# Table of Contents

1. Introduction .......................... 1
   1.1. Audience and Goals .................. 1
   1.2. Architecture Overview ............... 1
   1.3. Document Conventions ............... 2

   2.1. Upgrading From a Previous Version .... 3
   2.2. JDK ................................ 3
   2.3. Linux ................................ 3
   2.4. Cassandra .......................... 3
      2.4.1. Important: Backups ............... 4
      2.4.2. Download ......................... 4
      2.4.3. Deployment ....................... 4
      2.4.4. Storage options ................. 4
      2.4.5. Configuration .................... 5

2.5. FrontLine Server .................... 6
   2.5.1. Download ......................... 7
   2.5.2. Launch ............................ 7

3. Automated Installation With Ansible .. 13
   3.1. Requirements ....................... 13
      3.1.1. Configuring a proxy ............... 13
   3.2. Using the installer .................. 14
      3.2.1. Downloading and integrity checking ... 14
      3.2.2. Configuring the installer ........... 15
      3.2.3. Running the installer ............... 15
      3.2.4. Running the installer locally ......... 15
      3.2.5. Running FrontLine .................. 16
   3.3. Installation Layout ................. 16
   3.4. Troubleshooting ..................... 17

4. Deploying FrontLine on Kubernetes ...... 18
   4.1. Getting FrontLine’s Docker image ...... 18
   4.2. Getting FrontLine’s injectors Docker image . 18
   4.3. Setup Cassandra ..................... 18
   4.4. Setup FrontLine ..................... 20
      4.4.1. Setup RBAC (optional) ............... 20
      4.4.2. Setup Docker Hub credentials as a secret (Optional) ... 20
      4.4.3. Setup FrontLine .................... 21

5. Injectors Deployment ................... 23
   5.1. Injector requirements ............... 23
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 WebApp:
  - A rich web UI
  - A public REST API
- The FrontLine Extensions:
  - A Grafana Datasource for querying FrontLine metrics
  - Continuous Integration plugins for Jenkins, Bamboo and TeamCity or any solution that can trigger a bash script
- Gatling injectors to be used like standard Gatling OSS, with extra features so FrontLine can
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**: 1.11.1 at the time this document was edited
Chapter 2. Manual Installation

2.1. Upgrading From a Previous Version

- Shut FrontLine process down
- Perform a backup of your Cassandra data (eg /var/lib/cassandra directory, but might vary depending on how you’ve installed Cassandra)
- Download and unzip updated FrontLine bundle
- Copy conf directory content from previous installation to the new one
- Copy keys directory from previous installation (if it exists) to the new one (cp -r previous_frontline_folder/keys new_frontline_folder)
- Start new FrontLine process

2.2. 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.

2.3. 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.

2.4. Cassandra
2.4.1. Important: Backups

We recommend you to perform regular backups of your data, all the more prior to upgrading FrontLine to a new version.

For a simple single node setup (which we recommend if you're not experienced with operating Cassandra clusters), performing a Cassandra backup simply consists of make a copy (eg, tar.gz) of the directories that are configured in your `cassandra.yaml` file as the following entries:

- `data_file_directories`
- `commitlog_directory`
- `saved_caches_directory`

2.4.2. Download

Download and install Cassandra.

As of FrontLine 1.11.1, 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.

2.4.3. Deployment

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

For an initial evaluation, you can host FrontLine and Cassandra on the same host.

If hosting FrontLine and Cassandra on the same instance, consider a 8 cores (4 cores with hyper-threading) host with 16Gb of RAM and 20Go of disk space. Otherwise, considering a 4 cores (2 cores with hyper-threading) host with 8 Gb of RAM for each instance, and 20 Gb of disk space for Cassandra.

FrontLine itself requires little disk space, as only uploaded private keys are stored on FrontLine’s host.

2.4.4. Storage options

There are simple rules to follow when choosing the right type of instance to use for a Cassandra virtual machine:

- SSD is always best
- Avoid network disks

For any cloud provider, make sure you use a storage system that support at least 3000 IOPS as we encountered slow downs with 1500 IOPS.

