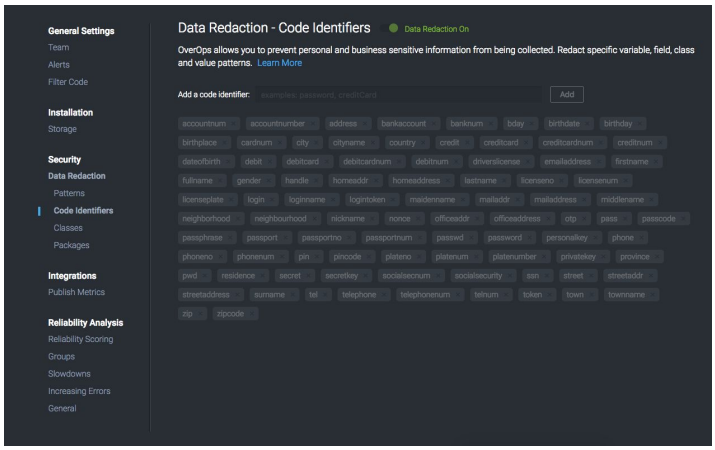
OverOps Security

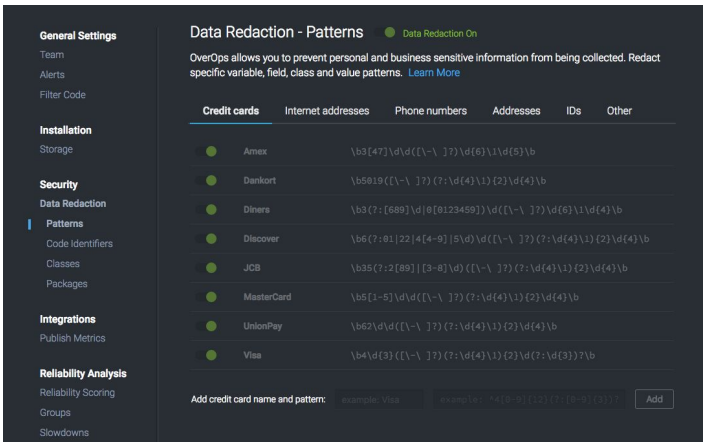
When it comes to your production environment, OverOps knows that security is of the utmost importance, so we take our security measures extremely seriously. Private information is redacted at the source, and all data is encrypted end-to-end from the monitored workstation to the analysis workstation (data is not unencrypted at analysis server). Segregated access control is available with SAML authentication option. Our cloud services are certified ISO 27001 and data locality can be addressed with a local storage option.

Table of Contents

Personally Identifiable Information, Data Redaction and Code Redaction

OverOps enables you to filter out any personally identifiable information (PII) and business sensitive information before it leaves your hosts. Three complementary modes are available for redacting variable data at run-time: pattern, identifier and class/method based filtering. Additionally, you can filter and redact entire code blocks at run-time.





Code Identifier-Based Filtering

This redaction method redacts all data collected from predefined variables, fields, classes and packages in your application. OverOps provides an extensive list of default variable and field names which are fully editable and customizable. This redaction mode is enabled by default.

Pattern-Based Filtering

This redaction method asynchronously scans all collected variable values against sets of predefined regular expression patterns, identifying values such as phone numbers, credit cards and email addresses. OverOps provides an extensive list of default regular expression patterns which are fully editable and customizable. This redaction mode is enabled by default.

Private Encryption

All source code and variable state collected is privately encrypted using a 256-bit AES Encryption key before leaving the production node. Only you have access to secret encryption keys which are not stored by OverOps.

Code Analysis

When viewing an error within OverOps, you can see the source code and variable values at the moment of occurrence. OverOps uses a combination of JVM-level signal detection and continuous code analysis in the cloud to determine and collect the right source code and variable state for each error.

All source code and variable data collected at run-time is encrypted using a strong 256 bit AES key privately generated for you during installation. Code and variable data collected on your machines is only uploaded to and stored by OverOps in the cloud in its encrypted private form. This ensures that it can only be viewed by you and your team using your private encryption key, and that it cannot be accessed by anyone else (including OverOps administrators).

