[INSERT LOGOS AS RELEVANT]

OPENLMIS SUPPORT REQUIREMENTS

[NAME OF COUNTRY/PROGRAM]

[INSERT DATE]

EXECUTIVE SUMMARY

Ensuring the infrastructure and processes are in place to set up, configure, deploy, and maintain the software, as well as provide end-user support is key to the success of any software implementation.

In order to facilitate the successful implementation of OpenLMIS, this document aims to provide information and recommendations for technical infrastructure and support covering various areas:

- Requirements for hosting and sustaining technical operation of OpenLMIS
- Support and maintenance processes at various levels, both technical and end-user support
- Roles and required qualifications/skills of support personnel
- Training needed to support the ongoing use of OpenLMIS

The basic requirements for a hosting a deployment of OpenLMIS version 3:

- A domain name to reach the installation
- An SSL certificate ensure secure web communication to OpenLMIS
- An application server (or instance)
- A database server (or instance) capable of running PostgreSQL for the various services
- Credentials with an SMTP server

Cloud hosting is recommended due to the reliability, security, and support commercial cloud hosting options offer for comparatively low cost, though local hosting is also an option so long as the necessary infrastructure can be provided and maintained.

Additionally, server and application monitoring and maintenance services are recommended. User sites require computers with recent operating systems (i.e. Windows 7-10) and a web browser (Mozilla Firefox or Google Chrome), along with sufficient internet connectivity and power.

Recommended technical support includes a four-tier approach, with each tier responsible for issues of increasing complexity. When issues are reported to the technical support team each level is responsible for investigation prior to escalating it to the next tier of support. The support tiers, roles, responsibilities, qualifications, and required training are summarized in the table below.

Support Tier	Role	Responsibilities	Qualifications	Training
Tier 1	Superusers (local staff)	• Act as local resource, can facilitate additional support	• Successful completion of superuser training	• System use • Basic troubleshooting • Issue reporting

	Technical officers (provincial staff)	• Respond to reported issues • Record/update issues in tracking system • Follow up to ensure users complete regular tasks • Support user trainings	• Experience providing user/customer support and training • Strong communication • Knowledge of /experience with health, supply chain, and/or OpenLMIS beneficial	• System features, use, and configuration • Basic troubleshooting • Support processes & responsibilities • Conducting user trainings
Tier 2	System administrators (central staff)	• Respond to reported issues with troubleshooting and resolution • Issue triage & management • Testing & QA • Engagement with OpenLMIS community	• Background/education in health, logistics, computer science, IT, informatics etc. • Knowledge of/experience in health and/or supply chain sectors highly recommended • Knowledge/experience with relevant tools	• Implementation features and workflows • OpenLMIS Core features • Reference data configuration • Advanced troubleshooting • Conducting user trainings • OpenLMIS Community engagement
Tier 3	Software Engineers (Central Staff and partners)	• Code modifications • Testing & QA • System administration and monitoring • System & server updates and maintenance • Engagement with OpenLMIS community	• Software engineering and/or database/system administration background/education • Experience and familiarity with OpenLMIS languages, tools, and principles • Knowledge of/experience with Agile development recommended	• OpenLMIS architecture, features, standards, & conventions • Reference data configuration requirements & processes (required data element, API calls) • Support processes & responsibilities • System monitoring & maintenance • OpenLMIS Community engagement
Tier 4	OpenLMIS Core team	• Core feature development • Respond to support requests • Provide guidance for development	N/A (Core team includes a variety of roles and personnel)	N/A

		approaches/ design		

The support team should also have standard processes for issue reporting, tracking, triage, and management, as well as release management and deployment processes. It is highly recommended that relevant tools be implemented to support these processes, such as Atlassian Jira and/or Service Desk.

Ensuring the appropriate infrastructure, support team, processes, tools, and resources are implemented and maintained will help ensure that an implementation of OpenLMIS is deployed and operates smoothly and effectively.

Table of Contents

ACRONYMS & ABBREVIATIONS

3G	Third generation (wireless mobile telecommunications technology)
API	Application programming interface
AWS	Amazon Web Services
FAQ	Frequently Asked Questions
IT	Information Technology
MoH	Ministry of Health
OpenLMIS v3	Open Logistics Management Information System version 3
OS	Operating system
RDS	Relational Database Service
REST	Representational state transfer
SLF4J	Simple Logging Facade for Java
SSL	Secure sockets layer
SMTP	Simple mail transfer protocol
SCM	Software change management
i18n	Internationalization and localization

1. INTRODUCTION

Ensuring the infrastructure and processes are in place to set up, configure, deploy, and maintain the software, as well as provide end-user support, is key to the success of any software implementation.