For AWS, avoid burstable instance types on AWS EC2 (T2), they don’t offer good performance for Cassandra deployments:
• C5d are equipped with NVMe SSD, which are great for installation running Cassandra and FrontLine on the same virtual machine
• I3 are better for virtual machines reserved for Cassandra

In Azure, use instances with the s prefix:

• FsV2 are great for installations running Cassandra and FrontLine on the same virtual machine
• LsV2 are better for virtual machines reserved for Cassandra

2.4.5. 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.

Network Access

The FrontLine host needs network access to:

• your Cassandra cluster
• your source repository (if you want to build from sources)
• your binary repositories (if you want to download pre-packaged simulations), 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.

Do NOT use a load balancer on a Cassandra cluster. The Cassandra network driver already does its own load balancing, adding a load balancer on top of it would break cluster communication.

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:

```plaintext
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):

```plaintext
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.

### 2.5. FrontLine Server
2.5.1. 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/frontline-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.

2.5.2. 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 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, FrontLine will ask for it when you launch it or when your current license is expired
http {
  port = 10542 ①
  cookieMaxAge = 604800 ②
  ssl { ③
    #certificate = "/path/to/domain.crt" ④
    #privateKey = "/path/to/domain.key" ⑤
    generateSelfSignedCertificate = false ⑥
  }
  proxy { ⑦
    #host = ""
    #port = 80
  }
}

① FrontLine HTTP bind port
② Cookies max-age in seconds (default: 7 days)
③ SSL configuration, activated if both certificate and privateKey are uncommented and points to valid files, or if generateSelfSignedCertificate is true.
④ Path to the certificate (or full chain) file. Must be an X.509 certificate chain file in PEM format.
⑤ Path to the private key file. Must be a PKCS#1 or PKCS#8 private key file in PEM format.
⑥ For testing purpose, you can make FrontLine produce a self signed certificate
⑦ Optional HTTP proxy, enabled when both host and port are filled. This proxy will be used for every HTTP request to Cloud providers APIs and on-demand injectors.

injector {
  httpPort = 9999 ①
  enableLocalPool = false ②
  kubernetes {
    disableTrustManager = 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 password for the FrontLine superAdmin account. FrontLine will create a new password when you launch it for the first time.
key for encrypting cookies. Must be 128, 192 or 256 bit (not bytes) long. FrontLine will create a new secretKey when you launch it for the first time.

cassandra {
    localDataCenter = datacenter1
    contactPoints = [{
        host = localhost
        port = 9042
    }]
    keyspace = gatling
    replication = "{'class':'SimpleStrategy', 'replication_factor': 1}"
    batchGroupingSize = 25
    credentials {
        #username = "hello"
        #password = "world"
    }
    runsCleanup {
        #timeOfDay = "15:10"
        #maxRunsBySimulation = 30
        #maxRunAge = 100
    }
}

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 <7>.
7 The max age for the runs, in days. Can be combined with <6>.
The LDAP configuration, use this part of the config only if you want to enable LDAP based user management.

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

The port used to access your LDAP server.

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

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

The password of the above technical account.

You can override default attribute names in LDAP.

The connect timeout to your LDAP.

The response timeout when searching your LDAP.

The objectClass of your users if they have one. Used to filter out search results.
Your TLS configuration for LDAP (you don’t need this part if you use plain LDAP)
Choose what will be the format of your trust store/key store. Can be either PEM or JKS
The configuration that will be used if you chose "PEM" in the format
Path to the server certificate if your LDAP certificate is not signed by a JDK trusted CA
Path to the client certificate if you need mutual authentication
Path to the client private key if you need mutual authentication. The key format must be PKCS8
The configuration that will be used if you chose "JKS" in the format
Path to the trust store containing the server certificate if your LDAP certificate is not signed by a JDK trusted CA. Optional, will use JDK's default if undefined.
Password for the trust store
Path to the key store containing client certificate and private key if you need mutual authentication. Optional, will use JDK's default if undefined.
Password for the key store

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

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](https://hocon$configurator.org) 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...
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.