To offload work from the local JVM in order to efficiently analyze errors, OverOps converts bytecode loaded by the application (e.g .jar, .war. class files) into an abstract graph structure which it analyzes in the cloud. The graph structure does not contain symbols, values or operators, and cannot be executed or reverse engineered. This conversion process, which runs on your machine, includes removing all jar, package, class, field, method, and variable names (both from your code and any Java or 3rd party frameworks), as well as removing all logical and numeric operators, number and string constants, and code attributes.

The graph is used by OverOps' central analysis service to correlate events across your cluster and determine which code fragments and variable values are required to analyze each error, as well as the fastest way of collecting those so in order to maintain low CPU and IO overhead.

As each binary artifact is uniquely identified by its MD5 signature, the conversion process only happens once per deployed artifact (e.g. .jar, .war) across your entire cluster. This enables OverOps to analyze code differentially, and further reduce overhead from your machines.

Data Encryption

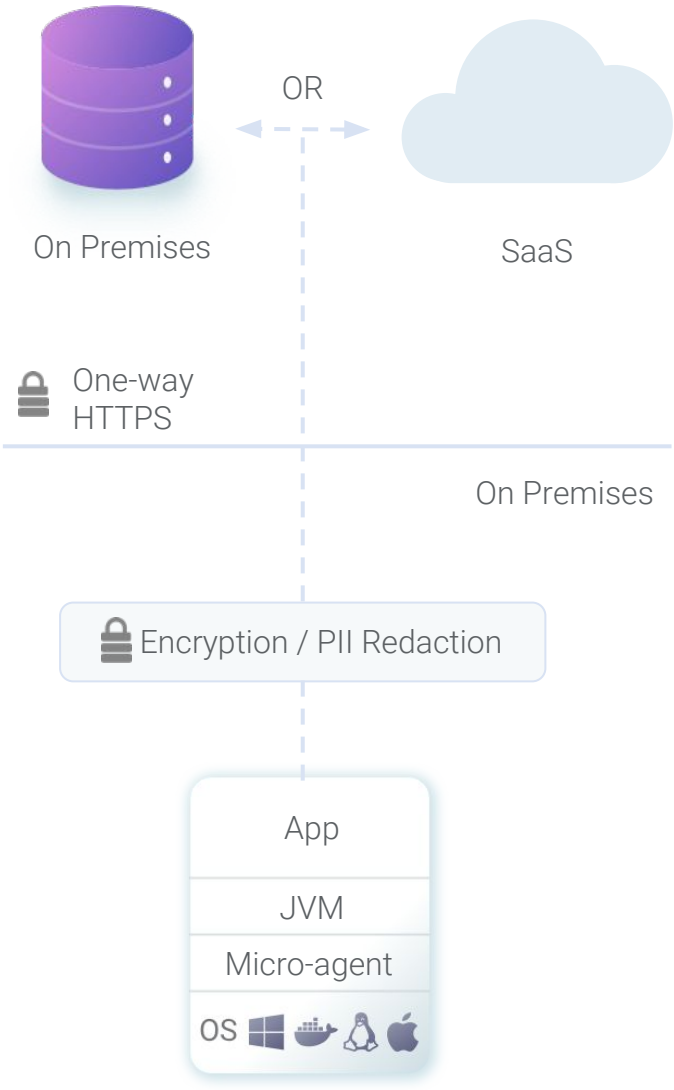
OverOps encrypts all source code and variable data collected at runtime using a strong 256-bit AES key privately generated for you during installation. Code and variable data collected on your machines is only stored outside of the monitored machine in its encrypted private form. This ensures that it can only be viewed by you and your team using your private encryption key, and that it cannot be accessed by anyone else (including OverOps administrators).

Key generation may be done by OverOps for convenience, but it can be also done locally compiling from the code and generating a key that would never be available to OverOps server (since it is only stored in monitored server and browser, not the analytics nor storage server from OverOps). The code for generating your own key is found here: https://github.com/takipi/keygen. When using a locally generated key, this needs to be manually shared with the users that need access as described in the data decryption section.

