will be able to download artifacts, typically if your organization enforces repository mirrors.

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
  - JCenter repository (sbt and gradle users only): https://jcenter.bintray.com/
  - Gradle plugins portal: https://plugins.gradle.org
  - or instead, the internal mirrors your organization might be enforcing
- 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`.

Rack and Data Centers

Cassandra new java driver uses data center as a mandatory value to initialize the client. Users who want to transition to a properly scaled cluster might need to adjust.

From the default configuration file, we can read:

```
endpoint_snitch -- Set this to a class that implements IEndpointSnitch.

The snitch has two functions:
- it teaches Cassandra enough about your network topology to route requests
efficiently
- it allows Cassandra to spread replicas around your cluster to avoid correlated
failures. It does this by grouping machines into "datacenters" and "racks." Cassandra
will do its best not to have more than one replica on the same "rack" (which may not
actually be a physical location)

CASSANDRA WILL NOT ALLOW YOU TO SWITCH TO AN INCOMPATIBLE SNITCH ONCE DATA IS INSERTED
INTO THE CLUSTER.

This would cause data loss. This means that if you start with the default
SimpleSnitch, which locates every node on "rack1" in "datacenter1", your only options
if you need to add another datacenter are GossipingPropertyFileSnitch (and the older
PFS).  From there, if you want to migrate to an incompatible snitch like Ec2Snitch you
can do it by adding new nodes under Ec2Snitch (which will locate them in a new
"datacenter") and decommissioning the old ones.
```

Cassandra default endpoint snitch is `SimpleSnitch`. Which defaults values of data center and rack to "datacenter1" and "rack1". They are hardcoded and therefore can't be changed.

When using other snitches (E.g., `GossipingPropertyFileSnitch`), Cassandra will use the content of the `cassandra-rackdc.properties` configuration file to build it's network topology or the private IP of the machine (E.g.: Ec2Snitch)

When scaling out, consider these two cases to migrate your (single) instance to a proper cluster configuration:

- Single instance (straighforward)
- Multiples instances of any snitches to any others (probably incompatible) snitches

First method being something like (also applies to cluster name):

```
update system.local set data_center = 'dc1' where key = 'local';
update system.local set rack = 'rc1' where key = 'local';
```

Second method implies (as stated in Cassandra documentation) to add instances with proper configuration and removing the old ones as you go in order to avoid data loss.

2.4. FrontLine Server

2.4.1. Download

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

On launch, FrontLine will create or update the FrontLine schema in the Cassandra database.

2.4.2. Launch

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

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

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

FrontLine 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, you should not edit this configuration directly from this file, but from the web interface

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

① FrontLine 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
  enableLocalPool = false ②
  disableKubernetesTrustManager = true ③
}
```

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

② Enable local injector pool (not for production use)

③ When connecting to your kubernetes API, determine if you want a true trust manager to be used to validate your certificate. Disabled by default.

```
security {
  superAdminPassword = gatling ①
  secretKey = "MUST BE CHANGED!" 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
  localDataCenter = datacenter1 ②
  contactPoints = [{
    host = localhost
    port = 9042
  }]
  keyspace = gatling
  replication = "{'class':'SimpleStrategy', 'replication_factor': 1}"
  batchGroupingSize = 25
  credentials { ③
    #username = "hello"
    #password = "world"
  }
  runsCleanup { ④
    #timeOfDay = "15:10" ⑤
    #maxRunsBySimulation = 30 ⑥
    #maxRunAge = 100 ⑦
  }
}
```

① Can be adapted to your current Cassandra cluster configuration.

② The local data center your contact points belong to. Cassandra's value with SimpleStrategy is "datacenter1".

③ The username/password credentials for connecting to Cassandra

④ You can configure daily cleanups for your runs in this part.

⑤ The hour of the daily cleanup, mandatory to activate the feature. The format is ISO 8601 (e.g.: 17:45).

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

⑦ 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.

Injector Deployment Credentials

Check section 4 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

FrontLine 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. Automated Installation With Ansible

The installer can be run from anywhere. However, it can only install Gatling FrontLine on the following distributions:

- Amazon Linux 1

Images of Gatling FrontLine published to the AWS Marketplace are made using this installer. The directory layout will be the same.

3.1. Requirements

Ansible will be used to perform the installation. You'll need:

- Python 2.7.7+ or 3.5+
- Ansible 2.7.6+
- An AWS EC2 instance running the Amazon Linux 1 distribution, its instance type must be at least `t2.large`.

In case you don't already use Ansible, you can download it from here. You do not need any Ansible knowledge to use this installer.

If you do want to know more about Ansible, you can check its user guide.

3.1.1. Configuring a proxy

Ansible will use the shell's proxy when running the script in your computer.

If you need to specify a proxy for the remote machine on which Gatling FrontLine will be installed, you can add environment variables at the installer level:

frontline.yml

```
 - hosts: all

   vars:
     # ...

   roles:
     # ...

