Hybrid Deployment Installation

Release 4.12.5

Document Version 1.1

March 14, 2018

343 Sansome Street, SF, CA. 94104 - www.OverOps.com

Contents

Introduction

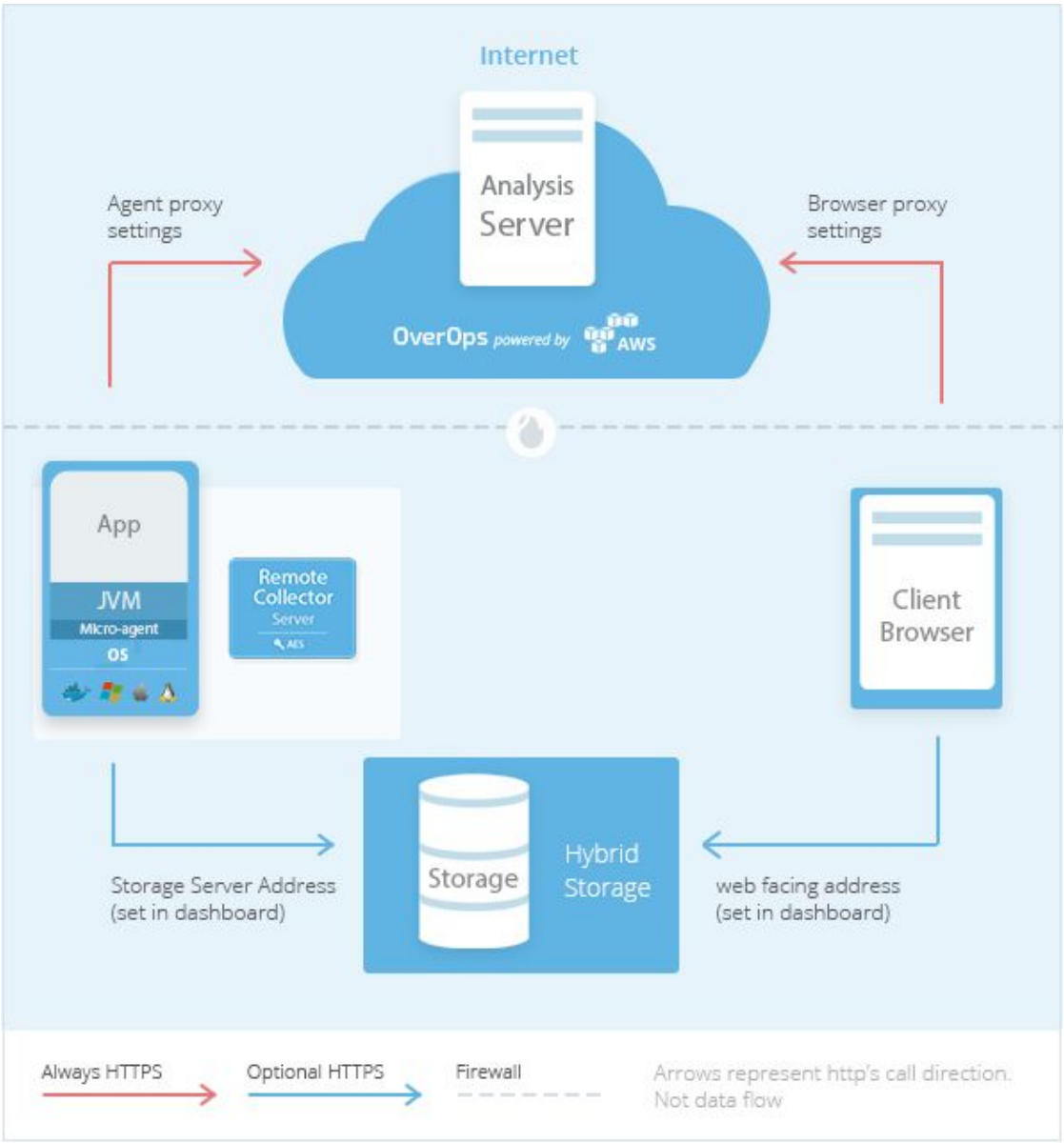
1.1 Overview

OverOps is a tool to identify and locate code failure in production. It allows tracing errors across the source code, determine the variables state, and debug level log statements for any error across the entire call stack. OverOps can be installed in SaaS, Hybrid, or full On-premises environments, this document deals with the hybrid deployment.

In an OverOps Hybrid deployment, the source code and variable state data are stored locally while the statistic analysis is hosted in the cloud. The OverOps central analysis engine is used to aggregate metrics and correlate events between JVMs in the environment. The local Storage Server does not connect to or get accessed from the cloud and can be completely separate from the public Internet.