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

If using Git to clone repositories using SSH protocol, Git 2.3 is the minimal version supported.

Build Tool Client (typically maven, gradle or sbt)

If you intend to have FrontLine build tests from sources, then 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.

Maven 3.3.9 is the minimal version supported. This is the version shipped in Debian 9 (Stretch).
Chapter 3. Automated Installation With Ansible

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

3.1. Requirements

The installer can be run from anywhere.

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

- Python 2.7.7+ or 3.5+
- Ansible 2.7.6+
- An instance/VM (at least t2.large or equivalent) running on supported Linux distributions

Supported Linux distributions are:

- Amazon Linux 1/2
- CentOS 7
- Debian 8/9/10
- RHEL (RedHat Enterprise Linux) 7

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:
3.2. Using the installer

3.2.1. Downloading and integrity checking

You can download the installer here:

```
REPLACE_WITH_YOUR_REPOSITORY_URL/frontline-installer/REPLACE_WITH_LATEST_FRONTLINE_VERSION/frontline-installer-
REPLACE_WITH_LATEST_FRONTLINE_VERSION.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=1.11.1
repo_url=REPLACE_WITH_YOUR_REPOSITORY_URL

archive_name=frontline-installer-${version}.zip
archive_url=${repo_url}/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, you'll need to fill in your instance's SSH connection info and provided UUID in `configuration.yml`.

`configuration.yml`:

```yaml
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
  vars:
    frontline:
      uuid: REPLACE_WITH_YOUR_OWN_UUID
```

You can also modify in `configuration.yml` whether you want to install build tools (Maven, Gradle, sbt), Nginx or kubectl (required for Kubernetes pools).

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 1/2, 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 the installer locally

⚠️ This is only meant for scenarios in which you can’t run Ansible remotely. Running Ansible locally isn’t the common use case.

In case you don’t have the necessary tools to run Ansible remotely, i.e. running Ansible on your machine in order to install Gatling FrontLine on another machine, you can launch Ansible directly on the machine that will host Gatling FrontLine.

First, you need to copy the inventory `configuration.yml` file vars inside the playbook `frontline.yml` file, as such:
---

- hosts: all

  vars: # part you need to copy
  ...

  roles:
  ...

Then, you will be able to run Ansible directly on the host you intend to install Gatling FrontLine in:

```bash
ansible-playbook
  -b --ask-become-pass
  -c local
  -i localhost,
  frontline.yml
```

### 3.2.5. Running FrontLine

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

You can control them with the `service/systemctl` command:

```bash
# On SysV-based distributions
sudo service {cassandra|frontline|nginx} {start|stop}
# On systemd-based distributions
sudo systemctl {start|stop} {cassandra|frontline|nginx}
```

- 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/lib/cassandra
/var/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

Systemd unit files:

/etc/systemd/system/cassandra.service
/etc/systemd/system/frontline.service

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
Chapter 4. Deploying FrontLine on Kubernetes

If you're using Kubernetes-based injector pools, it is recommended to run FrontLine itself inside Kubernetes too:

- Less configuration is required than when connecting to a cluster from the outside
- It is not necessary to open firewall rules so that FrontLine can contact injectors

4.1. Getting FrontLine’s Docker image

FrontLine's image is hosted as a private image on Docker Hub.

Please contact our support and provide us with your Docker Hub username so we can grant you access.

4.2. Getting FrontLine’s injectors Docker image

FrontLine’s injector image is publicly accessible on Docker Hub.

4.3. Setup Cassandra

This manifest sets up a single-node Cassandra cluster, along with a service to expose it
apiVersion: v1
kind: Service
metadata:
  name: cassandra
spec:
  ports:
  - name: tcp
    port: 9042
  selector:
    app: cassandra

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: cassandra
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      containers:
      - name: cassandra
        image: cassandra:3.11
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: 2
            memory: 3Gi
        ports:
        - containerPort: 9042
        volumeMounts:
        - mountPath: /var/lib/cassandra
          name: cassandra-data
        securityContext:
          capabilities:
            add:
              - IPC_LOCK
        volumes:
        - name: cassandra-data
          # Prefer PersistentVolumeClaims for durability
          hostPath:
            path: <local storage path for Cassandra data>
4.4. Setup FrontLine

4.4.1. Setup RBAC (optional)

If your cluster has RBAC enabled, this manifest configure the necessary permissions for FrontLine:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: frontline
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: frontline-manage-injectors
rules:
- apiGroups: [""
  resources: ["namespaces"]
  verbs: ["get"]
- apiGroups: [""
  resources: ["pods", "pods/exec"]
  verbs: ["create","delete","get","list","patch","update","watch"]
- apiGroups: ["extensions"]
  resources: ["ingresses"]
  verbs: ["create", "delete", "get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: frontline-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: frontline-manage-injectors
subjects:
- kind: ServiceAccount
  name: frontline
```