1.1 BACKGROUND

[insert relevant background information] here such as:

-project funders, parters,

-timeline

-assessments completed to date

1.2 OBJECTIVES

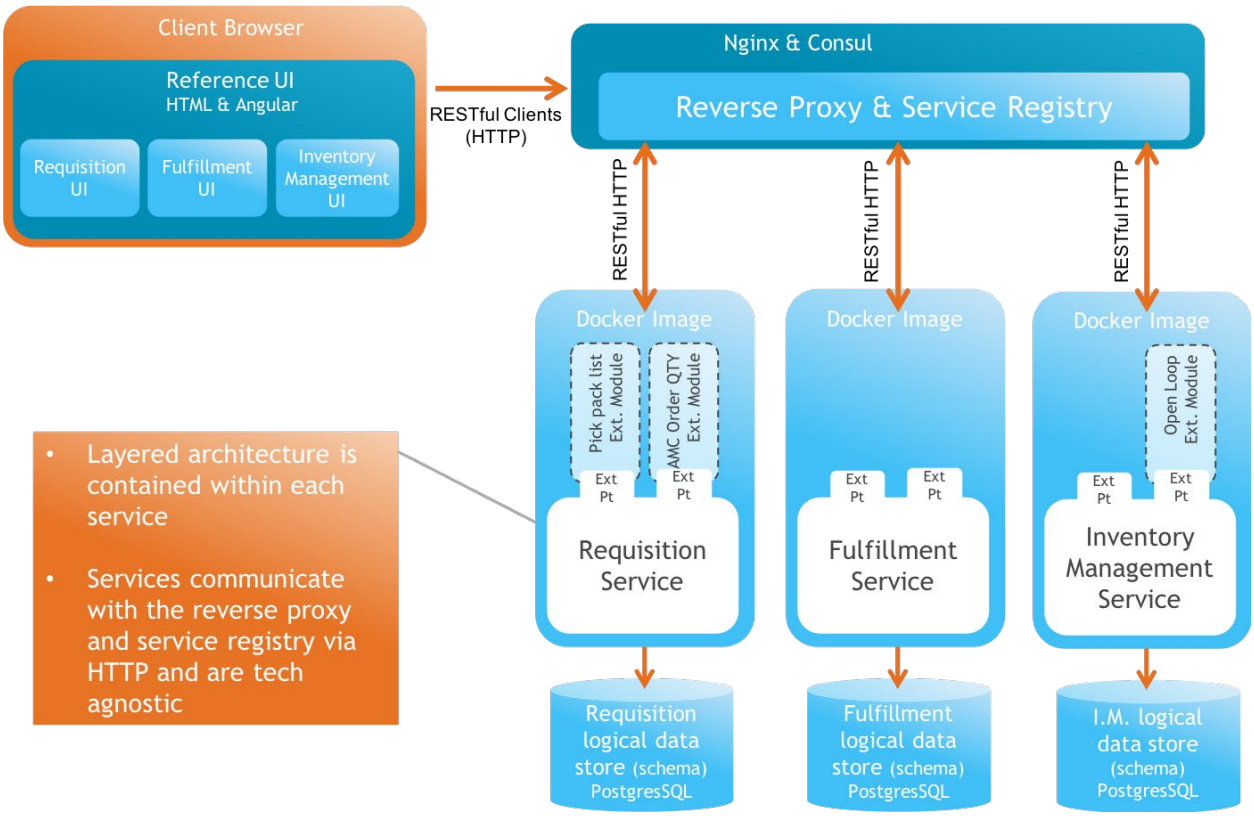
In order to facilitate the successful implementation of OpenLMIS, this document aims to provide information and recommendations for technical infrastructure and support covering various areas:

- Requirements for hosting and sustaining technical operation of OpenLMIS
- Support and maintenance processes at various levels, both technical and end-user support
- Roles and required qualifications/skills of support personnel
- Training needed to support the ongoing use of OpenLMIS

2. TECHNICAL INFRASTRUCTURE

OpenLMIS v3 is built using a microservices architecture designed to be modular, secure, scalable, and deployable without custom code, as well as supporting extensions (customizations) while maintaining backward compatibility and shared value. Figure 1 presents a high-level architecture diagram to provide a general understanding of the OpenLMIS v3 architecture. Resources for more detailed information and diagrams are provided in Annex 2.

Figure 1. OpenLMIS v3 Architecture Overview



In order to set up, configure, deploy, and continue to run OpenLMIS version 3, there are a number of requirements for the technical infrastructre supporting the software. This section outlines those requirements, including hosting, monitoring and maintenance, and user-site infrastructure. It should be noted that the specifics of some requirements, such as the hard drive and memory needed, will vary dependening on the size and scope of a given implementation.

2.1 HOSTING

Any OpenLMIS implementation will require several instances of OpenLMIS to be deployed and used for various purposes (see Table 1). The different instances are depoyments of the same application used for a variety of purposes.