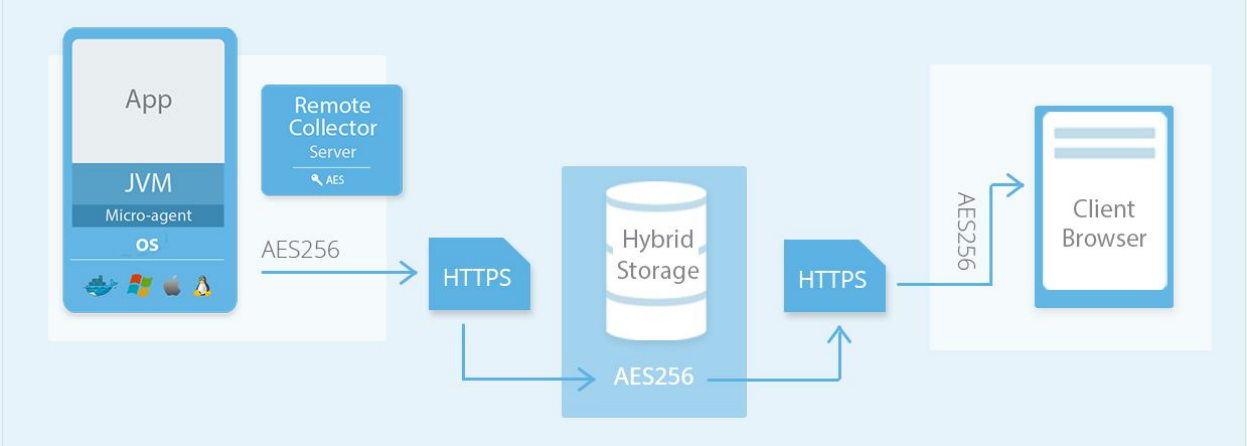
Storage of sensitive information only within a local network allows for compliance with data residency regulations. During error analysis, information is retrieved by the user browser directly from the local Storage Server.

The hybrid installation includes installing a Storage Server and one or more Agents behind a private network for additional privacy and compliance. By default, the communication takes place over an HTTP transport layer, in this model the objects transferred remain encrypted as they were at the Collector and get unencrypted at the browser. For additional security and to avoid browser warnings, use HTTPS for this transport to allow for server signature certification.



OverOps Hybrid Architecture

OverOps offers multiple layers of security to protect the privacy of the collected source code, the variables and data relating to your application. All code and variables are encrypted using the secret encryption key. This encryption is applied on the Agent/Collector by the source JVM and the code is sent to the Storage already encrypted. The code is decrypted at the browser for user display. The encryption used for the code and variables is AES 256-bit.



At startup, the Agent/Collector asks the web application where to store data, therefore, the Storage Server must be installed before the Collector/Agent to avoid having to generate a new key. For more details about OverOps security features, see: OverOps Security Overview.

1.2 Glossary

The following table describes the key terms in the OverOps installation and product:

Agent (micro-agent)	The component installed alongside a JVM that monitors the customer applications.
Collector	The endpoint to which Agents report. The Collector can be installed on the same machine as an Agent or a remotely on a separate machine.
Remote Collector	A Collector installed on a remote machine, separate from the Agent.
SaaS Deployment	A Software-as-a Service deployment option where only a Agent and Collector components are installed. All other components are administered by OverOps as a service.
Hybrid Deployment	A extension to the SaaS deployment option designed to address data locality concerns where a Storage Server is installed behind your company Firewall. With a Hybrid deployment, both source code and variable state data are physically stored and managed by you offering an additional layer of security to the SaaS deployment.
On-Premises Deployment	A deployment option in which the entire OverOps infrastructure is installed on-premise.

Installation Key	A unique identifier and a means of encrypting data to limit access to authorized individuals.
Dashboard	The OverOps Dashboard serves as the main hub to detect, prioritize and fix critical errors in your staging and production application(s).
Automated Root Cause	Displays single events, including stack frames and variable state values that led to a specific error, as well as the distribution of that error over time.
Event (snapshot)	An event instance that OverOps captured, displayed on the dashboard and the Automated Root Cause page. OverOps records event information such as: exceptions, log errors / warnings, HTTP errors, etc.

Prerequisites

The Hybrid environment includes three components: the Storage Server, the Agent and one or more Collectors. The Storage Server requires a separate machine and the Agent is installed on the JVM of the monitored machine. The main Collector, however, can be installed either on the monitored or a remote machine. Multiple Remote Collectors can be installed.

2.1 Hardware Requirements

The following resources are required for a hybrid installation.

Storage Server

- 2 Core Modern CPU
- 4 GB RAM
- 100 GB Disk space

Remote Collector (optional)

- Minimum 4 GB RAM (8 GB recommended)
- Minimum 50 GB Disk Space

2.2 Software Requirements

The following software is required for a hybrid installation.

Storage Server

- Linux
- Java 7 or above
- sudo privileges.

Remote Collector

- Linux
- Java 6 or above
- sudo privileges.

2.3 Network Requirements

Storage Server

Open the following *inbound* TCP/HTTP(S) ports for the Storage Server. These ports are defaults and may be changed to different ones in the *settings.yml* file. This enables network access for the Collector and for the user browser within the local network.