To display the source code for target methods that are related to an error, the relevant pieces from the converted bytecode graph are decompiled in the cloud into a source code template which does not contain any symbology, operator values or literals.

The code template is in turn sent back to the OverOps collector process on your server, where it is mapped and reconstructed into source code (using the original bytecode which resides on your machine). The reconstructed source code is encrypted on your machine using the private AES encryption key (known only to you). The encrypted source code is then stored for later viewing by authorized users on storage server (on the cloud or locally).

Storing the source code related to each specific error at the moment of occurrence ensures that even if you deploy new code to your servers, you will still have access to the exact code and variable state in the future.

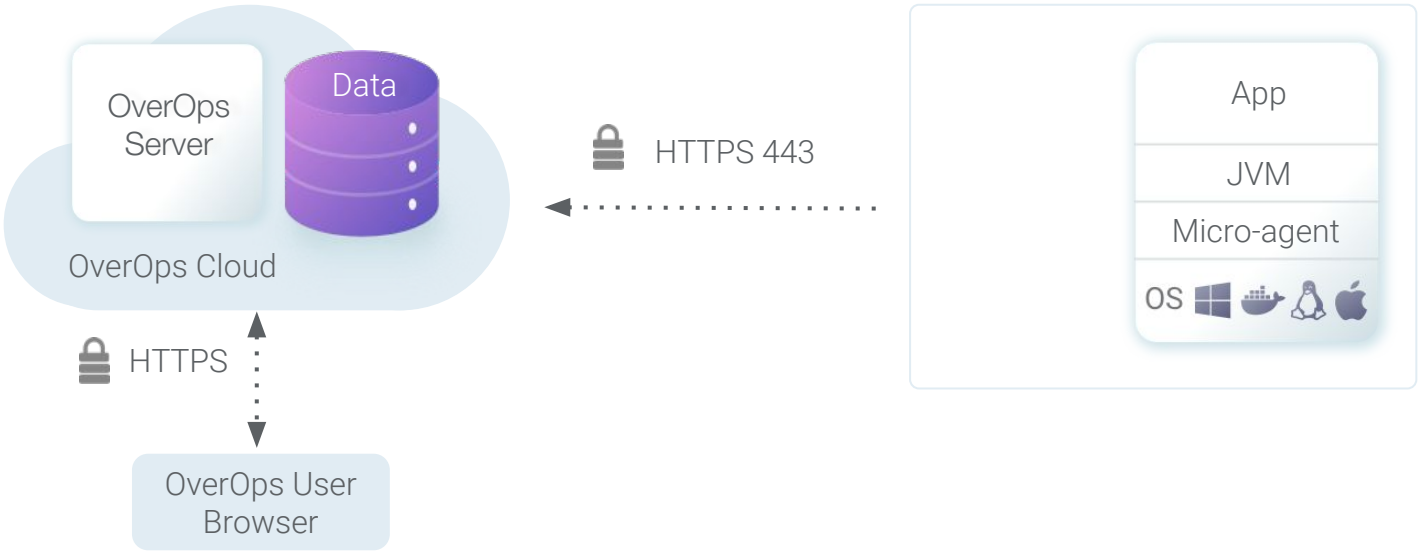


OverOps Deployment Models & Data Storage

OverOps provides three distinct modes for storing data collected from your JVMs: SaaS, on-prem data and on-premises.