Table 1. OpenLMIS Instances

Instance	Purpose	Required or Recommended?
Development	Software engineers use this instance for active development before moving code changes into the test/staging instance. Development may occur for several purposes: • System customization • Feature development	Required

	• Bug fixes • Incorporating OpenLMIS Core releases	
Test/UAT	Completed code changes are deployed in this instance for testing prior to release to production; also used for reproducing issues/defects	Recommended***** - separate instance recommended for large deployments with active development beyond basic operations and maintenance support
Training	End-users can access this system for training and orientation.	Recommended***** - separate instance recommended for large deployments with frequent training activities.
Production	The live instance in use	Required

*******A deployment is required to have at least one instance to use for testing/staging and training. Depending on the size and scope of the implementation (amount of data, number of users, frequency of training, level of active development) one instance could be used for both purposes.*

The basic requirements for OpenLMIS deployment are:

- A domain name to access the installation (e.g. https://validation.ao.openlmis.org)
    - There are no specific requirements for the domain name or format, though it is highly recommended to obtain a domain from the MoH so it is clear that the system is the official and legitimate information system.
    - Given the need for multiple instances, it is recommended to use simple variations of the same domain
- An SSL certificate ensure secure web communication to OpenLMIS (recommended to be obtained from MoH with the domain name)
- An application server (or instance) capable of running Docker Machine (as well as Compose, etc.) with sufficient bandwidth, processing power, memory, and storage to run many (6+) Services and associated utilities
- A database server (or instance) capable of running PostgreSQL for the various services
    - Automatic regular database backups should be set up
- Credentials with an SMTP server to process and send e-mails notifications and alerts

These requirements can be met either by a cloud hosting service or with local (in-country) infrastructure. Further details, as well as the benefits/disadvantages of each option are provided below.

Cloud Hosting

The OpenLMIS Core team uses Amazon Web Services (AWS), and therefore has the most documentation of specifications and needs for AWS hosting. OpenLMIS, however, does not require AWS hosting and could be hosted on a different service, such as Microsoft Azure. Cloud hosting, of some type, is the recommended option for several reasons:

- **Reliability & robustness**

    - Commercial cloud hosting platforms like AWS and Azure are extremely robust, with a large support staff and contractually guaranteed (and very high) uptimes/availability.
- **Scalability**
    - Commercial cloud hosting's extensive infrastructure allows for scaling quickly and easily. As need for space and services increase, it can be added with minimal effort.
- **Data security**
    - Hosting platforms provide significant guarantees related to data safety (the prevention of data loss) and security
    - Automatic database backups are provided
- **Cost**
    - Commercial hosting services are generally very low cost due to large economies of scale and can offer a very high level of service at a much lower cost than most individual organizations could realize on their own.
- **Ease of remote management**
    - In the event that any of the technical support and/or system administration and management is remote (not physically located in the same place as the local servers) with any regularity, ensuring remote access to the system is critical to adequate support. Commercial cloud hosting platforms provide such capabilities with their services, whereas a locally hosted instance would need to provide and maintain these capabilities as well as the rest of the technical infrastructure.

Local Hosting

While cloud hosting provides a number of advantages, as detailed in the previous section, some implementations may prefer to host the deployment locally for a variety of reasons including national policy and regulations. If an implementation does opt for local hosting, all of the basic requirements noted in the hosting section must be provided by the servers are hosted (see Annex 1 for additional information on server specifications). There is some additional infrastructure required for local hosting (services that are generally included in the contract for commercial cloud hosting):

- **Database backups** - An additional server in a separate location is required for storage of database backups. The backup server will require standard monitoring and maintenance.
- **Application server backup** - It is highly recommended that a parallel application server is deployed and maintained in the event the primary server encounters serious issues. The switch over from the primary server to the backup can be manual but it would be recommended that an automatic cutover is set up to minimize the impact to users.
- **Ability to scale** – Cloud hosting easily accommodates increasing the size and/or processing power of a hosting instance to support scale up. A locally hosted instance should begin with additional capacity, beyond the expected need, to provide a buffer and

allow for some scale up. The capacity of the servers must also be routinely reviewed to ensure that they are adequate and/or can be upgraded if/when necessary.

In addition to the basic infrastructure locally hosted instances must provide adequate security, monitoring, and maintenance to ensure the system operates smoothly and reliably, and that the data is safe and secure. Further, it is necessary to implement and support remote access for the OpenLMIS support team. An implementation could host OpenLMIS locally on their own, if they were to deploy and maintain their own servers. It would likely be costlier and less effective than cloud hosting. A local data center, either commercial or operated by the MoH/implementing organization, would be a preferred option, and is recommended for local hosting, assuming the capacity and capabilities required for support.