- 8080 used for HTTP access
- 8443 used for HTTPS access.

Remote Collector

For HTTPS *outbound* access, the following hosts must be whitelisted. This may be done using a proxy.

- backend.takipi.com
- prod1-sparktale.s3.amazonaws.com
- prod1-frontend-sparktale.s3.amazonaws.com

If whitelisting hostnames is not available, a fixed IP address can be used on a case-by-case basis.

3. Installation

This section describes how to install and setup the Storage Server to initially communicate over HTTP.

Note: The <STORAGE_IP> placeholder in this document stands for the IP address of the Storage Server.

3.1. Preparations

Prior to installation, carry out the following actions:

1. Create an OverOps account at https://app.overops.com.
2. Create a new key from the OverOps dashboard.

   An existing key may be used only if no data has been loaded to it. This prevents missed hits after the configuration is changed to hybrid.
3. Contact your OverOps support team and request that the OverOps key name be converted to hybrid. For this purpose, provide:
   a. Account name (the service owner's email)
   b. Service name (first part of the key, eg: S12345)
   c. Hostname or IP address of the Storage Server (<STORAGE_IP>).

      After installation, this may be self updated by admin users.

3.2. Installing the Storage Server

The Storage Server can work on a local disk or an AWS S3 bucket.

3.2.1. Installing the Storage Server on a Local Server

To install the Storage Server on a local server:

1. From the following URL, download the Storage Server installation file:

   ```
   wget
   http://app-takipi-com.s3.amazonaws.com/deploy/takipi-storage/takipi-storage-latest.tar.gz
   ```

2. Extract files to the /opt directory:

   ```
   sudo tar zxvf takipi-storage-latest.tar.gz -C /opt
   ```

3. Run installation:

   ```
   sudo chmod +x /opt/takipi-storage/etc/takipi-storage
   sudo cp /opt/takipi-storage/etc/takipi-storage /etc/init.d
   ```

   Edit the /etc/init.d/takipi-storage file and verify that it points to a valid Java installation

4. Set the Storage Server as a service, depending on the Linux distribution:

   - Ubuntu:

     ```
     sudo /usr/sbin/update-rc.d takipi-storage defaults
     ```

   - Other Linux:

     ```
     sudo /sbin/chkconfig takipi-storage on
     ```

5. Start the the Storage Server:

   ```
   sudo service takipi-storage start
   ```

6. Verify that the Storage Server is running:

   ```
   ps -ef | grep takipi-storage
   ```

   If service does not start check the logs and send to the OverOps team:

   ```
   less /opt/takipi-storage/log/takipi-storage.log
   ```

3.2.2. Installing the Storage Server on AWS S3

To install Hybrid Storage on an S3 bucket:

1. From Amazon Web Services, designate an S3 bucket for the OverOps hybrid storage: create, allocate or share an existing an S3 bucket with OverOps using a path, e.g. overops-<COMPANY>-storage.
2. From the following URL, download the Storage Server installation file:

   ```
   wget
   http://app-takipi-com.s3.amazonaws.com/deployx/s3/deploy/takipi-stor
   age/takipi-storage-latest.tar.gz
   ```

3. Extract files to the /opt directory:

   ```
   sudo tar zxvf takipi-storage-latest.tar.gz -C /opt
   ```

4. Run installation:

   ```
   sudo chmod +x /opt/takipi-storage/etc/takipi-storage
   sudo cp /opt/takipi-storage/etc/takipi-storage /etc/init.d
   ```

   Edit the /etc/init.d/takipi-storage file and verify that it points to a valid Java installation
5. Set the Storage Server as a service, depending on the Linux distribution:

   a. Ubuntu:

      ```
      sudo /usr/sbin/update-rc.d takipi-storage defaults
      ```

   b. Other Linux:

      ```
      sudo /sbin/chkconfig takipi-storage on
      ```

6. Generate an access key for the S3 API, or set up an ARN Policy to the bucket, see https://docs.aws.amazon.com/config/latest/developerguide/s3-bucket-policy.html.
7. In *settings.yml*, fill in the s3Fs configurations:

```
# If using attaching AIM Role to instance, leave accessKey and
secretKey empty
s3Fs:
    bucket: <BUCKET_NAME>
    pathPrefix: <FOLDER_IN_BUCKET>
    credentials:
      accessKey:
      secretKey:
```

