**SMILE CDR**

**2.0.1Basic Concepts**

Welcome to Smile CDR! Before getting started, let's spend a few minutes going over the basic concepts of how the software works.

**2.0.2Nodes and Modules**

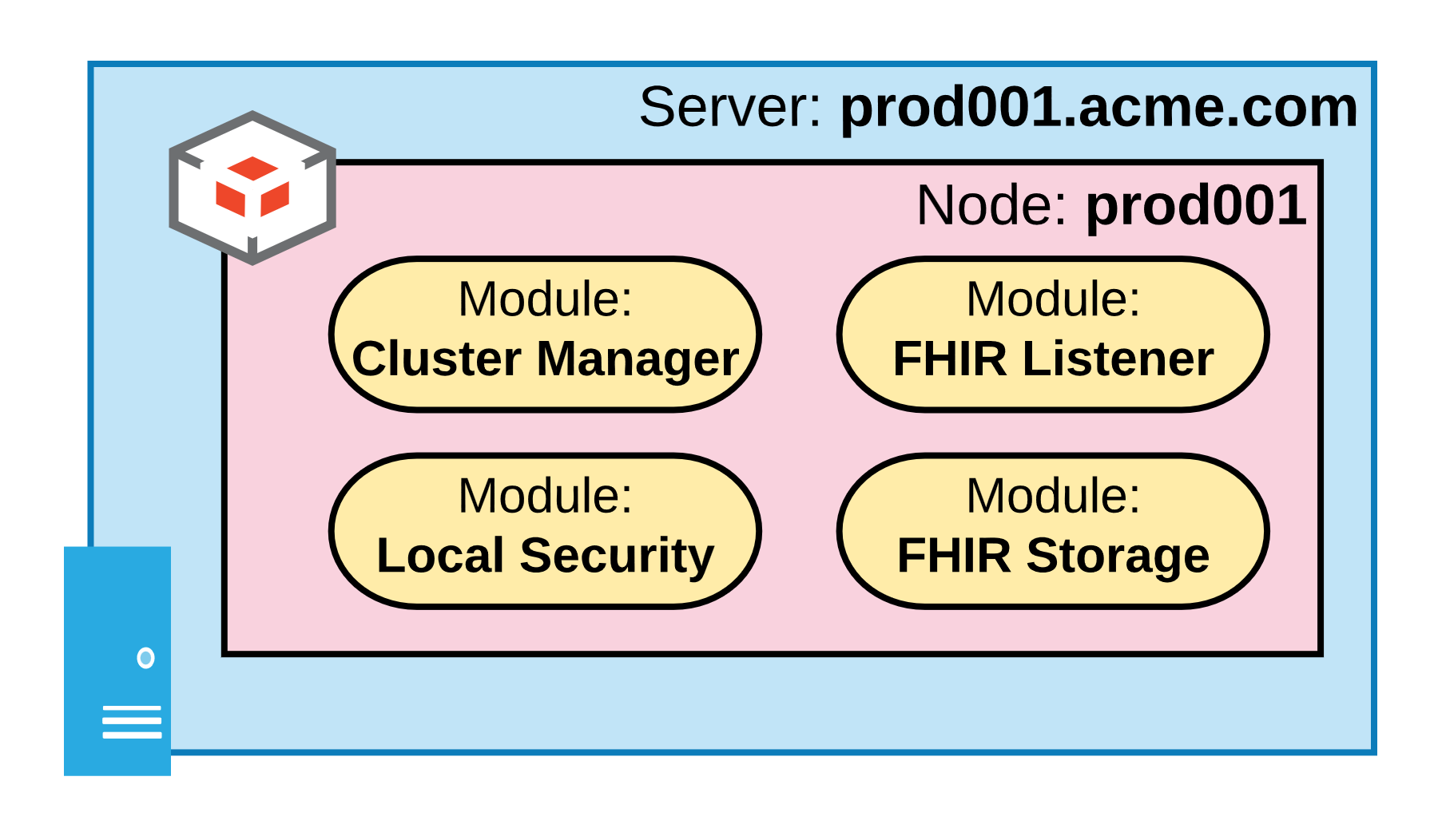
A Smile CDR deployment is divided into **nodes** and **modules**.

* A node is a single server process that runs on a server, and is executing the Smile CDR service (e.g. as a Linux daemon or as a Docker container)
* A module is a Smile CDR software module running within a node that handles a specific piece of functionality

Each module in a node has its own configuration, and performs a specific function within the CDR. As an example, the simplest version of a FHIR server built with Smile CDR would consist of several modules, including:

* A FHIR Listener module – that has been configured to listen on a given port, has TLS/HTTPS settings, security settings, etc.
* A FHIR Storage module – that has been configured with database settings, that handles saving and retrieving resources, etc.
* An Inbound Security module – that handles authenticating incoming requests

The following diagram shows a simple example of a server with a single node containing several modules.