2.2 MONITORING & MAINTENANCE

In addition to the technical infrastructure required to run OpenLMIS, there are additional tools that should be put in place to support ongoing operations and system maintenance. While not required, these systems allow more effective support and maintenance and are strongly recommended:

- Server monitoring – implementing a tool to monitor server use and performance are highly useful for the support team to ensure the system is operating properly and identify any issues.
    - Typical metrics tracked include disk space usage, network bandwidth usage, error or exception logs, CPU usage, etc.
    - A system that provides alerts and notifications to the support team when an issue arises, or set threshold is met, is recommended to ensure potential issues are addressed before becoming a problem and/or are resolved quickly.
        - Scalyr is used for server monitoring by the OpenLMIS Core team and OpenLMIS implementation in Malawi
        - Most cloud hosting services include reporting capabilities on many of the typical server monitoring metrics, such as those noted previously.
- Google Analytics – Google Analytics is used by the OpenLMIS Core team as well as the Malawi implementation to track and monitor usage and site traffic, including the most commonly accessed pages and number of active users throughout a specified time period. While the information provided by Google Analytics is less critical than that provided by the server monitoring system, it can be useful for system management and evaluation purposes (and is free and low effort to set up).

2.3 INFRASTRUCTURE AT USER SITES

Any location where users are expected to access and use OpenLMIS require some basic infrastructure:

- **Computer**
    - Operating system: Windows 7-10 (other, similarly recent, OSs are also acceptable)
    - Screen resolution: 1000x600 / 1300x975 or higher
    - Monitor: there is no specific requirement for monitor size, however a 15-inch monitor or larger is recommended to ensure the best user experience
- **Web browser**
    - Chrome (v52 or higher) or Firefox (v49 or higher)
        - These are recommended for optimal user experience and supported by the OpenLMIS Core team (Core code changes are tested on both)
        - Other browsers (Internet Explorer, Safari, etc.) may also work but are not supported and therefore may be suboptimal

- **Internet connectivity**
    - OpenLMIS is a web-based system requiring an internet connection.
        - If using a mobile data network, a minimum 3G connection is required. Dongles or mobile hotspots and airtime should be provided (the amount of airtime will vary depending on the frequency of use and amount of data communicated)
- **Power**
    - Sufficient electricity to run the computer and internet (if applicable) to perform the required functions. Ideally a primary and secondary power source should be available.

3. TECHNICAL SUPPORT

The support needs of an information system like OpenLMIS encompass a variety of tasks from development, to system monitoring, to responding to end-user questions. Establishing a support team and standard processes to ensure that the system is operating smoothly, as intended, and is meeting the expectations of the end-users is critical to a successful deployment. This section outlines the requirements and recommendation for the personnel, processes, and tools to support an OpenLMIS deployment.

3.I PROCESSES

In addition to the personnel, there are a few key processes that should be in place to govern system support and ensure support is effective, efficient, and satisfactory for end-users. Key processes include:

- Issue Reporting
- Triage
- Release Management

A description and recommendations for each process follows.

Issue Reporting & Escalation

A standard process for issue reporting helps ensure that users are aware of how to contact support and report issues, as well as that the support team logs issues in a consistent manner, enabling triage and tracking.

Users should have multiple channels by which to contact the support team and report issues:

- Phone
    - Dedicated line/number
    - WhatsApp groups
- E-mail - to dedicated support address, actively monitored by the support team)
- Web-based – online help desk, embedded in the system