8. Start the the Storage Server:

   ```
   sudo service takipi-storage start
   ```

9. Verify that the Storage Server is running:

   ```
   ps -ef | grep takipi-storage
   ```

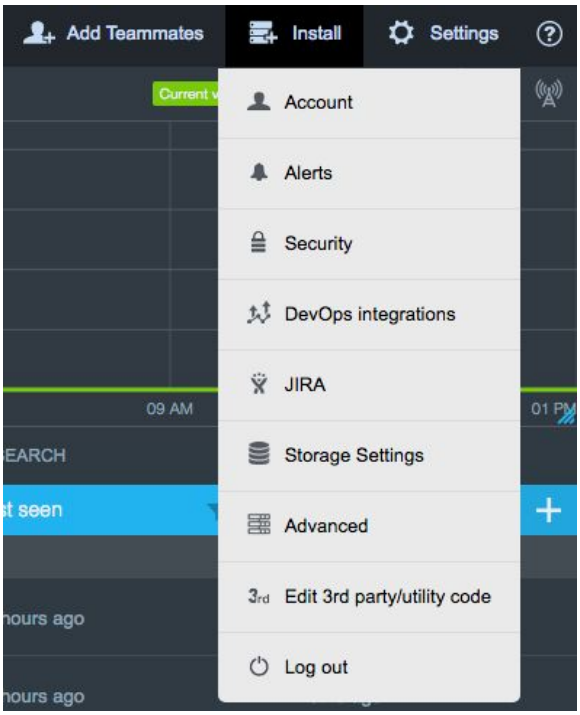
   If service does not start check the logs and send to the OverOps team:

   ```
   less /opt/takipi-storage/log/takipi-storage.log
   ```

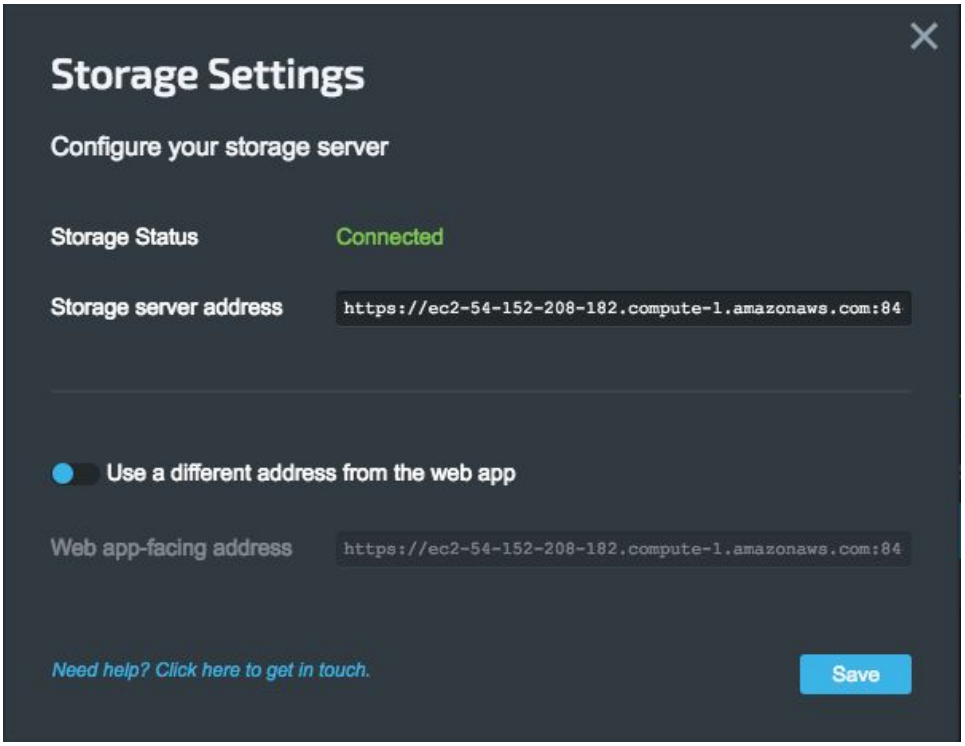
3.3 Enabling Storage Server Connectivity over HTTP

To configure the Storage Server with HTTP:

1. Login to OverOps and from **Settings**, click **Storage Settings**.



2. From the Storage Settings dialog box, set the web application and the Storage Server addresses.
3. When the browser is using a different IP address/hostname to access the server, check the checkbox and set it in the **web app-facing address** field.

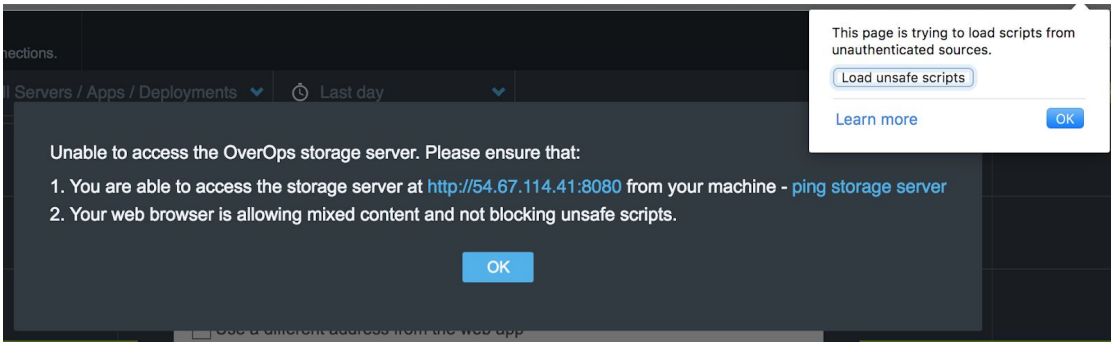


Note: The address format is: **http://<IP_ADDRESS>:<PORT>**

Example: http://127.0.0.1:8080

4. Enable unsecured scripts to load data by clicking **Load unsafe scripts**.

Note: You no longer need to enable unsecured scripts after Section 4 Enabling HTTPS



