MiaRec

Installation Guide

MiaRec, Inc.

Table of contents

1. Installation Guide

This documentation describes the procedures required to install and configure MiaRec call recording solution.

The following topics are covered:

2. Hardware Requirements

2.1 Overview

MiaRec solution has flexible architecture supporting various deployment scenarios depending on number of users and requirements to high availability and redundancy.

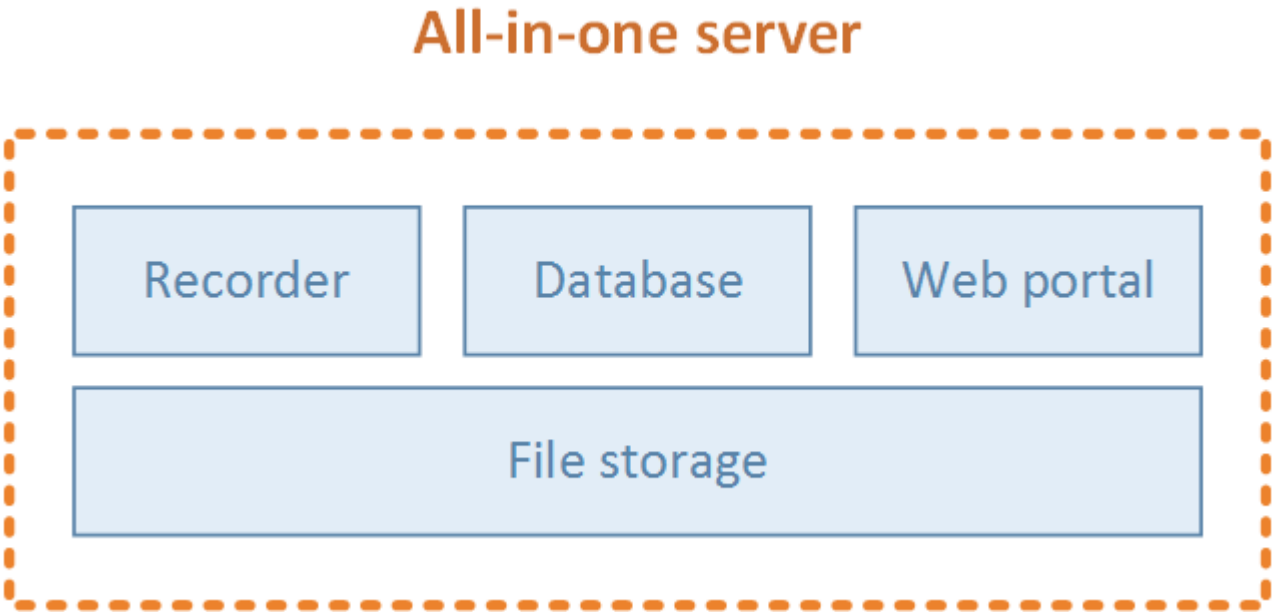
- **All-in-one server**. All components (recorder, database, web portal, storage) are deployed to a single server
- **Decoupled architecture (multiple servers)**. Each component is deployed to a dedicated server for redundancy and load balancing purposes.

"All-in-one" configuration is recommended for up to 2,000 users.

The distributed configuration is recommended for more than 2,000 users.

2.2 All-in-one server

This article provides hardware recommendations for "all-in-one" setup, where all software components (recorder, database, web portal and storage) are deployed in a single server.



"All-in-one" configuration is recommended for deployments up to 2,000 users. For larger deployments we recommend to use decoupled architecture (multiple servers).

Recommended hardware configuration for recording 50-500 users

Physical or virtual server with the following minimum hardware specification:

- **CPU** - Intel CPU quad-core or better. Frequency at at least 20GHz.
- **Memory** - 16 GB or more.
- **Storage**:
  - Two high speed disks (at least 10,000 rpms HDD or preferably SSD) in RAID 1 configuration for storing operating system, program files and database data. Disk space requirements - at least 300GB.
  - High capacity disk array (local or NAS/SAN) in RAID 5/6 configuration for storing audio mp3 files and, optionally, log files. Disk space requirements - 0.24 MB/minute of recording

  For example, in average a business user makes 10 calls per day with a duration 5 minutes. This will end up to 1,000 minutes per user per month (assuming 20 working days). One month of storage for 500 users will require 120 GB of disk space.
- **OS** - Windows Server 2012, 2016, 2019 (64-bit) or Linux RedHat/Centos 7.x

Recommended hardware configuration for recording 500-1,000 users

Physical or virtual server with the following minimum hardware specification:

- **CPU** - Intel CPU six-core or better. Frequency at least 2.3GHz.
- **Memory** - 32 GB or more.
- **Storage**:
  - Two high speed disks (at least 10,000 rpms HDD or preferably SSD) in RAID 1 configuration for storing operating system, program files and database data. Disk space requirements - at least 600GB.
  - High capacity disk array (local or NAS/SAN) in RAID 5/6 configuration for storing audio mp3 files and, optionally, log files. Disk space requirements - 0.24 MB/minute of recording

  For example, in average a business user makes 10 calls per day with a duration 5 minutes. This will end up to 1,000 minutes per user per month (assuming 20 working days). One month of storage for 1,000 users will require 240 GB of disk space.
- **OS** - Windows Server 2012, 2016, 2019 (64-bit) or Linux RedHat/Centos 7.x

Recommended hardware configuration for recording 1,000-2,000 users

Physical or virtual server with the following minimum hardware specification:

- **CPU** - Intel CPU hex-core or better. Frequency at least 2.3GHz.
- **Memory** - 64 GB or more
- **Storage**:
  - Two high speed disks (at least 10,000 rpms HDD or preferably SSD) in RAID 1 configuration for storing operating system, program files and database data. Disk space requirements - at least 1,000 GB.
  - High capacity disk array (local or NAS/SAN) in RAID 5/6 configuration for storing audio mp3 files and, optionally, log files. Disk space requirements - 0.24 MB/minute of recording.

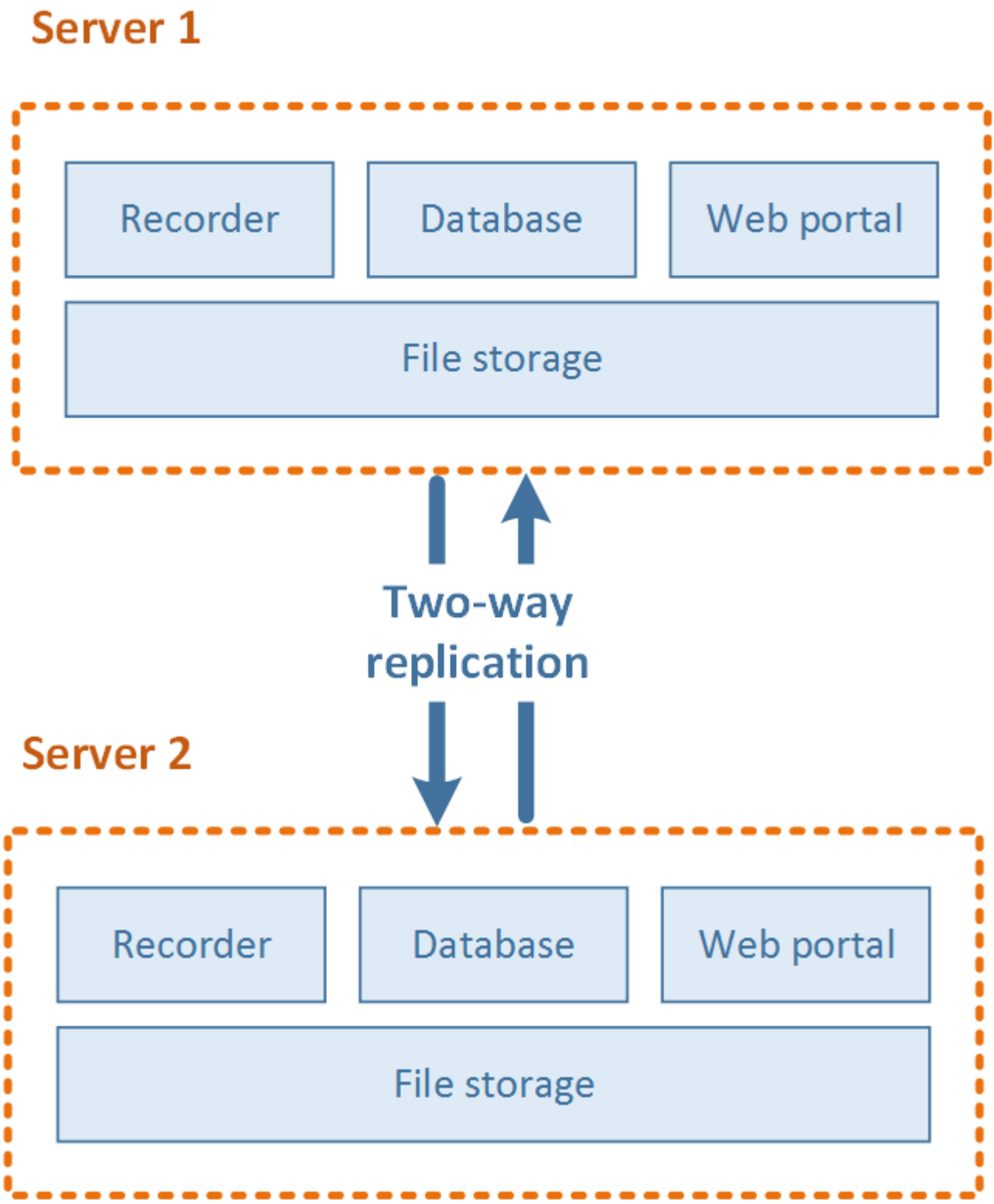
  For example, in average a business user makes 10 calls per day with a duration 5 minutes. This will end up to 1,000 minutes per user per month (assuming 20 working days). One month of storage for 2,000 users will require 480 GB of disk space.
- **OS** - Linux RedHat/Centos 7.x

More than 2,000 users

For larger deployments we recommend to use decoupled architecture (multiple servers).

High availability and redundancy

MiaRec supports High Availability setup using advanced multi-master asynchronous replication between multiple "all-in-one" servers. More details about data replication



2.3 Decoupled architecture

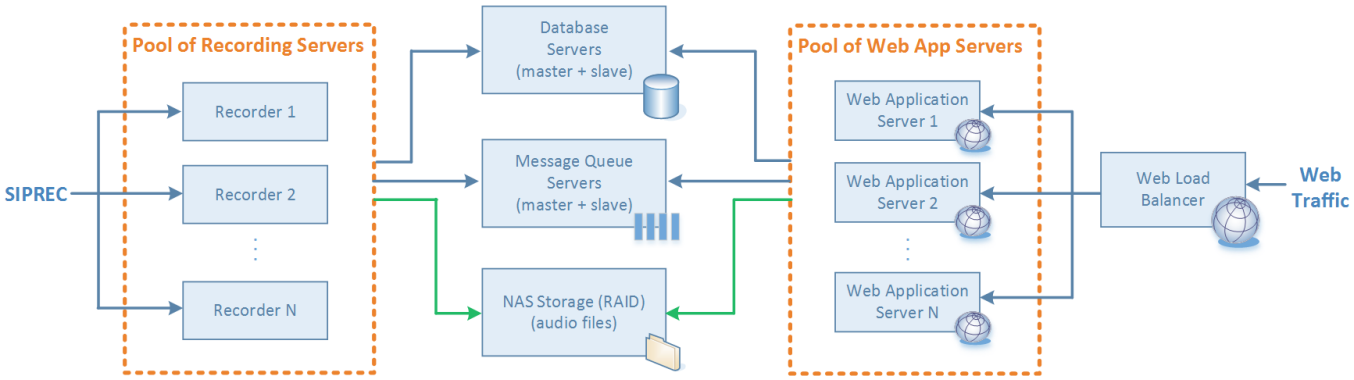
Within MiaRec's decoupled architecture, each software component (recorder engine, database, web portal, storage) is deployed on a dedicated server. As an option, the components may be duplicated to achieve full redundancy and/or scalability.

Decoupled architecture is recommended for recording 2000 or more users.

The following diagram shows the extreme case when at least two copies of each component are deployed on their own dedicated server (master/slave or multi-master) to achieve full redundancy.

Besides such extreme cases, MiaRec supports other configurations with a partial share of hardware resources with some other components. For example, for a small-scale deployment in a hosted environment we recommend you isolate a recording server as the minimum requirement. The rest of the components may share hardware resources on the second server. This two-server setup provides a good balance between security (isolation of a critical recording server) and cost (sharing of hardware resources by other components).

Nowadays a virtualization is a preferred deployment method for new software. In a virtual environment it is significantly cheap to spin up additional VMs and isolate components from each other to achieve reliability, security and scalability.



Such architecture allows you to achieve the following goals:

- **Redundancy**: All components are duplicates to eliminate single-point-of-failure issues. Some of these components support master/master, others support master/slave configuration with a floating ip-address mechanism.
- **Performance**: The software components do not ontend for the same server resources (CPU, Memory, I/O, etc.)
- **Scalability**: Multiple recording and web servers can be deployed for load balancing purposes. Additional server could be easily added to the solution to cover customer growth. MiaRec software architecture provides an almost linear scalability of individual components. For example, if the bottle-neck is a web portal, then you just need to spin up an additional VM with web application.
- **Reliability**: The components are isolated from each other. In a hosted environment, it is important to isolate recording servers from web servers in order to prevent potential disruption of service due to occasional spikes in web traffic. With such architecture, the issues with some of components are not propagated to other components. In the worse case, you may have slowdown of the web portal, but the recording process will not suffer from such issues, and you will not lose any recordings due to CPU/disk/network overload.
- **Security**: In a hosted environment, it is important to keep recording and database servers in a private network isolated from end-user facing web servers. A potential breach of the web server will not spread to other servers.

2.3.1 Hardware specification recommendations

Different components have different requirements to hardware. For example, MiaRec recording server benefits the most from multiple CPU cores and does not benefit at all from additional memory (for example, recording of 500 concurrent sessions consumes less than 1GB of memory, but requires 16-core CPU). The database server benefits the most from SSD disks with a high speed random access. The web portal doesn't benefit from SSD disks, but it benefits from additional memory.

Below you will find recommendations on the hardware specification of each individual component.

Recording server hardware requirements

We recommend one recording server (or virtual machine) for each 500 concurrently recorded session (equivalent to approximately 5,000 users in a Hosted PBX environment). MiaRec recording engine has exceptional performance, and can record 1,000 and more concurrent session on a single server; we recommend you keep an average load of 500 concurrent sessions per server in order to have enough room for potential spikes in load.

When using audio file encryption, the recommendations are one server per 250-300 concurrently recorded session.

SMALL SERVER CONFIGURATION (ABOUT 1,000 USERS PER RECORDER SERVER):

CPU	**4 cores or more. Frequency of at least 2.26GHz.**
Memory	16 GB or more
Storage	• 2 hard disks using RAID 1 for storing OS, binary files and log files. Minimum free disk space is 300GB (for log files). • 2 high speed hard disks (10K or 15K RPM) using RAID 1 for temporary storage of audio files for in-progress calls. Minimum free disk space is 300GB. (*)

LARGE SERVER CONFIGURATION (ABOUT 10,000 USERS PER RECORDER SERVER):

CPU	**12 cores or more. Frequency of at least 2.26GHz.**
Memory	32 GB or more
Storage	• 2 hard disks using RAID 1 for storing OS, binary files and log files. Minimum free disk space is 300GB (for log files). • 2 high speed hard disks (10K or 15K RPM) using RAID 1 for temporary storage of audio files for in-progress calls. Minimum free disk space is 300GB. (*)

(*) - For performance reasons it is recommended that you store audio files for in-progress calls locally on the server. The audio file will be moved to the network attached storage at the end of each call.

In addition to performance reasons, this solution provides another layer of protection to prevent network failures. In case there are network connection issues due to the NAS, the recorder process may continue to record calls, and store audio files locally till the connection to the NAS server is recovered.

Database server requirements

One or two database servers (PostgreSQL) in master/slave configuration using floating ip failover mechanism.

CPU	**2 cores or more. Frequency of at least 2.26GHz.**
Memory	32 GB or more
Storage	Solid state drives (SSDs) using RAID 1 or RAID 10 with free space 3GB for each 1M records stored in database

Web application server requirements

One or more web application servers are required for load balancing and redundancy purposes.

Each of the servers host web portals as well as worker processes for task manager. The task manager is used to execute various maintenance tasks like export, backup, replication, retention, etc. The workers on multiple web application servers form the pool of workers, i.e. the tasks are automatically distributed over multiple application servers for redundancy and load balancing purposes.

The recommended number of web servers depends on anticipated pages/s web requests load.

For a hosted PBX environment a rough estimate is one web server per 5,000 users.

CPU	4 cores or more. Frequency of at least 2.26GHz.
Memory	16 GB or more
Storage	2 hard disks using RAID 1 for storing OS, binary files and log files. Minimum free disk space is 150GB (for log files).

Web load balancer requirements

The web load balancer (HAProxy) is required when two or more web servers are deployed.

The load balancer server itself may be duplicated to eliminate a single point of failure situation. Switchover between load balancing servers is implemented using floating ip mechanism.

CPU	**2 cores. Frequency of at least 3.00GHz.**
Memory	4 GB
Storage	Storage is not critical because HAProxy is mostly CPU consuming process (single thread). 64GB of disk storage for OS, application binary files and logs should be enough.

Message Broker server requirements

One or two servers in master/slave configuration for message queue system. The message queue system is used for internal communication between various components of MiaRec solution.

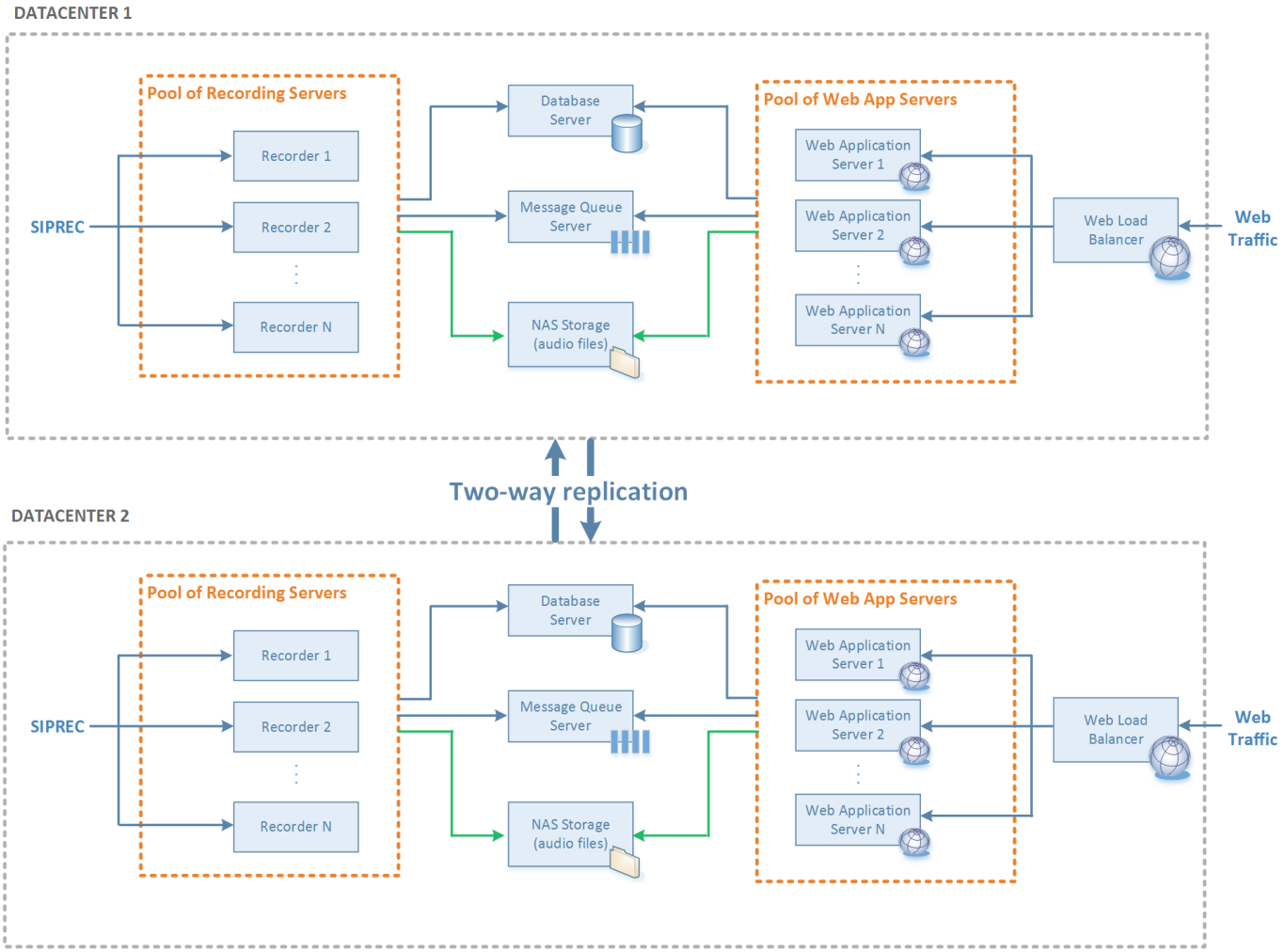
CPU	**2 cores or more. Frequency of at least 2.26GHz.**
Memory	16 GB or more
Storage	• 2 hard disks using RAID 1 for storing OS and binary files (64GB) • 2 high speed hard disks (10K or 15K RPM) using RAID 1 for persistent storage of messages with free space at least 64GB.

2.3.2 Network attached storage (NAS) for audio files

MiaRec stores audio files in compressed MP3 format. Default compression settings are 0.24MB/minute of recording.

2.4 Decoupled with GEO-redundancy

MiaRec supports advanced multi-master asynchronous application-level replication between datacenters. It is quite unique on the market because other vendors mostly support either master/slave or master/master synchronous or SAN-based replication (expensive and not suitable for GEO-redundancy).



2.5 Disk space requirements

MiaRec supports following formats for audio files:

Format	Size per minute	Hours per TB
MP3 (stereo 32kbps) - default	**0.24** MB/minute	**72,818** hours/TB
MP3 (mono 16kbps)	**0.12** MB/minute	**146,636** hours/TB
WAV (stereo)	**1.92** MB/minute	**9,102** hours/TB
WAV (mono)	**0.96** MB/minute	**18,204** hours/TB

2.5.1 Example of disk space requirements calculations

Assumptions:

- In average, a business user makes 10 calls per day with a duration 5 minutes. This results into 1,000 minutes per user per month (assuming 20 working days).
- File format is MP3 stereo 32kbps, i.e. 0.24MB/minute

Approximate disk space requirements (see assumptions):

Total users	30 days storage	1 year storage	3 year storage	7 year storage
50	12 GB	144 GB	432 GB	1,000 GB
100	24 GB	288 GB	864 GB	2,000 GB
200	48 GB	576 GB	1,728 GB	4,000 GB
500	120 GB	1,440 GB	4,320 GB	10,000 GB
1,000	240 GB	2,880 GB	8,640 GB	20,000 GB
2,000	480 GB	5,760 GB	17,280 GB	40,000 GB

2.5.2 Screen recording storage requirements

Screen recording compression is configurable under **Administration -> Screen Recording -> Screen Recording Settings**.

A default bitrate is 256kbps, which is the best balance between video quality and file size.

Bitrate	Size per minute	Hours per TB
256kbps	**1.92** MB/minute	**9,102** hours/TB

3. MiaRec Architecture

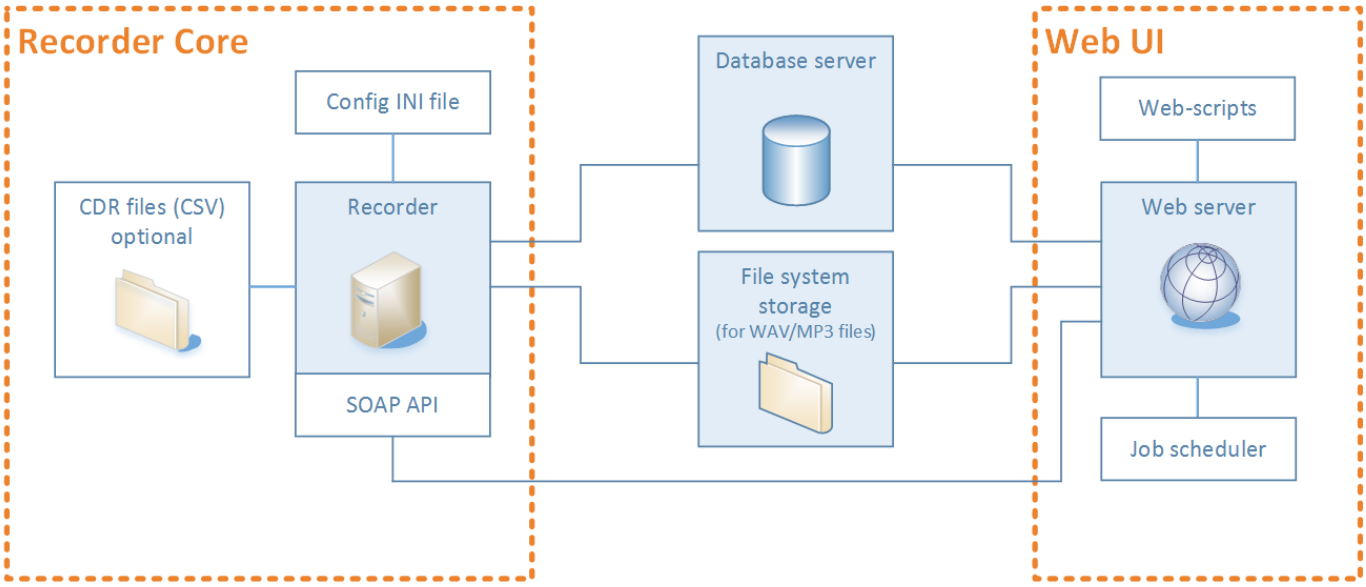
The MiaRec call recording solution consists of multiple components:

- Recorder
- Database
- Web server
- Job scheduler

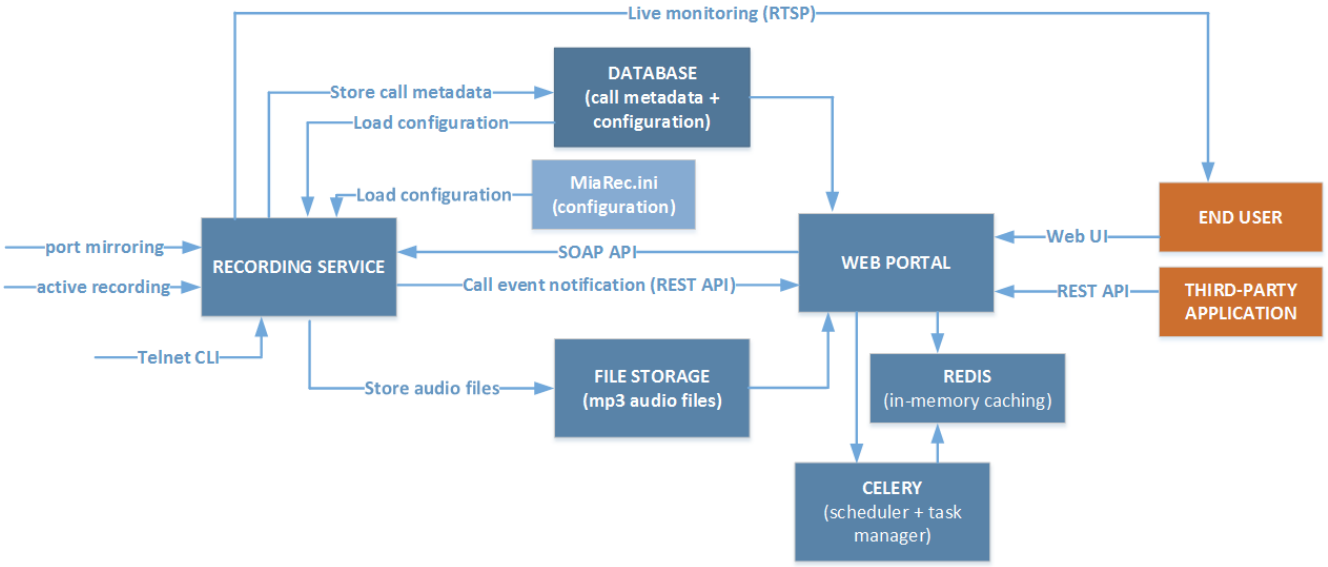
All these components may be deployed on a single server, in case of small size project, or distributed over multiple dedicated servers, optionally, with duplication/redundancy.

Below you can find a simplified as well as detailed diagram of MiaRec architecture.

3.1 Simplified diagram



3.2 Detailed diagram



3.2.1 Recording service

The Recording service is a major component of MiaRec solution.

It captures call recordings from phone system via passive (port mirroring) or active recording method (SIPREC, Cisco BIB, Avaya TSAPI+DMCC etc).

Voice data is stored in WAV/MP3 format on local file system or NAS/SAN.

Call metadata is stored in database and/or textual CSV files (for backup/fail-over purposes).

REST API is used for retrieving of real-time information about call recordings and changing of recorder's behavior, for example, trigger on-demand recording, pause/resume recording, reload configuration etc.

Live monitoring feature is based on RTSP protocol for streaming real-time audio to end user.

Recorder service notifies a web-portal about call events in real-time (call begin/finish) via REST API. Such notification is used to trigger some post-processing tasks like continuous call replication, grouping of multi-segment calls into single interaction etc.

Telnet CLI is used for troubleshooting and monitoring purposes.

Recorder service loads own configuration from MiaRec.ini file at first. From INI file it reads database connection settings (host, port, login etc) and then load configuration from database.

By design the recorder service is independent from other components. It doesn't depend on web-portal component at all. And it continues to record calls even if database is down. In this case, call metadata will be stored in textual CSV files, which may be imported into database when the latter is up again.

3.2.2 File storage

Audio files are stored either locally or on a network-based storage device.

Additionally, MiaRec supports two-phase file storage to improve performance and provide fault-tolerance. When call recordings starts, the recorder creates audio file on local disk array (usually high-speed). When call recording completes, audio file is moved automatically to long-term storage (network-based or high volume but low-speed disk array). If network-based storage is not available, file move operation will fail, but the audio file itself will be successfully stored on local disk array. Such architecture protects from occasional issues with network.

Two-phase file storage architecture is used also to improve performance during active recording phase. When call is in progress, the recorder flushes periodically data to disk by small portions. If there are hundreds of concurrent call recordings, then it causes high IOPS (input-output operations per second) on disk array. In this case, usage of local high-speed disk array is highly recommended. When call completes, its audio file is moved to long-term storage. Such move operation will trigger a single disk write operation per call. IOPS on long-term storage is significantly lower in this case.

3.2.3 Database

Database is used for storing call metadata, recorder configuration as well as web-portal data.

3.2.4 Web portal

Web portal provides access to call recordings to end-users.

Additionally, web portal implements REST API which may be used by third-party applications for accessing call recordings and other resources (like users, groups, roles etc).

3.2.5 Celery scheduler

Celery is a job scheduler and background task manager. It executes such jobs like backup, replication, ldap user synchronization etc.

4. Installation

4.1 Ansible-based installation on Linux

4.1.1 Overview

MiaRec uses Ansible IT automation engine to deploy its software components on Linux system. This guide provides step-by-step instructions for both initial deployment as well as update of MiaRec software.

What is Ansible?

Ansible is an automation tool for provisioning, application deployment, and configuration management.

Ansible uses playbooks written in the YAML language for orchestration. For more information, see Ansible - Intro to Playbooks.

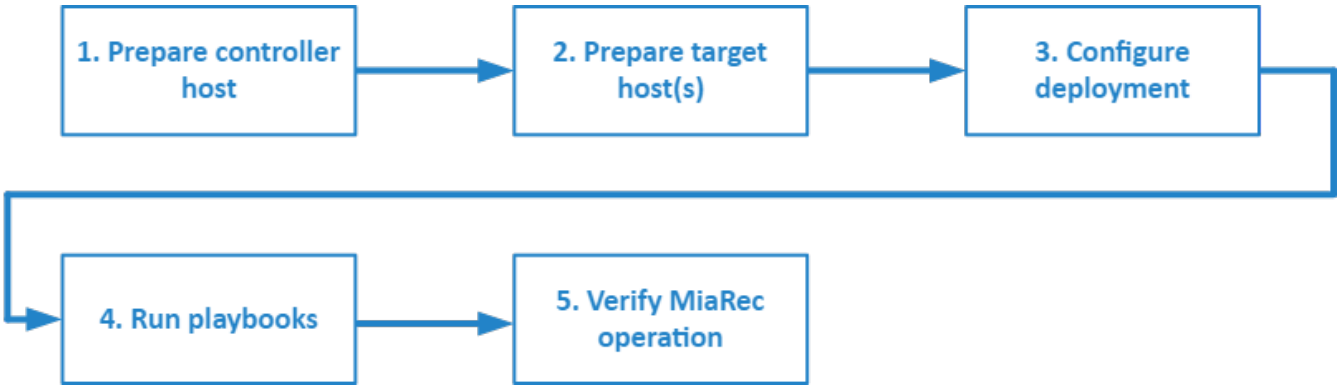
Compared with other server configuration management DevOps tools, Ansible doesn't require agents to be installed on the managed servers. Instead, Ansible manages the IT infrastructure by using SSH protocol to communicate the managed resources. This dramatically simplifies the configuration of managed systems for two reasons—no process daemons need to run on the remote servers to communicate with a central controller and IT administrators aren't required to manage or maintain agents on each managed node.

Ansible can communicate with multiple managed nodes at the same time. This allows to easily deploy various software components, like database, web server, recorder on multiple dedicated servers using a single command.

Comparing to manual installation commands, Ansible allows to build a completely reproducible server configuration. It is a good practice to test Ansible playbooks towards the staging environment and after verification apply the same configuration to the production environment.

Installation workflow

The following diagram shows the general workflow of an MiaRec installation using Ansible.

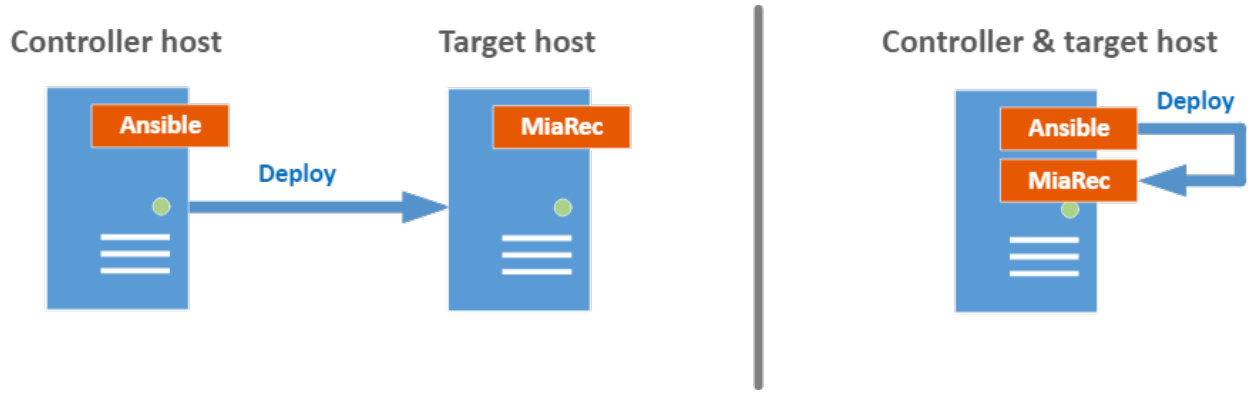


In the next chapters, each of these steps is described in details.