SaaS

In SaaS mode, data collected from your JVMs is redacted for PII and encrypted locally using your private encryption key before it is sent to the OverOps Cloud to be stored for later viewing and analysis by users. OverOps uses Amazon AWS S3 storage, for more information around the security of your data on Amazon S3 please visit here: https://aws.amazon.com/s3/faqs/#security

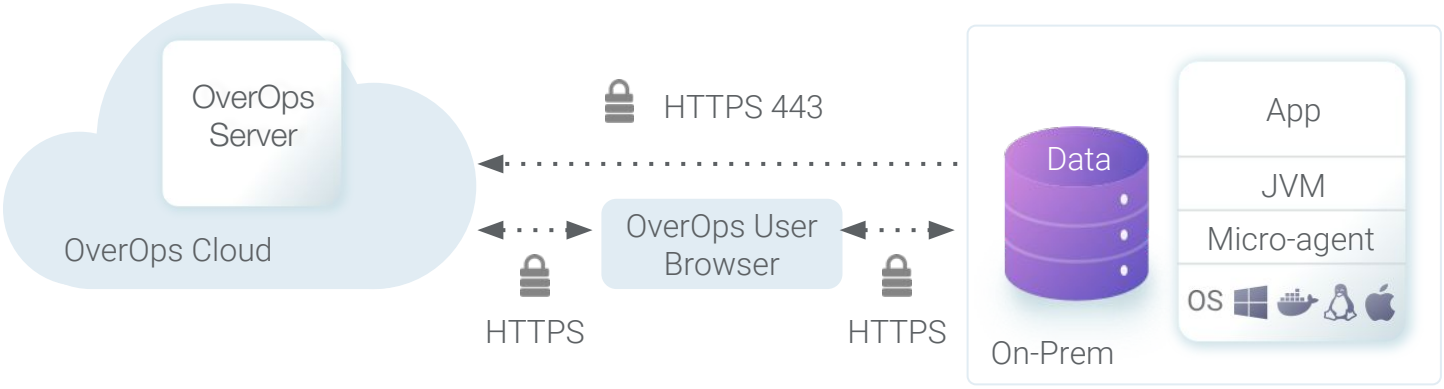


All communications between OverOps' daemon process and the analysis server, hosted on AWS, are made over HTTPS on outbound port 443. OverOps does not require that you open an inbound port for communications.

On-Prem Data Mode

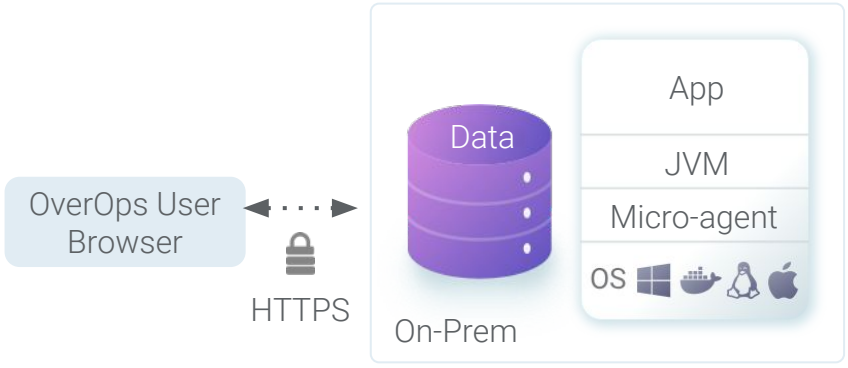
In on-prem data mode, data collected from your JVMs is locally redacted for PII and encrypted using your private encryption key before it is stored in a server that resides behind your firewall. The cloud-based central analysis engine is used only to aggregate metrics and correlate events between different JVMs in your environment.

When viewing an error analysis, information is retrieved directly into the user's web browser from the on-premises storage server without leaving your firewall and domain.The on-premises storage server does not need to connect to the cloud and can be completely separate from the public internet.



Full On-Premises Mode

Similar to on-prem data mode, In full on-premises mode data collected from your JVMs is locally redacted for PII and encrypted using your private encryption key before it is stored locally on-premises behind your firewall. When viewing an error analysis, information is retrieved directly into the user's web browser from the on-premises storage server without leaving your firewall and domain.



User and Access Point Control

Access control is essential in production environments. OverOps enables you to manage user access to the OverOps Dashboard.

Role Management

OverOps enables the administrator of each cluster (identified by its installation key) to control which team members can access error analyses collected from monitored machines.

User Account Management

We natively support Single-Sign On from Google or Github accounts. We also support integrations with SAML to sync with your identity management of choice.