3.4 Verifying Storage Server Installation:

1. Verify that the machine with the JVM you want to monitor is communicating with the Storage Server:

   ```
   curl -I http://<STORAGE_IP>:8080/storage/v1/diag/ping
   ```

   The expected output is:

   ```
   HTTP/1.1 200 OK
   Date: Wed, 08 Nov 2017 22:19:15 GMT
   Content-Type: text/plain
   Content-Length: 0
   ```

2. Check that the web application is communicating with the Storage Server: from the browser, go to: http://<STORAGE_IP>:8080/storage/v1/diag/ping

   An **OK** message is displayed.

Note: If you encounter an 'Unable to access the OverOps storage server' message on the OverOps Dashboard, despite working communication to the Storage Server via cURL or URL, it may be due to an enabled AdBlock extension. In this case, a 'net::ERR_BLOCKED_BY_CLIENT' error message is displayed in the browser network console.

3.5 Installing the Remote Collector

The Remote Collector can also be installed on the Storage Server. This does not change the Collector installation process.

To install a Remote Collector:

1. To create a machine-specific installation script, in the following one-liner, replace the <HOSTNAME> and <INSTALLATION_KEY> placeholders:

   ```
   wget -O - -o /dev/null http://get.takipi.com | sudo bash /dev/stdin
   -i --sk=<INSTALLATION_KEY> --listen_on_port=<COLLECTOR_PORT>
   ```

   Note: The one-liner can be found in the web application by clicking **Add Servers.**

2. Verify that takipi-service is running:

   ```
   ps -ef | grep takipi
   ```

3.6 Installing Agent for Local Collector

To install the Agent on monitored machines and restart JVM:

1. To create a machine-specific installation script, in the following one-liner, replace the <INSTALLATION_KEY> placeholder with your OverOps installation key.
2. Run the installation file on a machine with JVM to monitor:

   ```
   wget -O - -o /dev/null http://get.takipi.com | sudo bash /dev/stdin
   -i --sk=<INSTALLATION_KEY>
   ```

3. To restart the JVM, run the JVM with an -agentpath option:

   ```
   java -agentlib:TakipiAgent -jar your.jar
   ```

3.7 Installing Agent for Remote Collector

To install the Agent on monitored machines and restart JVM:

1. To create a machine-specific installation script, in the following one-liner, replace the placeholders:
   <REMOTE_COLLECTOR_HOST> the host on which you installed the Collector,
   <REMOTE_COLLECTOR_PORT> the the port you opened on the Collector Host
   <INSTALLATION_KEY> your OverOps installation key.
2. Run the installation file on a machine with JVM to monitor:

   ```
   wget -O - -o /dev/null http://get.takipi.com | sudo bash /dev/stdin
   -i --sk=<INSTALLATION_KEY> --daemon_host=<REMOTE_COLLECTOR_HOST>
   --daemon_port=<REMOTE_COLLECTOR_PORT>
   ```

3. To restart the JVM, run the JVM with an -agentpath option:

   ```
   java -agentlib:TakipiAgent -jar your.jar
   ```

4. Enabling HTTPS

This section describes how to setup Storage Server communication over HTTPS. Make sure that you have completed all of section 3 Installation before starting. If you are using a publicly signed certificate, skip section 4.1, and go to section 4.2 Switching to HTTPS using a Publicly Signed Certificate.

Storage Server may simultaneously serve both HTTP and HTTPS connections. Part of the deployment architectures involves deciding when to use each one and depends on the network topology.

In a secured connection, the browser does not warn against 'Unsecured scripts'. OverOps recommends using a publicly signed certificate to avoid having to manually accept the self-signed certificate. Connecting to the Collector using HTTPS requires the Collector to trust the Storage Server. When using self-signed certificates, this may require a manual step.

Since the data objects are encrypted at the source, even on HTTP mode, no sensitive data is transmitted unencrypted at any time.

4.1 Auto-Generating a Self-signed Certificate

OverOps provides a tool to generate a certificate.
To auto-generate a self-signed certificate:

1. From the browser, go to: https://storage-cert-gen.takipi.com/

   The certificate generating page opens:

2. Enter the hostname of the server to be trusted by the certificate and click **Generate**.

   The zip file is downloaded.

3. Extract the files into the chosen directory.

   The zip file contains three files:

   - *settings.yml* contains the path and the password to the keystore file

   - *takipi-storage.jks* is the Java Keystore file

   - *takipi-storage-certificate.pem* is the certificate.