4.4.2. Setup Docker Hub credentials as a secret (Optional)

If you're not mirroring FrontLine's image in your private registry, you'll need to setup your Docker credentials as a secret to pull FrontLine's image:
kubectl create secret docker-registry docker-hub-credentials \
--docker-server=<your-registry-server> \
--docker-username=<your-name> \
--docker-password=<your-pword> \
--docker-email=<your-email>

4.4.3. Setup FrontLine

This manifest sets up FrontLine, pre configured with your license key and admin credentials. You can then expose FrontLine using LoadBalancer/NodePort services, Ingress, etc...

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: frontline-conf
data:
  frontline.conf: |
    frontline.licenseKey = <YOUR FRONLINE LICENSE KEY>
    frontline.security.superAdminPassword = <YOUR SUPER ADMIN PASSWORD>
    frontline.security.secretKey = <YOUR ENCRYPTION SECRET KEY>
    frontline.cassandra.contactPoints = [{
      host = cassandra.default.svc.cluster.local
      port = 9042
    }]
  logback.xml: |
  <?xml version="1.0" encoding="UTF-8"?>
  <configuration>
    <contextListener class="ch.qos.logback.classic.jul.LevelChangePropagator">
      <resetJUL>true</resetJUL>
    </contextListener>
    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
      <encoder>
        <pattern>%d [%-5level] %logger{15} - %msg%n%rEx</pattern>
      </encoder>
      <immediateFlush>false</immediateFlush>
    </appender>
    <root level="INFO">
      <appender-ref ref="CONSOLE"/>
    </root>
  </configuration>

---
apiVersion: v1
kind: Service
metadata:
  name: frontline
spec:
  ports:
    - name: http
      port: 10542
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontline
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: frontline
template:
  metadata:
    labels:
      app: frontline
spec:
  # Required unless you mirror Frontline in your private registry
  imagePullSecrets:
    - name: docker-hub-credentials
  serviceAccountName: frontline
containers:
  - name: frontline
    imagePullPolicy: Never
    image: gatlingcorp/frontline:{revnumber}
    resources:
      requests:
        cpu: 2
        memory: 4Gi
    ports:
      - containerPort: 10542
    volumeMounts:
      - mountPath: /opt/frontline/conf
        name: frontline-conf
      - mountPath: /opt/frontline/keys
        name: ssh-keys
  volumes:
    - name: frontline-conf
      configMap:
        name: frontline-conf
    - name: ssh-keys
      hostPath:
        path: <local storage path for SSH keys>

Chapter 5. 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, spaces, dashes and underscores.

5.1. Injector requirements

5.1.1. Image

The hosts running the Gatling injectors must:

- run on Linux 64 bits, with Kernel >= 3.10
- have a JDK 8 or 11 installed (AdoptOpenJDK is recommended)
- have a passphrase-less SSH key
- /tmp not mounted with noexec

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

We also recommend that you disable automatic updates and automatic package repositories listing updates. This could eventually lead to upgrading the JDK while running, which could break your simulation run.

5.1.2. Network

The hosts running Gatling injectors must be reachable from Frontline:

- over SSH (port 22)
- over HTTP, on port 9999 (by default, configurable in frontline.conf)

5.1.3. CPU/Memory

Unless your simulation keeps huge amounts of data in memory (eg. for feeders with a lot of data), Gatling is mostly CPU-bound and IO-bound, and will take advantage of CPU and network-heavy hosts.