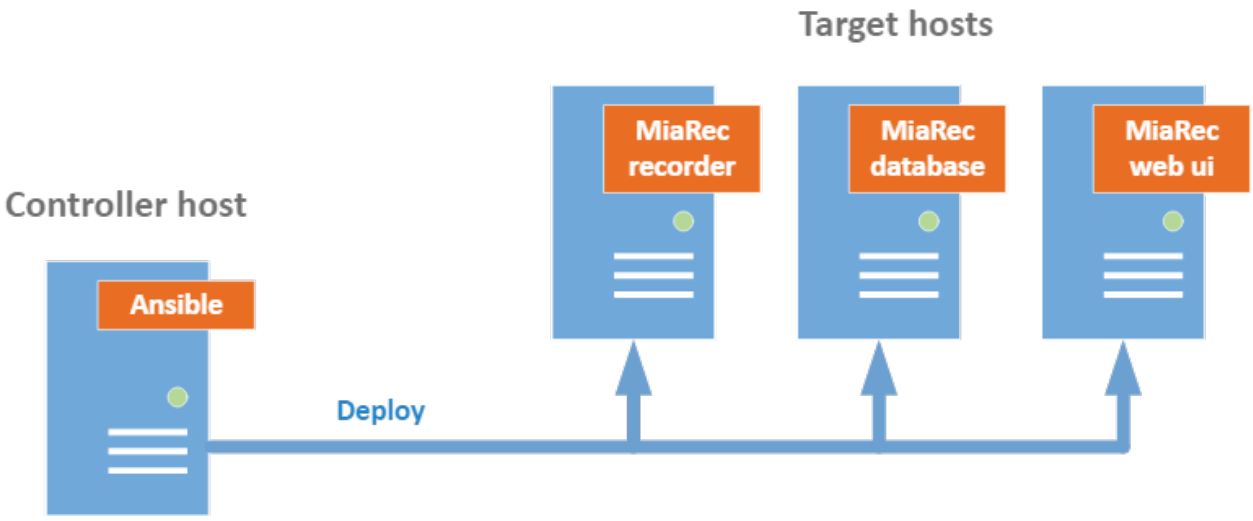
This guide refers to the following types of hosts:

- **Controller host**, which runs the Ansible playbook
- **Target hosts**, where Ansible installs MiaRec software components.

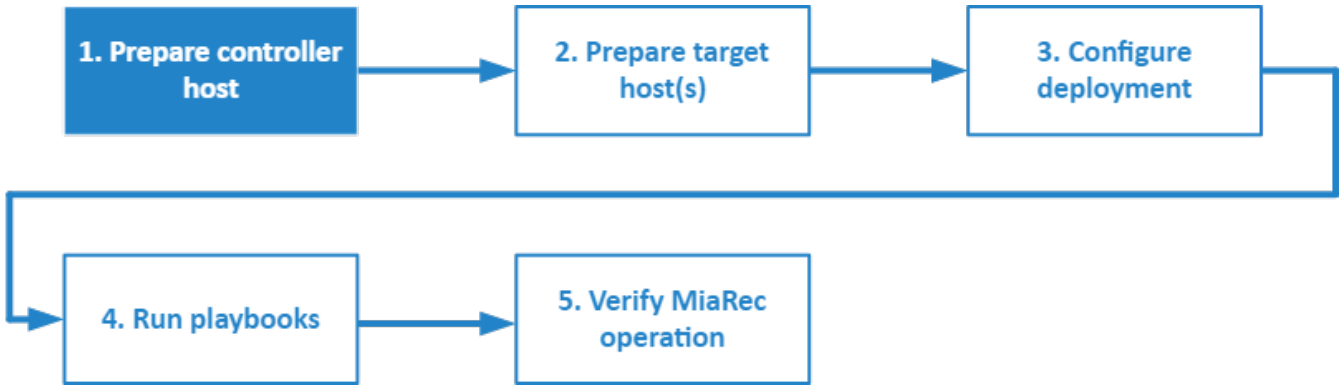
In simple scenarios, like "all-in-one" configuration when all MiaRec software components are deployed on a single host, the same host can be used for both Controller and Target roles, i.e. the Ansible playbook could be run to deploy MiaRec locally. The following diagram demonstrate a difference between these use cases: remote controller and local controller.



In more complex scenarios, like the deployment of MiaRec software components on multiple hosts, the Ansible playbook should be executed from a remote host. The following diagram shows how the remote controller host automatically deploys MiaRec on multiple servers.



4.1.2 1. Prepare controller host



When deploying MiaRec in "all-in-one" configuration on a single server, you can use the same host for both Controller and Target roles. In this case, the Ansible playbook will deploy MiaRec locally.

When deploying MiaRec on multiple servers, it is necessary to use a dedicated host for the Controller role.

Supported operating systems for the Controller host

MiaRec team officially supports the following operating system for the controller host:

- Centos 7 64-bit
- Centos 6 64-bit
- Ubuntu Server 14.04 (Xenial Xerus) LTS 64-bit
- Ubuntu Server 16.04 (Trusty Tahr) LTS 64-bit
- Windows 10 with Bash on Ubuntu *

(*) - The Windows 10 machine could be used solely for the Controller role. If you need to install MiaRec software on Windows operating system, then check the guide Installation on Windows.

It is possible to run Ansible playbook from Mac OSX and other operating systems. The complete list of the supported OSs is available in the official Ansible documentation. The MiaRec team provides technical support for the above mentioned OSs only.

Install Ansible on Ubuntu

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

Update package source lists:

```
sudo apt-get update
```

Upgrade the system packages and kernel:

```
sudo apt-get dist-upgrade
```

Reboot the host.

Install PIP (a tool for installing Python packages. Ansible is written in Python):

```
sudo apt-get install python3-pip
```

Install Ansible using PIP:

```
sudo pip3 install ansible
```

Verify Ansible version:

```
ansible --version
```

The output should be something like:

```
$ ansible --version
ansible 2.3.1.0
  config file =
  configured module search path = Default w/o overrides
  python version = 2.7.12 (default, Nov 19 2016, 06:48:10) [GCC 5.4.0 20160609]
```

Install Ansible on Centos 7

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

Upgrade the system packages and kernel

```
sudo yum upgrade
```

Reboot the host.

Install the Software Collections (SCL) repository. It is required for the latest version of Postgresql (11/12).

On Centos 7:

```
sudo yum install centos-release-scl
```

On RedHat Enterprise:

```
sudo yum-config-manager --enable rhel-server-rhscl-7-rpms
```

Install additional software packages if they were not installed during the operating system installation:

```
sudo yum install git
```

Install EPEL repository:

```
sudo yum install epel-release
```

Install Ansible:

```
sudo yum install ansible
```

Verify Ansible version:

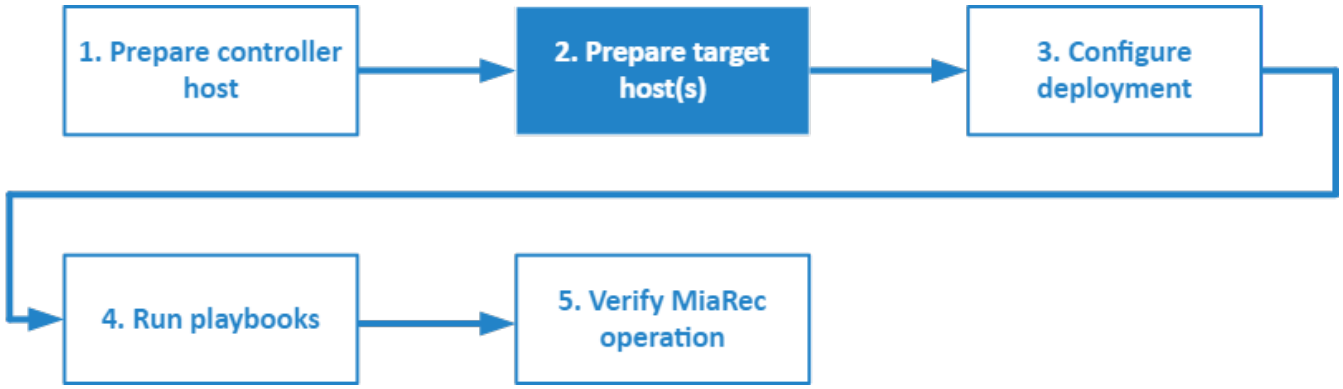
```
ansible --version
```

Install the MiaRec ansible scripts

Clone the latest stable release of the MiaRec-Ansible Git repository in the /opt/ansible-miarec directory:

```
git clone --recursive https://github.com/miarec/ansible-miarec /opt/ansible-miarec
```

4.1.3 2. Prepare target hosts



This section describes the installation and configuration of operating systems for the target host(s).

MiaRec team officially supports the following operating system for the controller host:

- Centos/RedHat 6 64-bit
- Centos/RedHat 7 64-bit
- Ubuntu Server 14.04 LTS 64-bit
- Ubuntu Server 16.04 LTS 64-bit

2.1. Configure the operating system (Ubuntu)

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

Update package source lists:

```
sudo apt-get update
```

Upgrade the system packages and kernel:

```
sudo apt-get dist-upgrade
```

Reboot the host to use the new kernel.

Install additional software packages if they were not installed during the operating system installation:

```
sudo apt-get install aptitude ntp ntpdate openssh-server acl python
```

Configure Network Time Protocol (NTP) in /etc/ntp.conf to synchronize with a suitable time source and restart the service:

```
service ntp restart
```

2.2. Configure the operating system (RedHat/Centos)

Install additional software packages and configure Network Time Protocol (NTP). Before you begin, we recommend upgrading your system packages and kernel.

Upgrade the system packages and kernel

```
sudo yum upgrade
```

Reboot the host to use the new kernel.

Install the Software Collections (SCL) repository. It is required for the latest version of PostgreSQL (11/12).

On Centos 7:

```
sudo yum install centos-release-scl
```

On RedHat Enterprise:

```
sudo yum-config-manager --enable rhel-server-rhscl-7-rpms
```

Install additional software packages if they were not installed during the operating system installation:

```
sudo yum install ntp ntpdate openssh-server
```

Configure Network Time Protocol (NTP) in /etc/ntp.conf to synchronize with a suitable time source and start the service:

```
# On RedHat/Centos 7 (SystemD)
sudo systemctl enable ntpd.service
sudo systemctl start ntpd.service
```

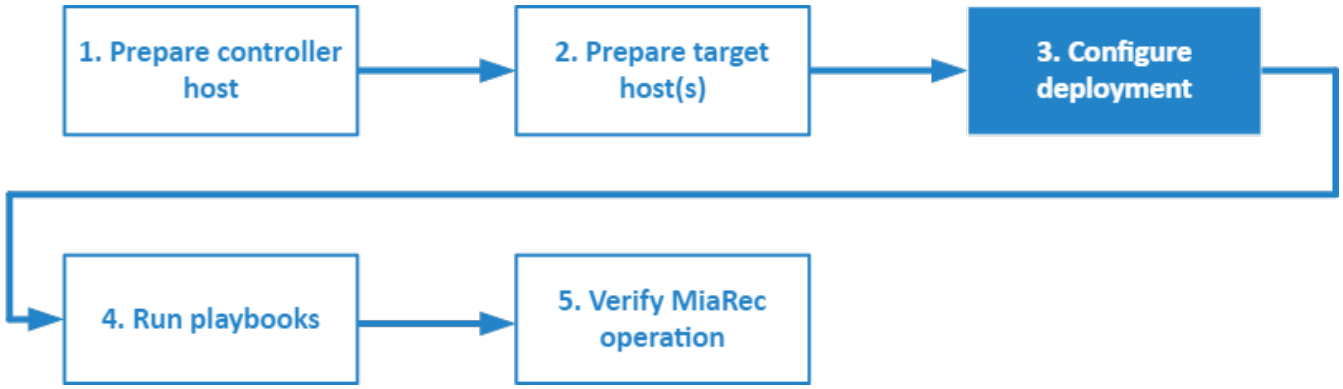
2.3. Configured password-less access (optional, recommended)

Skip this step if the same host is used as a controller and target host.

Use the following instructions to setup password-less access from the Ansible controller to the target hosts.

- How to set up ssh keys

4.1.4 3. Configure deployment



This section describes the configuration of MiaRec deployment. Such configuration should be done on the Ansible controller host.

3.1. Create inventory file (hosts)

The Ansible inventory file is an INI-formatted file that defines the hosts and groups of hosts upon which commands, modules, and tasks in playbooks operate.

Create the file `/opt/ansible-miarec/hosts` and add entries for every server you want to manage with Ansible (the Inventory File is highly configurable, see the Ansible documentation for more information):

```
vim /opt/ansible-miarec/hosts
```

EXAMPLE 1. LOCAL INSTALLATION (ALL-IN-ONE):

For local installation (when Ansible is running on the same host as MiaRec software), create the following `hosts` file:

```
[all]
; --------------------------------
; All-in-one host
; Parameters:
;   - private_ip_address => ip address to access the host from other components (for example, web application needs to connecto to database)
; --------------------------------

miarec ansible_connection=local private_ip_address=127.0.0.1


[all:vars]
; ------------------------------
; Version of installed packages
; ------------------------------
miarecweb_version   = x.x.x.x
miarec_version      = x.x.x.x
miarec_screen_version = x.x.x.x
miarec_livemon_version = x.x.x.x
postgresql_version  = 12
python_version      = 3.8.18
redis_version       = 5.0.10

install_pgbouncer = true

; Use command `pwgen -s 32 1` to generate random password
secret_db_password = <GENERATE-RANDOM-PASSWORD-HERE>

; Use command `pwgen -s 32 1` to generate random password
miarecweb_secret = <GENERATE-RANDOM-PASSWORD-HERE>


[recorder]
miarec

[screen]
miarec

[db]
miarec

[redis]
miarec

[web]
```

```
miarec

[celery]
miarec

[celerybeat]
miarec

[livemon]
miarec
```

EXAMPLE 2. REMOTE INSTALLATION VIA SSH (ALL-IN-ONE):

If you are running Ansible playbook from the Controller host over SSH, create the following `hosts` file (replace 1.2.3.4 ip-address with the target host address):

```
[all]
; --------------------------------
; All-in-one host
; Parameters:
;   - ansible_ssh_host => ip address to access the host using Ansible
;   - ansible_root     => root account to login to server. Usually, 'root', but for Ubuntu it may be 'ubuntu'
;   - private_ip_address => ip address to access the host from other components (for example, web application needs to connecto to database)
;                           For 'all-in-one' setup, the private_ip_address should be set to '127.0.0.1' as all communication is done internally
; --------------------------------

miarec ansible_host=1.2.3.4 ansible_port=22 ansible_user=root private_ip_address=127.0.0.1


[all:vars]
; ------------------------------
; Version of installed packages
; ------------------------------
miarecweb_version   = x.x.x.x
miarec_version      = x.x.x.x
miarec_screen_version = x.x.x.x
miarec_livemon_version = x.x.x.x
postgresql_version  = 12
python_version      = 3.8.18
redis_version       = 5.0.10

install_pgbouncer = true

; Use command `pwgen -s 32 1` to generate random password
secret_db_password = <GENERATE-RANDOM-PASSWORD-HERE>

; Use command `pwgen -s 32 1` to generate random password
miarecweb_secret = <GENERATE-RANDOM-PASSWORD-HERE>

[recorder]
miarec

[screen]
miarec

[db]
miarec

[redis]
miarec

[web]
miarec

[celery]
miarec

[celerybeat]
miarec

[livemon]
miarec
```

EXAMPLE 3. REMOTE INSTALLATION VIA SSH TO MULTIPLE HOSTS (DECOUPLED ARCHITECTURE):

If you deploy MiaRec components on dedicated hosts, create the following `hosts` file (replace ip-adresses accordingly):

```
[all]

; --------------------------------
; All hosts
; Parameters:
;   - ansible_ssh_host => ip address to access the host using Ansible
;   - ansible_root     => root account to login to server. Usually, 'root', but for Ubuntu it may be 'ubuntu'
;   - private_ip_address => ip address to access the host from other components (for example, web application needs to connecto to database)
;                           For 'all-in-one' setup, the private_ip_address should be set to '127.0.0.1' as all communication is done internally
; --------------------------------

rec1.miarec  ansible_ssh_host=192.168.88.11  private_ip_address=192.168.88.11  ansible_user=root
```

```
rec2.miarec  ansible_ssh_host=192.168.88.12  private_ip_address=192.168.88.12  ansible_user=root
db.miarec    ansible_ssh_host=192.168.88.15  private_ip_address=192.168.88.15  ansible_user=root
redis.miarec ansible_ssh_host=192.168.88.16  private_ip_address=192.168.88.16  ansible_user=root
web1.miarec  ansible_ssh_host=192.168.88.21  private_ip_address=192.168.88.21  ansible_user=root
web2.miarec  ansible_ssh_host=192.168.88.22  private_ip_address=192.168.88.22  ansible_user=root


[all:vars]
; ------------------------------
; Version of installed packages
; ------------------------------
miarecweb_version   = x.x.x.x
miarec_version      = x.x.x.x
miarec_screen_version = x.x.x.x
miarec_livemon_version = x.x.x.x
postgresql_version  = 12
python_version      = 3.8.18
redis_version       = 5.0.10

install_pgbouncer = true

; Use command `pwgen -s 32 1` to generate random password
secret_db_password = <GENERATE-RANDOM-PASSWORD-HERE>

; Use command `pwgen -s 32 1` to generate random password
miarecweb_secret = <GENERATE-RANDOM-PASSWORD-HERE>

[recorder]
rec1.miarec
rec2.miarec

[screen]
rec1.miarec
rec2.miarec

[db]
db.miarec

[redis]
redis.miarec

[web]
web1.miarec
web2.miarec

[celery]
web1.miarec
web2.miarec

[celerybeat]
web1.miarec

[livemon]
web1.miarec
web2.miarec
```

In this example, we define two remote machines `miarec1` and `miarec2` and then place them into group `miarec`. Ansible playbook is executed against whole group.

3.2 Edit the version info in the inventory file

The `hosts` file contains the version of to be installed packages.

You need to edit at least the following parameters:

- miarecweb_version
- miarec_version
- miarec_screen_version
- miarec_livemon_version

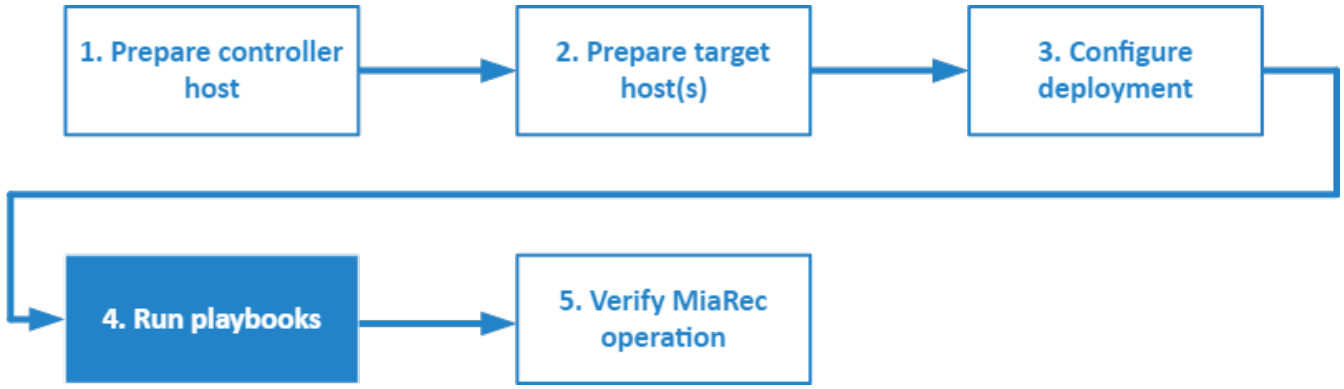
To get the latest MiaRec version, contact your MiaRec representative.

Example:

```
[all:vars]
; ------------------------------
; Version of installed packages
; ------------------------------
miarecweb_version   = 1.1.1.1
miarec_version      = 2.2.2.2
miarec_screen_version = 3.3.3.3
miarec_livemon_version = 4.4.4.4
postgresql_version  = 12
```

```
python_version      = 3.8.18
redis_version       = 5.0.10
```

4.1.5 4. Run playbooks



MiaRec playbooks

MiaRec installation process is split on three playbooks:

1. The `prepare-hosts.yml` Ansible foundation playbook installs the infrastructure services (PostgreSQL database, Redis, Apache web server, Python) and configures firewall (iptables). You need to run this playbook only once.
2. The `configure-firewall.yml` playbook configures **iptables** firewall on target host(s). It is optional playbook, but for security reasons, it is recommended to run it. Alternatively, the firewall can be configured manually. You need to run this playbook when firewall rules change.
3. The `setup-miarec.yml` playbook installs the MiaRec services, including web portal (miarecweb), recorder (miarec) and screen recording contoller (miarec_screen). Run this playbook for initial installation as well as for subsequent updates.

4.1. Run prepare-hosts.yml playbook to provision the server(s)

The playbook prepare-hosts.yml will install the required packages, like PostgreSQL database, Apache web server, Redis, Python, opens appropriate ports in firewall, etc. Normally you need to run this playbook only once when you prepare the system for MiaRec installation.

In case of remote installation, it is necessary to establish trust relationships between the controller and target machines. When speaking with remote machines, Ansible by default assumes you are using SSH keys. SSH keys are encouraged but password authentication can also be used where needed by supplying the option `--ask-pass`. You need to supply also the option `--ask-sudo-pass` if you are connecting to the remote server as non-root user.

When using password-less authentication (or when running Ansible locally on target host), you can simply run the following command:

```
cd /opt/ansible-miarec
ansible-playbook -i hosts prepare-hosts.yml
```

When using password authentication, you can run the following command and you will prompted to enter the password for SSH connection:

```
ansible-playbook -i hosts prepare-hosts.yml --ask-pass
```

Confirm satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP *********************************************************************
...
miarec               :  ok=79   changed=42   unreachable=0    failed=0
```

4.2. Run configure-firewall.yml playbook to enable iptables on the server(s)

> **CAUTION!** MiaRec installer uses **iptables** as a default firewall. It will be enabled automatically on the target system and the other firewall software, if any, will be disabled. For example, on Centos 7, **firewalld** will be disabled. On Unbuntu 16.04, **ufw** will be disabled.

Alternatively, you can skip this step and configure firewall for MiaRec manually.

Run playbook:

```
ansible-playbook -i hosts configure-firewall.yml
```

4.3. Run setup-miarec.yml playbook to install or update MiaRec software

The playbook setup-miarec.yml will install the MiaRec software components (recorder, web portal, etc.). You need to run this playbook during initial installation as well as during upgrade of MiaRec to the new version.

To install/update MiaRec, run the following command:

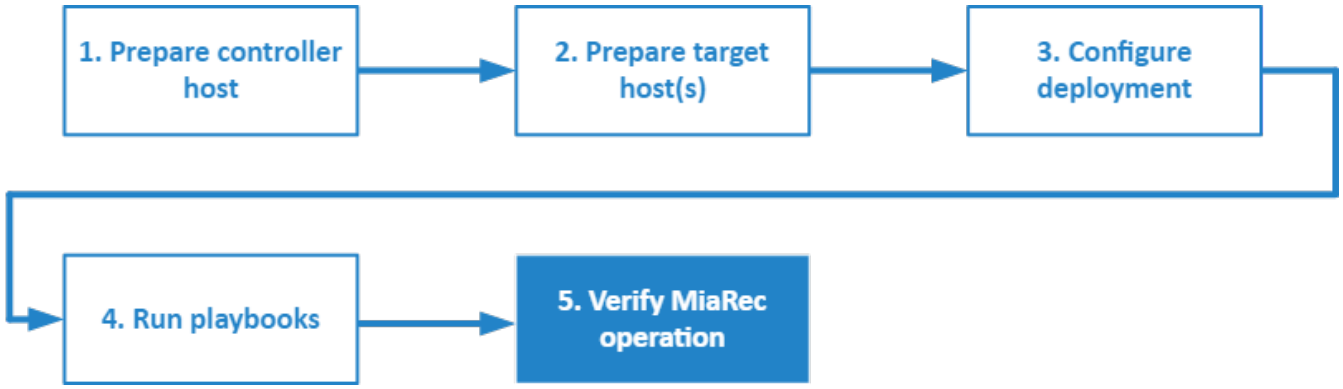
```
ansible-playbook -i hosts setup-miarec.yml
```

Confirm satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP *********************************************************************
...
miarec               :  ok=38   changed=25   unreachable=0    failed=0
```

\

4.1.6 5. Verify MiaRec operation



Use web browser to access MiaRec web portal. Navigate to **Administration -> Maintenance -> System Log** to check the errors.

Configure appropriate recording interface in **Administration -> System -> Recording Interfaces** and make a few test calls. Verify that calls are recorded.

It is recommended to reboot the target machine and verify all services are up and running after system reboot.

```
shutdown -r now
```

- **PostgreSQL database**:

  ```
  service postgresql-9.5 status
  ```

  * **Redis cache** (use ping command. It should print PONG if success):

  ```
  /opt/redis/bin/redis-cli ping
  ```

  * **Apache web server**

  ```
  service httpd status
  ```

  * **Celery task manager**

  Centos 6 (init.d):

  ```
  service celeryd status
  ```

  Centos 7 (SystemD):

  ```
  systemctl status celeryd
  ```

  * **Celery beat scheduler**

  ```
  service celerybeat status
  ```

  * **MiaRec recorder**

  Centos 6 (Upstart):

  ```
  initctl status miarec
  ```

  Centos 7 (SystemD):

  ```
  systemctl status miarec
  ```

  * **MiaRec scree recorder**

  Centos 6 (Upstart):

  ```
  initctl status miarec_screen
  ```

  Centos 7 (SystemD):

  ```
  systemctl status miarec_screen
  ```

4.2 VMWare OVA template-based installation

MiaRec OVA template is a pre-installed virtual machine with operating system (Centos 7.3 64-bit) and MiaRec call recording software.

This OVF template is ideal solution for trial. It is easy to import it into VMWare ESX/ESXi or VMWare Workstation.

4.2.1 1. Request URL to OVA template

To request URL to OVA template, please contact your MiaRec representative.

You need this URL for the next step.

4.2.2 2. Installation instructions for VMWare ESXi

Login to ESXi via VMWare vCenter Client.

From the menu select **File -> Deploy OVF Template...**

Inside the opened dialog enter the URL to OVA file, which you received from the MiaRec representative.

Choose a name for this VM, select the desirable disk format (we commend "Thin provisioning" for trial and "Thick provisioning" for production deployment) and click "Finish" to start deploying of the virtual machine.

4.2.3 3. Console/ssh access to the MiaRec server

Login to console as a **root** user. A default password is **miarec**

It is highly recommended to change a root password as soon as possible. Login to the system as a root and type the following command to change own password:

```
passwd
```

4.2.4 4. Determining the MiaRec ip-address (when using DHCP)

Login to MiaRec virtual machine as root and execute:

```
ip a
```

It will show the ip-address assigned to the first network interface. User that ip-address to access MiaRec web-portal or SSH.

4.2.5 5. MiaRec web portal access

You need to know the ip-address of virtual machine (see the above step).

Type in the web-browser the URL `http://VM-IP-ADDRESS`

You will be asked to create the admin account if you access the web portal the first time.

4.2.6 6. Trial license

You need to request 30-day trial license for MiaRec call recorder. Open MiaRec web-portal and go to menu **Administration -> Maintenance -> Recording License**. Click on "Edit license" link. You will see "Computer Id" value. Send it to support@miarec.com to receive a trial license code.

4.2.7 7. Multi-tenant mode [optional]

Navigate to menu **Administration -> Customization -> Multitenancy** to enable multi-tenancy in MiaRec.

4.2.8 8. Configure network

By default, the MiaRec OVA template is configured with DHCP. For testing purposes, you can keep using DHCP-based network configuration. For a production environment, we recommend to configure static ip-address as described below.

Assign static ip-address (if not using DHCP)

By default MiaRec virtual machine is configured to obtain own ip-address from DHCP server. If you would like to use static ip-address for MiaRec VM, then edit file `/etc/sysconfig/network-scripts/ifcfg-ens33`

```
vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

Change `BOOTPROTO=dhcp` to `BOOTPROTO=none` and add the following lines to this file:

```
IPADDR=x.x.x.x
GATEWAY=y.y.y.y
DNS1=8.8.8.8
DNS2=8.8.4.4
```

Replace x.x.x.x and y.y.y.y with desired machine ip-address and gateway.

Restart network interface:

```
service network restart
```

Test the network connection:

```
ping 8.8.8.8
```

Update /etc/hostname and /etc/hosts

If you would like to access MiaRec web-portal via dns name rarther than ip-address, then edit files `/etc/hostname` and `/etc/hosts`.

Example of `/etc/sysconfig/network`

```
HOSTNAME=miarec.example.com
```

Example of `/etc/hosts`

```
127.0.0.1             miarec.example.com localhost
```

Restart network:

```
service network restart
```

Restart MiaRec recorder service

If the network configuration is updated, then you need to restart miarec recorder service:

```
service miarec restart
```

4.3 Deploying MiaRec on Amazon AWS (up to 2,000 users)

4.3.1 1. Network architecture

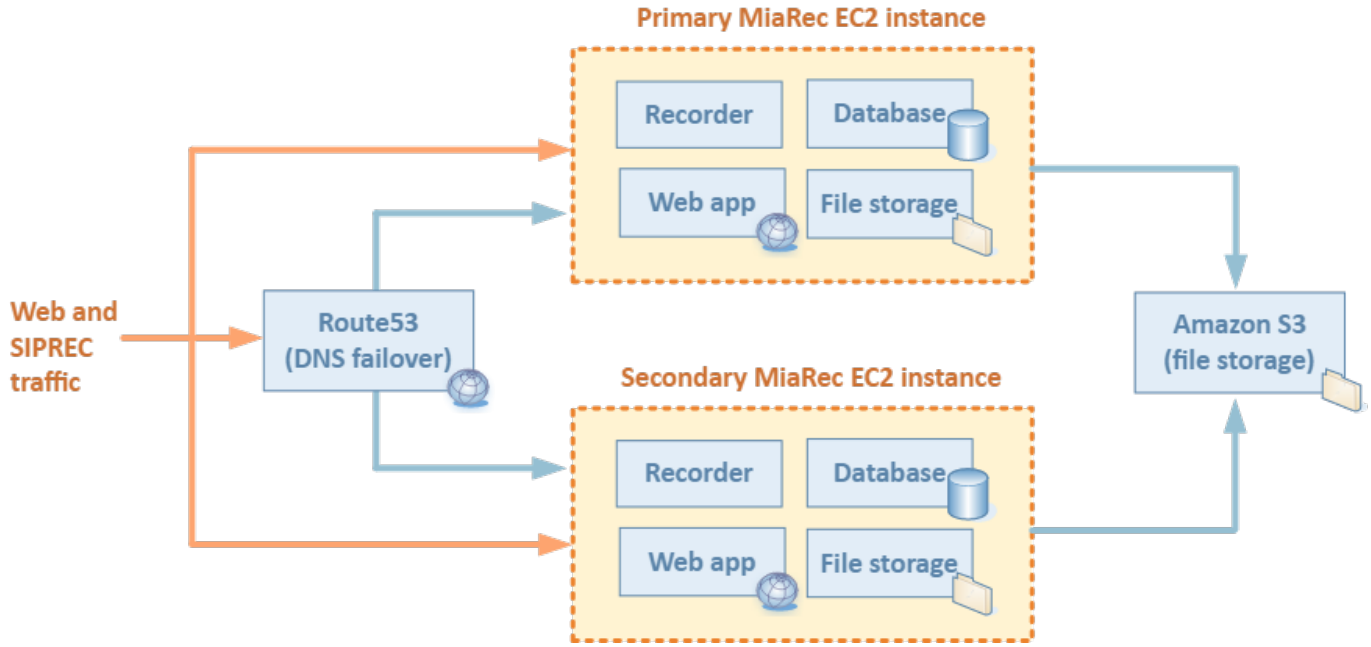
This guide provides step-by-step instructions for deployment of MiaRec call recording software on Amazon AWS.

MiaRec on AWS can leverage many of the services that are designed with High Availability. Depending on anticipated capacity, we recommend two types of HA architecture:

- **Basic HA** with two EC2 instances in all-in-one configuration (*). Recommended for up to 2,000 subscribers.
- **Advanced HA** with dedicated EC2 instances for each of components. Recommended for more than 2,000 subscribers.

(*) All-in-one configuration means recorder, database, web application and other components are hosted on the same virtual server.

This guide describes the Basic HA setup for recording of up to 2,000 subscribers. The recommended network architecture for MiaRec on AWS is shown in the following diagram.



Components:

- Two **EC2 instances** (Virtual Servers) hosting MiaRec software in all-in-one configuration.
- **S3** (Cloud Storage) for long term storage of recorded audio/video files.
- **Route 53** service (Managed Cloud DNS) for DNS failover

High Availability characteristics:

- **Server Redundancy**. In the proposed architecture, we have two independent MiaRec servers (EC2 instances). These instances are configured in Hot Standby mode. When the primary server fails, the secondary server is switched into operation. The instances are deployed in different Availability Zones (covered in this guide) or Regions.
- **Data Redundancy**. Configuration and call metadata is stored in database. The recorded files are stored in file system. Database data is replicated asynchronously between two MiaRec instances. The audio/video files are uploaded to Amazon S3 storage, which provides replication and redundancy.

- **Auto Failover mechanism**. Amazon Route 53 service monitors the health and performance of MiaRec instances. Using DNS failover, it can route the web traffic from an unhealthy instance to a healthy one. For SIPREC traffic, we recommend to use DNS SRV-based failover if the phone platform supports it (DNS SRV-based failover will minimize downtime). If DNS SRV is not supported by phone platform, then it is possible to use Router 53 DNS failover mechanism for SIPREC traffic.

Frequently asked questions:

Why not use DNS round-robin for web traffic load balancing?

Round Robin DNS works by responding to DNS requests with a list of potential IP addresses corresponding to several web servers that host identical resources. The DNS server returns a list of IP addresses in random order.

DNS round robin works the best for load balancing. Failover scenarios are not recommended due to how browsers handle multiple addresses. Modern browsers will choose one of IP addresses and if it cannot connect, the browser will try the next server. the process is user-transparent, and occurs only if the first server tried times out, and only for the first page requested from our site in any browser session. If one of the servers is down for long time, then 50% of users will receive its IP address the first in a list, they will experience high load time (about 3 minutes) for the first page. Older browsers (like IE7) do not support switching to the second IP address, i.e. 50% of users on old browsers will not be able to access the web server (many web-sites today do not support old web browsers, so, this fact probably can be ignored).

Another drawback of DNS round robin technique comes from how MiaRec replicates data between instances. DNS round robin works the best with stateless servers. In this guide, we build stateful MiaRec server, which includes database in the same machine. Data between machines is replicated asynchronously. Asynchronous replication has a lot of advantages comparing to synchronous replication, but there is the expected delay in data synchronization. For example, MiaRec synchronizes only the completed calls. The in-progress calls will be visible on of servers only. If using DNS round robin, then 50% of users will not be able to see in-progress calls.

DNS SRV vs DNS failover for SIPREC traffic

If the phone platform supports DNS SRV for SIPREC, then it should be used as a preferred method. With DNS SRV, the phone platform receives a list of IP-addresses with priorities. The IP-addresses will be tried in particular order depending on record priority/weight. If one of servers does not respond, then the second one will be tried automatically.

The phone platform determines that server is not available based on its SIP response. DNS SRV-based failover time is shorter than DNS failover. The Amazon Route 53 DNS failover uses health checks, which are good for web servers, but they cannot check health of SIP devices.

Depending of implementation, the phone platform can preemptively monitor each of IP-addresses using SIP keep alive mechanism (usually, send SIP OPTIONS message) and make failover time even shorter.

Why upload audio/video files to S3 instead of storing them locally on EC2 instance?

Short answer: costs and reliability.

According to official documentation "Amazon EBS volumes are designed for an annual failure rate (AFR) of between 0.1% - 0.2%, where failure refers to a complete or partial loss of the volume, depending on the size and performance of the volume." Amazon S3 is designed to deliver 99.999999999% durability (documentation).

Amazon EBS volume data is replicated across multiple servers in an Availability Zone. With Amazon S3 data is automatically distributed across a minimum of three physical facilities that are geographically separated by at least 10 kilometers within an AWS Region, and Amazon S3 can also automatically replicate data to any other AWS Region.

EBS costs are $0.10 per GB/month (US East Region, General Purpose SSD). S3 storage costs are $0.023 per GB/month (US East Region, first 50TB).

How to scale this architecture?

The architecture described supports vertical scaling, i.e. the EC2 instance can resized to larger CPU/memory values. You must stop your Amazon EBS–backed instance before you can change its instance type.

How big is a delay in synchronization between servers?

Data is replicated asynchronously. The replication process can be configured to start either by schedule or continuously. In the latter case, call recording metadata and file will be replicated as soon as call completes. Other configuration data, like users/groups/roles will be replicated as soon as the next recording is replicated or up to 1 minute after a change, if call traffic is low.

What happens if the primary server is down?

This is probably the most important question in this guide.

When the primary server is down, then failover mechanism is initiated. Amazon Route 53 service monitors the health and performance of MiaRec instances. If it detects that one the primary server is unhealthy, then DNS records are updated to point to the secondary server. The web traffic will be automatically loaded to the secondary server.

Failover for SIPREC traffic is managed independently of web traffic failover. The phone platform reds a list of IP addresses of recording servers from DNS SVR record. It automatically route SIPREC traffic to the secondary server if the primary server doesn't respond to SIP requests.

If the server is down completely, then both SIPREC and web traffic will be automatically routed to the second instance. This scenario is straight forward. But there is a potential situation when only one of software components experiences issues. For example, web server is not responding on the primary server, but recorder service is fully functioning. In this case, the failover occurs only for web traffic. Such loosely coupled architecture (independence in web and siprec failover events) has some pros and cons. Good thing in MiaRec architecture is it allows the recorder service to continue recording of calls even if other components, like database and web server are completely down. This guarantees that the most critical part of call recording platform is as robust as possible. If, for example, both servers experience problems with the web component, the recording process is not affected at all. In order to detect such enormous situations with partial failure, it is necessary to utilize monitoring tools like CloudWatch, Zabbix or similar.

How much data may be lost in case of primary server is down?

Data between servers is replicated asynchronously by schedule or continuously. In the later case, call metadata will be uploaded to other server as soon as call completes.

If there were in-progress call recordings on the primary server before it died, then such calls will be recorded only partially. If disk storage of the primary server is recovered later, then it is possible to recover the first portion of media for such in-progress calls. The replication process will be restored automatically after the server is alive again.

If the disk storage of the primary server is unrecoverable, then data for in-progress recordings as well as data of not-uploaded yet recordings is lost.

4.3.2 2. Create VPC

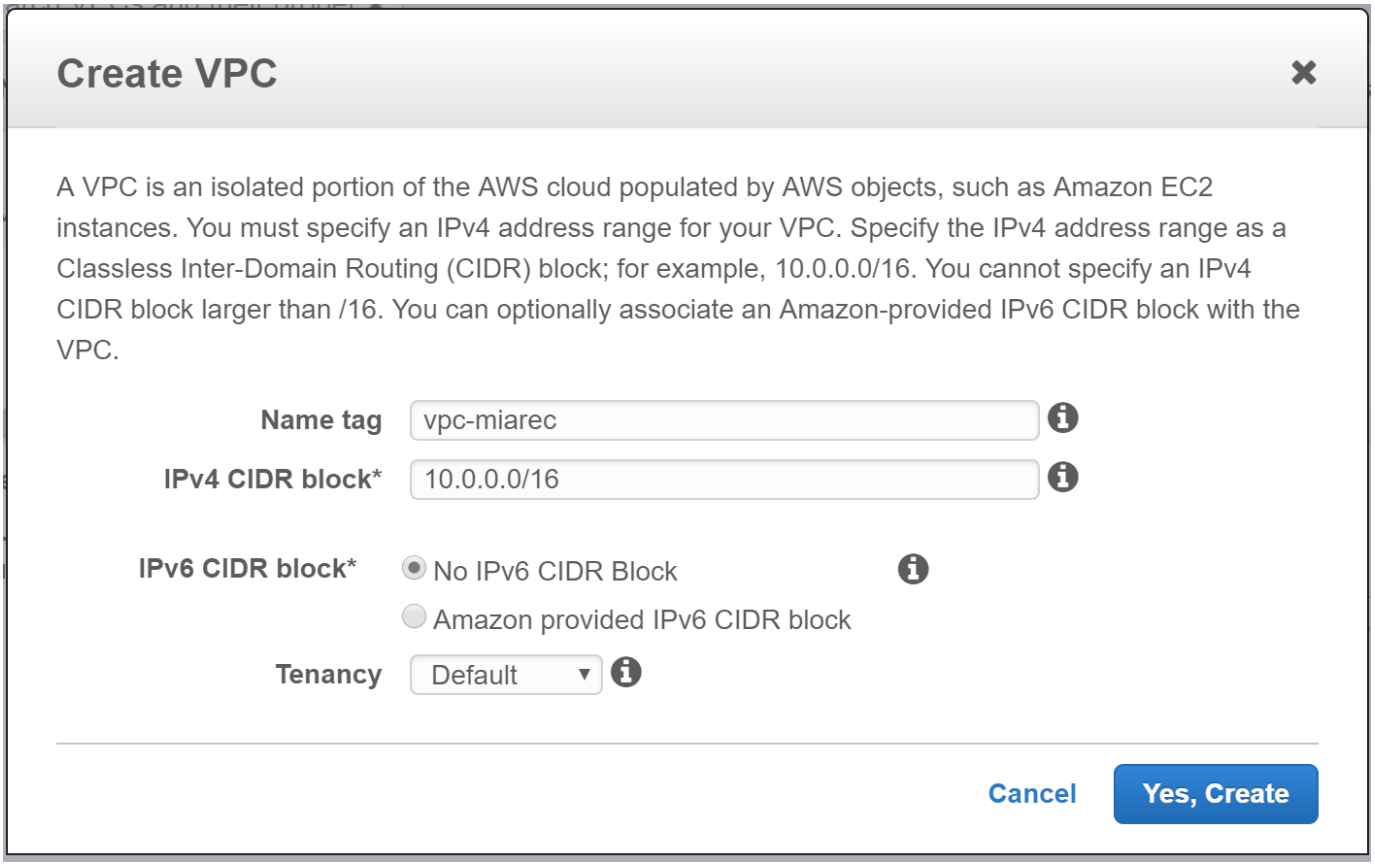
A virtual private cloud (VPC) is a virtual network that closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of Amazon Web Services (AWS). You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

In this guide, we will create a dedicated VPC for MiaRec cluster.

Create VPC

To create a VPC:

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the dashboard, choose **Your VPC** and click **Create VPC** button.
3. Choose the name which will help you to identify it later in the console.
4. We use `10.0.0.0/16` for the **CIDR block** and leave tenancy as default if we don't require dedicated hardware. For more information about IPv4 and IPv6 addressing, see IP Addressing in Your VPC
5. Click **Yes, Create**.

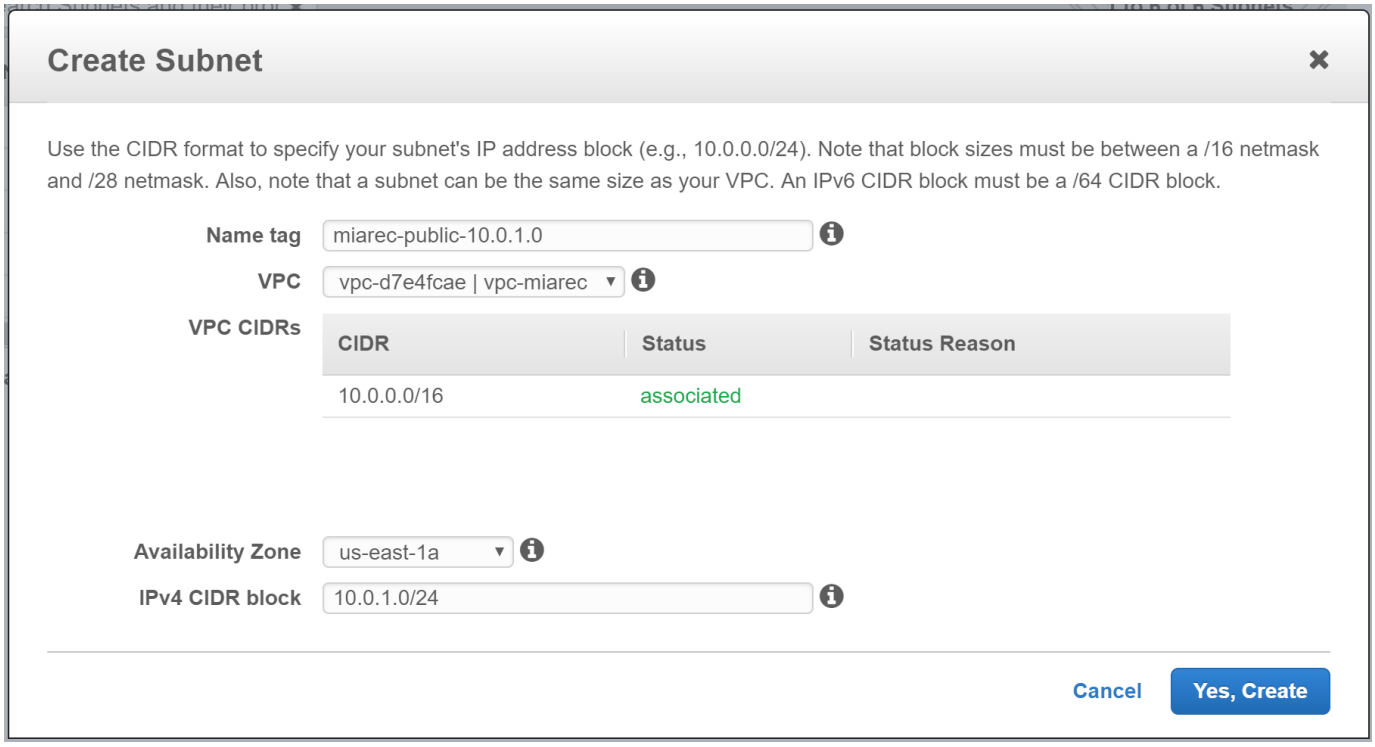


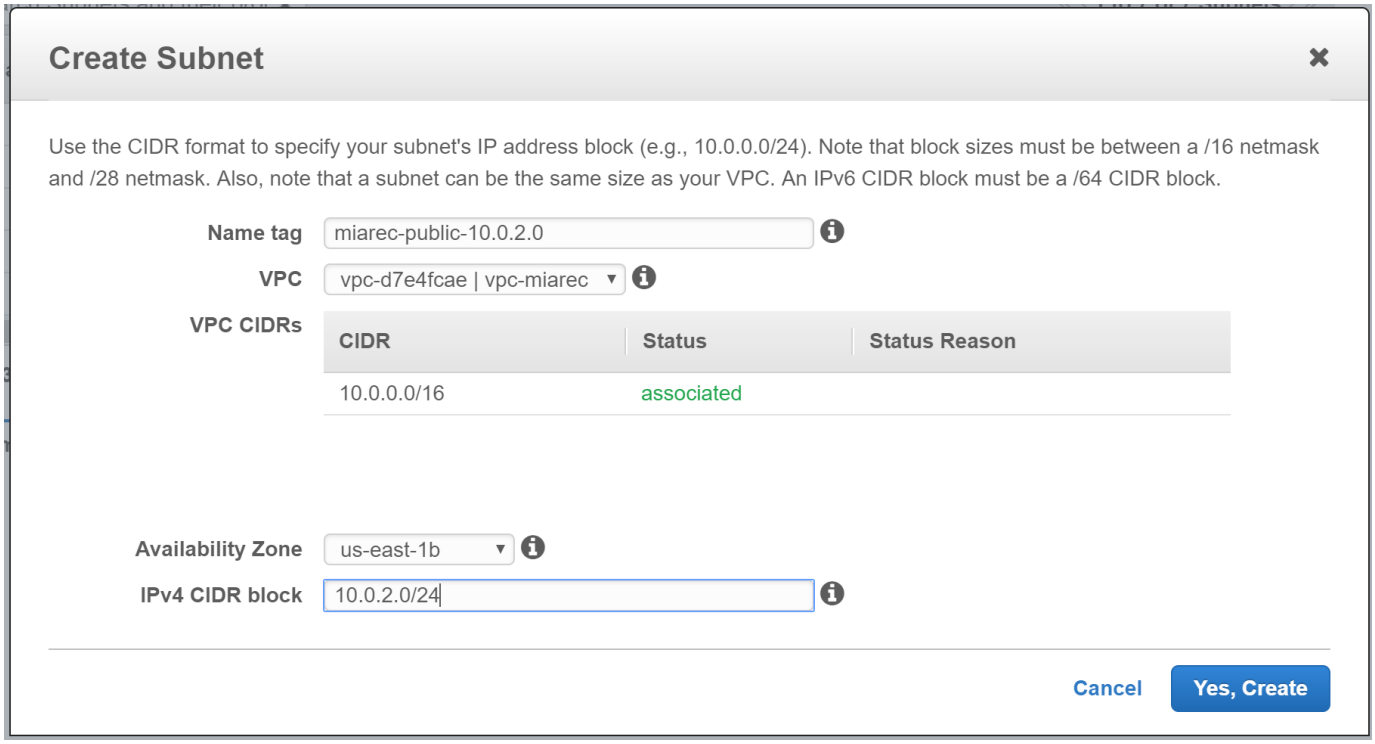
Create subnets

Now let's create two subnets in different Availability Zones. We will deploy two MiaRec instances in different Availability Zones for redundancy. An Availability Zone is a logical data center in Amazon AWS. Each zone has redundant and separate power, networking and connectivity to reduce the likelihood of two zones failing simultaneously.

To create subnets:

1. In the VPC Dashboard, choose **Subnets** and click **Create Subnet** button.
2. Choose the name
3. Associated this subnet with the previously created VPC.
4. Select different different Availability Zones for each of subnets.
5. We use `10.0.1.0/24` for one subnet and `10.0.2.0/24` for the second





In this example, we created two subnets:

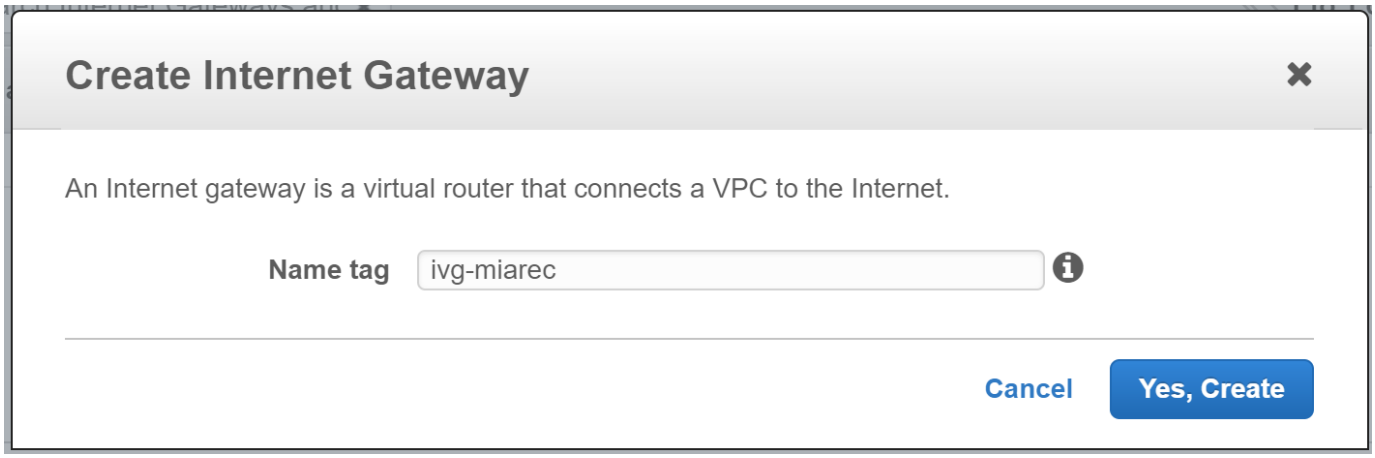
Subnet name	Availability Zone	IPv4 CIDR block
miarec-public-10.0.1.0	us-east-1a	10.0.1.0/24
miarec-public-10.0.2.0	us-east-1b	10.0.2.0/24

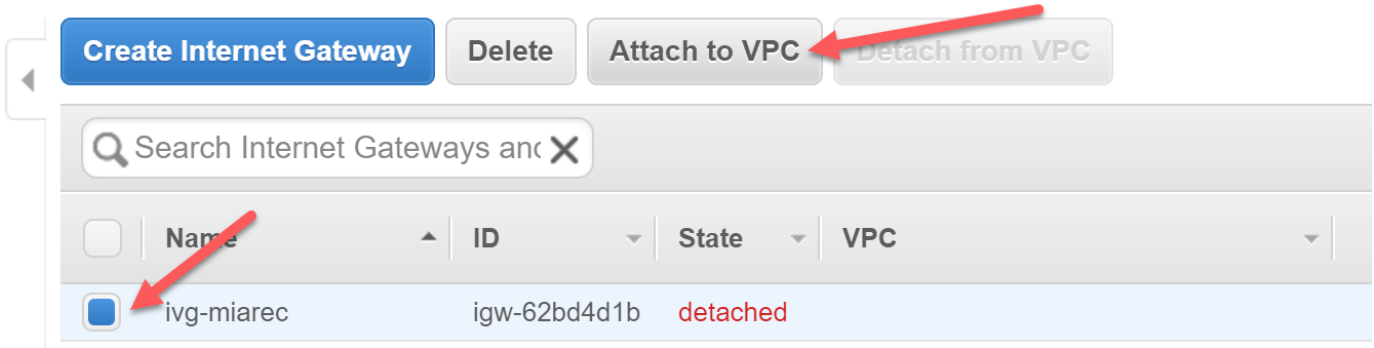
Create Internet Gateway

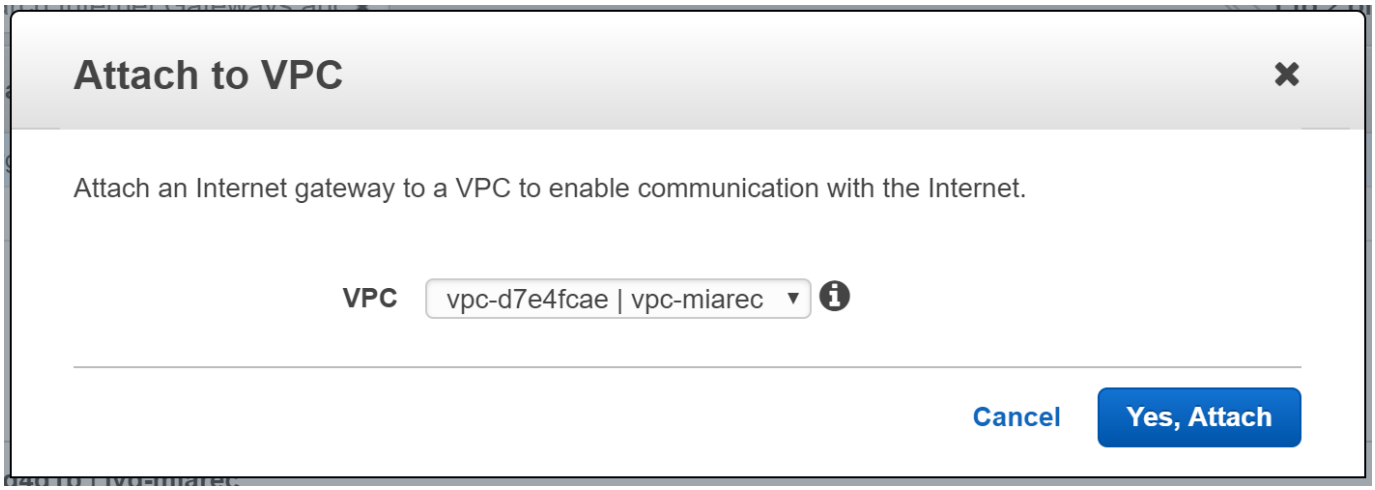
Up to now all our subnets are private. We need to create **Internet Gateway**. An Internet gateway is a virtual router that connects a VPC to the Internet. An Internet gateway serves two purposes: to provide a target in your VPC route tables for Internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses.

To create Internet Gateway:

1. In the VPC Dashboard, choose **Internet Gateways** and click **Create Internet Gateway** button.
2. Choose the name
3. Click **Yes, Create**
4. Select the newly created Internet Gateway from the list and click **Attach to VPC** to associate it with your MiaRec VPC.
5. Click **Yes, Attach**



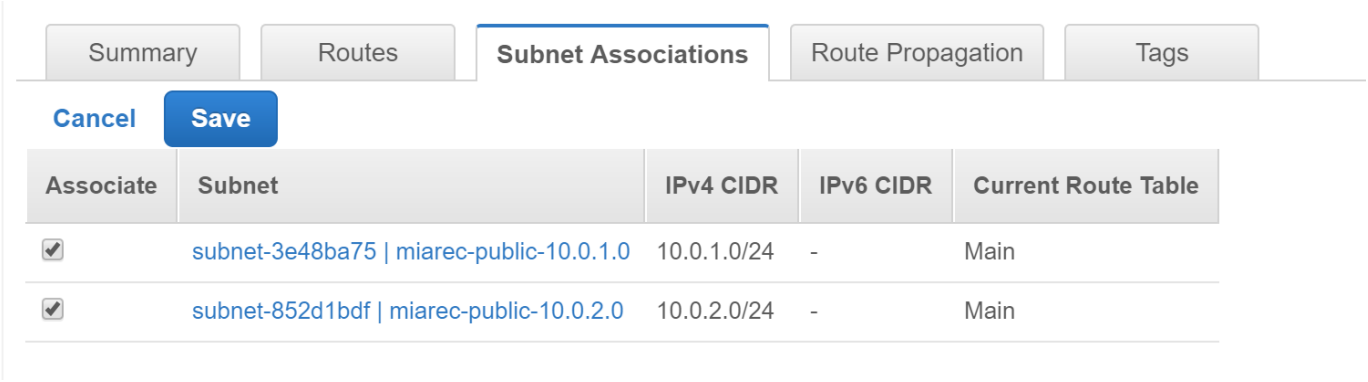


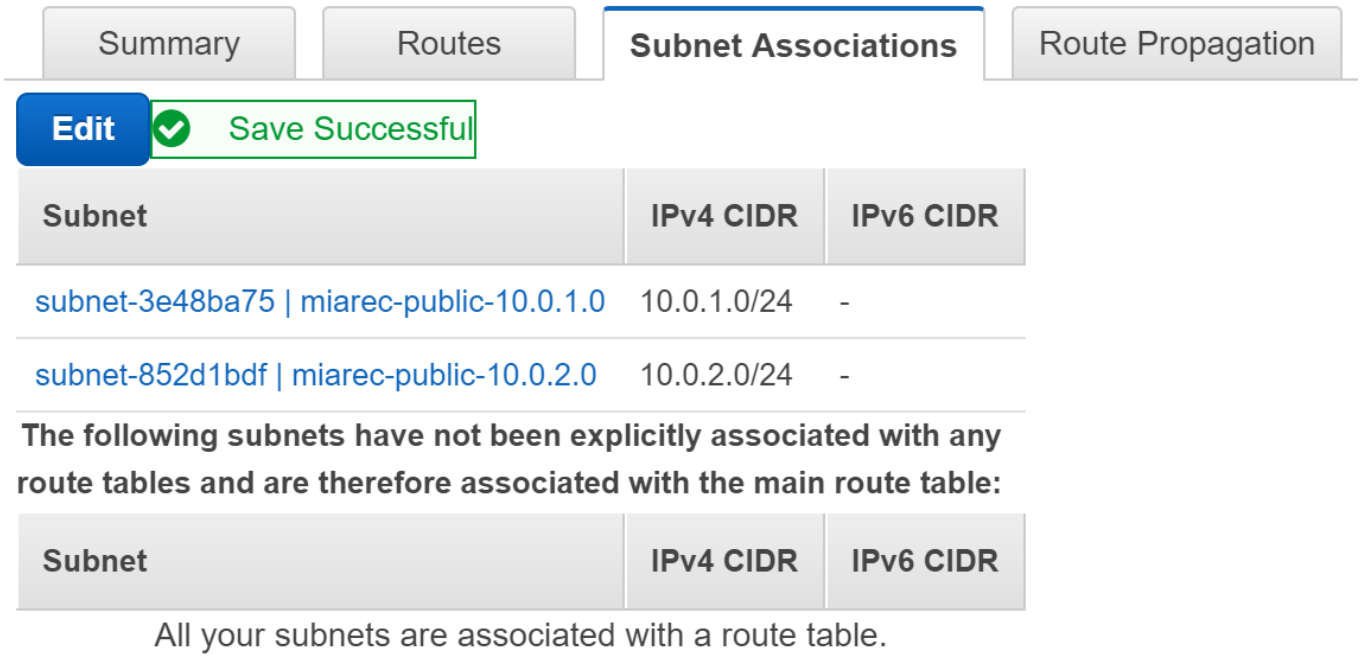


Associate subnets with Route Table

Now we need to associate the subnets with Route Table.

1. Navigate to **Route Tables** in VPC Dashboard.
2. Select the existing route table associated with your newly created VPC.
3. Press the **Subnet Associations** tab on the bottom section. Click **Edit**.
4. Select the subnets and click **Save.**

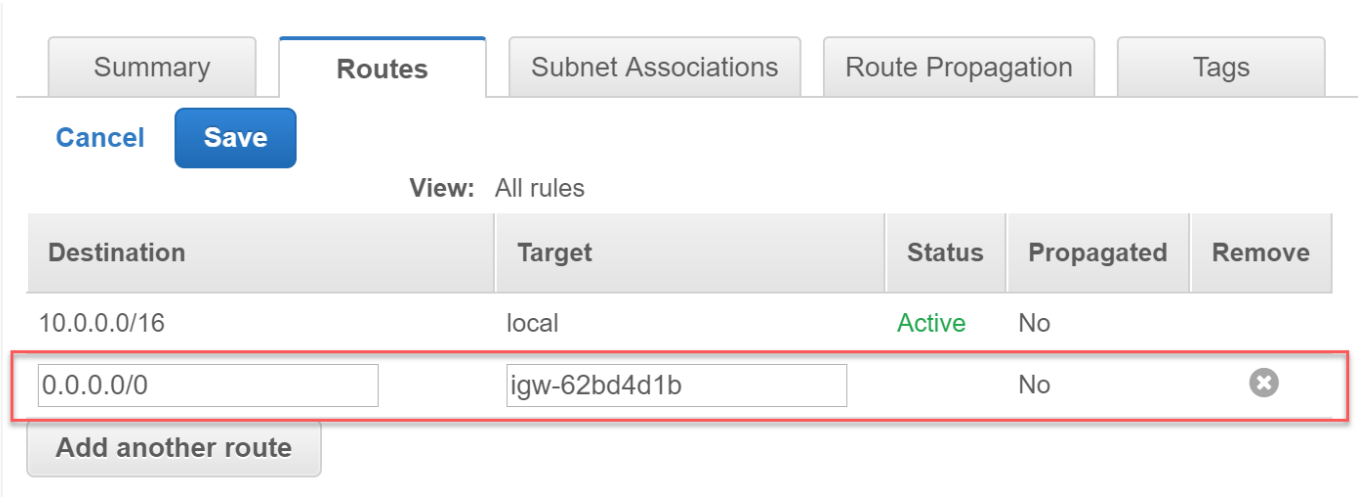




Configure default gateway

We need to add a custom route table for destination 0.0.0.0/0 and Internet Gateway as a target. This will allow our machines to communicate to public Internet, for example, to download software updates.

1. Navigate to **Route Tables** in VPC Dashboard.
2. Select the existing route table associated with your newly created VPC.
3. Press the **Routes** tab on the bottom section. Click **Add another route**
4. Create **Destination** `0.0.0.0/0` with the newly created Internet Gateway as a **Target**.



4.3.3 3. Create EC2 instances

We are going to launch two EC2 instances and install MiaRec software on them. These two instances will be created in different Availability Zones for redundancy.

To create EC instance:

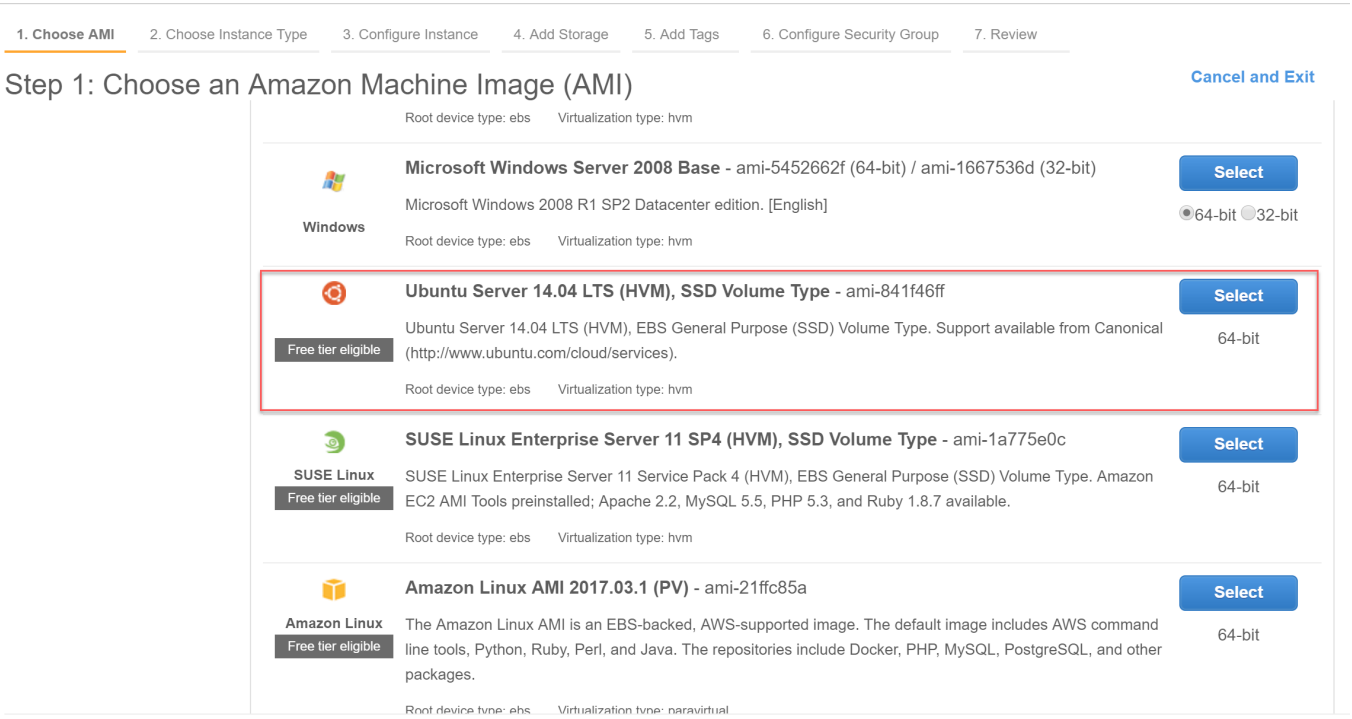
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2
2. Select **Instances** in the left pane and click **Launch Instance**

Step 1. Choose an Amazon Machine Image (AMI)

Select **Ubuntu Server 14.04 LTS, EBS General Purpose (SSD) Volume Type**.

MiaRec supports the following operating systems:

- Centos 6
- Centos 7
- Ubuntu Server 14.04 LTS
- Ubuntu Server 16.04 LTS



Step 2. Choose an Instance Type

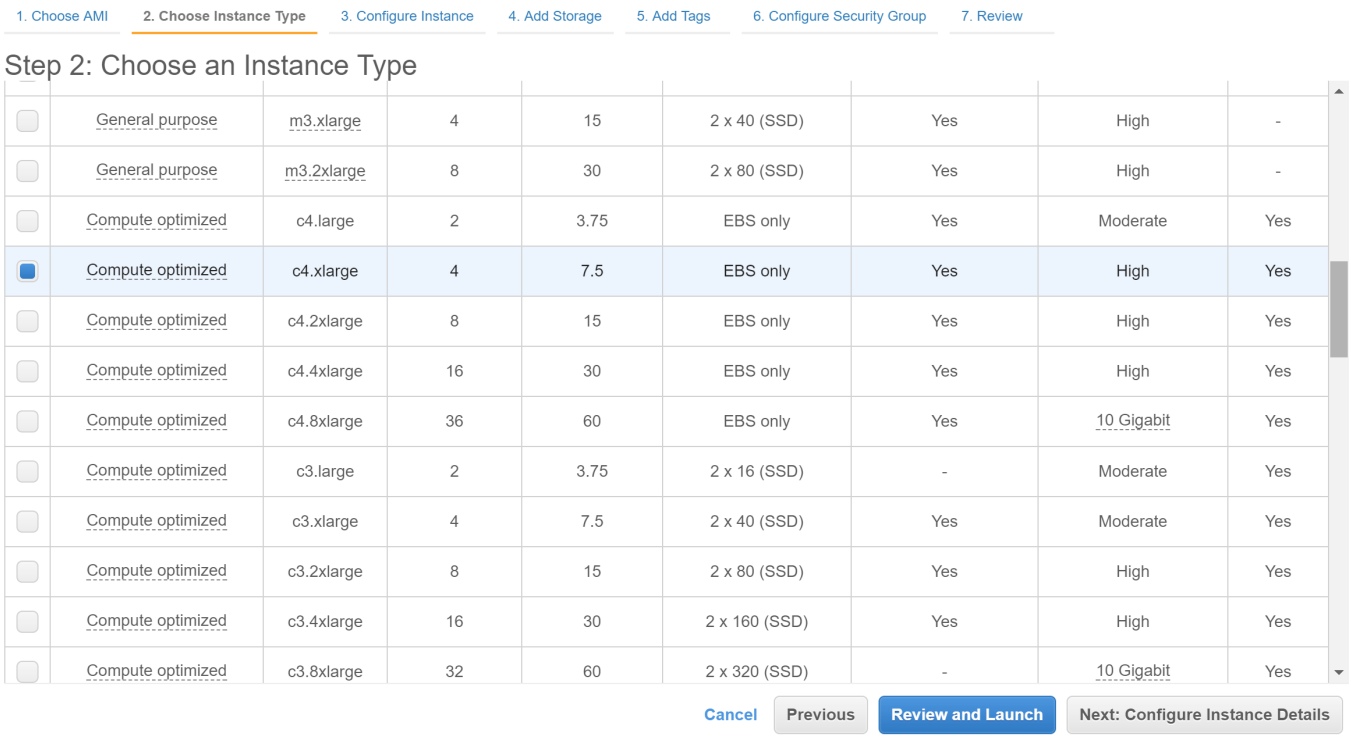
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity.

For MiaRec, we recommend Compute Optimized instances. Refer to the following table for instance type recommendations. These recommendations are based on average system usage (10 calls per day per user, 5 minutes average call duration). Actual hardware requirements may be differ in your case.

Max subscribers	Instance Type	vCPU	Memory (GiB)	Storage	On-demand, Monthly *	1-Year Term, No Upfront, Monthly *
250	c4.large	2	3.75 GiB	EBS only	$72.00	$45.99
500	c4.xlarge	4	8 GiB	EBS only	$143.28	$91.98
1,000	c4.2xlarge	8	15 GiB	EBS only	$286.56	$183.96
2,000	c4.4xlarge	16	31 GiB	EBS only	$573.12	$367.92

(*) - The provided pricing as of data of article (Septempter, 2017) for US-East region, Linux host (Centos/Ubuntu/Amazon Linux).

More than 2,000 users? We recommend to use a decoupled architecture instead of all-in-one setup.

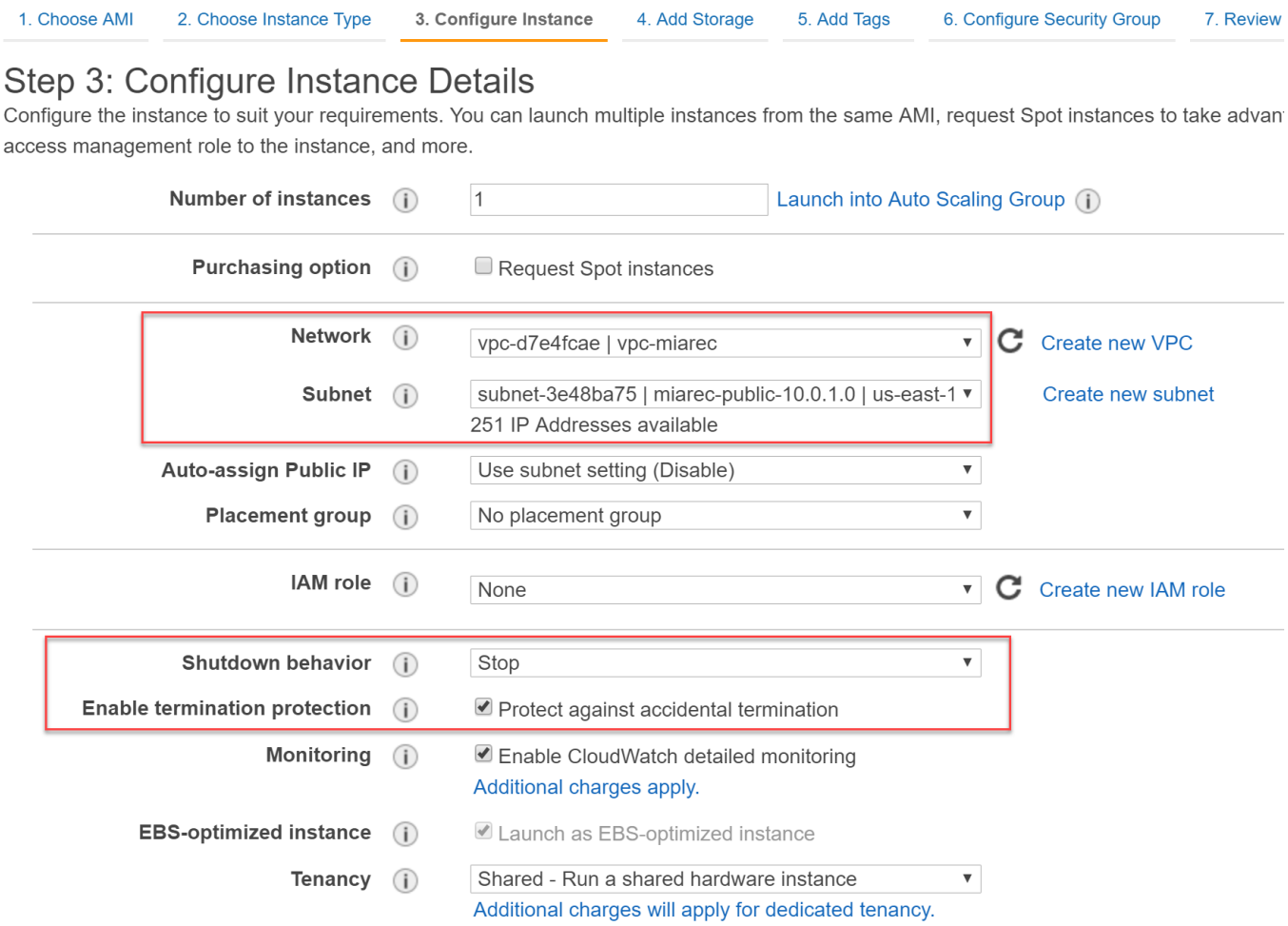


Step 3. Configure Instance Details

Choose separate subnets for each of two MiaRec instances. This will allow to deploy them in different Availability Zones for redundancy.

Shutdown behavior should be set to **Stop**.

We recommend to **Enable termination protection** as a protection from accidental deletion of server.



Step 4. Add Storage

Specify the desired disk storage size for EBS volume.

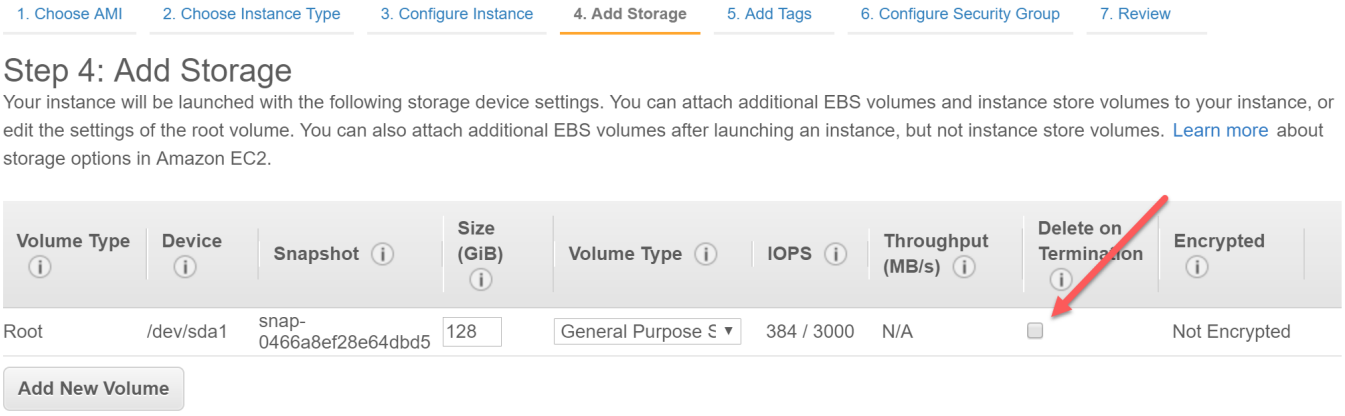
- As a **Volume Type** select **General Purpose SSD** as a minimum. For high load, it is possible to select **Provisioned IOPS SSD** (it is more expensive, but provides guaranteed I/O performance).

Imporant!. Uncheck **Delete on Termination**. This will allow you to detach this EBS volume from EC2 instance and attach to new one, for example, with better hardware specs.

Disk storage will be used for:

- OS and application files
- Database data files, approximately 3GB per 1 million records in database
- Application logs
- Temporary location for audio files (before the files are uploaded to S3 for long term storage). 0.24 MB/minute in MP3 stereo format. We recommend to keep available disk space for at least 3 days of data. In case of issues in upload process to S3, it gives enough time to administrator to troubleshoot and fix issue. This will make the system less dependent on S3 availability.

Number of users	Avg calls/ day/user	Avg duration	Total minutes/day	Storage/ day	Recommended EBS volume
50	10	5 min	75,000 min	18 GB	100 GB
100	10	5 min	150,000 min	36 GB	150 GB
250	10	5 min	375,000 min	90 GB	320 GB
500	10	5 min	750,000 min	180 GB	600 GB
1,000	10	5 min	1,500,000 min	360 GB	1,200 GB
2,000	10	5 min	3,000,000 min	720 GB	2,400 GB



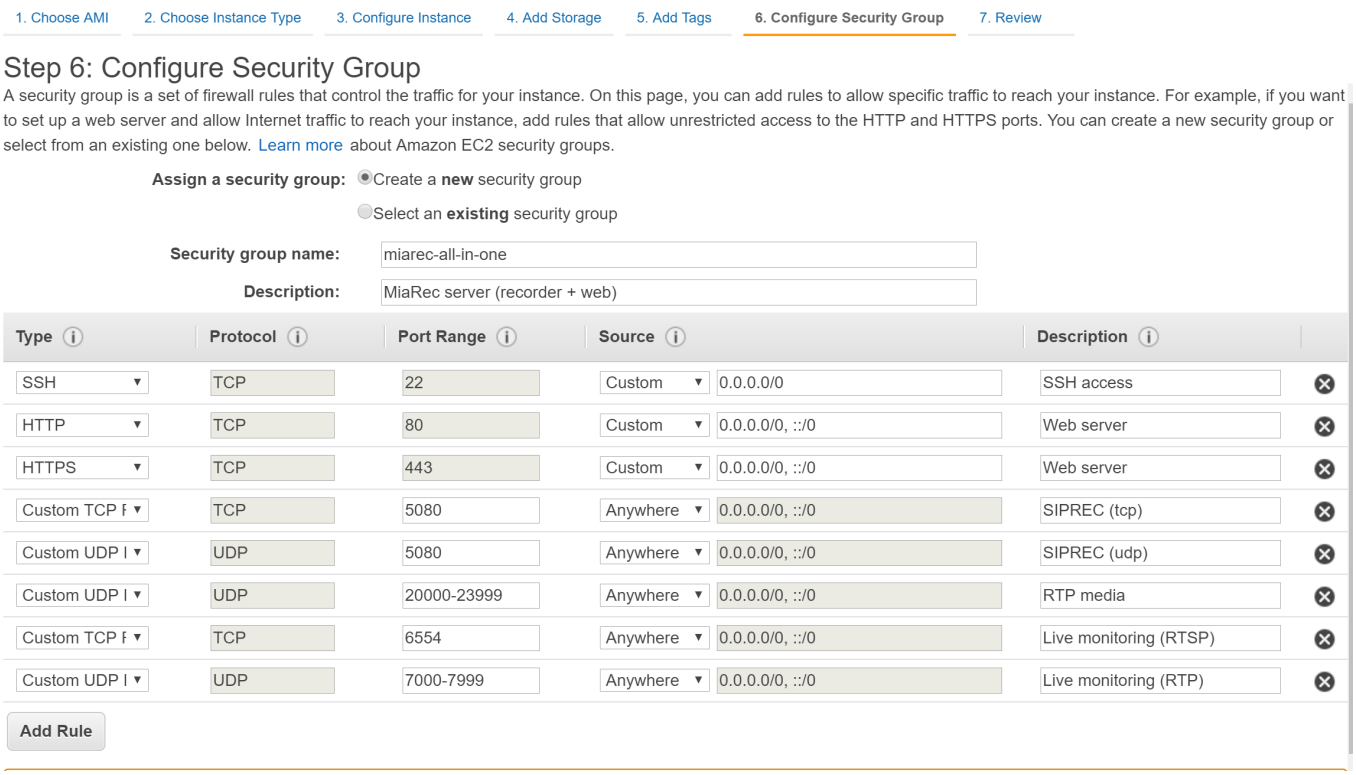
Step 6. Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance.

MiaRec application requires the following ports to be opened:

- TCP 22 for SSH inbound connection
- TCP 80 and 443 for web server
- TCP 6554 and UDP 7000-7999 for live monitoring (optional)
- TCP/UDP 5080 for SIPREC signaling and UDP 20000-23999 for RTP media (these port values can be changed in MiaRec web admin portal)

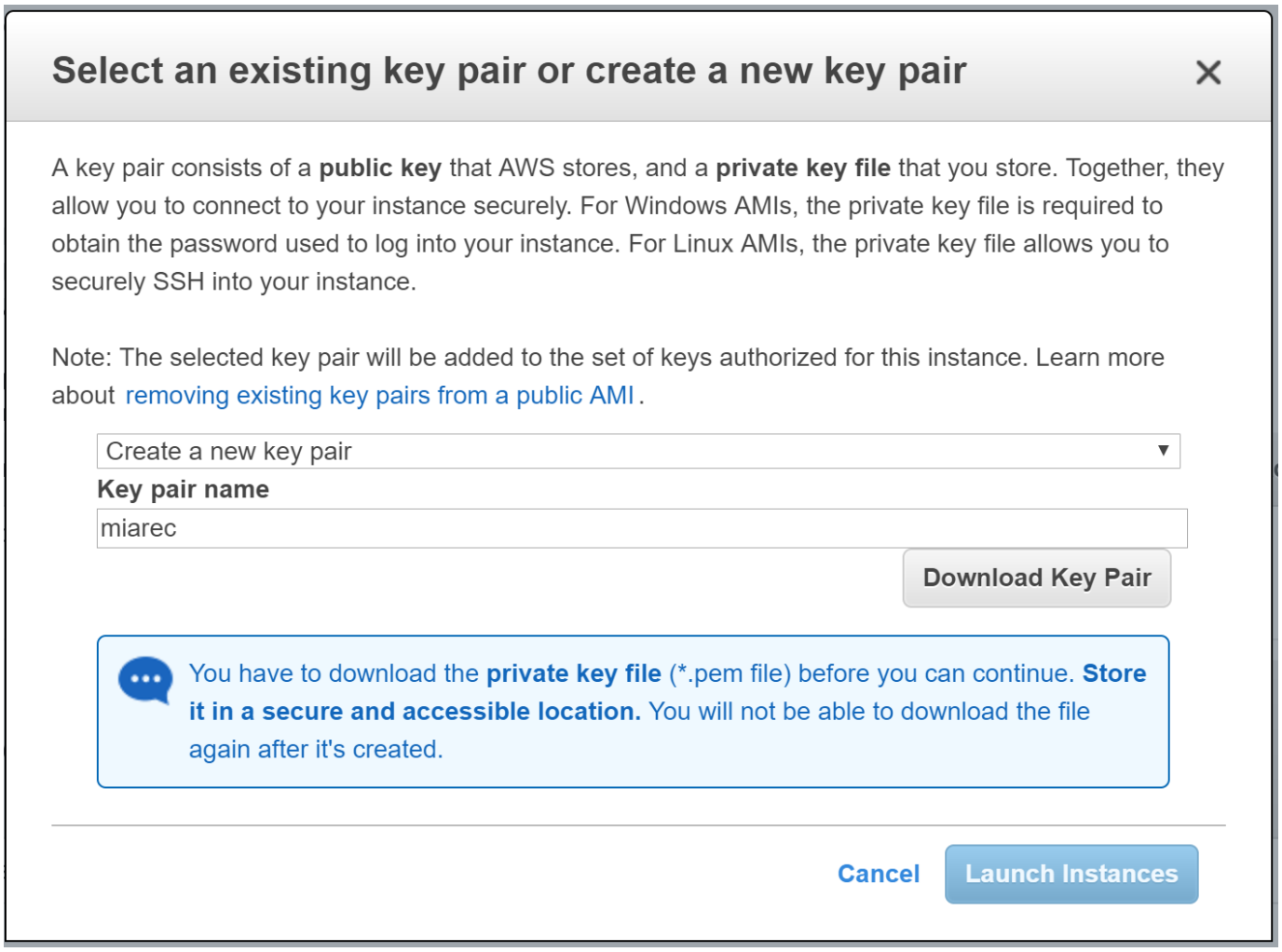
Important!. In the following example, SIPREC and RTP ports are opened to all sources (0.0.0.0/0). For security reasons, access to these ports should be limited to your phone only. Specify there the IP-addresses, from which your phone system sends SIPREC and RTP traffic.



Create SSH keys

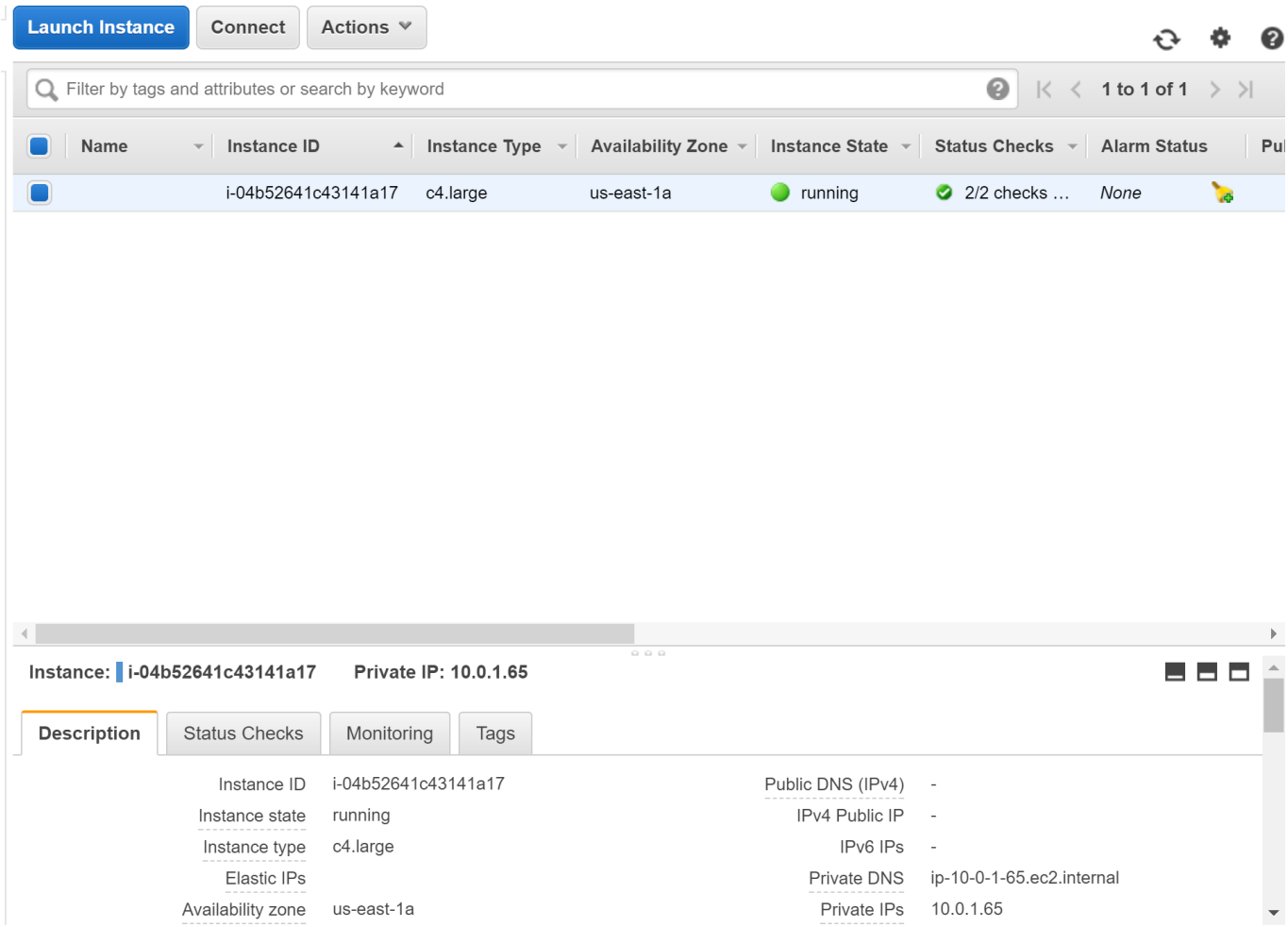
When you launch an instance, you should specify the name of the key pair you plan to use to connect to the instance. You can use Amazon EC2 to create your key pair. Alternatively, you could use a third-party tool and then import the public key to Amazon EC2.

If you use Amazon to create your key pair, then you have to download the private key file (*.pem file) and store it in a secure and accessible location. You will use this key to access the instance via SSH.



Check status of running instances

Navigate to **Instances** section of EC2 Dashboard to see your new instance running.



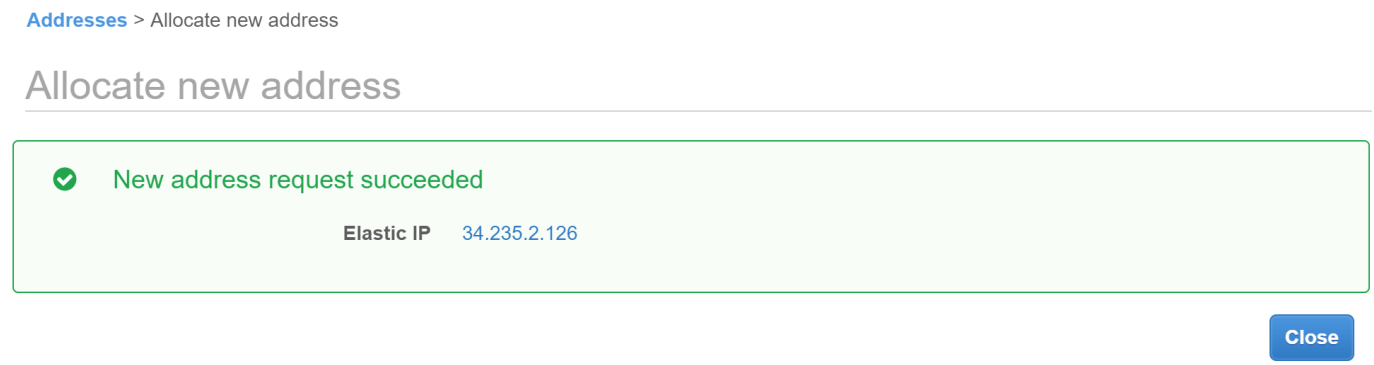
4.3.4 4. Configure Elastic IP address

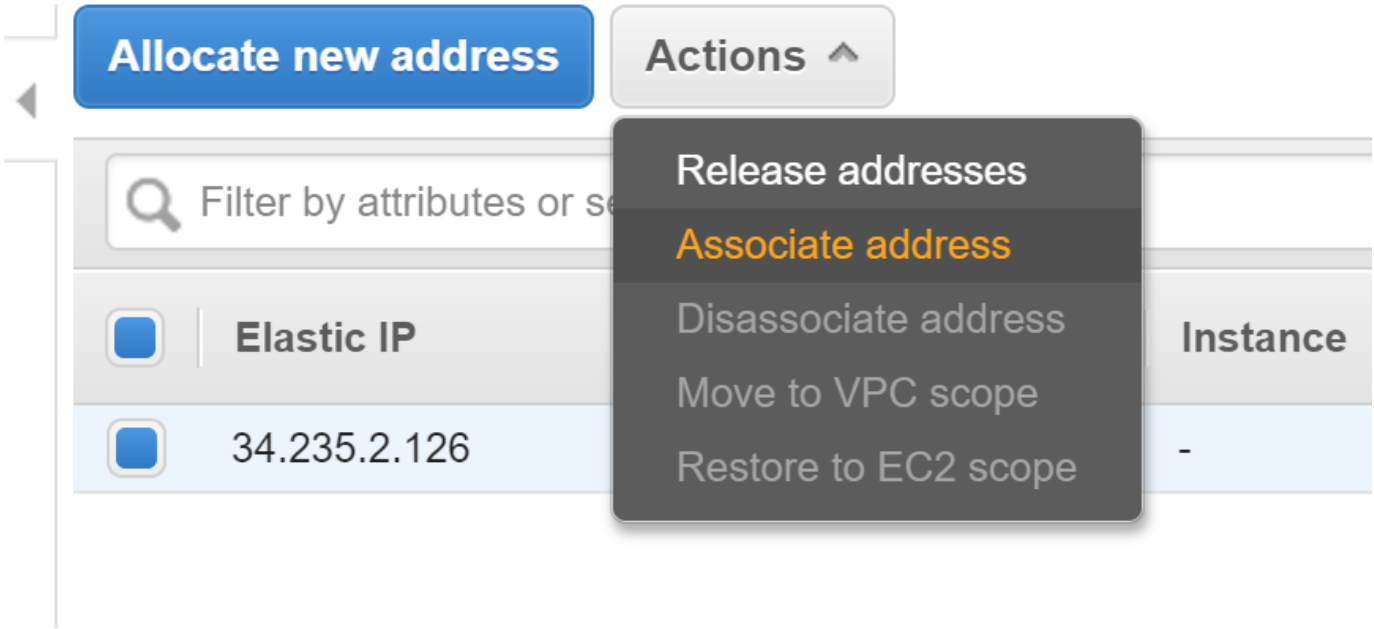
An Elastic IP address is a public IPv4 address, which is reachable from the Internet. An Elastic IP address is associated with your AWS account. With an Elastic IP address, you can mask the failure of an instance or software by rapidly remapping the address to another instance in your account.

We need to create an Elastip IP address for each MiaRec instance as both of them will be accessible from the Internet.

To allocate Elastic IP address:

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2
2. Select **Elastic IP address** in the left pane and click **Allocate new address**
3. Once the IP address is allocated, select it in a list and choose **Associate address** from **Actions** drop-down.
4. Associate the IP address with EC2 instances.
5. Repeat these steps for the second EC2 instance.





4.3.5 5. Install MiaRec software on EC2 instance

We will deploy MiaRec to Amazon EC2 instances using Ansible provisioning tool.

You need to follow the step-by-step instructions "Installation on Linux using Ansible" with a few additions described below.

You can run Ansible playbook from:

- Any Linux host with Internet access (Ubuntu, Centos, Redhat). Mac OS X should work as well, but not tested by us.
- Windows 10 with Linux subsystem
- Another EC2 instance.

Copy SSH private key to Ansible Controller machine

You need to use previously created key to access the instance via SSH. Check Connecting to Your Linux Instance Using SSH for details.

Copy the private key to the Ansible Controller machine, for example, to `~/.ssh/aws-key.pem`.

Use the chmod command to make sure that your private key file isn't publicly viewable:

```
chmod 400 ~/.ssh/aws-key.pem
```

Test SSH connection to EC2 instance:

```
ssh -i ~/.ssh/aws-miarec.pem ubuntu@34.235.2.126
```

If everything is ok, you should see the following message the first time you connect to it:

```
The authenticity of host '34.235.2.126 (34.235.2.126)' can't be established.
ECDSA key fingerprint is SHA256:uTU/hyG+7qy1Aq0CxliKekYDWXZI0EEAaPkmXuA9K9M.
Are you sure you want to continue connecting (yes/no)?
```

Enter `yes`. You should connect to the instance now and see Ubuntu welcome message like:

```
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-125-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Fri Sep 22 01:32:04 UTC 2017

System load:  0.0               Processes:           103
Usage of /:   0.6% of 125.86GB   Users logged in:     0
Memory usage: 2%                IP address for eth0: 10.0.1.65
Swap usage:   0%
```

Create Inventory file (/opt/ansible-miarec/hosts)

In the step [3. Configure deployment] of instructions, you will need to create `hosts` file. If you selected Ubuntu as OS for EC2 instance, then create the following file:

```
miarec1 ansible_ssh_host=1.1.1.1 ansible_ssh_private_key_file=~/.ssh/aws-key.pem ansible_port=22 ansible_user=ubuntu
miarec2 ansible_ssh_host=2.2.2.2 ansible_ssh_private_key_file=~/.ssh/aws-key.pem ansible_port=22 ansible_user=ubuntu

[miarec]
miarec1
miarec2
```

Where:

- Two hosts are defined inside group `miarec`. Ansible playbook will be executed against all hosts in group. By default, installation will be done simultaneously.
- Replace 1.1.1.1 and 2.2.2.2 with public ip-addresses of your EC2 instances (these should point to Elastic IP addresses created previously).
- The parameter `ansible_ssh_private_key` specifies the location of SSH private key for connecting to EC2 instance (in our example, it is `~/.ssh/aws-key.pem`).
- The parameter `ansible_user` is set to `ubuntu` for Ubuntu system. For Centos, you need to set it to `root`.

Now, test Ansible connection to AWS using the command:

```
ansible miarec -m shell -a "uname -a"
```

This command will connect to all hosts in group `miarec` in inventory file and print the output of command `uname -a`.

You should see something like:

```
miarec1 | SUCCESS | rc=0 >>
Linux ip-10-0-1-65 3.13.0-125-generic #174-Ubuntu SMP Mon Jul 10 18:51:24 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux

miarec2 | SUCCESS | rc=0 >>
Linux ip-10-0-2-73 3.13.0-125-generic #174-Ubuntu SMP Mon Jul 10 18:51:24 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

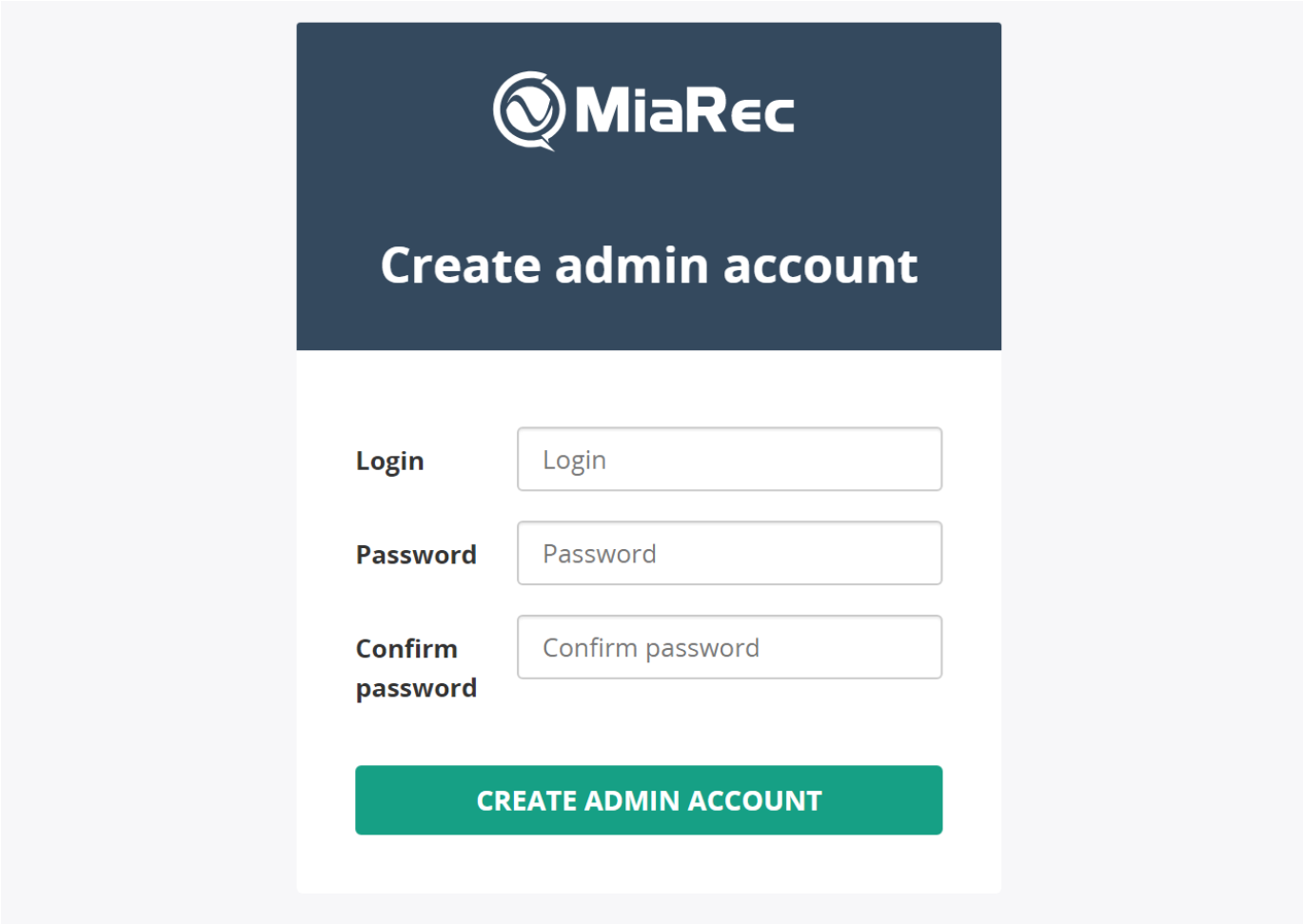
Deploy MiaRec to EC2 instances

Follow instructions to deploy MiaRec on EC2 instances using Ansible.

Verify MiaRec web portal and create admin account

Navigate in web browser to the Elastic IP address of each of MiaRec instances, like **http://1.2.3.4**.

When you access MiaRec web portal the first time, it will ask you to create admin account. You need to create unique accounts for these two servers, like "**admin**" and "**admin2**". This is necessary to prevent conflicts during synchronization between servers.



4.3.6 6. Configure Route 53 DNS Failover for web traffic

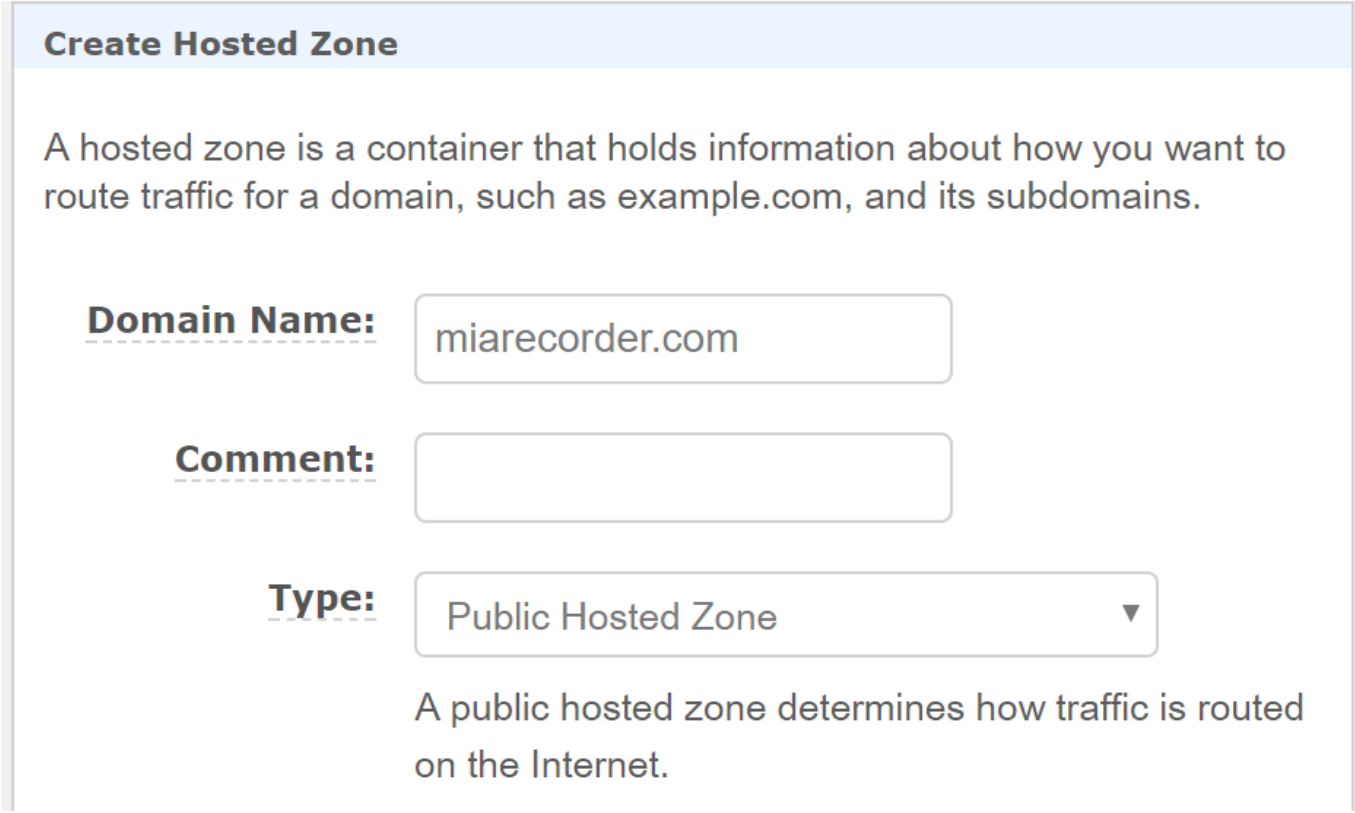
Amazon Route 53 service monitors the health and performance of MiaRec instances. Using DNS failover, it can route the web traffic from an unhealthy instance to a healthy one.