4. Open the *settings.yml* file and merge the HTTPS parameters into the existing *settings.yml* file on the Storage Server:

```
applicationConnectors:
      - type: http
        port: 8080
      - type: https
        port: 8443
        keyStorePath: /opt/takipi-storage/<STORAGE_KS_FILE>
        keyStorePassword: <STORAGE_KS_PASSWORD>
        validateCerts: false
```

5. Move the *takipi-storage.jks* file in the keystore directory according to the keystore path.

6. Restart the Storage Server:

   ```
   sudo service takipi-storage stop ; sudo service takipi-storage start
   ```

7. Optional: From the browser, import the *takipi-storage-certificate.pem* file. This procedure is different for each browser.
8. Verify the secured connection by entering the https://<STORAGE_HOST>:8443/storage/v1/diag/ping URL in the browser.

   The output for a secured connection is: **OK**

4.2 Using a Publicly-signed Certificate

When using a publicly-signed certificate, you have to switch HTTP to HTTPS. Before switching to HTTPS, verify that the standard HTTP port is up and running.

To switch to HTTPS:

1. From the keystore directory, copy the *takipi-storage.jks* file to the Storage Server directory in /opt/takipi-storage.
2. Update the Storage Settings file /opt/takipi-storage/settings.yml with the HTTPS settings. Add or uncomment the highlighted lines to match the following:

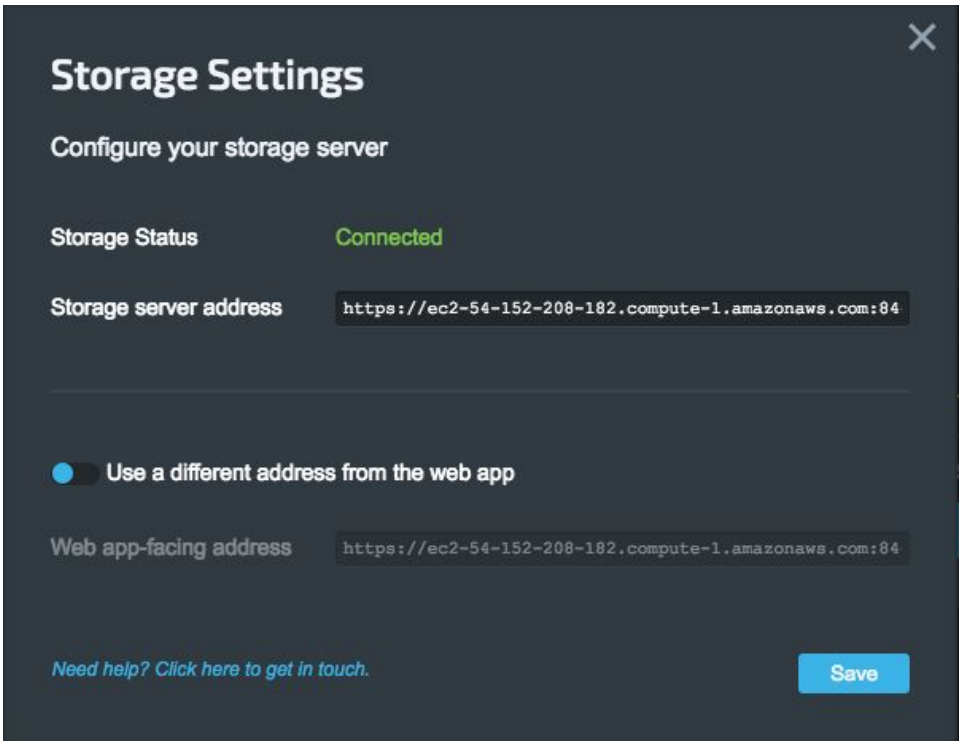
   ```
   applicationConnectors:
         - type: http
           port: 8080
         - type: https
           port: 8443
           keyStorePath: /opt/takipi-storage/<STORAGE_KS_FILE>
           keyStorePassword: <STORAGE_KS_PASSWORD>
           validateCerts: false
   ```

3. Restart the Storage Server:

   ```
   sudo service takipi-storage stop ; sudo service takipi-storage start
   ```

4. From the OverOps Dashboard, click Settings and select Storage Settings.

5. From Storage Settings dialog box, update the access settings for the Web app-facing address to use HTTPS and the new port. The Storage Server address should continue to use HTTP, see Collector over HTTPS.



Storage Server address format is: https://<STORAGE_IP>:<STORAGE_PORT>
e.g. https://127.0.0.1:8443
When the browser accesses the server with a different IP address/hostname, click the checkbox and set it in the **web app-facing address** field.