Therefore, we recommend the following instances:

- AWS: c5.xlarge (4 cores) or c5.2xlarge (8 cores)
- GCE: n1-highcpu-4 (4 cores) or n1-highcpu-8 (8 cores)
- Azure: F4 (4 cores) or F8 (8 cores)

We strongly discourage the use of CPU-burstable instance types:

- AWS: T2/T3 instances
• GCE: f1-micro, g1-small
• Azure: Bs instances

These instance types target light workloads, but sporadic peaks of activity where they can benefit from a temporary increase in CPU performance Gatling is more likely to use 100% of the CPU cores and will not benefit from CPU bursts.

5.2. Certified images

We provide certified, pre-built images for AWS, Azure, GCE & Docker. Those images are available for both JDK 8 & 11 and includes all recommended optimisations. If you can't use our certified images, we provide the templates from which our certified images are built:

• For AWS & GCE: frontline-injector-playbook (requires Ansible & Packer)
• For Docker: frontline-injector-docker-image

5.3. 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.

5.4. 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
5.5. On Demand

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

5.5.1. AWS

Gatling Corp 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 and config files. See Set up AWS Credentials and Region for Development

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

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

① ONLY REQUIRED WHEN SETTING INSTANCE PROFILE ON INJECTORS
5.5.2. 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

![i] The GCE Account used must have the instanceAdmin role.

5.5.3. 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.

![i] 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.

5.5.4. 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.

5.5.5. Microsoft Azure (On-premises license and Azure Marketplace only)

Gatling Corp provides certified images that you choose in the FrontLine Microsoft Azure configuration. This image 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.

There are requirements before creating an Azure pool:

- Create a virtual network.
- Create an image by following the Azure documentation if not using certified images.
- Create and save a SSH key pair without password.

There are two way to provide Azure credentials to your FrontLine instance:
• Get the credentials from Microsoft Azure: follow this link [https://www.inkoop.io/blog/how-to-get-azure-api-credentials/](https://www.inkoop.io/blog/how-to-get-azure-api-credentials/) and save the subscription ID, tenant ID, client ID and client secret.

• Use User Assigned Managed Identities.

⚠️ In both cases, the Azure User used must have the Contributor permission.

### Create a User Assigned Managed Identity

Identities can be created using either the Azure Portal or the Azure CLI.

⚠️ Role should be set under the subscription scope, otherwise Contributor will not suffice. It can't be set under the resource group scope as FrontLine will start Azure virtual machines in a new resource group each time a simulation is launched.

Here are the creation steps:

• Go to [https://portal.azure.com/#create/Microsoft.ManagedIdentity](https://portal.azure.com/#create/Microsoft.ManagedIdentity), set resource name, resource group and location.

• Go to your subscription, click on Access Control (IAM), then Add role assignment.

• Role should be set to Contributor.

• Set "Assign access to" to User assigned managed identity and select the name of your Managed Identity.

• The virtual machine hosting Gatling FrontLine needs to be assigned this identity.

### 5.5.6. Kubernetes / OpenShift

There are requirements before creating a Kubernetes/OpenShift pool:

• Docker Hub should be reachable from your infrastructure, otherwise you'll need to build the injector image and push it to your private registry.

• A service account able to manage pods and services (either attached to FrontLine or for its token).

Additionally, if FrontLine is deployed outside Kubernetes/OpenShift:

• The Kubernetes API should be reachable by FrontLine.

• If using the NodePort mode, firewall rules must be added so that FrontLine can reach Kubernetes nodes on the configured Kubernetes NodePort range (by default, 30000-32767).

⚠️ If your cluster uses RBAC, you'll need a role with the following permissions for FrontLine’s service account:
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: frontline-manage-injectors
rules:
  - apiGroups: [""]
    # "services" can be skipped if FrontLine is deployed inside Kubernetes
    resources: ["pods", "pods/exec", "services"]
    verbs: ["create", "delete", "get", "list", "patch", "update", "watch"]