Prerequisites:

- Your domain name has to be managed by Amazon Route 53, otherwise it will not be possible to use DNS Failover. You can register new domain name for your MiaRec HA cluster or use existing one.

Create Hosted Zone

1. Sign in to the AWS Management Console and open the Amazon Route 53 console at https://console.aws.amazon.com/route53/.
2. In the navigation pane of the Route 53 console, choose **Hosted zones**, and then choose **Create Hosted Zone**.
3. Enter the registered domain name into **Domain Name**. In this guide, we use domain `miarecorder.com` as an example.



Create A-records for MiaRec servers (miarec1 and miarec2)

We need to create DNS A-record for each of our MiaRec servers. In this example, we use "miarec1" and "miarec2", but you can name it whatever you want.

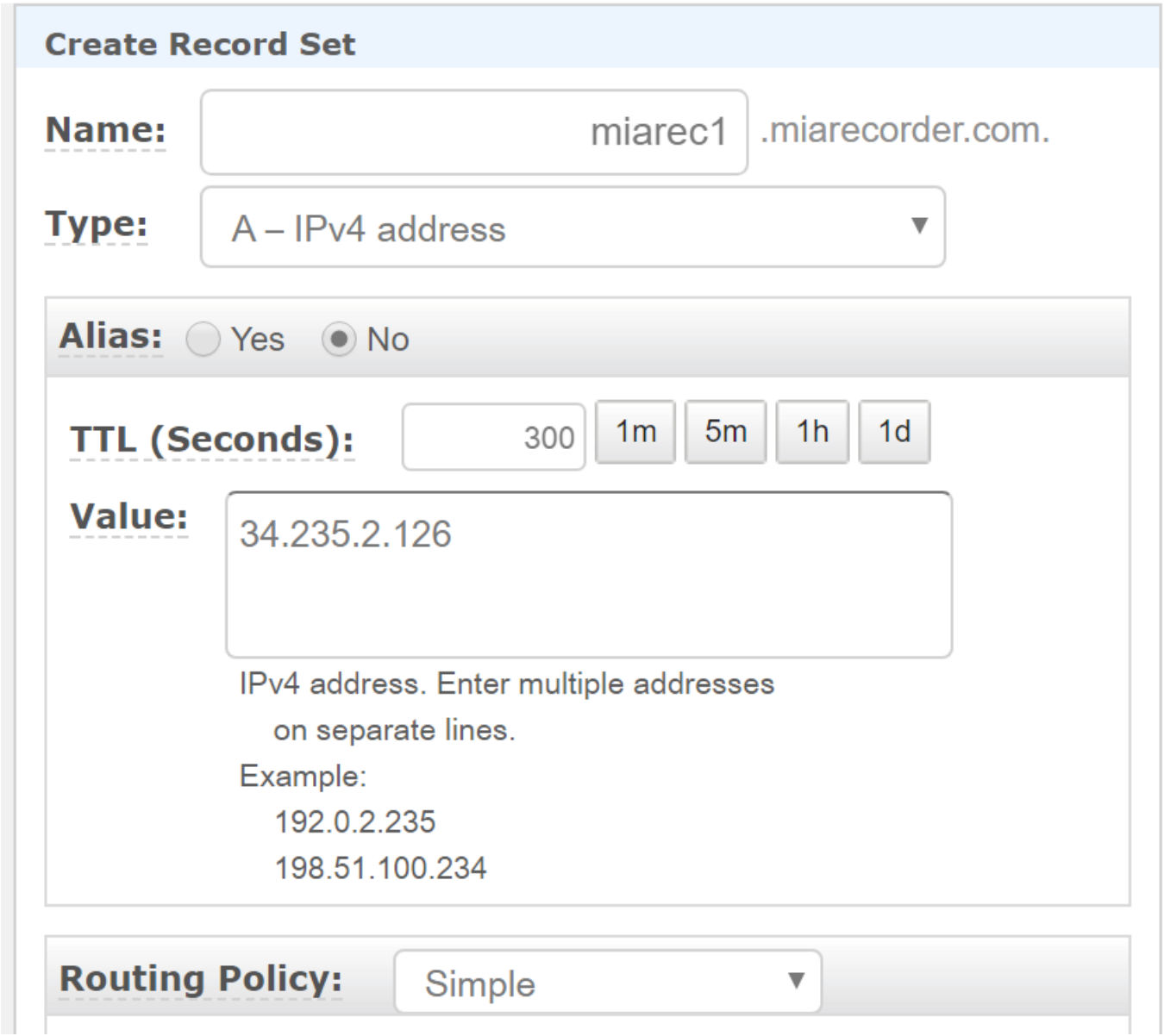
Name	Type	Alias	TTL	Value	Routing Policy
miarec1	A	No	300	x.x.x.x	Simple
miarec2	A	No	300	y.y.y.y	Simple

Where:

- **x.x.x.x** is the Elastic IP address (public) of the first MiaRec instance
- **y.y.y.y** is the Elastic IP address (public) of the second MiaRec instance

To create A-records:

1. In the navigation pane of the Route 53 console, choose **Hosted zones**, select the domain name, and then choose **Create Record Set**.
2. Choose **A - IPv4 address** for **Type**.
3. Enter Elastic IP address of the MiaRec EC2 instance into **Value** field.
4. Choose **Simple** for **Routing Policy**
5. A default **TTL** value is ok
6. Repeat these steps for the second MiaRec EC2 instance.

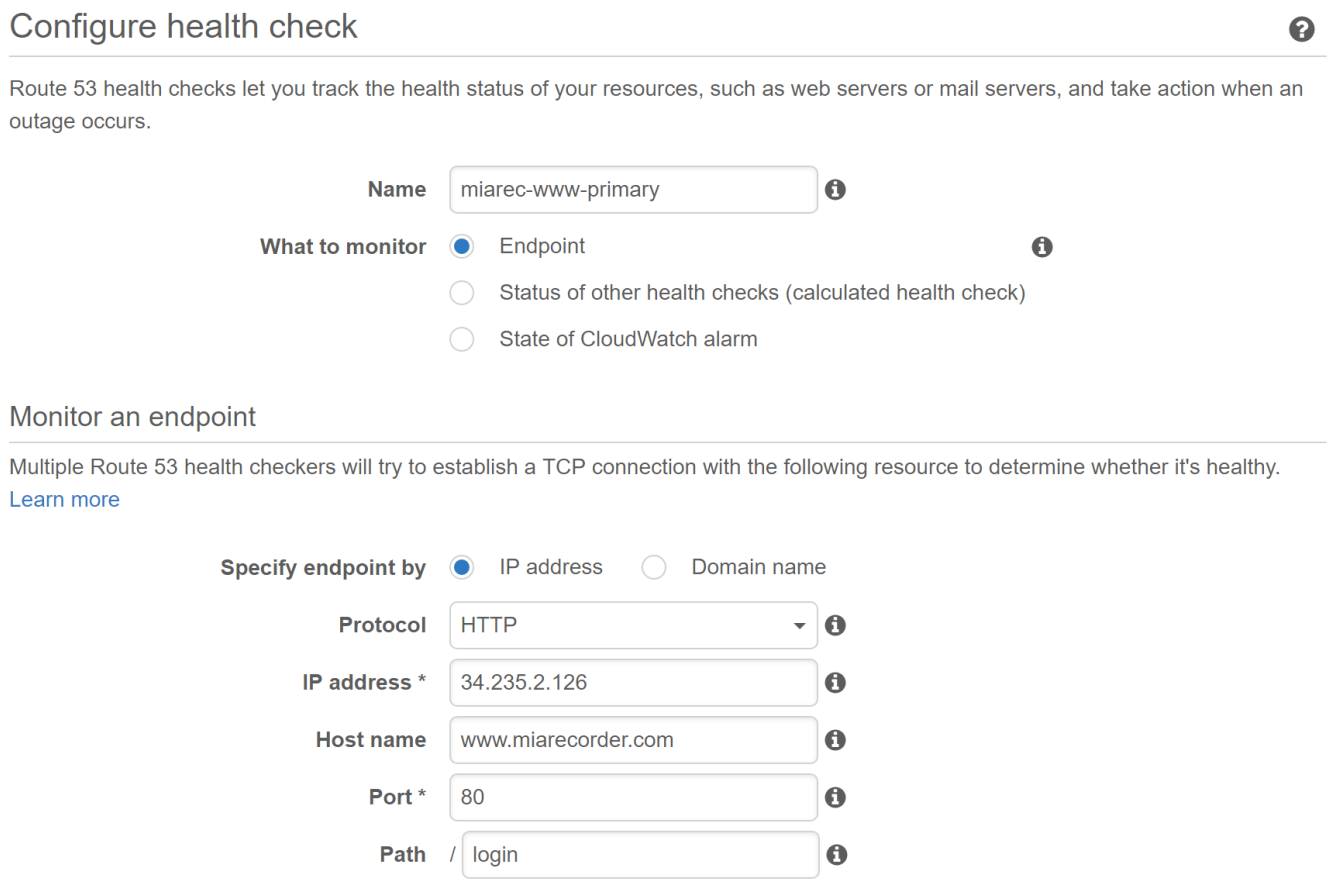


Create Health checks for MiaRec web servers

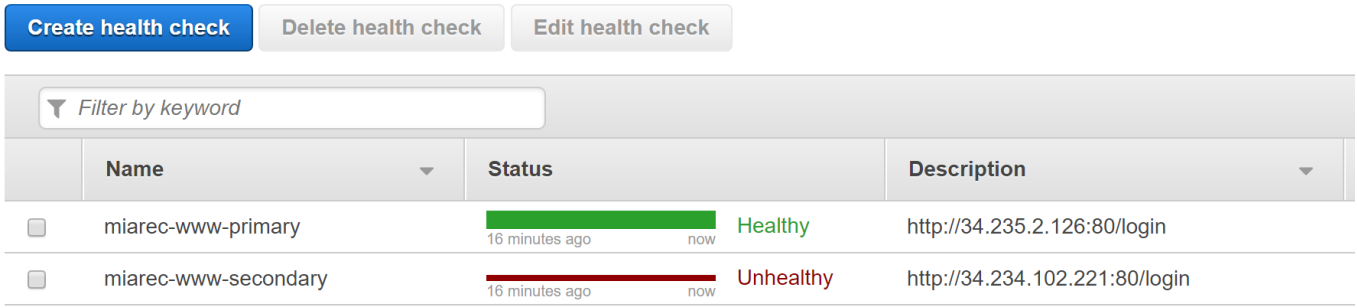
For DNS Failover, we need to configure health checks for each of servers.

For each MiaRec instance:

1. In the navigation pane of the Route 53 console, choose **Hosted checks**, select the domain name, and then choose **Create health check**.
2. Choose a convenient name, like "miarec-www-primary" and "miarec-www-secondary".
3. Choose **Domain name** from **Specify endpoint by** options.
4. Choose **HTTP** for **Protocol** (this option should be HTTPS if HTTP is disabled)
5. Enter the DNS name of the MiaRec EC2 instance, in our example, domain names are "miarec1.miarecorder.com" and "miarec2.miarecorder.com".
6. Choose **80** for **Port**
7. Enter "login" for **Path** (this health check will verify if login page is accessible).
8. Keep other settings as default.
9. Repeat these steps for the second MiaRec EC2 instance.



In a couple of minutes, you should be able to see health check report as shown in the following screenshot.



Create DNS A-record for web traffic (with failover)

Now, we are going to create DNS A-record, which will be used by end users for accessing MiaRec web portal, something like "**recordings.mycompany.com**". We will configure DNS Failover for this record. Amazon Route 53 services will route web traffic to the secondary server, when health check for the primary server returns error.

Create two records with the following settings:

Name	Type	Alias	TTL	Value	Routing Policy	Failover Record Type	Set ID	Health Check to Associate
recordings	A	No	60	x.x.x.x	Failover	Primary	recordings-Primary	miarec-www-primary
recordings	A	No	60	y.y.y.y	Failover	Secondary	recordings-Secondary	miarec-www-secondary

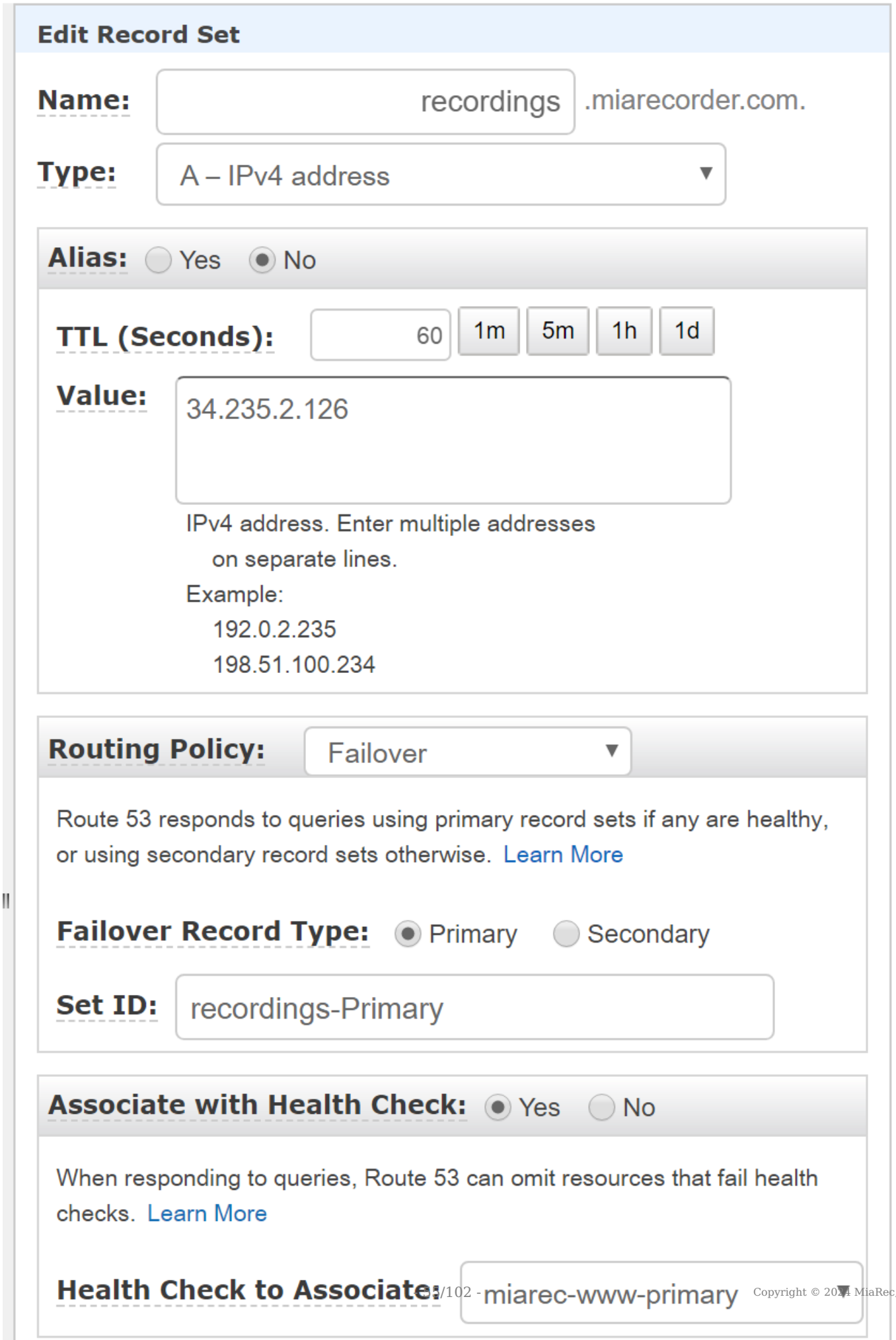
Where:

- **x.x.x.x** is the Elastic IP address (public) of the first MiaRec instance
- **y.y.y.y** is the Elastic IP address (public) of the second MiaRec instance

To create A-records with DNS Failover, repeat for each MiaRec instance:

1. In the navigation pane of the Route 53 console, choose **Hosted zones**, select the domain name, and then choose **Create Record Set**.
2. Choose **A - IPv4 address** for **Type**.
3. Select **60** for **TTL**. Recommended value is between 30 to 60 seconds. Each client caches DNS records. In case of failover, the web browser may attempt to access the unhealthy instance until cache expires.
4. Enter Elastic IP address of the MiaRec EC2 instance into **Value** field.
5. Choose **Failover** for **Routing Policy**
6. Choose **Primary** for the first instance and **Secondary** for the second instance for **Failover Record Type**
7. Choose a convenient name for **Set ID**
8. Associate this record to the corresponding health check, created previously.
9. Repeat these steps for the second MiaRec EC2 instance.

Below screenshots, demonstrate configuration of Record Set for both MiaRec instances.



Edit Record Set

Name: recordings .miarecorder.com.

Type: A – IPv4 address

Alias: Yes No

TTL (Seconds): 60 1m 5m 1h 1d

Value: 34.196.49.59

IPv4 address. Enter multiple addresses on separate lines.

Example:

192.0.2.235

198.51.100.234

Routing Policy: Failover

Route 53 responds to queries using primary record sets if any are healthy, or using secondary record sets otherwise. Learn More

Failover Record Type: Primary Secondary

Set ID: recordings-Secondary

Associate with Health Check: Yes No

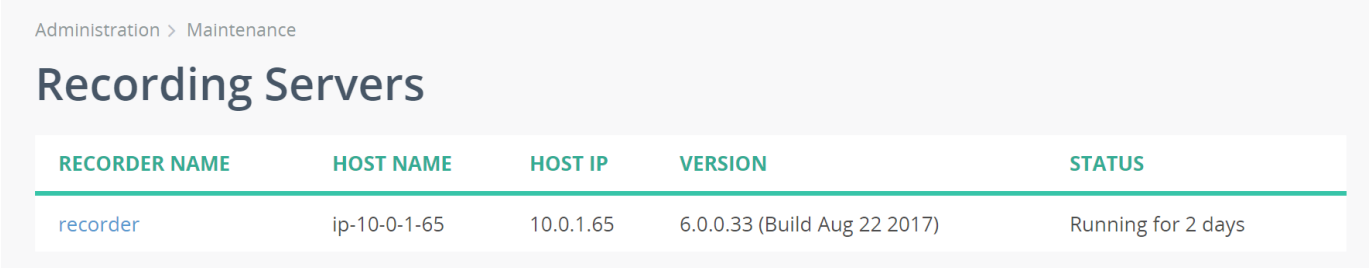
When responding to queries, Route 53 can omit resources that fail health checks. Learn More

Health Check to Associate: miarec-www-secondary

Test DNS failover

Navigate in web browser to **http://recordings.yourdomain.com**.

Login as administrator and navigate to **Administration -> Maintenance -> Recording Servers**. You should see the private IP address of this instance. This information allows you to determine on which instance you are now. The primary instance should be in a subnet 10.0.1.x and the secondary in 10.0.2.x.

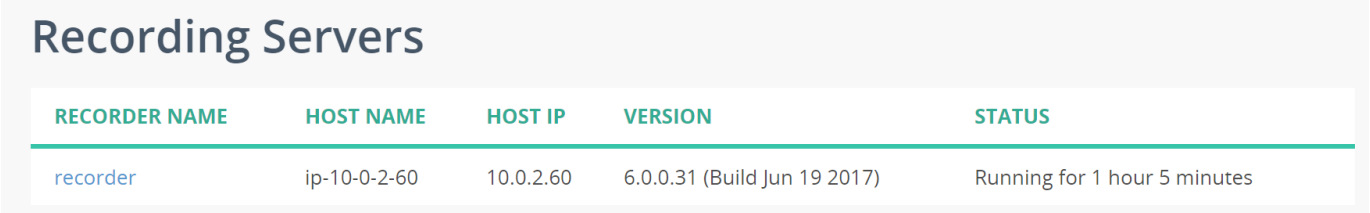


Now, simulate failure using one of the following methods:

- Stop Apache web server using SSH (on Ubuntu the command is `sudo service apache2 stop`, on Centos it is `sudo service httpd stop`).
- Shutdown the server via SSH using command `sudo shutdown -h now`
- Stop instance via **Actions** menu in **Amazon EC2 Dashboard**.

Try to access MiaRec web portal using web browser. It may take a few minutes for Amazon Route 53 to detect server failure (by default, it checks the server health every 30 seconds and requires at least 3 consecutive failures before the server is marked unhealthy). Once the server is marked unhealthy, the domain name **http://recordings.yourdomain.com** is automatically routed to the secondary MiaRec instance IP-address. If the TTL value is reasonably small (no more than 60 seconds), then failover should shortly after that.

Within 3-4 minutes, you should be able to access the MiaRec web portal again. Login as administrator and navigate **Administration -> Maintenance -> Recording Servers** to check on which instance you are now.



Now, restore the primary server and verify if web-traffic is routed back to it after 3-5 minutes.

4.3.7 7. Configure DNS SRV for SIPREC traffic

The SRV record is a Domain Name System (DNS) resource record that is used to identify servers that host specific services. Each of servers is assigned a priority and weight, which allows to use DNS SRV for failover, load balancing or both. In this guide, we use DNS SRV for failover purposes only. For load balancing, we recommend to use different architecture (decoupled).

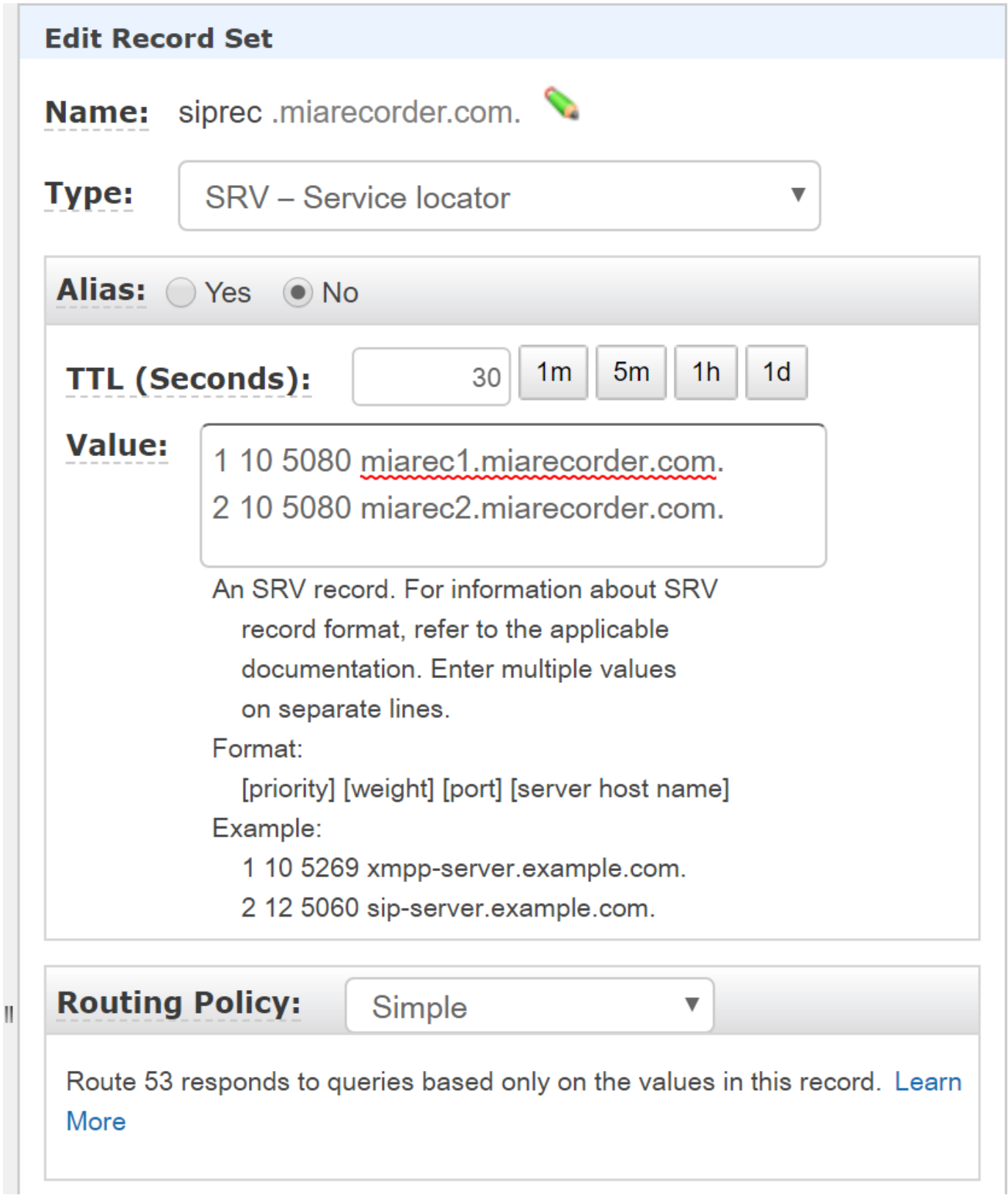
To create DNS SRV records in Route 53:

1. In the navigation pane of the Route 53 console, choose **Hosted zones**, select the domain name, and then choose **Create Record Set**.
2. Choose **SRV - Service locator** for **Type**.
3. Enter the following data into **Value** field:

```
1 10 5080 miarec1.yourdomain.com
2 10 5080 miarec2.yourdomain.com
```

   Here, we configure two servers with priority 1 and 2 correspondingly. In this configuration the phone platform will always use the miarec1.yourdomain.com server (priority 1) unless it is not healthy. 4. A default **TTL** value is ok 5. Choose **Simple** for **Routing Policy**

The following screenshot demonstrates the configuration of DNS SRV:



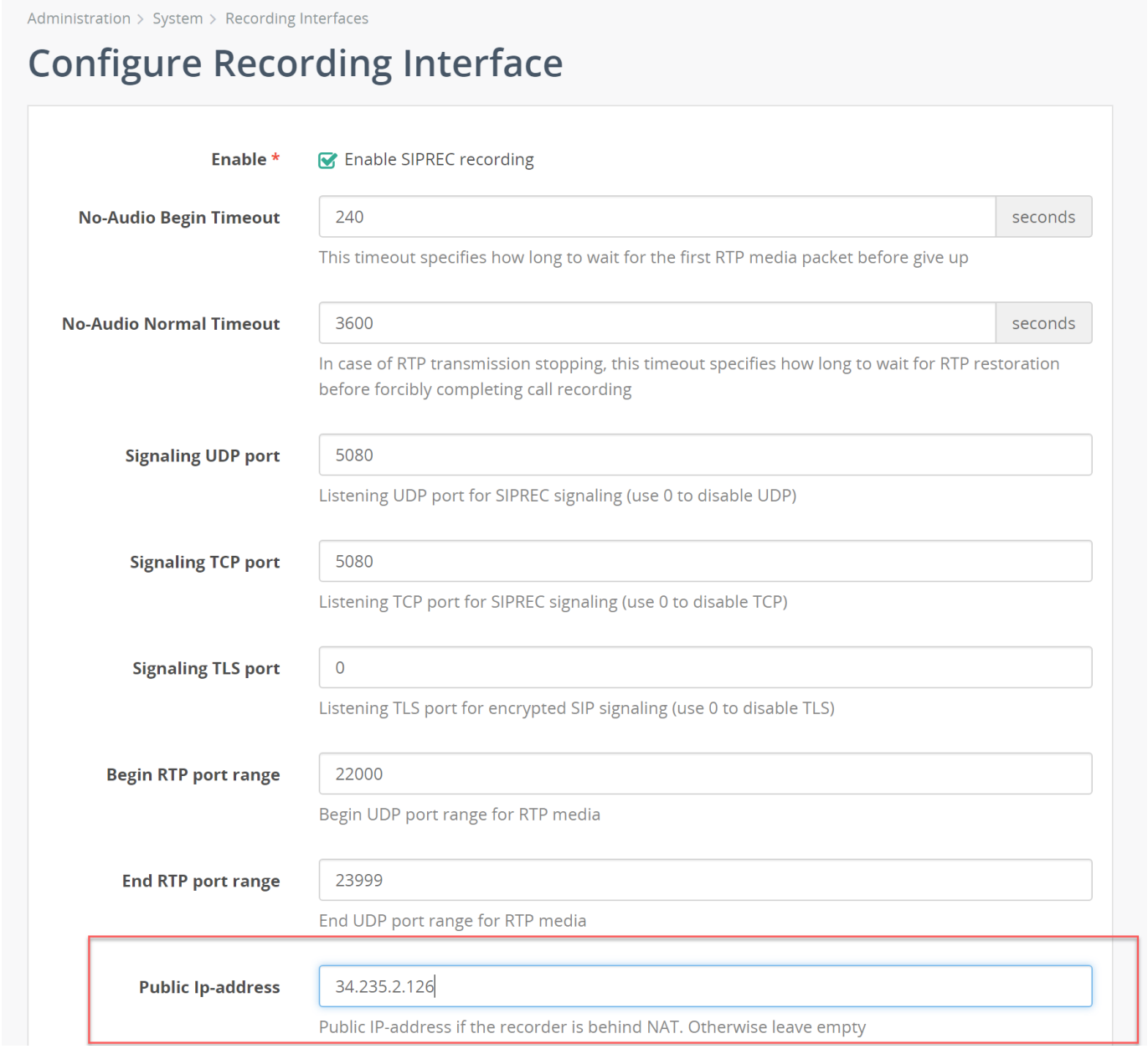
4.3.8 8. Configure SIPREC recording

Configure SIPREC recording interface in MiaRec

We need to configure public ip-address in each of MiaRec instances. MiaRec will advertise this ip-address to the phone platform in SDP media description info (ip-address and port on which it expects to receive RTP packets from the phone platform).

Navigate in MiaRec web portal to **Administration -> System -> Recording Interfaces -> SIPREC -> Configure**.

Configure the Elastic IP address in each of two MiaRec instances. See below screenshot for details:



Configure SIPREC recording interface in your phone platform

Refer to the corresponding documentation for your phone platform.

References:

- [BroadWorks SIPREC configuration]
- [Metaswitch SIPREC configuration]

Test recording

Make some test calls and locate the recordings in MiaRec web portal.

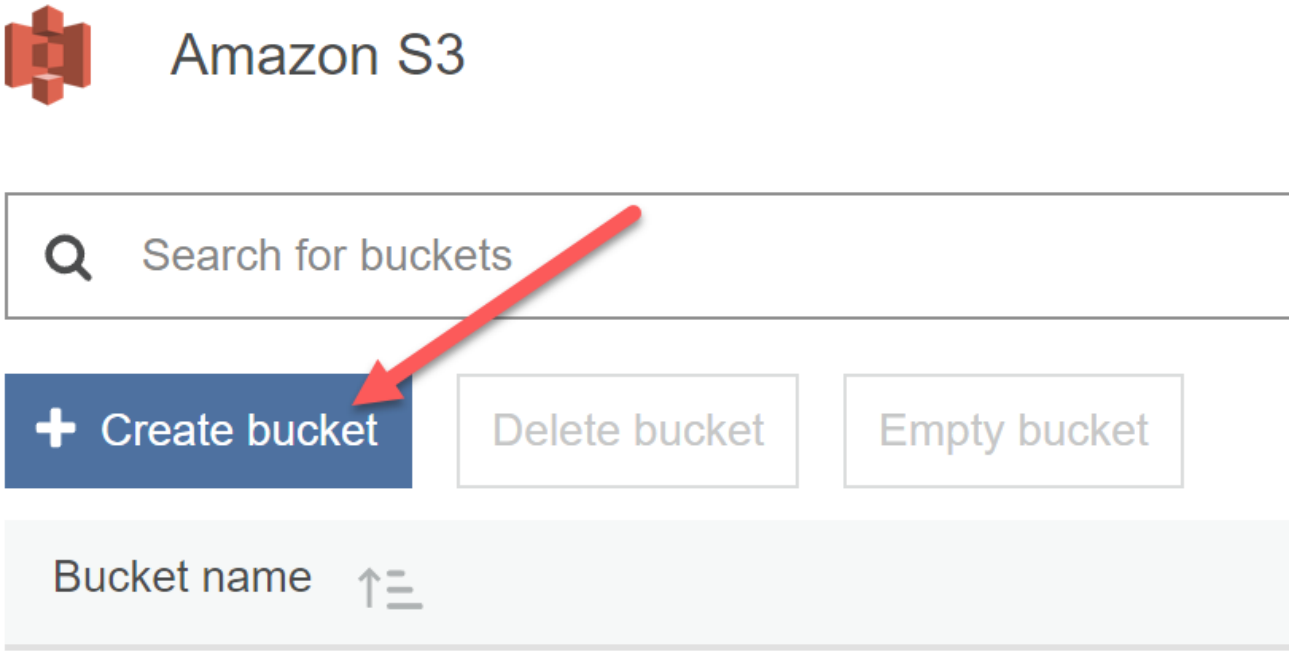
Test SIPREC failover

Simulate failure on the primary server (shutdown instance or stop "miarec" service) and verify if recording is switched over to the secondary instance.

4.3.9 9. Configure automatic file relocation to Amazon S3

1. Create an S3 bucket

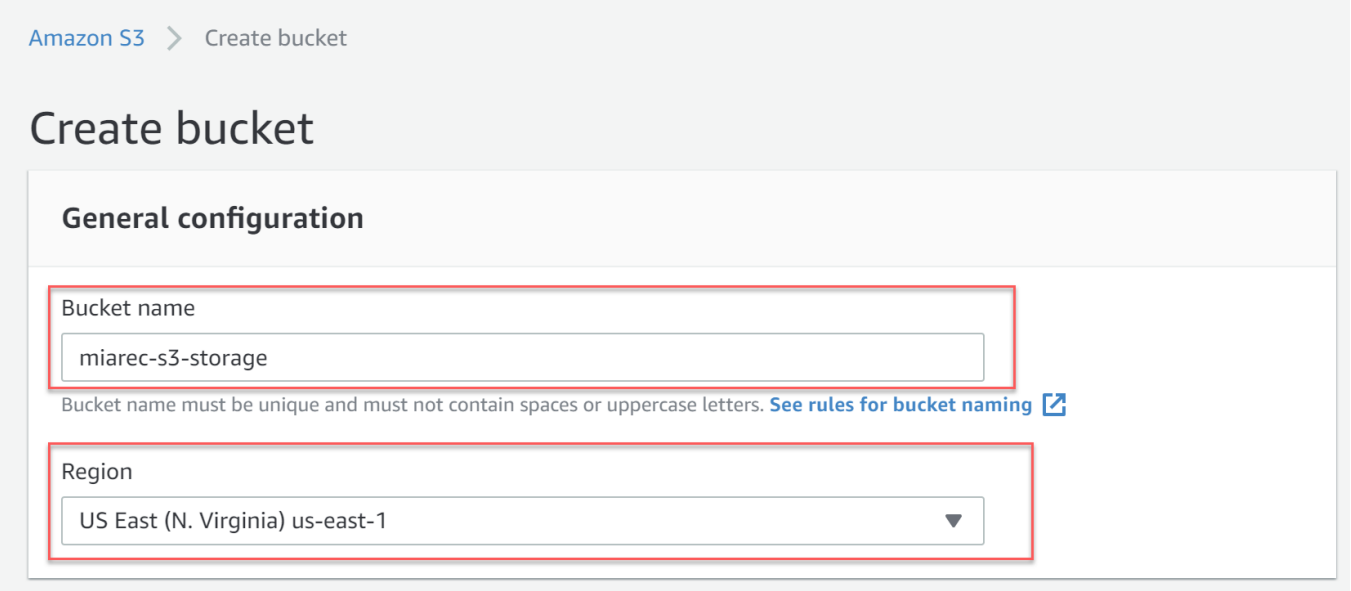
1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.