6. In the browser, import the *takipi-storage-certificate.pem* file. This procedure is different for each browser.

4.3 Using an HTTPS Proxy

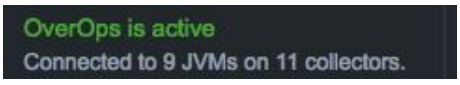
Alternatively, the Storage Server can run only on HTTP and set up an HTTPS proxy to offload the encryption and keep the keystore in a single place. This requires a number of considerations:

1. The proxy must be set at the root of the virtualhost, i.e **https://<VIRTUAL_HOSTNAME>/**
2. The URL in the dashboard must be updated to one of the proxies. This can be done for both access from the browser (web app-facing address) and access from the monitored servers running the Collector process.
3. If connecting from the Collector using HTTPS, the server running the Collector must have a trusted certificate.

4.4 Verifying the HTTPS Setup

1. From the browser, go to the the Storage Server diagnostic URL
   https://<STORAGE_IP>:<STORAGE_HTTPS_PORT>/storage/v1/diag/ping
   E.g: https://127.0.0.1:8443/storage/v1/diag/ping
   If the certificate is not signed, indicate that you trust the Storage Server when accessing it from the browser.
2. Verify connectivity from the Collector host using cURL:
   ```
   curl -I -k
   https://<STORAGE_IP>:<STORAGE_HTTPS_PORT>/storage/v1/diag/ping
   ```

Once the connection is established from the web application to the Collector and the Storage Server, the **OverOps is active** status appears on the top left corner of the Dashboard:



The Storage Server is installed and active.

5. Installing Rootless Agent

For security reasons, the OverOps Agent can also be installed without root access.

To install the OverOps Agent on monitored machines, without root access:

1. Download the OverOps Agent:

   With Wget:

   ```
   wget
   https://s3.amazonaws.com/app-takipi-com/deploy/linux/takipi-agent-latest.tar.gz
   ```

   With cURL

   ```
   curl -LO
   https://s3.amazonaws.com/app-takipi-com/deploy/linux/takipi-agent-latest.tar.gz
   ```

2. Unpack it to any directory.

   A **takipi** folder is created.

3. From the takipi folder, edit the *takipi.properties* file and set the following parameters:

   - **serverName**: the machine name, usually this is the output of echo `hostname`.

   - **masterHost**: the Collector host.

   - **masterPort**: the Collector port.

4. Start your application with -agentpath option:

   ```
   java -agentpath:/home/user/takipi/lib/libTakipiAgent.so -jar
   sample.jar arg1
   ```

5. Advanced Settings

This section describes alternative configuration and additional functionalities for the Hybrid installation.

5.1 Changing Default Ports

If the default ports are already in use, change the ports on which the server in *settings.yml* file.

To change default ports:

- In the */opt/takipi-storage/settings.yml* file, under applicationConnectors, change HTTP and HTTPS ports as required:

```
applicationConnectors:
      - type: http
        port: 8080 # Change here
      - type: https
        port: 8443 # Change here
        keyStorePath: /opt/takipi-storage/storage.jks
        keyStorePassword: secret
        validateCerts: false
```

5.2 Creating a Certificate from a JVM Keystore (Optional)

You can also create a self-signed certificate from the JVM Keystore.

To export the certificate and load it to the Collector's JVM keystore to be trusted (optional):

1. On the Storage Server, extract the .cer certificate for the keystore. By default, the certificate name is generated from the domain name, e.g. *takipi.cer*:
   keytool -export -keystore <STORAGE_KEYSTORE_FILE> -alias <DOMAIN> -file <DOMAIN>.cer
   Enter keystore password: <STORAGE_KEYSTORE_PWD>
2. Add the certificate to each of the Collector services:
   a. To check which Java is the currently in use, from the *takipi.properties* configuration file, copy the path to the JVM library, highlighted:

```
cat /opt/takipi/takipi.properties
takipiHome=/opt/takipi
baseUrl=https://backend.takipi.com/
libraryPath=/opt/takipi/lib
installTime=1497372540096323
jvmPath=/usr/lib/jvm/java-8/jre/lib/amd64/server/libjvm.so
serverName=my_server_name
...
```

b. Using the copied path, move to the security directory of the current Java:

```
$ cd /usr/lib/jvm/java-8/jre/lib/security
```

c. Copy the <DOMAIN>.cer to the servers running the Collector, using secure file transfer (SCP, SFTP, etc.):

```
sudo cp /opt/takipi-storage/mydomain.cer .
```

Example to copy from the Storage Server to the local Collector:

```
$ scp -i "kp_file.pem"
ec2-user@ec2-storage_server:/opt/takipi-storage/keystore.jks .
```

d. Import the <DOMAIN>.cer as a trusted certificate (the default JVM password is 'changeit'):

```
sudo keytool -import -alias <DOMAIN> -file <DOMAIN.cer>
-keystore cacerts
Enter keystore password: <JVM_KS_PWD>
...
Trust this certificate? [no]: yes
```

2. Restart the Collectors:

```
sudo /opt/takipi/etc/takipi-stop && sudo
/opt/takipi/etc/takipi-start
```

5.3 Health

OverOps provides a metrics-based mechanism to display the connectivity between the Collector and the Storage Server. This mechanism utilizes StatsD protocol and is part of the Publish Metrics feature https://support.overops.com/hc/en-us/articles/218438597-Publish-Metrics.