+  environment:
+    http_proxy: http://proxy.bos.example.com:8080
+    https_proxy: http://proxy.bos.example.com:8080
```

3.2. Using the installer

3.2.1. Downloading and integrity checking

You can download the installer here:

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

We suggest you download and check the integrity of the installer by doing the following:

download.sh:

```
#!/bin/bash

# The two variables you must change
version=REPLACE_WITH_LATEST_FRONTLINE_VERSION
repo_url=REPLACE_WITH_YOUR_REPOSITORY_URL

archive_name=frontline-installer-${version}-bundle.zip
archive_url=${repo_url}/content/repositories/releases/io/gatling/frontline/frontline-
installer/${version}/${archive_name}

wget ${archive_url}
wget ${archive_url}.sha1

echo "$(cat ${archive_name}.sha1)  ${archive_name}" | sha1sum     -c -
# For MacOS users:                                  shasum -a 1 -c -
```

If you have aliases on `echo` and/or `cat`, you can prefix them with an anti-slash to make sure you are using the original command instead, as such: `\echo`, `\cat`.

3.2.2. Configuring the installer

After unzipping the installer, here are the next things you need to do before running it.

Things you need to do before running the installer:

- Fill in the `frontline.hosts.yml` file
- Fill in your provided UUID in `frontline.yml`

frontline.hosts.yml:

```
all:
  hosts:
    frontline:
      ansible_host: REPLACE_WITH_YOUR_FRONTLINE_HOST
      ansible_ssh_private_key_file: REPLACE_WITH_YOUR_PATH_TO_PRIVATE_KEY_FILE
      ansible_user: REPLACE_WITH_YOUR_FRONTLINE_HOST_USER
```

frontline.yml:

```
frontline:
  version: REPLACE_WITH_LATEST_FRONTLINE_VERSION
  uuid: REPLACE_WITH_YOUR_OWN_UUID
```

3.2.3. Running the installer

Type in `./installer.sh` and wait for the installation to end.

The script will ask for a sudo password. On Amazon Linux 1, the default user (`ec2-user`) is able to sudo without any password, so you can just type in `<Enter>` twice.

The script is idempotent. It means you can run it multiple times without compromising your previous installation. It also means you can start over a previous failed run and continue on.

The installer cannot be used to upgrade to a new version yet.

3.2.4. Running FrontLine

Services will be configured for each installed components of Gatling FrontLine. They will automatically start on boot.

You can control them with the `service` command:

```
sudo service {cassandra|frontline|nginx} {start|stop}
```

Gatling FrontLine depends on Cassandra, it will wait on its availability when starting.

Nginx reverse proxy to Gatling FrontLine, but will still start if it is not available.

3.3. Installation Layout

Two users will be created, `cassandra` and `frontline`, that will be used by, respectively, Cassandra and Gatling FrontLine.

If you want a file to be access by Gatling FrontLine (E.g.: private keys), make sure to properly modify its `group:user` to `frontline:frontline`.

Installation and configuration directories:

```
/opt/cassandra
/opt/frontline
```

Nginx is installed using the packager of the distribution.

All other dependencies (I.e.: builders), are also installed in `/opt`.

Versions are installed in their own directories and linked to `/opt/cassandra` and/or `/opt/frontline`. Previous configuration files won't be overwritten on update.

Home and data directories:

```
/var/opt/cassandra
/var/opt/frontline
```

SystemV configuration files:

```
/etc/sysconfig/cassandra
/etc/sysconfig/frontline
```

Any changes to the `PATH` of each services can be pushed in these files.

SystemV services files:

```
/etc/init.d/cassandra
/etc/init.d/frontline
```

Logging directories:

```
/var/log/cassandra
/var/log/frontline
```

3.4. Troubleshooting

If anything goes wrong during the installation. You can turn on Ansible logging by modifying the following line in the `frontline.yml` file, switching the value of `no_log` from `True` to `False`:

```
no_log: False
```

3.5. Ansible roles

In Ansible, recipes are called roles, while sets of recipes are named playbooks.

The installer is a playbook with multiples roles.

Here, we will describe each roles and their configuration.

3.5.1. Main configuration and optimizations roles

```
roles:
  - check_compatibility
  - disable_automatic_upgrades
  - upgrades
  - optimizations
  - hostname
  - motd
```

First, we make sure the distribution is compatible with the installation script.

On Amazon Linux 1, most security packages are updated on startup, including the bundled JDK7. As we don't any any conflicts, we disable the automatic upgrade for Java, and keep the everything else.

Then, we upgrade every installed packages.

We change the hostname of the instance to `frontline` to avoid conflicts with custom VPC inside AWS that have DNS resolution disabled.

Finally, we update the MOTD (Message Of The Day) of the instance that is shown when you log inside the instance.

3.5.2. Java role

```
roles:
  - adoptopenjdk
```

We install our own JDK to avoid conflict with any other installed JDK.

This role can be removed if you want to use something else. If you do so, you'll need to make sure a JDK is available from the `cassandra` and `frontline` users `PATH`.