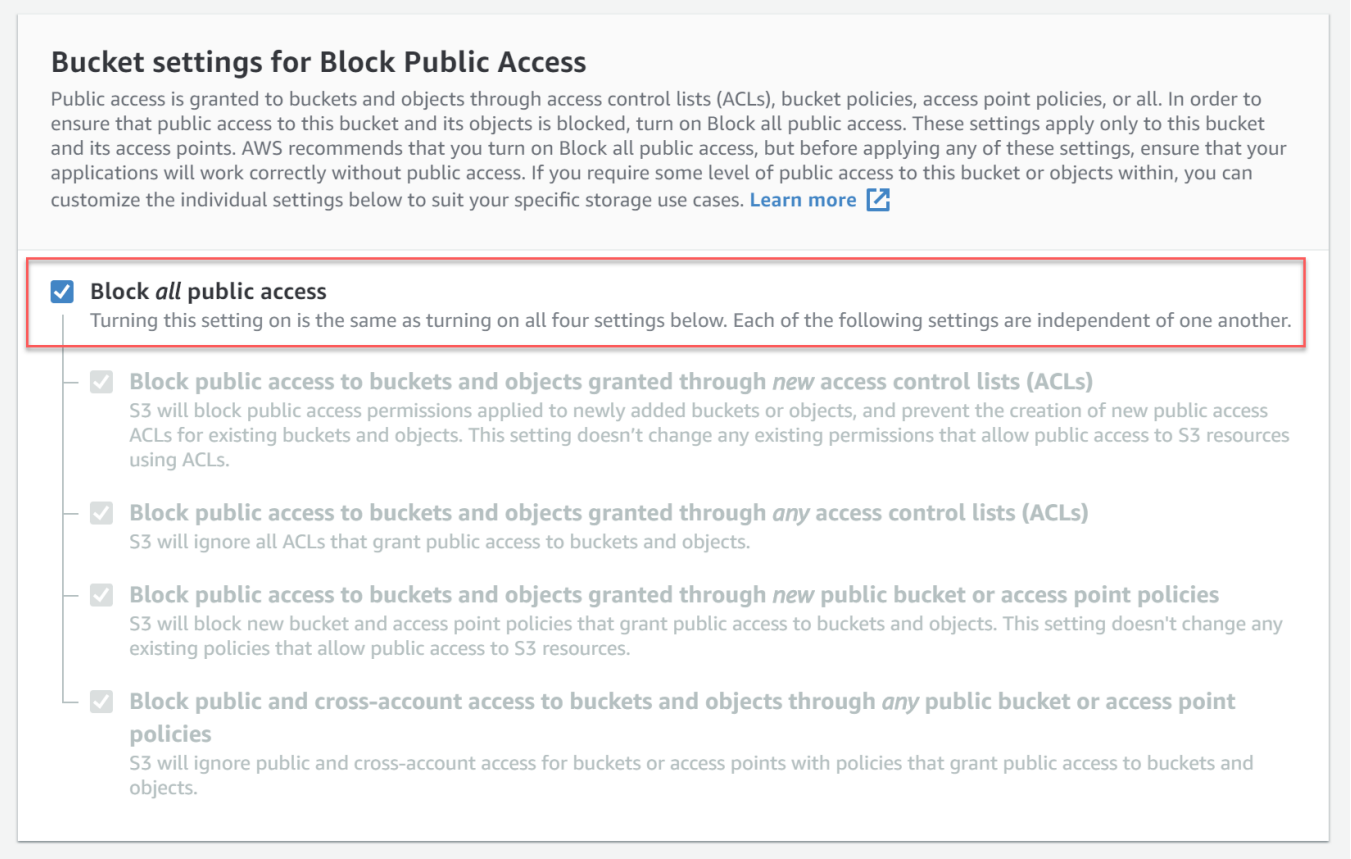
3. In the Bucket name field, type a unique DNS-compliant name for your new bucket. (The example screen shot uses the bucket name **miarec-s3-storage**. You cannot use this name because each S3 bucket names must be unique.) Create your own bucket name using the follow naming guidelines:
   - The name must be unique across all existing bucket names in Amazon S3.
   - After you create the bucket you cannot change the name, so choose wisely.
   - Choose a bucket name that reflects the objects in the bucket because the bucket name is visible in the URL that points to the objects that you're going to put in your bucket.

For information about naming buckets, see Rules for Bucket Naming in the Amazon Simple Storage Service Developer Guide.

1. For Region, choose the region where you want the bucket to reside. It is recommended to chose the region that is closest to your end-users. This will provide them the best performance during playback.



2. In the section **Bucket settings for Block Public Access**, make sure that public access is blocked.

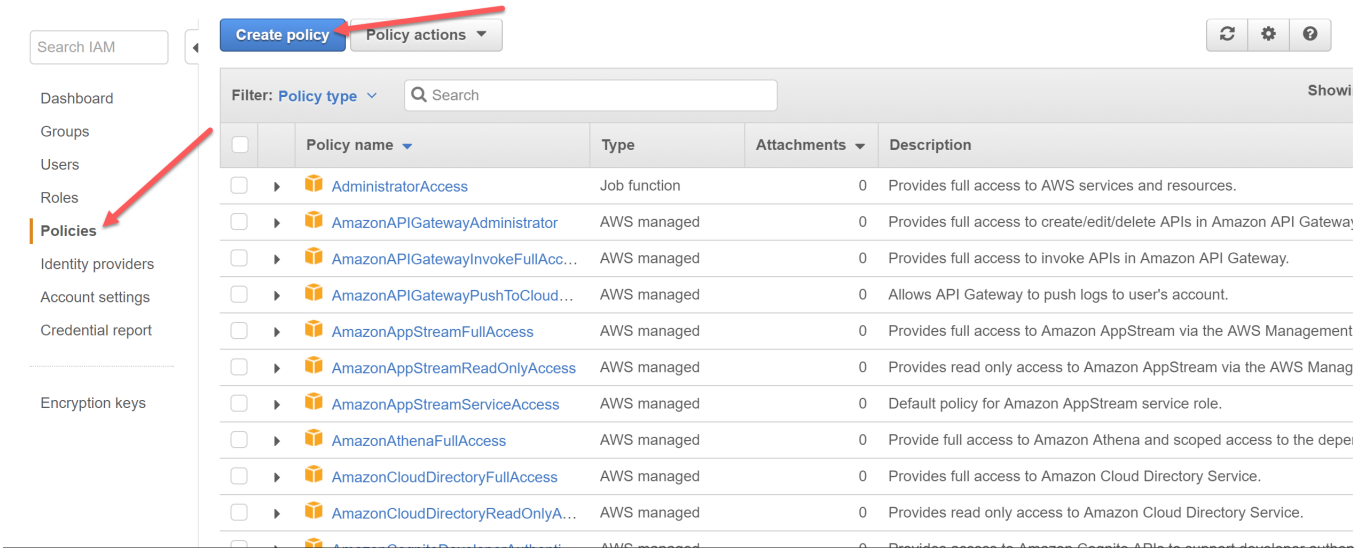


3. Click **Create bucket** in the last screen.

2. Create policy that grants access to the S3 bucket

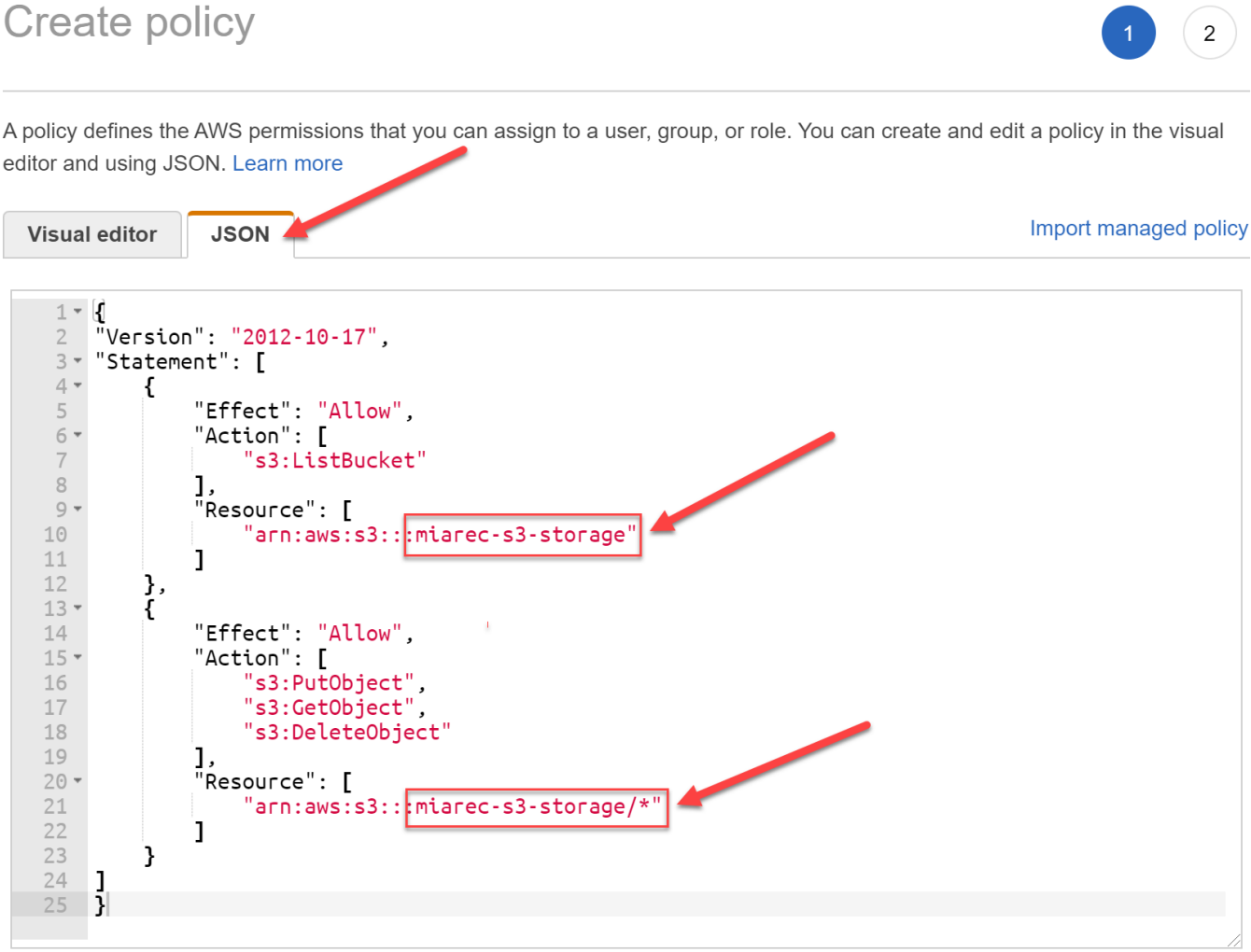
1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, click **Policies** and then click **Create Policy**.

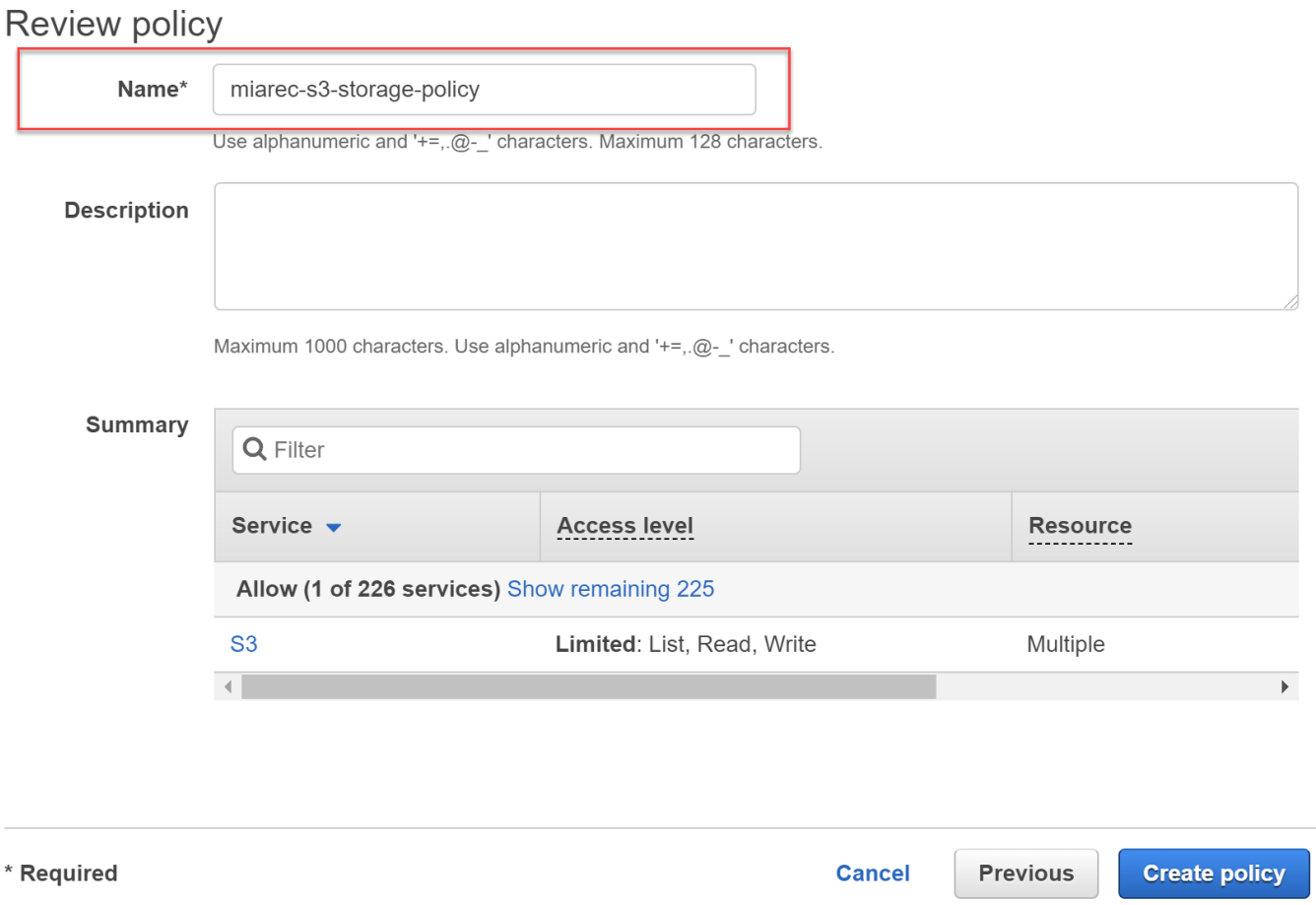


3. Select **JSON** tab, copy the following access policy and paste it into the **JSON** field. **Do not forget to replace `miarec-s3-storage` with your bucket name!!!**.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-s3-storage"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws:s3:::miarec-s3-storage/*"
        ]
    }
]
}
```

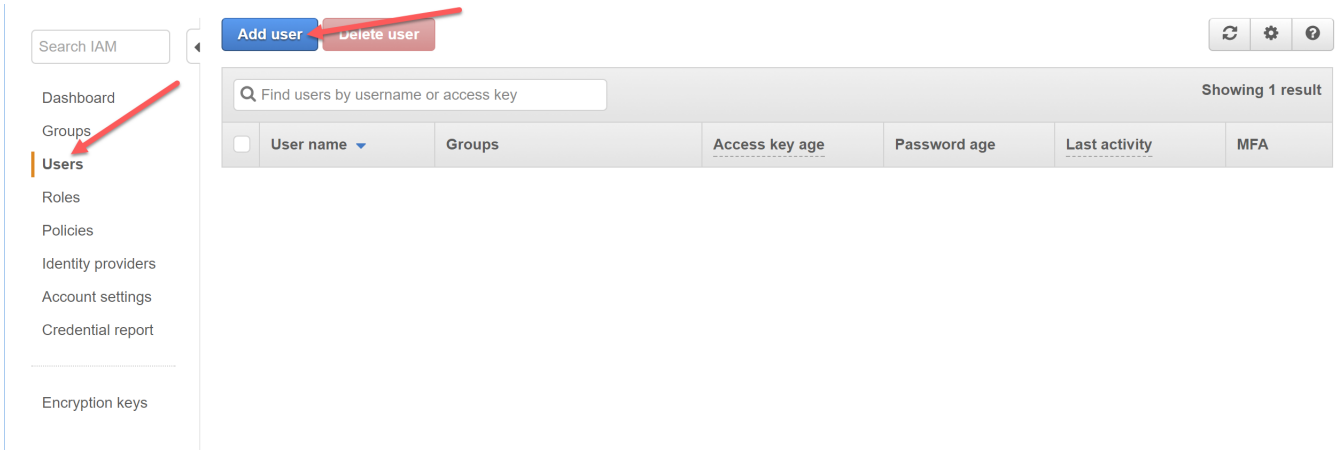


In the step **Review policy**, choose a descriptive name for the policy and click **Create policy** button.

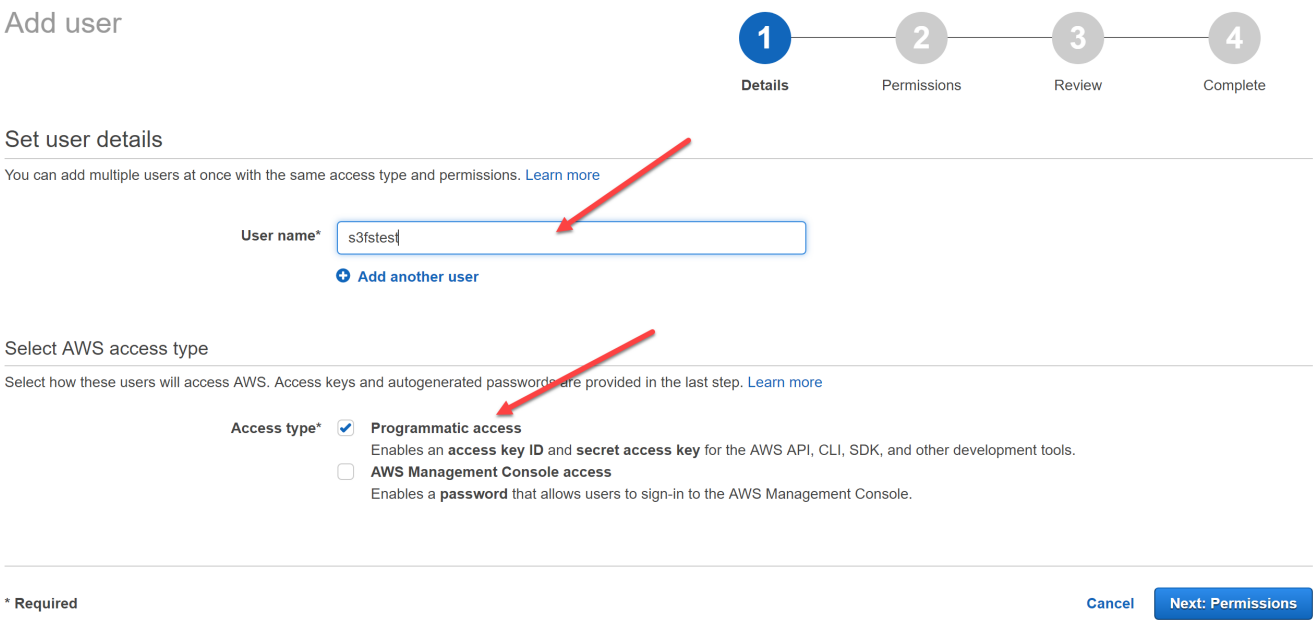


3. Create IAM user

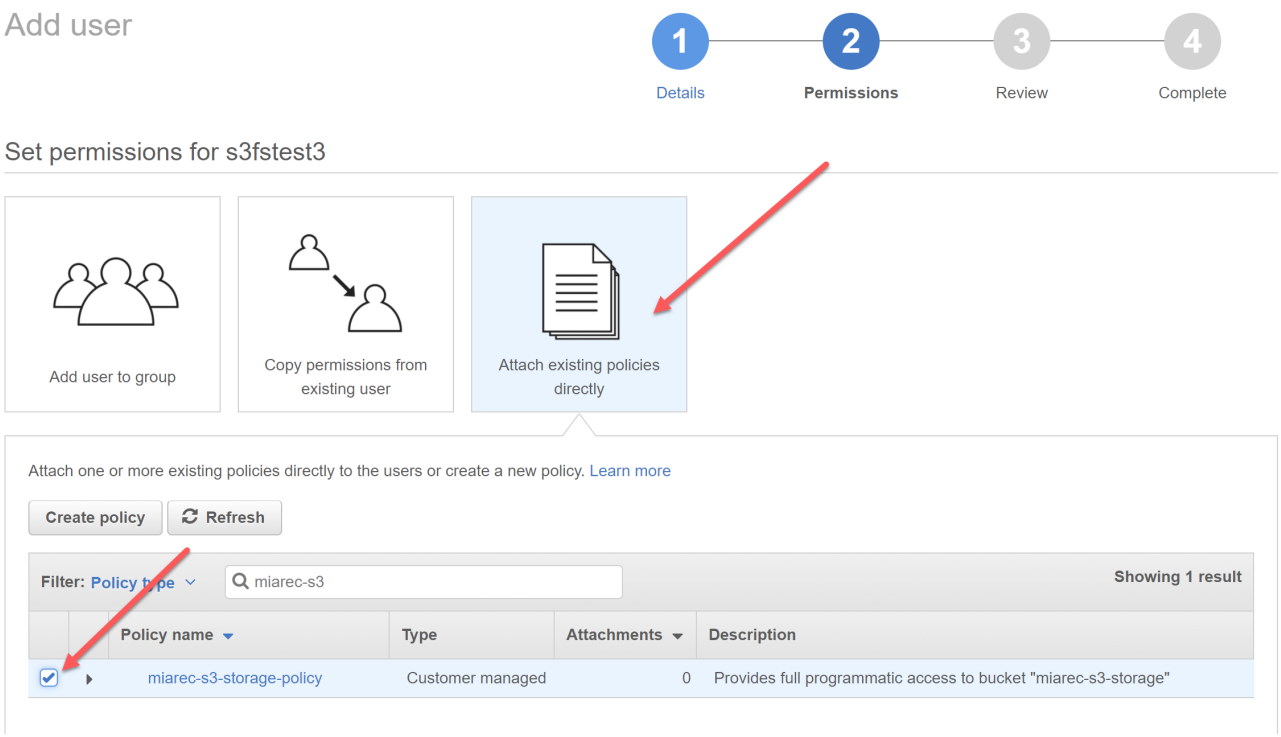
1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane on the left, click **Users** and then click **Add user**.



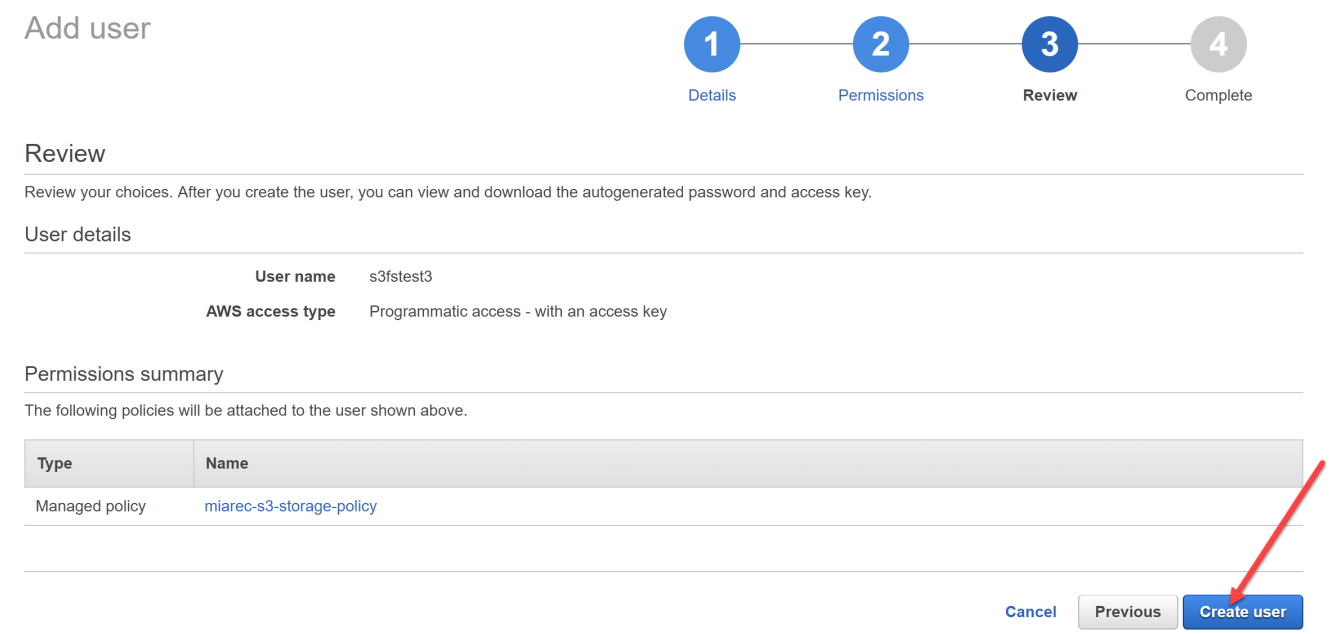
3. On **Details** screen, choose **User name** and enable **Programmatic access**.



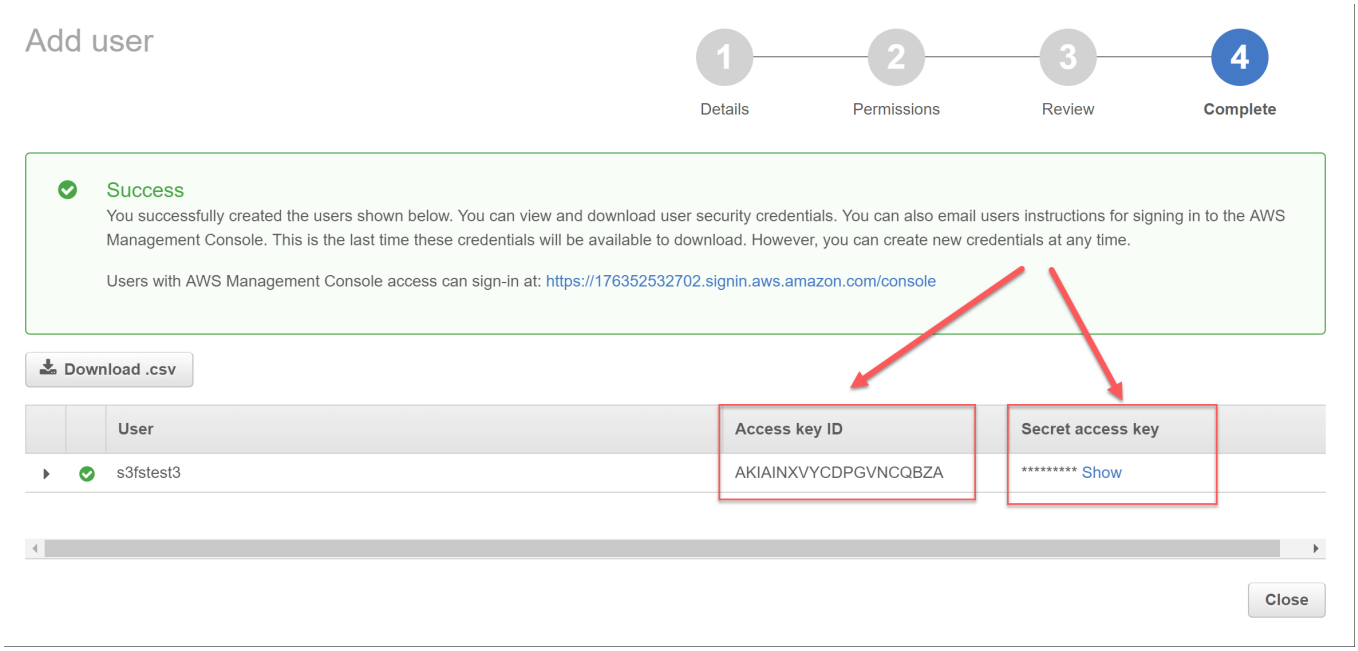
4. On **Permissions** screen, select **Attach existing policies directly** and then select the previously created policy from the list. Use the search box to find the policy by name.



5. Review the settings and click **Create user**.



6. On **Complete** screen, copy **Access Key ID** and **Secret access Key** and store them in secure place. We will use it for configuring storage target in MiaRec.

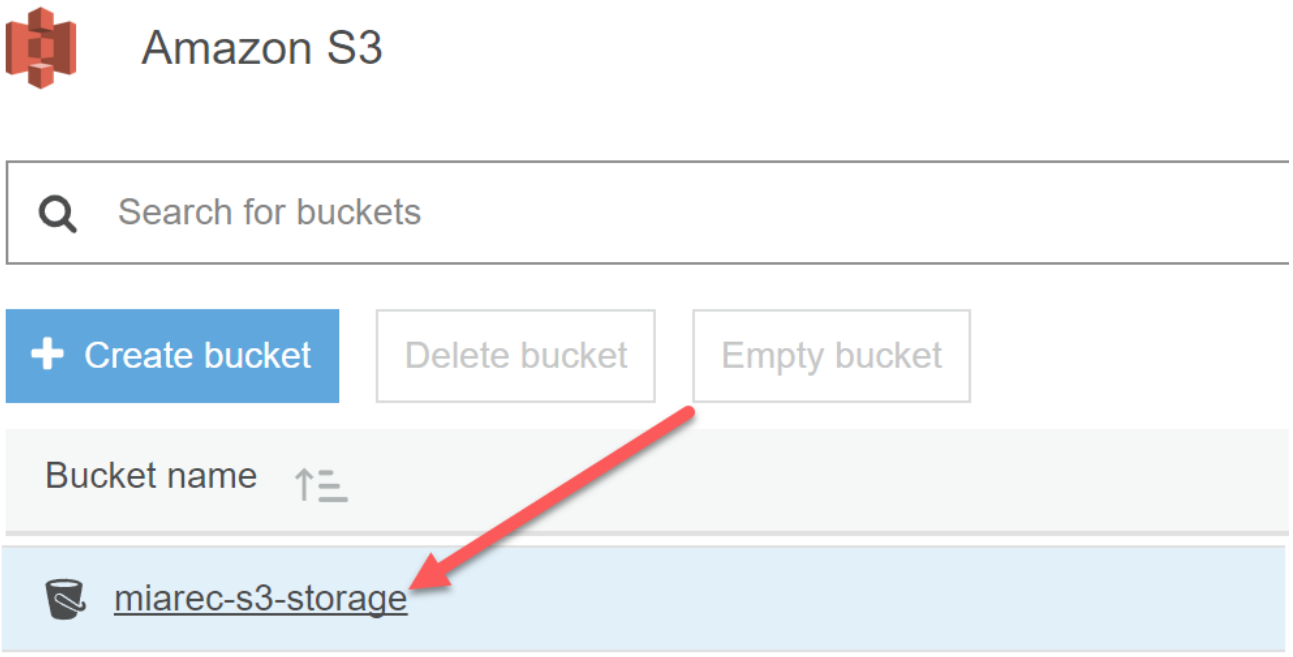


4. Add Cross-Origin Resource Sharing (CORS) configuration to an S3 bucket

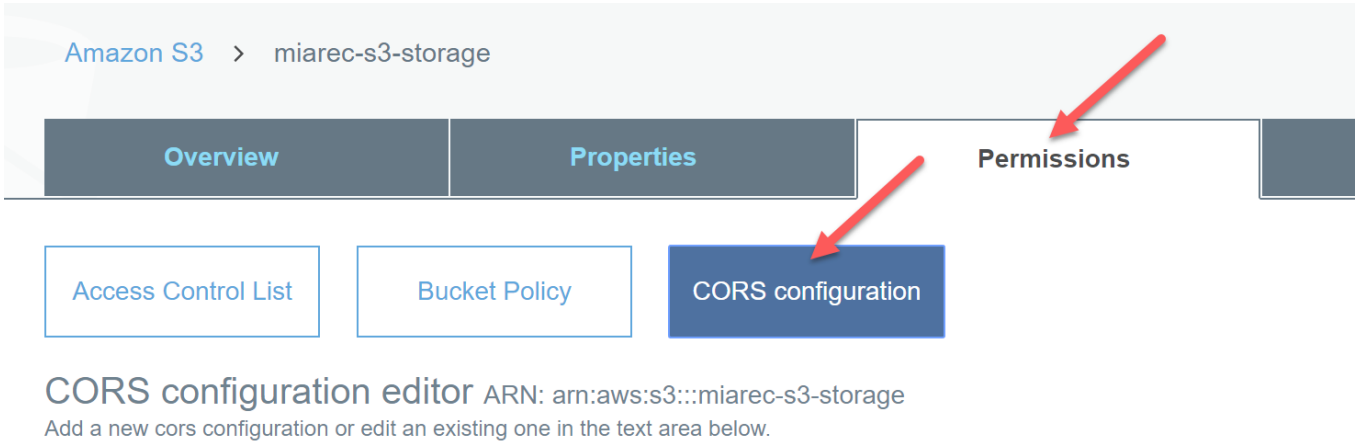
Cross-Origin Resource Sharing (CORS) allows client web applications that are loaded in one domain to interact with resources in another domain. This configuration is required for our setup because MiaRec web application is accessible using one domain (for example, https://recorder.example.com), but audio files are located at Amazon S3 domain (https://s3.amazonaws.com)

To add a CORS configuration to an S3 bucket:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to create a bucket policy for.

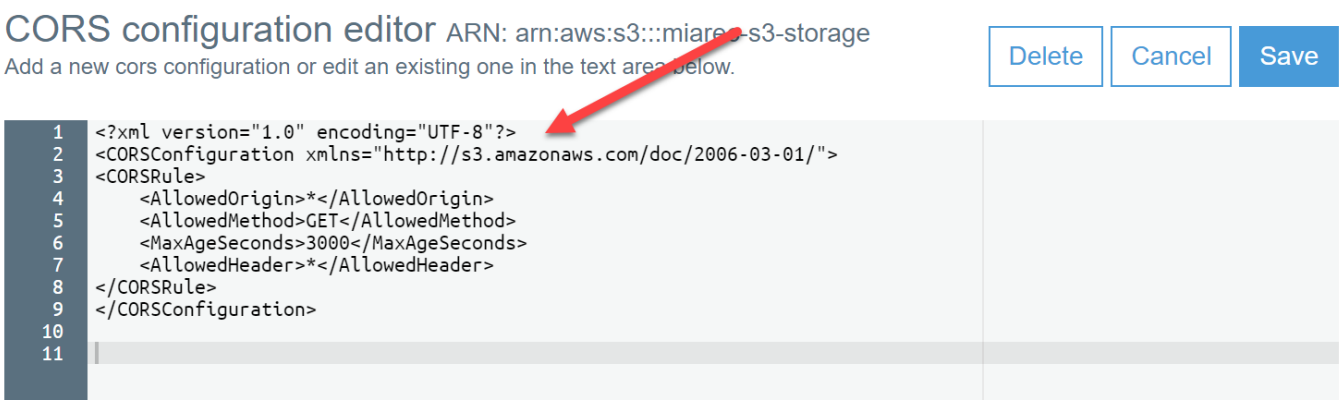


3. Choose **Permissions**, and then choose **CORS configuration**.



4. Copy the following CORS configuration and paste it into the **CORS configuration editor** field:

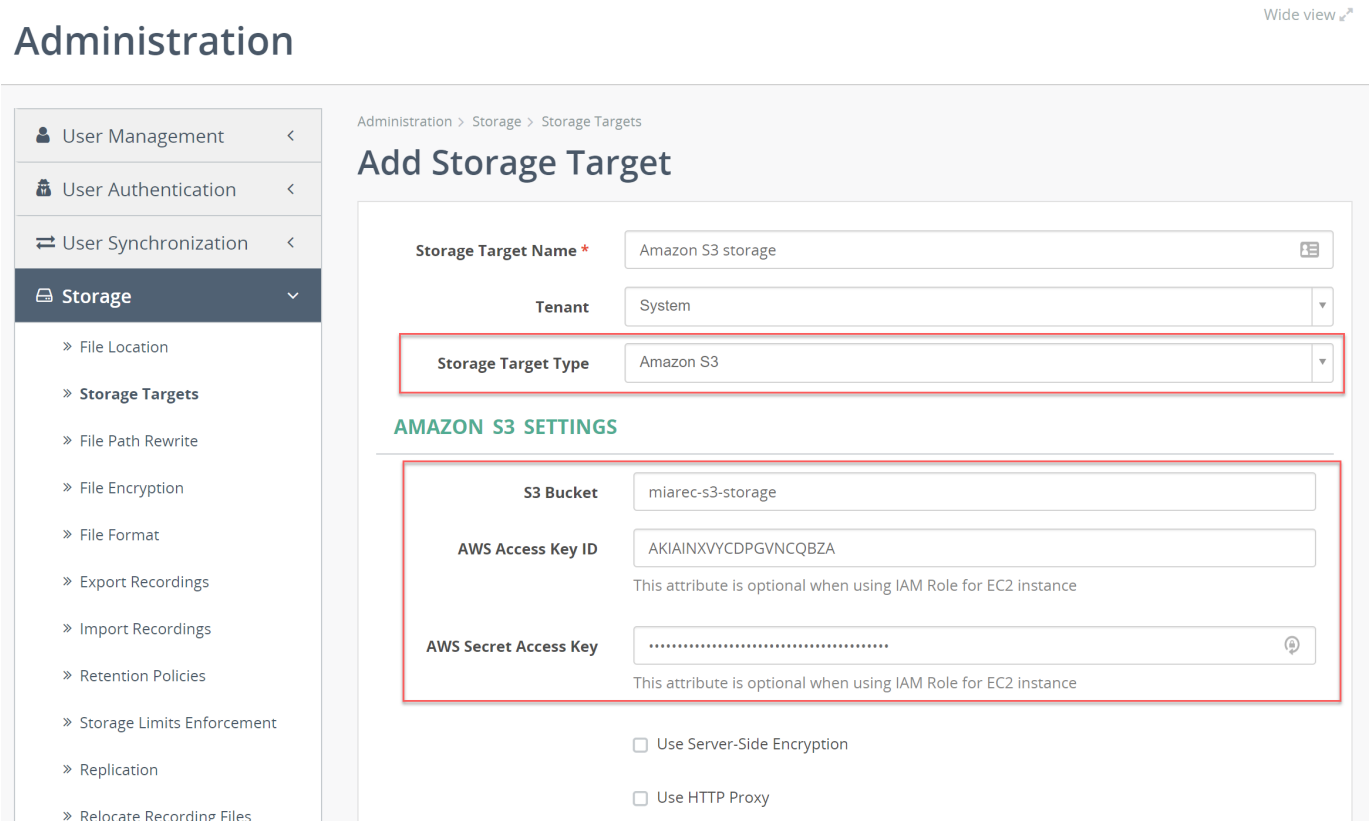
```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 3000
    }
]
```



Choose **Save**.

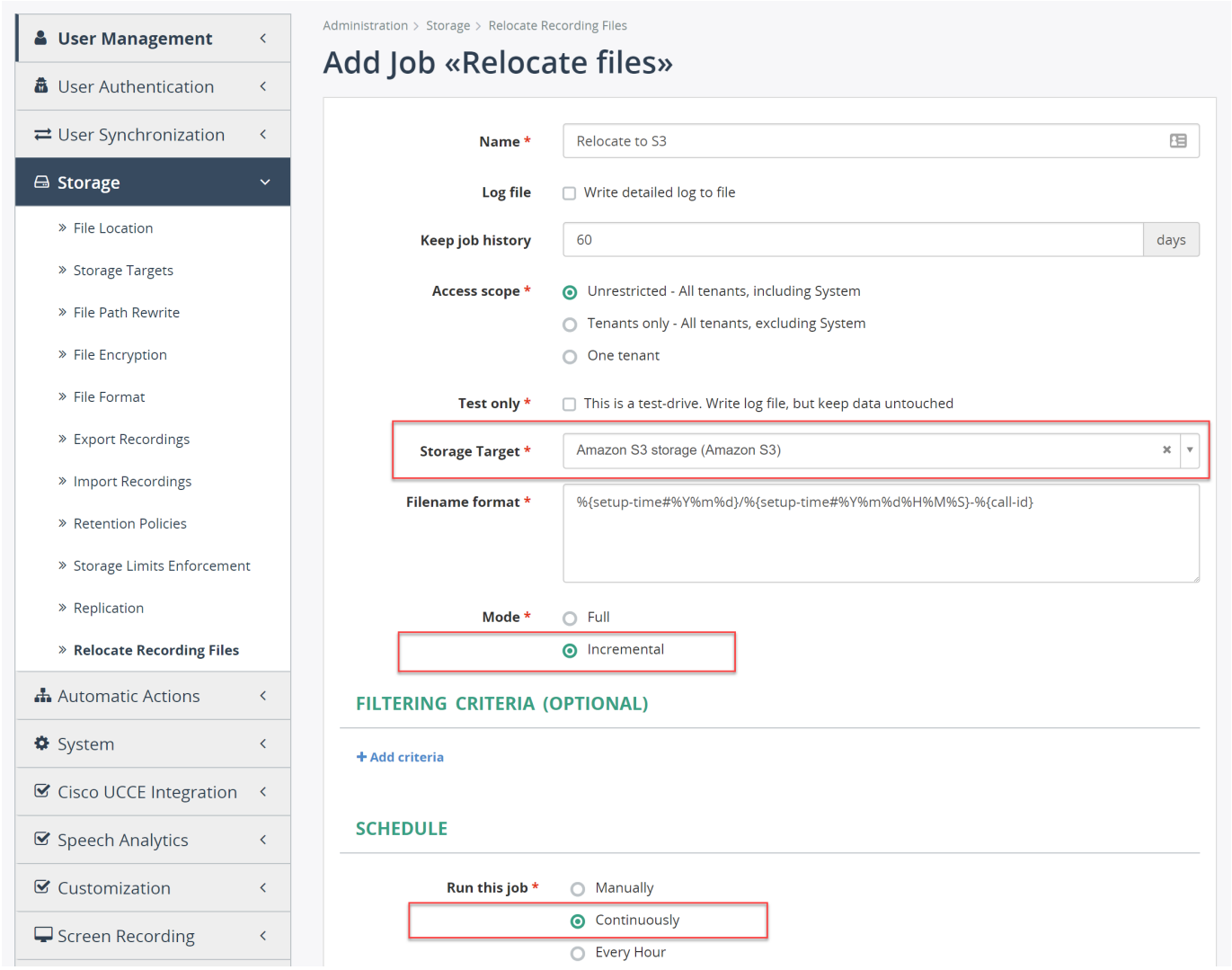
5. Configure Storage Target in MiaRec

1. Navigate in MiaRec web portal to **Administration -> Storage -> Storage Targets** and choose **Add**.
2. Select **Amazon S3** in **Storage Target Type**. Configure **S3 Bucket**, **AWS Access Key ID** and **AWS Secret Access Key** accordingly (as configured in the previous steps).

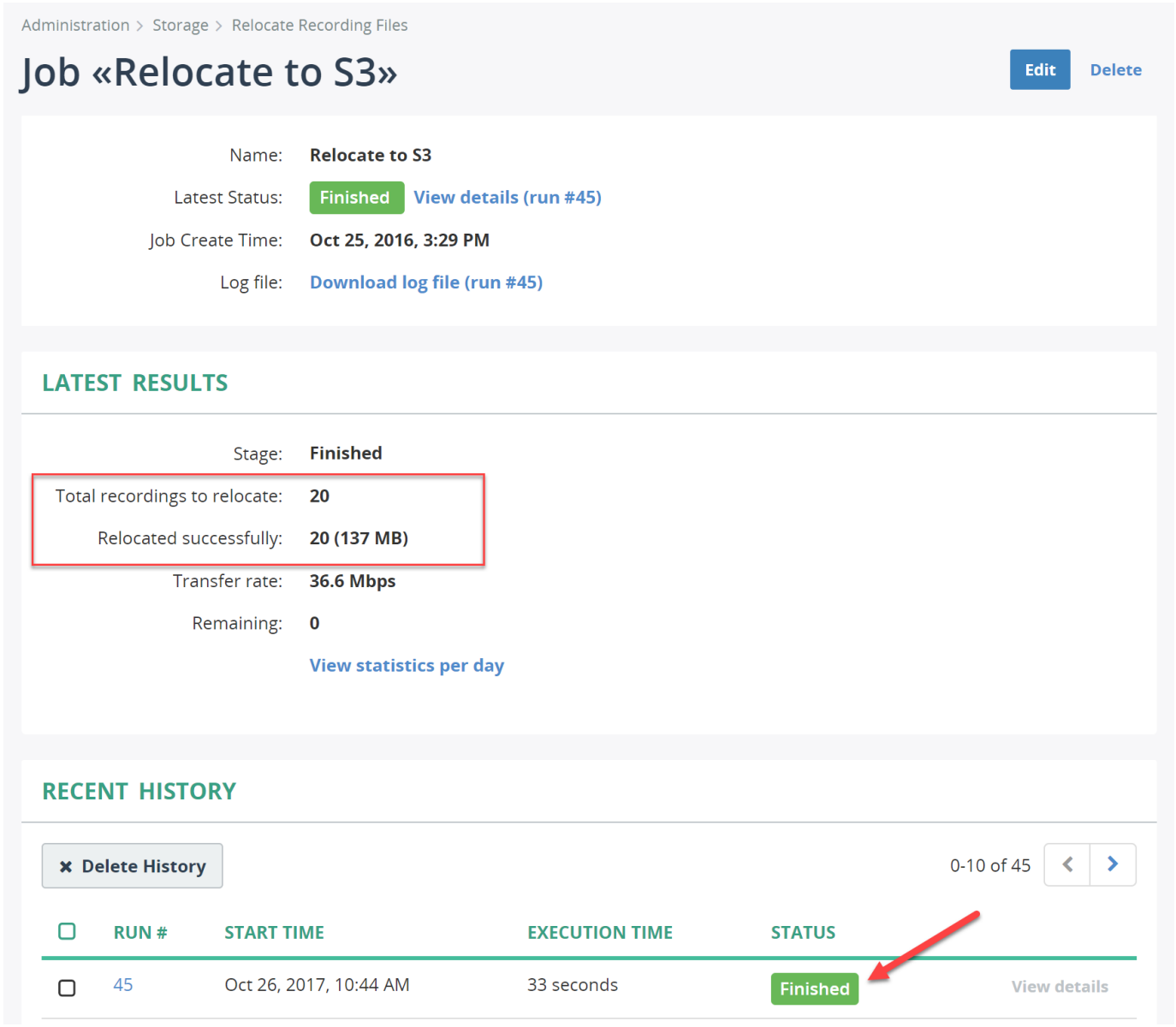


6. Configure automatic file relocation to S3 storate target

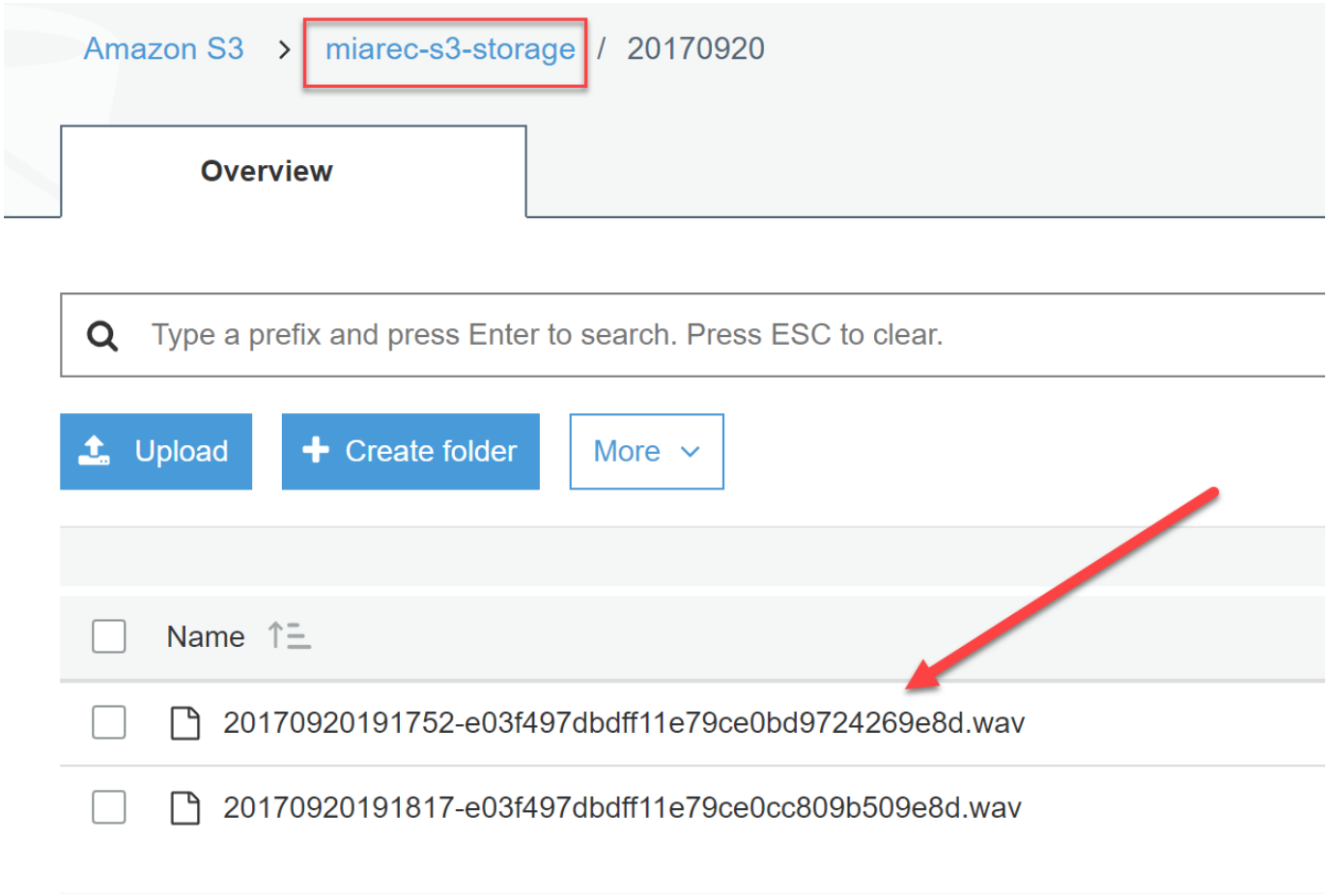
1. Navigate in MiaRec web portal to **Administration -> Storage -> Relocate Recording files** and choose **Add job**.
2. Configure **Storage Target**. Change **Mode** to **Incremental**. Select scheduler setting **Run this job** to **Custom (crontab)** and schedule it to run every 15 minutes by using `*/15` in the **Minute** attribute.



Make a few test calls and check status of this job. It is expected that files are automatically relocated to S3.



Navigate to Amazon S3 console and verify that files are located there:



4.3.10 10. Configure MiaRec replication

Documentation is in progress... Please, check later or contact our Support Team anytime.

4.3.11 11. Configure HTTPS for web server

Documentation is in progress... Please, check later or contact our Support Team anytime.

4.3.12 12. Configure CloudWatch monitoring

Documentation is in progress... Please, check later or contact our Support Team anytime.

4.3.13 13. Disaster recovery plan

Documentation is in progress... Please, check later or contact our Support Team anytime.

4.4 Installation on Windows

Installer for Windows operating system can be obtained from download page. The installer includes all required software (recorder, database, web server etc).

4.4.1 Step 1. Install all Windows updates

Before installing MiaRec, it is important to install the latest updates to Windows.

- Go to **Start - Control Panel - Windows Update**
- Check for the updates.
- Install all updates.
- Restart your system.

4.4.2 Step 2. Install KB2999226 update on Windows 7, 8, 2008, 2012

- Download update from https://support.microsoft.com/en-us/kb/2999226
- Install KB2999226 update

  If the installation of Windows update fails with error **"The update is not applicable to your computer"**, then some of the prerequisite updates are missing (see details here). On Windows 8.1 or Windows Server 2012 R2, you need to install KB2919355 first. Then you can install KB2999226. * verify if KB2999226 update is installed. Run the following PowerShell command:

  ```
  get-hotfix KB2999226
  ```

  Expected result:

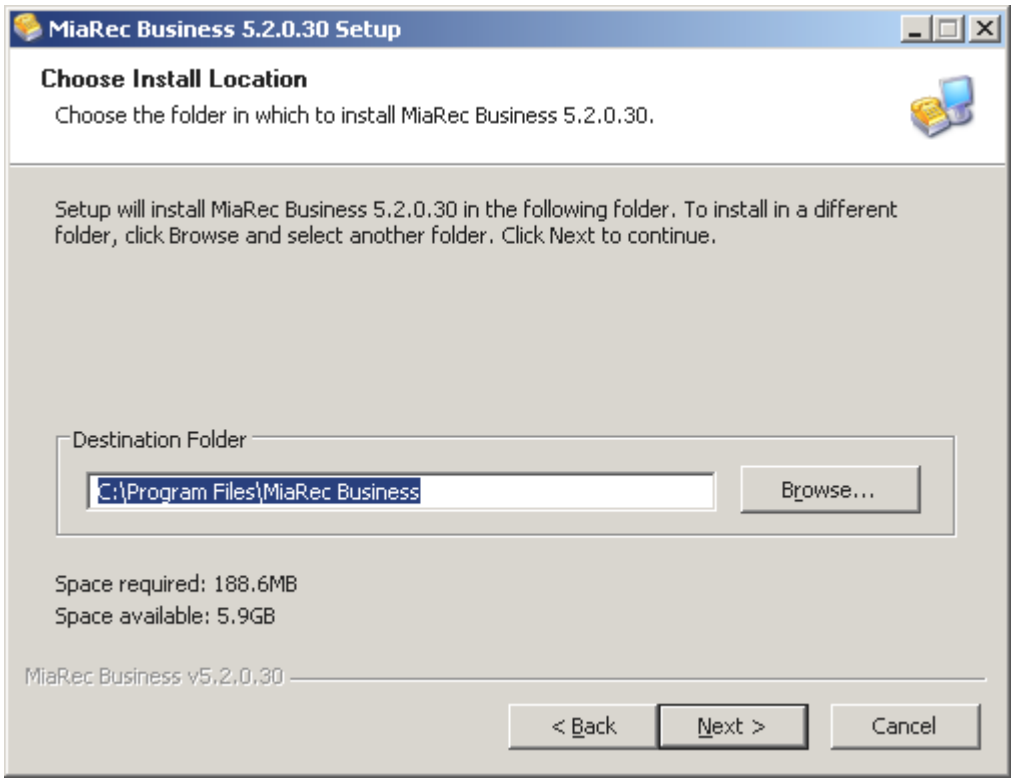
  ```
  PS C:\Users\Administrator> get-hotfix KB2999226

  Source        Description      HotFixID      InstalledBy          InstalledOn
  ------        -----------      --------      -----------          -----------
  WIN-UU4U5V... Update           KB2999226     WIN-UU4U5VSGI1L\A... 12/4/2017 12:00:00 AM
  ```

4.4.3 Step 3. Start MiaRec installer

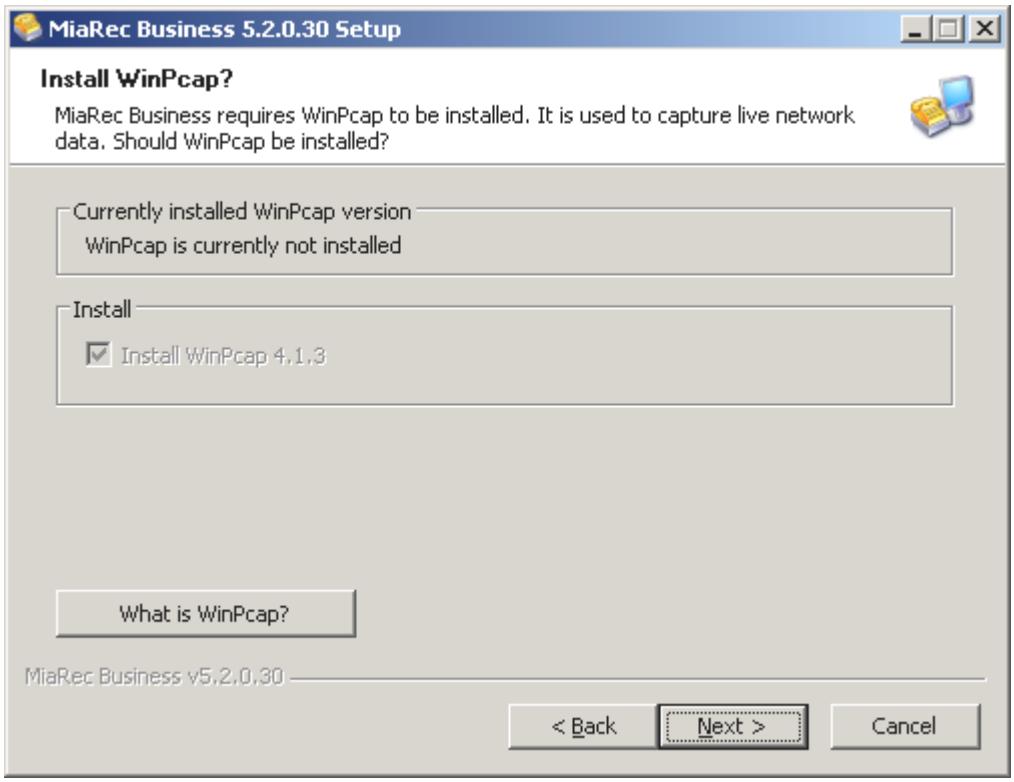
4.4.4 Step 4. Select destination folder

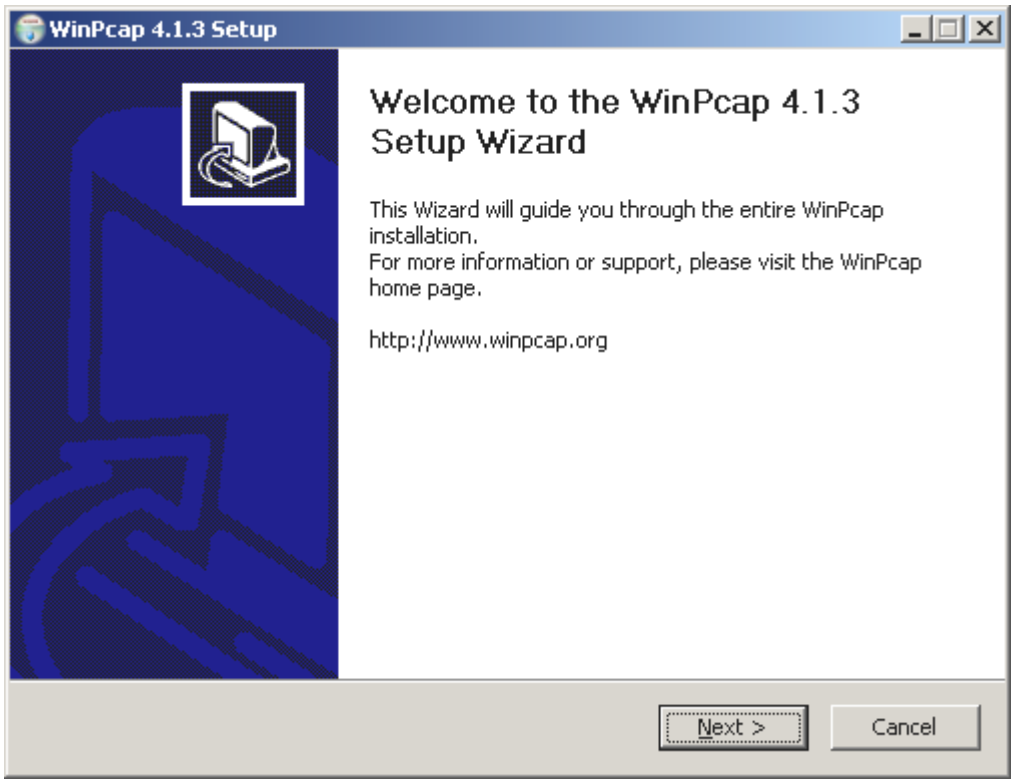
By default MiaRec is installed into **C:\Program Files** (on 32-bit system) or **C:\Program Files (x86)** (on 64-bit system). You can select different location for MiaRec files.



4.4.5 Step 5. Install WinPcap

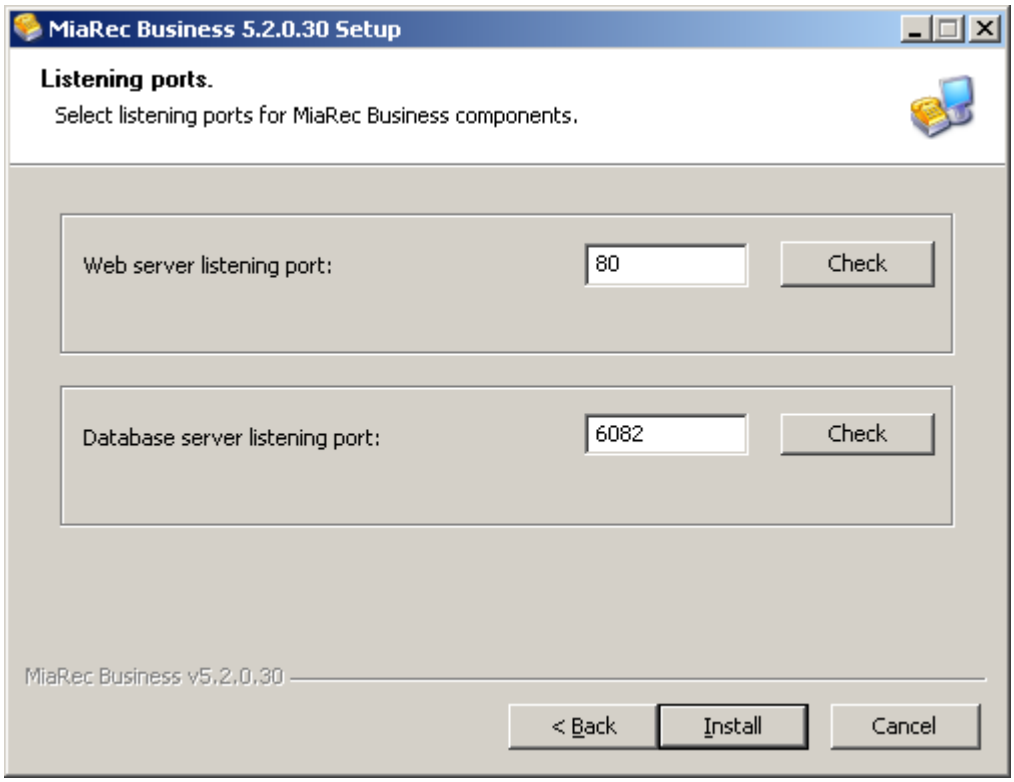
MiaRec requires WinPcap network driver. If it is not available on your system, then install it now.



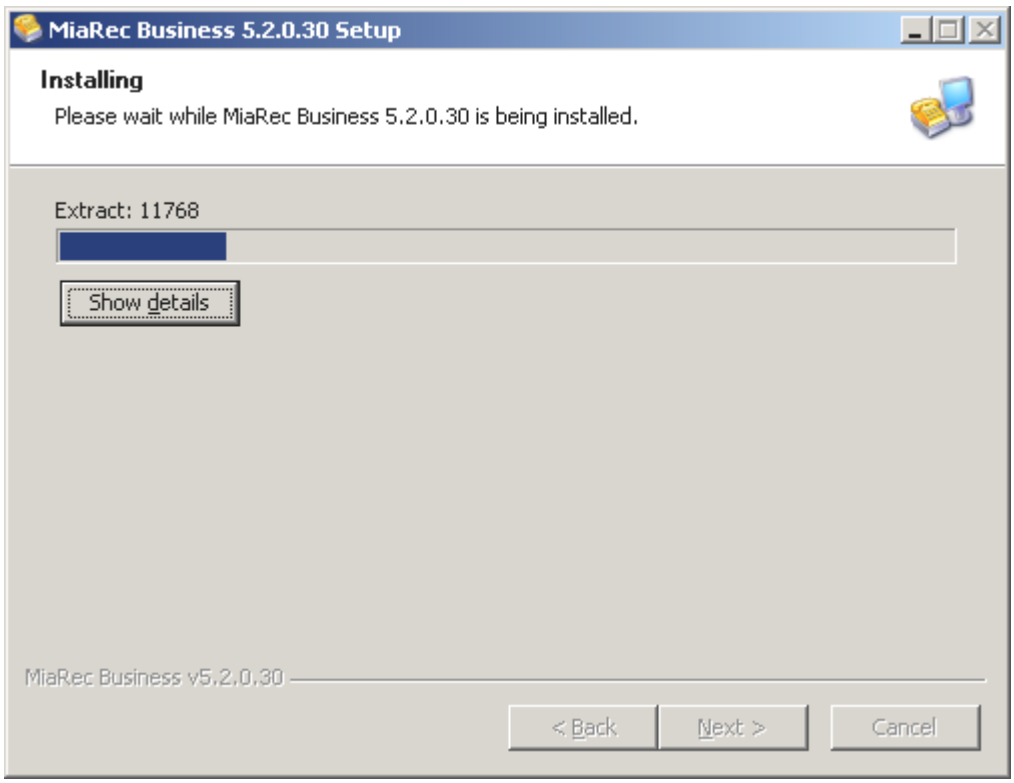


4.4.6 Step 6. Select ports for Web server and database

If port 80 is busy by other application (for example, IIS), then select another port, for example, 8080.

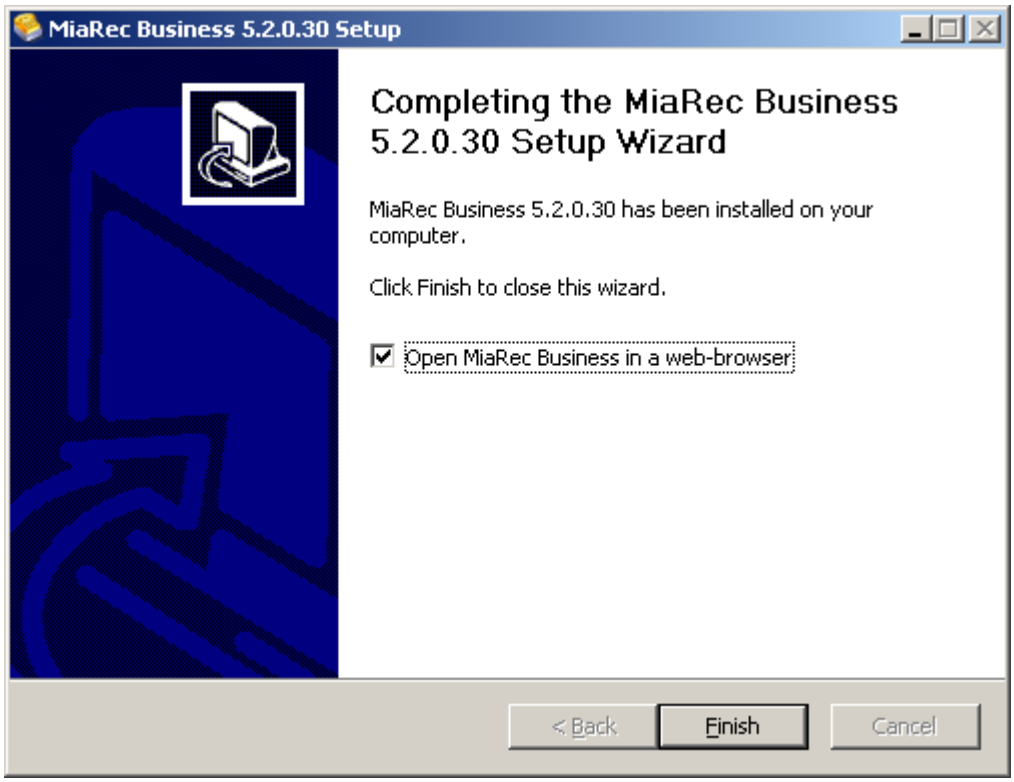


4.4.7 Step 7. Proceed to installation process



4.4.8 Step 8. Open MiaRec web portal in browser

When installation finished, open MiaRec web portal in browser.



5. Update

5.1 Ansible-based update on Linux

5.1.1 1. Update MiaRec playbooks

From time to time, we update playbooks for MiaRec installation/update. It doesn't occur with every software release, but sometimes we introduce new features to our product, that require update of installation scripts as well.

The MiaRec Ansible playbooks are hosted on public GitHub repository. To see if there were any changes to the playbooks, you can check the commit history.

Note, it will not harm to execute below commands even if there were no any changes to playbooks.

To load the latest version of playbooks, execute the following command on the Ansible controller machine:

```
cd /opt/ansible-miarec
git pull
git submodule update -i --recursive
```

Explanation:

- `git pull` command will load the latest version of the top project
- `git submodule update ...` command will load the latest version of the sub-projects (submodules).

> If you see the error **"Your local changes to the following files would be overwritten by merge"**, then some of local files have been edited manually on your server and the same files have been updated in the MiaRec github repository. You can run `git diff` command to see exactly what changes were made to the local files. To revert changes, you can execute command `git checkout -- FILE_NAME`. Where, **FILE_NAME** is the name of the file to revert changes.
>
> Then try again to pull the latest version from the MiaRec repository.

5.1.2 2. Update to Python 3.11.x

Navigate in MiaRec UI to **Administration -> Maintenance -> Version**. If a version of Python is older than 3.11.x, then do the following to update it.

Note, you do not need to update python if the currently installed version is already 3.11.x and a difference is only in the last part of a version like 3.11.1 and 3.11.7.

On the Ansible controller machine, in the inventory file `/opt/ansible-miarec/hosts`, change a version of Python to:

```
python_version      = 3.11.7
```

Then run the `prepare-hosts` playbook to install python:

```
cd /opt/ansible-miarec
ansible-playbook -i hosts prepare-hosts.yml -t python
```

This playbook will install Python 3.11.7 on the server.

Confirm the satisfactory completion with zero items `unreachable` or `failed`:

```
PLAY RECAP *********************************************************************
...
miarec               :  ok=38   changed=8   unreachable=0    failed=0
```

5.1.3 3. Update the MiaRec version info in the inventory file

On the Ansible controller machine, edit the version info in the inventory file `/opt/ansible-miarec/hosts`.

Example of this file:

```
[all:vars]
; ------------------------------
; Version of installed packages
; ------------------------------
miarecweb_version   = x.x.x.x
miarec_version      = y.y.y.y
miarec_screen_version = z.z.z.z
miarec_livemon_version = w.w.w.w
```

Contact your MiaRec representative to receive the latest version info.

5.1.4 4. Run playbook

To update MiaRec, run the following command:

```
cd /opt/ansible-miarec

ansible-playbook -i hosts setup-miarec.yml
```

When using password authentication, then add `--ask-pass` to the above command, like:

```
ansible-playbook -i hosts setup-miarec.yml --ask-pass
```

Confirm the satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP *********************************************************************
...
miarec               :  ok=38   changed=25   unreachable=0    failed=0
```

5.1.5 Resolve issue "Upgrade is stuck at the task Upgrade database layout"

If the upgrade process is stuck at the task "Upgrade database layout" for too long (more than 15 minutes), then do the following:

1. Terminate the upgrade process (Ctrl+C)
2. Stop gracefully all celery jobs (“gracefull” means that celery deamon will be stopped when all scheduled jobs complete their execution):

   ```
   service celeryd stop
   service celerybeat stop
   ```

   3. Stop Apache web server:

   ```
   service httpd stop
   ```

   4. Re-run Ansible upgrade playbook:

   ```
   ansible-playbook -i hosts setup-miarec.yml
   ```

   5. The Celery and Apache services should be started automatically as a part of Ansible upgrade process.

   ```
   service celeryd start
   service celerybeat start
   service httpd start
   ```

Why the upgrade process may stuck at the task "Upgrade database layout"?