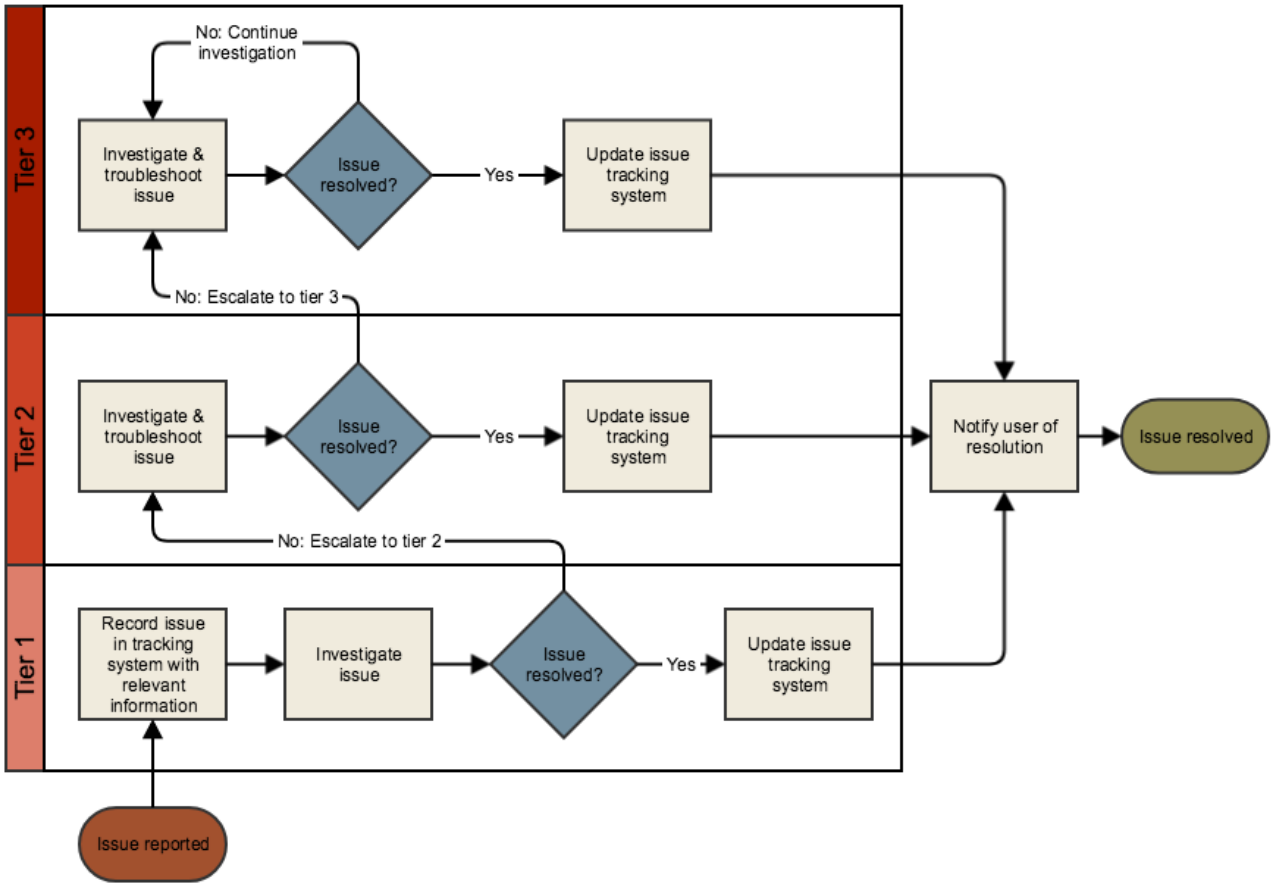
- Slack?

Support personnel (see description and Table 5. in section III.2) should collect a standard set of information about reported issues to enable troubleshooting, such as the user/facility/program etc. affected, how the issue is occurring, and the frequency with which it is occurring. The issue and relevant information should be logged (see Issue Reporting Tools section) and investigated. Once reported and logged each issue should be investigated and escalated as needed for a resolution based on the complexity and in accordance with responsibilities of each tiers of support (see Figure 2). Once a resolution is identified, the user(s) reporting and affected by the issue should be notified of the resolution. The notification channels and process will vary based on the tools used for issue reporting, tracking, and management. Users could be notified through several means:

- Phone call / WhatsApp group update
- Individual e-mail
- Mass e-mail to all OpenLMIS users
- Update to customer help desk portal

The specific user communication channels, processes, and responsibilities (for example, Tier 3 support may be responsible for updating the issue in the help desk, but Tier 1 may be responsible for calling the user) should be defined in the issue reporting and escalation SOPs.

Figure 2. Issue Escalation Process



OpenLMIS Core Issues v. Implementation Issues

Issues reported to the implementation's technical support team will range from updating a user permissions, to updating translations, to addressing a bug, to requests for new features. Many of these issues will be handled exclusively by the implementation support team. In some cases, however, the implementation technical support team can involve the OpenLMIS Core team:

- Report a bug in the Core product
- Seek feedback from Core/the Community on design of a fix or feature
- Requesting/advocating for a new feature in the Core team