```
# Download AdoptOpenJDK to avoid conflicts with (possibly) existing java installation.
# This can be skipped by commenting the adoptopenjdk role below.
adoptopenjdk:
  # Do mind that Cassandra support of Java 11 is still experimental
  major: 8 # Possibles values are: [8, 11], should be kept to 8 as Cassandra support
of Java 11 is still experimental.
  flavor: jdk # Possible values are: [jdk, jre], should be kept to JDK so that
FrontLine can compile your Gatling scenarios.
  heap_size: normal # Possibles values are: [normal, large]
```

3.5.3. Git and builders roles

```
roles:
  - git
  - gradle
  - maven
  - sbt
```

Git is installed to provide repository fetching capability to Gatling FrontLine.

Depending on which builder you use for your projects, you can disable the other ones.

Note that each builders export their `PATH` in an Ansible variable that is used later by the Cassandra and Gatling FrontLine role.

```
# This can be skipped by commenting the gradle role below.
gradle:
  mirror: https://services.gradle.org/distributions
  version: 5.2.1
# This can be skipped by commenting the maven role below.
maven:
  mirror: https://www.apache.org/dist
  version: 3.6.0
```

Sbt does not need any configuration nor version.

Finally, Gradle, Maven and SBT roles will prefetch any needed dependencies in order to speed up the first simulation you will start on your new Gatling FrontLine installation.

3.5.4. Cassandra and FrontLine roles

```
roles:
  - cassandra
  - frontline
```

This will install Cassandra and Gatling Frontline with the previously shown directory layout.

Cassandra version can be modified directly in the `frontline.yml` file:

```
# Cassandra is the database we use
# Mirror link can be changed from here in case it is no longer available.
cassandra:
  mirror: https://www.apache.org/dist
  version: 3.11.4
```

Gatling FrontLine configuration should be modified with your own credentials:

```
frontline:
  version: REPLACE_WITH_LATEST_FRONTLINE_VERSION
  uuid: REPLACE_WITH_YOUR_OWN_UUID
```

3.5.5. Nginx Reverse Proxy role

Nginx only purpose is to reverse proxy Gatling FrontLine to the port 80.

```
roles:
  - nginx
```

It can be skipped or modified if you want to provide your own configuration (SSL, etc.)

```
# Nginx will be used to proxy FrontLine to the HTTP port 80, as Gatling FrontLine
doesn't run as root.
# This can be skipped by commenting the nginx role below.
nginx:
  reverse_proxy:
    name: frontline
    host: http://127.0.0.1:10542
    server_name: _
  user: nginx
```

Chapter 4. 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.

4.1. 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

4.2. Tuning

We recommend that you tune your OS for maximum performance. Please check the Gatling documentation.

4.3. 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.

4.4. 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

4.5. On Demand

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

4.5.1. AWS

GatlingCorp provides certified AMIs that you choose in the FrontLine AWS configuration. This AMI will be used as a base for your injectors. However, you can still build a custom one with a JDK from 8 to 11 installed, a key pair without password configured and the port 22 and 9999 opened.

You'll also need to 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

4.5.2. 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.

4.5.3. 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.

4.5.4. 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.

4.5.5. Microsoft Azure (On-premises license only)

There are requirements before creating an Azure pool:

- Get the credentials to Microsoft Azure: follow this link https://www.inkoop.io/blog/how-to-get-azure-api-credentials/ and save the subscription ID, tenant ID, client ID and client secret.
- 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.

4.5.6. Kubernetes / OpenShift

There are requirement before creating a Kubernetes/Openshift pool:

- Your Kubernetes API should be reachable by FrontLine
- A node which relay Nodeport (kube-proxy process running) should be reachable by FrontLine
- A namespace/project to spawn injectors inside
- A service account for FrontLine (You will need the token of that account)
- This service account must have an edit role on the namespace that will be used (FrontLine will create and destroy pods and services)
- Dockerhub registry should be reachable to fetch injector docker image. Otherwise you will need to push a frontline injector image inside one of your reachable registry. The docker image is buildable from sources available at https://github.com/gatling/frontline-injector-docker-image
- If you are using EKS or GKE, you need to setup your security group/firewall to enable FrontLine to reach one of your node on your configured Kubernetes Nodeport range (by default this range is 30000-32767). Additionally for EKS, you need to enable nodes to contact each other on the 9999 injector port and 10022 ssh port.