We continuously add new features to our product. Some features require changes to database layout, like add column, table, etc. Some changes to database layout may require an exclusive lock on affected tables. The upgrade process may stuck waiting for a lock to be acquired. If other services are actively accessing database, it may take too long to acquire the lock.

Note, above instructions instruct how to stop web server and celery services only. The recording service doesn't have to be stopped. It continues to record calls even when web server and/or celery services are down. The recorder service also accesses the database, but it does not prevent acquiring a lock on table because recorder doesn't use transactions.

5.2 Migrate from manual to Ansible-based setup

This guide provide instructions for migration of manually-installed MiaRec software to Ansible-based setup.

Why migrate?

- It simplifies future updates. Ansible-based installation/update is a lot simpler. Basically, you can update software in one command.
- Manual installation is deprecated. New features will be supported in Ansible-based setup only as it is easier to maintain various distributives (Centos/RedHat/Ubuntu) and their versions using a single Ansible playbook. Manual installation is also error-prone as it requires manual copy/paste of many commands.

Note, this guide assumes all-in-one setup of MiaRec, i.e. all components (database, recorder, web server) are installed on the same host. Ansible playbooks will be run from the same host (although it is possible to run playbook from a remote host).

Installation overview

1. Install Python 2.7 (required for Centos 6)
2. Install Ansible
3. Download the MiaRec ansible playbooks
4. Check existing versions of PostgreSQL and Python
5. Create inventory file (hosts)
6. Run setup-miarec.yml playbook to update MiaRec software

5.2.1 1. Install Python 2.7 (required for Centos 6 only)

> This step is required for Centos 6 only as it has old version of Python 2.6. Skip this step if you are using Centos 7 or Ubuntu Server.

Verify if Python 2.7 is available on the server using the following commands:

```
$ which python2.7
/usr/local/bin/python2.7

$ python2.7 --version
Python 2.7.12
```

If Python 2.7 is installed (as shown above), then skip this step. Otherwise, install Python 2.7.

Centos 6 comes pre-installed with older version of Python (2.6). Ansible doesn't work on such old version.

We are not going to replace the older version of Python with newer. Instead, the newer version will be installed in parallel. A system-default Python will remain the same (if you call `python --version`, you still see version 2.6), but new version will be available by calling the exact name `python2.7 --version`.

Install the required packages for building python:

```
yum install zlib-devel bzip2-devel xz-devel openssl-devel sqlite-devel expat-devel
```

Download the latest stable Python 2.7 source code files from https://www.python.org/downloads/source/:

```
wget https://www.python.org/ftp/python/2.7.14/Python-2.7.14.tgz
```

Extract source code:

```
tar -xzvf Python-2.7.14.tgz
```

Build Python binaries:

```
cd Python-2.7.14
./configure --prefix=/usr/local --enable-shared LDFLAGS="-Wl,-rpath /usr/local/lib"
make
```

Install Python using alternative installation option (`altinstall`). Normally, one would use “make install”; however, in order not to override system defaults - replacing the Python already used by the system - we will use make altinstall.

```
make altinstall
```

This will install python into /usr/local/bin with name, which contains version, like `/usr/local/bin/python2.7` .

5.2.2 2. Install Ansible on Centos 6/7

Download PIP installer script and run it (PIP is a tool for installing Python packages. Ansible is written in Python):

```
wget https://bootstrap.pypa.io/get-pip.py
python2.7 get-pip.py
```

Install Ansible using PIP:

```
pip2.7 install ansible
```

Verify Ansible version:

```
ansible --version
```

The output should be something like:

```
$ ansible --version
ansible 2.3.1.0
  config file =
  configured module search path = Default w/o overrides
  python version = 2.7.14 (default, Nov 19 2016, 06:48:10) [GCC 5.4.0 20160609]
```

Verify that ansible version is 2.2+ or higher and python version is 2.7. If the python version shows 3.x then the installation of Ansible is not correct. Contact the MiaRec representative for support.

5.2.3 3. Install Ansible on Ubuntu

Update package source lists:

```
sudo apt-get update
```

Install PIP (a tool for installing Python packages. Ansible is written in Python):

```
sudo apt-get install python-dev python-pip
```

Install Ansible using PIP:

```
sudo pip install ansible
```

Verify Ansible version:

```
ansible --version
```

The output should be something like:

```
$ ansible --version
ansible 2.3.1.0
  config file =
  configured module search path = Default w/o overrides
  python version = 2.7.12 (default, Nov 19 2016, 06:48:10) [GCC 5.4.0 20160609]
```

Verify that ansible version is 2.2+ or higher and python version is 2.7. If the python version shows 3.x then the installation of Ansible is not correct. Contact the MiaRec representative for support.

5.2.4 4. Download the MiaRec ansible playbooks

Clone the latest stable release of the MiaRec-Ansible Git repository in the /opt/ansible-miarec directory:

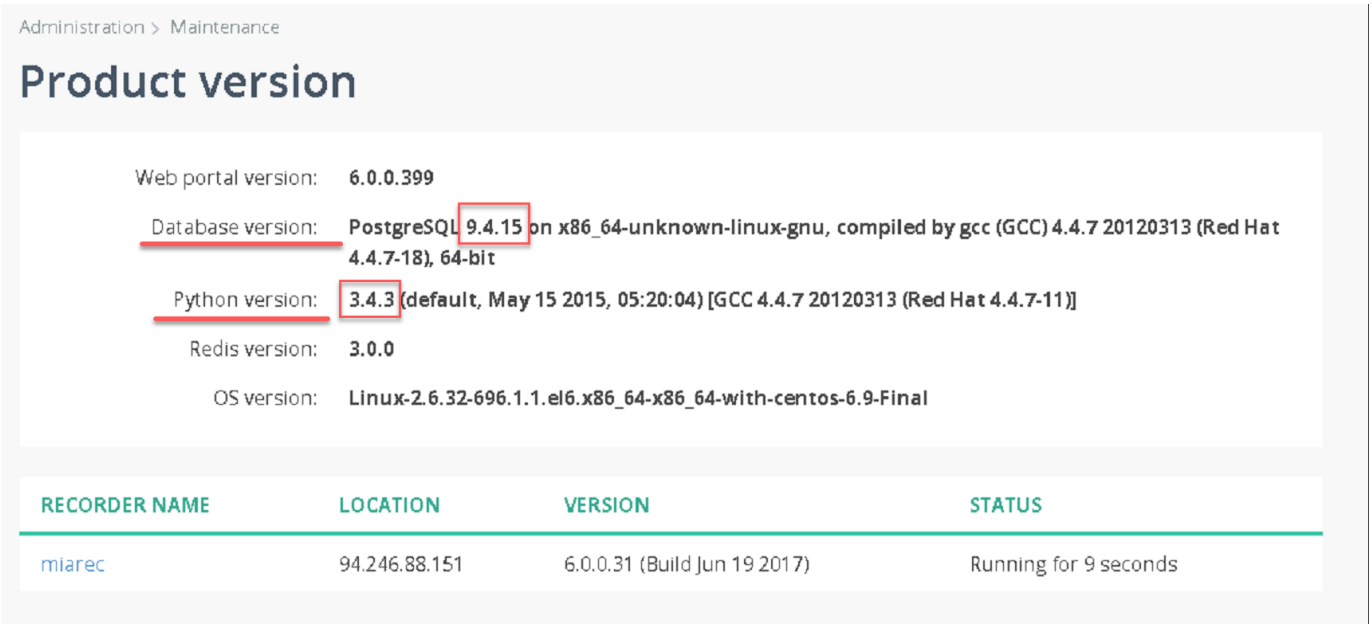
```
yum install git

git clone --recursive https://github.com/miarec/ansible-miarec /opt/ansible-miarec
```

If you receive `HTTP failed` error, then run `yum install nss` to update the root SSL certificates on your server.

5.2.5 5. Check existing versions of PostgreSQL and Python

Navigate in MiaRec web interface to **Administration -> Maintenance -> Version** and notice the versions of PostreSQL database and Python. In our example, PostgreSQL version is 9.4.8 and Python is 3.4.3.



Alternatively, you can execute the following commands in console:

```
$ psql --version
psql (PostgreSQL) 9.4.8


$ python3.4 --version
Python 3.4.3
```

Most likely, Python 3.4.3 has been installed manually on this server. Note, this Python version (3.4.3) is different from the Python version 2.7 discussed above for Ansible. MiaRec software requires 3.4+, but Ansible requires 2.7. It is ok to have multiple version of Python on the same machine as long as they are installed into different directories using `make altinstall` command.

5.2.6 6. Create inventory file (hosts)

The Ansible inventory file is an INI-formatted file that defines the hosts and groups of hosts upon which commands, modules, and tasks in playbooks operate (the Inventory File is highly configurable, see the Ansible documentation for more information). This guide assumes that there is only one host (all-in-one setup).

First, check the latest release information by contacting your MiaRec representative and edit the following variables in the **hosts** file (discussed later):

- `miarec\_version`
- `miarecweb\_version`
- `miarec\_screen\_version`

Create `/opt/ansible-miarec/hosts` file:

```
vim /opt/ansible-miarec/hosts
```

Copy/paste the following content for the `hosts` file. Make sure `postgresql_version` and `python_version` variables are set to the previously identified versions. Note, the PostgreSQL version should be specified in short format `x.x`, but Python should be specified in full format `x.x.x`.

```
[all]
; --------------------------------
; All-in-one host
; Parameters:
;   - private_ip_address => ip address to access the host from other components
; (for example, web application needs to connecto to database)
; --------------------------------

miarec ansible_connection=local private_ip_address=127.0.0.1


[all:vars]
; -------------------------------
; Version of installed packages
; -------------------------------
miarecweb_version   = 6.0.0.xxx
miarec_version      = 6.0.0.xxx
miarec_screen_version = 1.1.0.xxx
postgresql_version  = 9.4
python_version      = 3.4.3


[recorder]
miarec

[screen]
miarec

[db]
miarec

[redis]
miarec

[web]
miarec

[celery]
miarec

[celerybeat]
miarec
```

5.2.7 7. Run setup-miarec.yml playbook to update MiaRec software

The playbook setup-miarec.yml will install/update the MiaRec software components (recorder, web portal and screen recorder).

```
cd /opt/ansible-miarec

ansible-playbook setup-miarec.yml
```

Confirm satisfactory completion with zero items unreachable or failed:

```
PLAY RECAP ********************************************************************
...
miarec               :  ok=38   changed=25   unreachable=0    failed=0
```

5.2.8 Verify MiaRec operation

Verify MiaRec after update

5.3 Critical software update (Daylight saving time)

5.3.1 Issue description

On a night, when clocks are turned forward 1 hour, the application scheduler incorrectly interprets system time and starts the scheduled tasks/jobs in a loop.

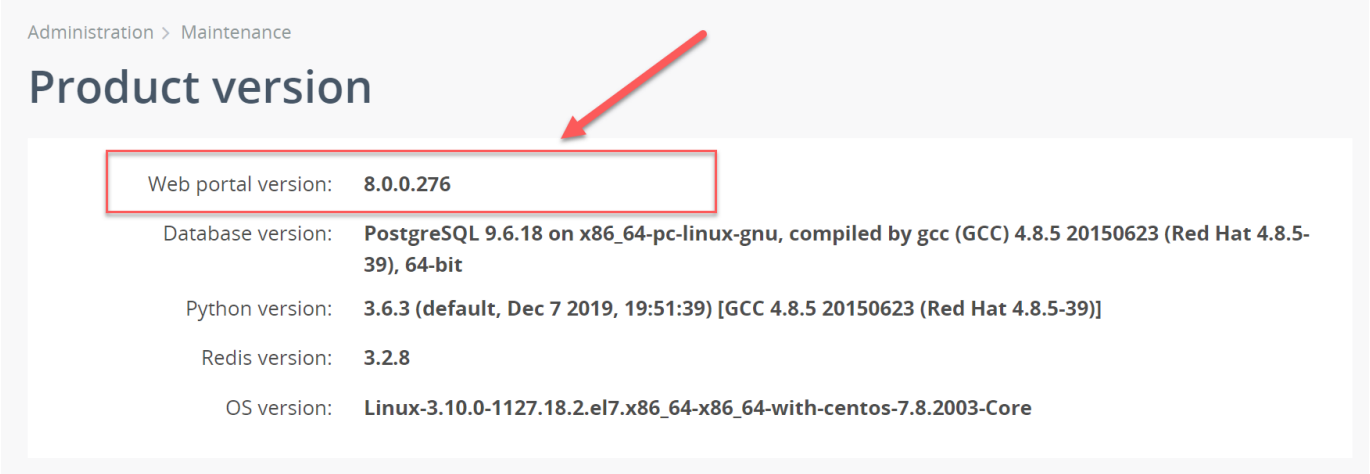
This causes excessive CPU usage and may cause "out-of-disk-space" issue due to grow of log file in size.

5.3.2 Affected versions

This bug existed in releases between:

- 6.0.0.1029 (released on August 20th, 2018) and
- 7.0.0.447 (released on March 27th, 2020).

To check a version of your MiaRec software, navigate in web portal to **Administration -> Maintenance -> Version**:



Important! The issue doesn't occur if operating system is configured with UTC timezone on system level.

To verify what timezone is configured on OS level, on Linux system, run the following command to verify system timezone:

```
date +"%Z %z"
```

If you see `UTC +0000` in the output, then your system is not affected by this bug.

You do not need to do anything if your system is not affected!

5.3.3 Mitigation

If your system is affected, then there are two options to mitigate the issue:

- Short-term solution
- Long-term solution

Option1. Short term solution

If you do not have time to apply a long term solution, then we recommend to do the following:

Before a night when clocks are turned forward 1 hour, stop the MiaRec's task scheduler service.

On Linux, run the following command via SSH:

```
service celerybeat stop
```

On Windows, navigate to **Control Panel -> Services** panel, and stop the service **MiaRec Celery Scheduler**.

After clocks are turned forward 1 hour, start the previously stopped service by running command `service celerybeat start` on Linux or by starting **MiaRec Celery Scheduler** service on Windows.

Explanation: The Celery Scheduler (celerybeat) service is responsible for triggering the scheduled tasks at the specified time (like every night, every hour, every 5 minutes etc). If this service is not running, then tasks are not run by schedule. It is safe to stop this service for a short period of time. Once it is started again, the scheduled tasks/jobs will catch-up and process the pending data. A "short period of time" could be hours or days depending on your system load. As a rule of thumb, if you record less than 500 users on the system, then you can safely stop the scheduler service for a few days, for example, before a weekend and then start on Monday.

Option 2. Long-term solution

For a long term solution, it is a necessary to update MiaRec software to the latest release. You can get the latest version by contacting your MiaRec representative.

5.3.4 Questions?

If you have any questions, contact our support team at support@miarec.com

6. Post-installation tasks

6.1 Firewall configuration

Open on the firewall the ports, which are used for accessing MiaRec from other computers on the network/Interenet

6.1.1 Open ports for MiaRec

MiaRec uses following ports, which should be opened on firewall:

Port	Description
80 (TCP)	MiaRec Web-portal (HTTP protocol) It is possible to change this port to other value during installation (for example, to 8080).
443 (TCP)	MiaRec Web-portal (HTTPS protocol)
6554 (TCP)	Live monitoring (RTSP signaling). If live monitoring is not used, then this port can be closed on firewall.
7000 - 7999 (UDP)	Live monitoring (RTP audio). If live monitoring is not used, then these ports can be closed on firewall.
5070 (TCP)	Cisco SIP trunk recording signaling (SIP protocol) - for Cisco UCM only
20000 - 21999 (UDP)	Cisco SIP trunk recording media (RTP protocol) - for Cisco UCM only
5080 (TCP, UDP)	SIPREC recording signaling (SIP protocol) - for SIPREC recording only
22000 - 23999 (UDP)	SIPREC recording media (RTP protocol) - for SIPREC recording only
32000 - 33999 (UDP)	Avaya DMCC recording media (RTP protocol) - for Avaya DMCC recording interface only
6091 (TCP)	Screen recording controller, unencrypted (optional)
6092 (TCP)	Screen recording controller, encrypted (TLS)

6.2 Enable https for miarec web portal

6.2.1 Setup free SSL certificate for MiaRec using Let's Encrypt (Centos 6/7)

This tutorial describes how to setup a free TLS/SSL certificate from Let's Encrypt on MiaRec server based on Centos 7 server running Apache as a web server.

SSL certificates are used within web servers to encrypt the traffic between the server and client, providing extra security for users accessing your application. Let’s Encrypt provides an easy way to obtain and install trusted certificates for free.

> **What is Let's Encrypt?** Let’s Encrypt is a free, automated, and open certificate authority managed by the non-profit Internet Security Research Group (ISRG). Major sponsors are the Electronic Frontier Foundation (EFF), the Mozilla Foundation, OVH, Akamai, Google and Cisco Systems. See this page for more on ISRG sponsors.

Step 1 - Enable EPEL repository in Centos 6/7

To use Certbot (described below), you must first enable the EPEL (Extra Packages for Enterprise Linux) repository and enable EPEL optional channel.

```
yum install epel-release
```

> **What is EPEL?** Extra Packages for Enterprise Linux (or EPEL) is a Fedora Special Interest Group that creates, maintains, and manages a high quality set of additional packages for Enterprise Linux, including, but not limited to, Red Hat Enterprise Linux (RHEL), CentOS and Scientific Linux (SL), Oracle Linux (OL).

Step 2 - Install Certbot

Install Certbot by running:

Centos 6:

```
cd /root
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
```

Centos 7:

```
yum install python-certbot-apache
```

What is Certbot? Certbot is an easy-to-use automatic client that fetches and deploys SSL/TLS certificates for webserver. Certbot was developed by EFF and others as a client for Let’s Encrypt. This client runs on Unix-based operating systems.

Step 3 - Configure Apache to serve .well-known/acme-challenge directory

The Apache web server should be configured properly to allow serving of the files inside the `/.well-known/acme-challenge` directory. In this tutorial, we will use directory `/var/www/html/.well-known` as a location for the Certbot's temporary files.

> **What is a purpose of .well-known directory?**
>
> To obtain SSL certificate, the Certbot client creates a temporary file in `${webroot-path}/.well-known/acme-challenge` directory. Then the Let’s Encrypt validation server makes HTTP requests to validate that the DNS for each requested domain resolves to the server running certbot. An example request made to your web server would look like:

```
66.133.109.36 - - [05/Jan/2016:20:11:24 -0500] "GET /.well-known/acme-challenge/HGr8U1IeTW4kY_Z6UIyaakzOkyQgPr_7ArlLgtZE8SX HTTP/1.1" 200 87 "-" "Mozilla/
5.0 (compatible; Let's Encrypt validation server; +https://www.letsencrypt.org)"
```

Create file `/etc/httpd/conf.d/letsencrypt-well-known.conf`:

```
vi /etc/httpd/conf.d/letsencrypt-well-known.conf
```

Copy-paste the following content to that file:

For Apache 2.4 (Centos 7):

```
<IfModule mod_proxy.c>
  ProxyPass /.well-known !
</IfModule>

Alias /.well-known/ "/var/www/html/.well-known/"

<Directory "/var/www/html/.well-known">
  Options None
  AllowOverride None
  Require all granted
</Directory>

<Location /.well-known/acme-challenge>
  Options None
  Require all granted
</Location>
```

For Apache 2.2 (Centos 6):

```
<IfModule mod_proxy.c>
  ProxyPass /.well-known !
</IfModule>

Alias /.well-known/ "/var/www/html/.well-known/"

<Directory "/var/www/html/.well-known">
  Options None
  Order allow,deny
  Allow from all
</Directory>

<Location /.well-known/acme-challenge>
  Options None
  Order allow,deny
  Allow from all
</Location>
```

Reload Apache:

```
service httpd reload
```

Step 4 - Obtain SSL certificates from Let's Encrypt server

Run the following command to obtain the certificate:

Centos 6:

```
./certbot-auto certonly --webroot -w /var/www/html/ -d miarec.example.com
```

Centos 7:

```
certbot certonly --webroot -w /var/www/html/ -d miarec.example.com
```

Important! Replace `miarec.example.com` with your MiaRec server DNS name.

If everything goes well, then you should see the following message:

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/miarec.example.com/fullchain.pem. Your cert will
   expire on 2017-08-06. To obtain a new or tweaked version of this
   certificate in the future, simply run certbot again. To
   non-interactively renew *all* of your certificates, run "certbot
   renew"
```

Note the location of the generated certificate files. In our example, it is `/etc/letsencrypt/live/miarec.example.com/`.

Step 5 - Install mod_ssl module for Apache

```
yum install mod_ssl
```

The module will automatically be enabled during installation, and Apache will be able to start using an SSL certificate after it is restarted. You don't need to take any additional steps for mod_ssl to be ready for use.

Step 6 - Configure Apache to use new SSL certificates

Edit file /etc/httpd/conf.d/ssl.conf

```
vi /etc/httpd/conf.d/ssl.conf
```

Modify the parameters `SSLCertificateFile`, `SSLCertificateKeyFile` and `SSLCertificateChainFile`. They should point to the public, private and CA certificate files correspondingly.

Example of configuration (replace `miarec.example.com` with your domain):

```
#   Server Public Key:
SSLCertificateFile /etc/letsencrypt/live/miarec.example.com/cert.pem

#   Server Private Key:
SSLCertificateKeyFile /etc/letsencrypt/live/miarec.example.com/privkey.pem

#   Server Certificate Chain:
SSLCertificateChainFile /etc/letsencrypt/live/miarec.example.com/chain.pem
```

Step 7 - Open port 443 on firewall

Add exclusion rule to firewall:

```
iptables -I INPUT 5 -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Save all rules into iptables configuration file:

```
service iptables save
```

Restart iptables service:

```
service iptables restart
```

Step 8 - Force HTTPS for all traffic except internal call event notification (recommended)

Create file `/etc/httpd/conf.d/miarec-ssl.conf`:

```
vi /etc/httpd/conf.d/miarec-ssl.conf
```

Copy/paste the following content into this file:

```
NameVirtualHost *:80
<VirtualHost *:80>
    RewriteEngine on
    RewriteCond %{HTTP_HOST}%{REQUEST_URI} !^127.0.0.1/notify
    RewriteRule ^/(.*) https://%{HTTP_HOST}/$1 [NC,R=301,L]
</VirtualHost>
```

Reload Apache:

```
service httpd reload
```

> **What is "127.0.0.1/notify" in the rewrite rule?** MiaRec uses internally the HTTP protocol for sending call event notifications from recorder engine to a web portal. The above rewrite rule will force HTTPS for all web traffic except internal communication between recorder and web portal.

Step 9 - Configure cron to automatically renew the certificate.

Let’s Encrypt CA issues short-lived certificates (90 days). This tutorial shows how to automatically renew the certificates using cron.

Edit file `/etc/crontab`:

```
vi /etc/crontab
```

Insert the following line to the end of file:

Centos 6:

```
27 5,21 * * * root /root/certbot-auto renew --quiet --no-self-upgrade --post-hook "apachectl graceful"
```

Centos 7:

```
27 5,21 * * * root certbot renew --quiet --no-self-upgrade --post-hook "apachectl graceful"
```

The example above will run the renew sub-command at 05:27 and 21:27 daily. You can change time to other values. If the certificates are updated, then apache is gracefully restarted.

Reload crond service:

Centos 6:

```
/etc/init.d/crond reload
```

Centos 7:

```
service crond restart
```

6.2.2 Setup free SSL certificate for MiaRec using Let's Encrypt (Ubuntu 14.04)

This tutorial describes how to setup a free TLS/SSL certificate from Let's Encrypt on MiaRec server based on Ubuntu 14.04 server running Apache as a web server.

SSL certificates are used within web servers to encrypt the traffic between the server and client, providing extra security for users accessing your application. Let’s Encrypt provides an easy way to obtain and install trusted certificates for free.

> **What is Let's Encrypt?** Let’s Encrypt is a free, automated, and open certificate authority managed by the non-profit Internet Security Research Group (ISRG). Major sponsors are the Electronic Frontier Foundation (EFF), the Mozilla Foundation, OVH, Akamai, Google and Cisco Systems. See this page for more on ISRG sponsors.

Step 1 - Install Certbot on Ubuntu 14.04

What is Certbot? Certbot is an easy-to-use automatic client that fetches and deploys SSL/TLS certificates for webserver. Certbot was developed by EFF and others as a client for Let’s Encrypt. This client runs on Unix-based operating systems.

To install Certbot, you must first enable the PPA repository maintained by the Certbot team:

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
```

Afterwards, update the package list to pick up the new repository's package information:

```
sudo apt-get update
```

And finally, install Certbot from the new repository with apt-get:

```
sudo apt-get install python-certbot-apache
```

Step 2 - Configure Apache to serve .well-known/acme-challenge directory

The Apache web server should be configured properly to allow serving of the files inside the `/.well-known/acme-challenge` directory. In this tutorial, we will use directory `/var/www/html/.well-known` as a location for the Certbot's temporary files.

> **What is a purpose of .well-known directory?**
>
> To obtain SSL certificate, the Certbot client creates a temporary file in `${webroot-path}/.well-known/acme-challenge` directory. Then the Let’s Encrypt validation server makes HTTP requests to validate that the DNS for each requested domain resolves to the server running certbot. An example request made to your web server would look like:

```
66.133.109.36 - - [05/Jan/2016:20:11:24 -0500] "GET /.well-known/acme-challenge/HGr8U1IeTW4kY_Z6UIyaakzOkyQgPr_7ArlLgtZE8SX HTTP/1.1" 200 87 "-" "Mozilla/
5.0 (compatible; Let's Encrypt validation server; +https://www.letsencrypt.org)"
```

Create file `/etc/apache2/sites-available/letsencrypt-well-known.conf`:

```
vim /etc/apache2/sites-available/letsencrypt-well-known.conf
```

Copy-paste the following content to that file:

For Apache 2.4:

```
<IfModule mod_proxy.c>
  ProxyPass /.well-known !
</IfModule>

Alias /.well-known/ "/var/www/html/.well-known/"

<Directory "/var/www/html/.well-known">
  Options None
  AllowOverride None
  Require all granted
</Directory>

<Location /.well-known/acme-challenge>
```

```
  Options None
  Require all granted
</Location>
```

Enable this configuration file:

```
sudo a2ensite letsencrypt-well-known.conf
```

Reload Apache:

```
sudo service apache2 reload
```

Step 3 - Obtain SSL certificates from Let's Encrypt server

Run the following command to obtain the certificate:

```
sudo certbot certonly --webroot -w /var/www/html/ -d miarec.example.com
```

Important! Replace `miarec.example.com` with your MiaRec server DNS name.

If everything goes well, then you should see the following message:

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/miarec.example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/miarec.example.com/privkey.pem
   Your cert will expire on 2017-12-26. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. To non-interactively renew *all* of your certificates, run
   "certbot renew"
```

Note the location of the generated certificate files. In our example, it is `/etc/letsencrypt/live/miarec.example.com/`.

Step 4 - Install mod_ssl module for Apache

The mod_ssl module is available in apache2-common package. Execute the following command at a terminal prompt to enable the mod_ssl module:

```
sudo a2enmod ssl
```

Enable HTTPS for Apache:

```
sudo a2ensite default-ssl
```

Step 5 - Configure Apache to use new SSL certificates

Edit file /etc/apache2/sites-available/default-ssl.conf

```
vim /etc/apache2/sites-available/default-ssl.conf
```

Modify the parameters `SSLCertificateFile`, `SSLCertificateKeyFile` and `SSLCertificateChainFile`. They should point to the public, private and CA certificate files correspondingly.

Example of configuration (replace `miarec.example.com` with your domain):

```
#   Server Public Key:
SSLCertificateFile /etc/letsencrypt/live/miarec.example.com/cert.pem

#   Server Private Key:
SSLCertificateKeyFile /etc/letsencrypt/live/miarec.example.com/privkey.pem

#   Server Certificate Chain:
SSLCertificateChainFile /etc/letsencrypt/live/miarec.example.com/chain.pem
```

Enable this configuration file and load Apache:

```
sudo a2ensite default-ssl.conf
sudo service apache2 reload
```

Step 6 - Open port 443 on firewall

If you are using **iptables** on this machine, then execute the following commands:

```
iptables -I INPUT 5 -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Save all rules into iptables configuration file:

```
service iptables save
```

Restart iptables service:

```
service iptables restart
```

If you are using **ufw** firewall, then execute the following commands:

```
sudo ufw allow https
```

Step 7 - Force HTTPS for all traffic except internal call event notification (recommended)

Create file `/etc/apache2/sites-available/miarec-ssl.conf`:

```
vim /etc/apache2/sites-available/miarec-ssl.conf
```

Copy/paste the following content into this file:

```
<VirtualHost *:80>
    RewriteEngine on
    RewriteCond %{HTTP_HOST}%{REQUEST_URI} !^127.0.0.1/notify
    RewriteRule ^/(.*) https://%{HTTP_HOST}/$1 [NC,R=301,L]
</VirtualHost>
```

Enable this configuration file and load Apache:

```
sudo a2ensite miarec-ssl.conf
sudo service apache2 reload
```

> **What is "127.0.0.1/notify" in the rewrite rule?** MiaRec uses internally the HTTP protocol for sending call event notifications from recorder engine to a web portal. The above rewrite rule will force HTTPS for all web traffic except internal communication between recorder and web portal.

Step 8 - Configure cron to automatically renew the certificate.

Let’s Encrypt CA issues short-lived certificates (90 days). This tutorial shows how to automatically renew the certificates using cron.

Edit file `/etc/crontab`:

```
vi /etc/crontab
```

Insert the following line to the end of file:

```
27 5,21 * * * root certbot renew --quiet --no-self-upgrade --post-hook "apachectl graceful"
```

The example above will run the renew sub-command at 05:27 and 21:27 daily. You can change time to other values. If the certificates are updated, then apache is gracefully restarted.

Reload cron service:

```
service cron reload
```

Verify if cron service is started:

```
service cron status
```

It should return something like:

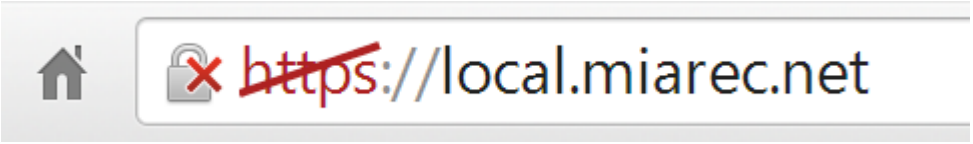
```
cron start/running, process 1105
```

6.2.3 Setup SSL certificate for MiaRec Web portal on Centos

In order to enable HTTPS (SSL) in MiaRec Web server, it is necessary to install SSL certificate. The certificate should be issued from a trusted Certificate Authority (like Verisign/Symantec, Comodo, GlobalSign, Digicert, GoDaddy etc).

The certificate is issued per domain name and can be used only with particular name. For example, if you install MiaRec on server and access it with address `https://rec.my-company.com`, then the SSL certificate should be issued to "rec.my-company.com" domain name.

Alternatively, the certificate can be self-signed. This means that instead of signing the certificate by Trusted Authority, you will sign it by your own certificate. In this case you will see in browser warning message that certificate is not trusted (means that it is not signed by trusted Certificate Authority), although the connection between client's web-browser and MiaRec server will be secure and encrypted:



You can generate the self-signed certificate using the following command line:

```
openssl req -new -newkey rsa:2048 -days 3650 -nodes -x509 -keyout server.key -out server.crt
```

This command will generate key/certificate pair and then sign it.

1. Install mod_ssl module for Apache

```
yum install mod_ssl
```

The module will automatically be enabled during installation, and Apache will be able to start using an SSL certificate after it is restarted. You don't need to take any additional steps for mod_ssl to be ready for use.

2. Install SSL private key and certificate

Copy your SSL private key to directory:

```
/etc/pki/tls/private/
```

Copy your SSL certificate to directory:

```
/etc/pki/tls/certs/
```

In some case you may need to copy also intermediary certificate of the company, which signed your certificate. Check their official instructions for Apache server.

3. Edit Apache configuration file (ssl.conf)

Edit file `/etc/httpd/conf.d/ssl.conf` and make sure that:

- `SSLCertificateFile` points to your certificate
- `SSLCertificateKeyFile` points to your private certificate
- `SSLCertificateChainFile` points to your certificate authority intermediary certificate (check your authority instructions)

```
#   Server Certificate:
SSLCertificateFile /etc/pki/tls/certs/miarec.example.com.crt

#   Server Private Key:
SSLCertificateKeyFile /etc/pki/tls/private/miarec.example.com.key
```

```
#   Server Certificate Chain:
SSLCertificateChainFile /etc/pki/tls/certs/CA.crt
```

4. Disable SSL protocol, allow TLS v1.2 only

It is recommended to disable SSL version 3.0 protocol, and force clients to use more secure TLS v1.2

Edit file `/etc/httpd/conf.d/ssl.conf`, locate the **SSLProtocol** line, if its commented out with a **#**, remove the hash (#) symbol and change it to the following:

```
SSLProtocol TLSv1.2
```

Now to increase the security strength we can also disable the weaker ciphers, located the **SSLCipherSuite** line, uncomment it and make it:

```
SSLCipherSuite HIGH:MEDIUM:!SSLv3:!kRSA:!RC4:!3DES
```

5 Disable TRACE method

Add the following line to the end of file `/etc/httpd/conf/httpd.conf`:

```
TraceEnable off
```

6. Open port 443 on firewall

Add exclusion rule to firewall:

```
iptables -I INPUT 5 -i eth0 -p tcp --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
```

Save all rules into iptables configuration file:

```
service iptables save
```

Restart iptables service:

```
service iptables restart
```

7. [Optional] Force HTTPS for all traffic except internal call events

Create file `/etc/httpd/conf.d/miarec-ssl.conf`:

```
vi /etc/httpd/conf.d/miarec-ssl.conf
```

Copy/paste the following content into this file:

```
NameVirtualHost *:80
<VirtualHost *:80>
    RewriteEngine on
    RewriteCond %{HTTP_HOST}%{REQUEST_URI} !^127.0.0.1/notify
    RewriteRule ^/(.*) https://%{HTTP_HOST}/$1 [NC,R=301,L]
</VirtualHost>
```

Reload Apache:

```
service httpd reload
```

> **What is "127.0.0.1/notify" in the rewrite rule?** MiaRec uses internally the HTTP protocol for sending call event notifications from recorder engine to a web portal. The above rewrite rule will force HTTPS for all web traffic except internal communication between recorder and web portal.

8. Restart Apache

```
service httpd restart
```