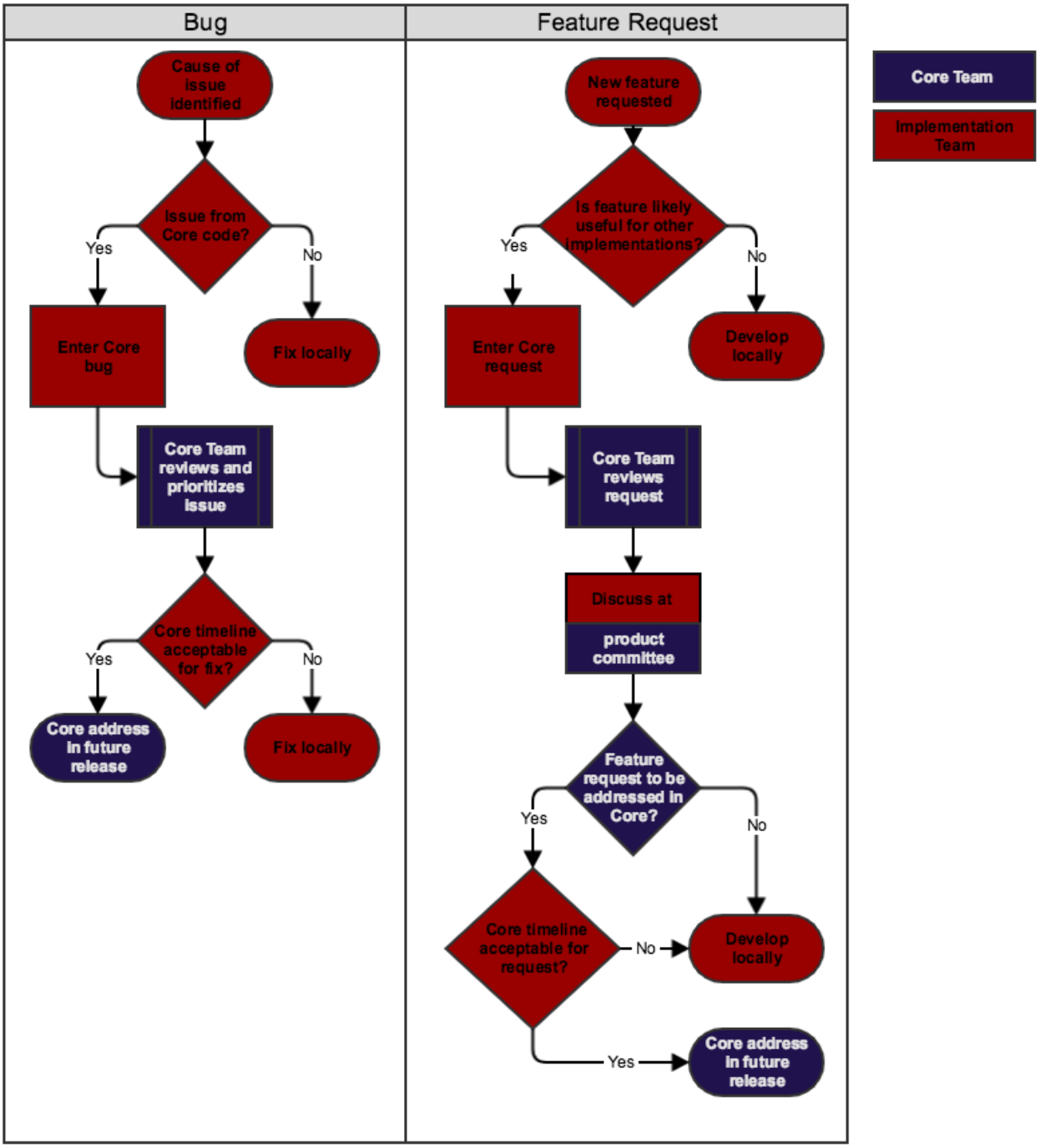
The OpenLMIS community uses JIRA for tracking bugs and all bugs must be submitted to the OLMIS project in order to be reviewed. To report a bug:

- Search for the/a similar bug (ensure you are not entering a duplicate)
    - If it exists, review the status and (if necessary) comment to reopen or raise the priority
- If it does not exist, open a new bug fill out all required information and include a detailed description and steps to reproduce the bug

- Include screenshots, error message text, stack trace, or any available reference material as an attachment

Figure 3 provides a general overview of the process of identifying whether or not to involved the Core team.

Figure 3. Core Team Involvement



See the Contributing to OpenLMIS information Annex 2 for more detailed information about the OpenLMIS Core teams issue reporting, triage, and management. This information also includes details about managing translations. OpenLMIS v3 includes translation keys and strings in each component, managed using Transifex. Community members can contribute translations to the Core product or, if it is a specific local translation, override the strings with custom translations (see Annex 2 for more details).

When bugs or feature requests are made to the Core team, they will be reviewed and prioritized along with the other items in the OpenLMIS Core backlog, to be addressed in a future release of OpenLMIS Core. An implementation can advocate for the Core team to prioritize their issues or requests by participating in the community forums and committees (the Product Committee in particular). Reporting an issue or making a request to the Core team, however, does not guarantee it will be high priority and/or addressed in the timeline required by the implementation. Even if an issue is a Core issue, the implementation development team can address the issue (or request) themselves, and contribute it back to the Core product. In such cases, the implementation team must determine whether or not it is feasible to wait for the Core team to address an issue.

OpenLMIS v3 includes translation keys and strings in each component, managed through Transifex. The community can contribute translation to the OpenLMIS Core Transifex projects or, for specific local translations, override the strings with custom translations (see Annex 2 for further detail).

Issue Triage, Tracking, & Management

The OpenLMIS support team is responsible for reviewing, troubleshooting, and triaging the issues reported. An issue triage team, ideally made up of three team members, should meet periodically (daily - weekly) to complete the following process*****:

1. Review all issues, starting with newly reported issues
    a) Review issue description, component, and troubleshooting attempted
2. Update issue type (bug, support request, new feature etc.) and support tier as necessary
3. Assign/update the issue's priority. Standard priority definitions should exist (see Tables 3 & 4) and based on the review of the ticket, the group determines the most appropriate for the issue.
4. Assign issues to support team members for further investigation/troubleshooting.
5. Update and escalate issue to tier 3 to refer to development team for additional investigation if the issue cannot be resolved in tier 2.