**2.0.3The Cluster Manager Module**

Every node has at least one module: the **cluster manager**. The cluster manager module is present even if you only have a single node (non-clustered deployment).

* It handles most of the basic bootstrap functionality of the system, including loading and saving module configuration, starting and stopping other modules, performance metrics tracking, health checks, etc.

**2.0.4Module Dependencies**

Within a node, a module can depend on other modules for functionality it needs. For example, the FHIR Listener module – that exposes an HTTP REST endpoint to which clients can connect – needs:

* A FHIR Storage module – that has been configured with database settings, that handles saving and retrieving resources, etc.
* An Inbound Security module – that handles authenticating incoming requests

There are different types of modules that can satisfy dependencies between modules. For example, the required Inbound security module might be a Local Inbound Security module, which is a security module that saves user accounts in Smile CDR's own database (including salted password hashes) so that users can be authenticated without needing to connect to an external user database. On the other hand, it could also be an LDAP Inbound Security module, which delegates password checks to an external LDAP directory.

**Platform Requirements**

This page lists requirements for installing Smile CDR. This applies only to self-hosted versions of Smile CDR. If you are using a cloud-hosted version then this page does not apply.

Note that these are the requirements that are known to work with Smile CDR. If you have other requirements in mind, they may work as well. Please [contact us](https://smiledigitalhealth.com/get-started) to discuss other options.

**2.1.2Server Requirements**

The following list defines the minimum server requirements for a server hosting Smile CDR. Smile CDR may be configured in a 1-node cluster if high availability is not needed, but 2+ nodes are recommended for better reliability.

* **Environment**: Smile CDR may be installed to physical or virtual servers, or to [Docker](https://www.docker.com/) containers.
* **CPU**: A 2-core X86-64 CPU (required); 4-core X86-64 CPU (recommended).
* **Memory**: At least 4 GB of available RAM is recommended.
* **Operating System**: Deployment on GNU Linux is recommended. Ubuntu and RHEL/CentOS are both known to work, but other distributions are likely also fine. We do not currently support running Smile CDR on the Microsoft Windows operating system.
* **Disk Space**: Smile CDR requires an environment with at least 10 GB of free disk space. Note that large installations may require significantly more.
* **Database**: Smile CDR can be used with any of the database platforms listed [below](https://smilecdr.com/docs/getting_started/platform_requirements.html#database-requirements).

**2.1.3Java Requirements**

Smile CDR is a Java application and requires a Java Virtual Machine (JVM) or Java Development Kit (JDK) installed. This is not required when using the Docker distribution of Smile CDR, since that distribution includes a JDK.

Smile CDR is supported for use against one of the following distributions of Java:

* OpenJDK 21.0.1+ **(Supported)**
* OpenJDK 17.0.1+ **(Supported)**

Smile CDR will run against a simple JRE (compiler tools are not required in order to use Smile CDR) but the JDK comes with many tools which can be useful for troubleshooting issues (e.g. jstack, jmap, etc.)

**2.1.3.1OpenJDK Distributions**

There are many available distributions of the Java Virtual Machine (JVM) and Java Development Kit (JDK). These include AdoptOpenJDK, Azul Zulu, Amazon Corretto, and RedHat OpenJDK. These distributions are all based on the same core OpenJDK codebase, but each vendor may add additional patches and fixes.

We recommend the [Amazon Corretto](https://aws.amazon.com/corretto/) JDK.

**2.1.3.2Non-OpenJDK Distributions**

Smile CDR may also work correctly against non-OpenJDK distributions of Java (e.g. IBM J9 JDK) but these are not well-tested and are not recommended.

**2.1.3.3Forcing an Unsupported JDK**

You can skip the Smile CDR minimum Java version check by adding the argument -Dskipjavaversioncheck=true to your setenv file.

**2.1.3.4Previous supported versions**

| **Supported Java Versions** | **Smile Version** |
| --- | --- |
| 21.0.1+ / 17.0.1+ | 2024.02 - Later |
| 17.0.1+ / 11.0.1+ | 2022.05 - 2023.11 |
| 17.0.1+ / 16.0.1+ / 11.0.1+ / 10.0.1+ / 1.8.0+ | 2022.02 |
| 16.0.1+ / 11.0.1+ / 10.0.1+ / 1.8.0+ | 2021.08 - 2021.11 |
| 11.0.1+ (recommended) / 10.0.1+ / 1.8.0+ | 2019.05 - 2021.05 |
| 11.0.1+ (recommended) / 1.8.0 (supported) | 1 - 2019.02 |

**2.1.4Database Requirements**

Smile CDR requires a **relational** database to use as a store for ***system data***, such as configuration. A **relational** database or **non-relational** database can be used to store ***FHIR resource information*** and should be setup using the **UTF-8** character set, case-sensitive, and the standard 'C' locale.