When the StatsD server is up and running and the Publish Metrics is enabled, diagnostic metrics are sent from the Collector indicating the Storage Server's status:

```
# Successful API calls count to the storage server from a collector host
overops.diagnostics.${HOST}.StorageCalls

# Failed API calls count to the storage server from a collector host
overops.diagnostics.${HOST}.StorageErrors
```

The metrics can be used in various ways, see: Publish Metrics page.

5.5 High Availability

The Storage Server is a stateless, extendable storage API application that officially supports a file system directory path and AWS S3 backend implementations. This section describes the different ways to achieve high availability in both the application and the data tier.

Data Tier

Path Backend (default)

For high availability, mount the storage directory on all hosts running the Storage Server using the local NAS or any other distributed file system solution.

By default, the storage directory is located in /opt/takipi-storage/storage, and alternative locations are configured in the *settings.yml* file in /opt/takipi-storage.

AWS S3 Backend

High availability is provided with AWS S3. S3 behaves like NAS, and allows multiple Storage Servers to access the same data by using the same S3 region and bucket. The code can be found at: https://github.com/takipi/takipi-storage/tree/s3-storage

Other backends

To add your own elastic file system implementation, extend the takipi-storage project, from https://github.com/takipi/takipi-storage.

Application Tier

The Storage Server can be deployed on multiple hosts running behind a load balancer, enabling both scale and redundancy. The HTTPS (TLS) can be terminated either at the load balancer or at the Storage Servers as described in section Enabling HTTPS.

Nginx Configuration Examples

Use the following examples to configure application tier high availability:

HTTP only, using multiple Storage Servers:

```
http {
  upstream storage {
    server storage1.example.com:8080;
    server storage2.example.com:8080;
    server 10.84.0.109:8080;
  }

  server {
    listen 80;
    server_name loadbalancer.example.com;

    location / {
      proxy_set_header      Host $host;
      proxy_set_header      X-Real-IP $remote_addr;
      proxy_pass            http://storage;
    }
  }
}
```

HTTP/HTTPS using multiple Storage Servers, terminating TLS on the load balancer:

```
http {
  upstream storage {
    server storage1.example.com:8080;
    server storage2.example.com:8080;
  }

  server {
    listen 80;
    server_name loadbalancer.example.com;
    return 301 https://$host$request_uri;
  }

  server {
    listen 443;
    server_name loadbalancer.example.com;

    ssl on;
    ssl_certificate /etc/nginx/storage.crt;
    ssl_certificate_key /etc/nginx/storage.key;

    ssl_session_timeout  5m;
    ssl_prefer_server_ciphers On;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+AES128:DH+AES:RSA+AESGC
M:RSA+AES:!aNULL:!MD5:!DSS;

    location / {
      proxy_set_header      Host $host;
      proxy_set_header      X-Real-IP $remote_addr;
```

```
      proxy_set_header    X-Forwarded-Proto https;
      proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_redirect      http:// https://;
      proxy_pass            http://storage;
    }
  }
}
```

HTTP/HTTPS using multiple Storage Servers, terminating TLS on the Storage Servers:

```
http {
  upstream storage-http {
    server storage1.example.com:8080;
    server storage2.example.com:8080;
  }

  upstream storage-https {
    server storage1.example.com:8443;
    server storage2.example.com:8443;
  }

  server {
    listen 80;
    listen 443;
    server_name loadbalancer.example.com;

    location / {
      proxy_set_header      Host $host;
      proxy_set_header      X-Real-IP $remote_addr;
      proxy_set_header    X-Forwarded-Proto $scheme;
      proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass            http://storage-$scheme;
      proxy_redirect off;
    }
  }
}
```

5.6 Backups

OverOps recommends setting up a recurring backup of the storage data to a remote location immediately after finishing the installation process. By default, the data is stored in the /opt/takipi-storage/storage directory or configured in the *settings.yml* file. All the

files in the data layer are immutable, therefore backup can be run regardless of the Storage Server activity. The backup procedure serves two different purposes: secure the data on a secondary server, or provide redundancy in case of a main Storage Server failover. Instructions below assume neither, for more information contact your support team.

To set up recurring backup:

1. From the Storage Server terminal, run:

   `crontab -e + 0 * * * * rsync -av -e ssh <PATH_TO_STORAGE_DIRECTORY> <USER>@<BACKUP_SERVER_IP>`

Where:

- **`crontab -e + 0 * * * *`** - indicates that the command is run hourly.
- **`rsync -av -e ssh /opt/takipi-storage/storage <USER>@<BACKUP_SERVER_IP>`** - indicates the backup server to which the content is synced.

  Note: The main Storage Server requires SSH access to the backup server. Refer to the relevant rsync man page for further enhancements and troubleshooting.

Example:

```
crontab -e
0 * * * * rsync -av -e ssh /opt/takipi-storage/storage jdoe@10.0.12.121
```