*******The process outlined is a sample recommended based on standard practices. Specific details such as the frequency of triage meetings, priority definitions, and issue categorizations may vary depending on the needs of the implementation and tools in use.*

The prioritization of an issue should consider both the impact of the issue (or the number or percentage of users affected) and the urgency (the severity of the issue). Both measures are important to identify the appropriate priority; a widespread but minor issue (extensive impact, low urgency) should not be prioritized above an issue that is moderate impact, affecting a subset of users, but critical for those users (i.e. they cannot complete their normal job functions).

Table 3. Impact & Urgency Definitions (Sample)

Impact	Definition
Extensive	Widespread issue – affects all or most users/facilities/functions
Significant	Large issue – affects many users/facilities/functions
Moderate	Limited issue – affects some users/facilities/functions
Minor	Localized issue – affects one or a small, specific group of users/facilities/functions

Urgency	Definition
Critical	Severely impedes work (crashing, data loss etc., no viable workaround)
High	Loss of function, impedes work (workaround may be available but is insufficient)
Medium	Loss of function, doesn't interrupt work and/or workaround is available
Low	No loss of function, cosmetic problem that doesn't interrupt work

Table 4 provides a matrix to display how the impact and urgency measures are combined to identify a priority: Blocker; Critical; Major; Minor; or Trivial.

Table 4. Priority Definitions based on Impact & Urgency

		Urgency			
		Critical	High	Medium	Low
Impact	Extensive	Blocker	Blocker	Critical	Major
	Significant	Blocker	Critical	Major	Minor
	Moderate	Critical	Major	Minor	Trivial
	Localized	Major	Minor	Trivial	Trivial

Release Management

Periodic system updates (releases) after the initial deployment are needed to implement code changes to the existing system, such as bug fixes, enhancements, or new features. The periodic

releases can include changes made specifically by the implementation development (tier 3 support) team as well as incorporate new releases of OpenLMIS Core.

OpenLMIS v3 is comprised separate, standalone and deployable 'services' or 'components'. A release of OpenLMIS will contain updated versions of the various services and components. This architecture allows implementations to mix and match different sets of these services, and extending (or customize) specific pieces, while still staying connected to the global OpenLMIS codebase. An implementation can receive updates, bug fixes, security and performance improvements, and new features from the global community.

Establishing and documenting a release management process for each implementation to make periodic updates will help ensure the deployment of updates are consistently handled properly. A release process document details the manner in which the OpenLMIS development team creates, publishes, and deploys a release of a new version of OpenLMIS. The release management process should set standards and procedures for several areas:

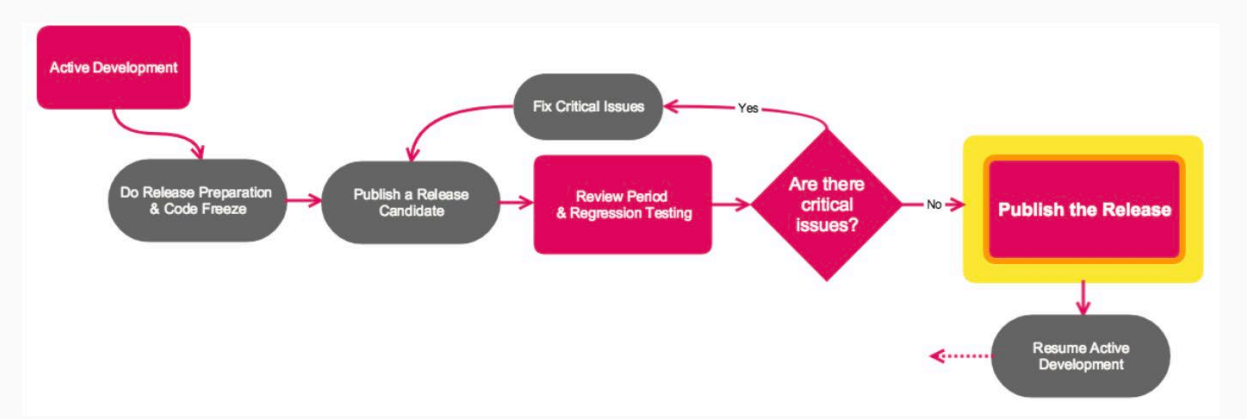
- Release process SOPs (including emergency/patch releases)
- Testing & QA
- Release schedule (including emergency/patch releases
- Building a release
- Deployment
    - Prerequisites
    - Approval
    - Deployment process (including emergency/patch releases)
    - Rollback
    - User notification & support

The release management process governs the standard procedures to follow for releases. It is recommended that the process include the creation of a release deployment plan for each individual release, to documents the specific activities, timelines, testing, review, approval, and necessary support for the release. Annex 2 includes links to release deployment plan and schedule templates included in OpenLMIS' implementor toolkit.

OpenLMIS Core Releases v. Implementation Releases

The Core team is responsible for the ongoing development of the Core product and an implementation team is responsible for development and management of their deployment of OpenLMIS including any extensions (customizations), UI changes, or localized language translations that are not part of the Core product. The OpenLMIS Core team follows their own release process, outlined in Figure 4, to create, test, and release new versions of OpenLMIS Core (see Annex 2 for more detailed information about the Core release process).

Figure 4. OpenLMIS Core Release Process



The OpenLMIS Core team includes a testing and review period that is open to the community, and actively encourages community members, especially existing implementations, to participate in the Core release candidate review. This can be completed in the Core testing instance, or by incorporating the new release into the local testing/UAT instance for testing. OpenLMIS-Malawi uses the latter process to provide feedback to the Core team regarding the release. Figure 5 provides an outline of the responsibilities for Core versus implementation teams in releases.

Figure 5. Core v. Implementation Release Responsibilities

OpenLMIS Core	Implementation

• **Create release candidate**
• **Testing & QA for Core Product:**
  • New Core features
  • Core language translations
  • Core backward compatability
  • Minimum performance standards (maintaining at least the previous performance levels)
• **Fix critical Core issues**
• **Publish Core release**
	• **Incoprorate Core releases into local instance**
	• **Participate in Core release testing and review and provide feedback**
	• **Create implementation-specific release candidate**
	• **Testing & QA for implementation instance:**
	  • Implementation-specific features, enhancements, and bug fixes
	  • Local langauage translations
	  • Implementation-specific backward compatability
	• **Deploy release**
	  • Deployment, user notification, support, and training as defined by the release management process and release deployment plan

3.2 TEAM

Supporting an OpenLMIS deployment with the processes outlined previously requires a variety of staff to meet the needs at each level (both administrative levels as well as support levels). Support is typically organized in tiers, with the complexity (and required skills and qualifications) of responsibilities increasing in higher tiers:

- **Tier 1**
  - Tier 1 is the first point of contact for user support and is responsible for responding to basic end-user issues and questions. Tier 1 personnel should have a solid general understanding of OpenLMIS and the intended processes. They can answer end-user questions about workflow and processes, as well as help resolve basic technical issues.
- **Tier 2**
  - Tier 2 support personnel are system experts with in-depth knowledge of OpenLMIS features, capabilities, and configuration, as well as the business processes and workflow. Tier 2 staff respond to more complex issues that those typically resolved by tier 1 including reference data configuration, user set up and permissions, and advanced troubleshooting. Tier 2 support do not make code changes or system /database administration escalated: these types of issues are escalated to tier 3.
- **Tier 3**
  - System and database administration including server configuration, monitoring, and maintenance (some support and maintenance may be provided by the cloud hosting service or data center, if a commercial service is used)
  - Software development, including fixing defects, making enhancements, new feature development, database changes, or merging OpenLMIS Core updates is the responsibility of the tier 3 support team, made up of software engineers.
- **Tier 4**
  - The OpenLMIS Core team/OpenLMIS Community can be considered a fourth tier of support. The specific implementation is not responsible for maintaining or funding the tier 4 support but to utilize the Core team's support the level two and three support staff should regularly engage with the OpenLMIS community.
  - Tier 4 support is not a requirement of deploying OpenLMIS v3, but strongly recommended as Core improvements and support will benefit the implementation.
  - Engagement would include attending community meetings (such as the technical or product committee meetings) and participating in the online community portals and forums. Engagement with the community will facilitate the ability of the level three support team to escalate feature requests, major issues, and other challenges to the Core team along with other implementations. Feedback about existing and desired features, enhancements, or issues encountered will continually improve the Core product, and extensions of the system for specific implementations can be

contributed back, ensuring ongoing support for those features from the Core team (see Annex 2 for additional resources about contributing back to the Core product).

Table 5 provides a breakdown of the roles, responsibilities, and necessary qualifications for each tier of support as well as where the relevant personnel would be located.

Table 5. Support Roles, Responsibilities, & Qualifications

Support tier	Role	Responsibilities	Qualifications	Location
Tier 1	Superusers	• Support other OpenLMIS users by ensuring processes are clear and basic questions answered • Act as local resource, can facilitate additional support	• Qualifications relevant for their primary position (i.e. pharmacist, warehouse clerk, program manager) • Successful completion of superuser training	Lower administrative levels (municipalities, districts, etc.)
	End User support (First point of contact)	• Respond to reported issues • Record/update issues in tracking system • Follow up to ensure users complete regular tasks • Support user trainings	• Experience providing user/customer support and training, ideally for software/technology • Strong communication • Knowledge of /experience with health, supply chain, and/or OpenLMIS beneficial OR demonstrated ability to learn system and relevant business processes	Provincial and/or Central
Tier 2	Technical Support Officers	• Respond to reported issues with troubleshooting and resolution • Issue triage & management • Testing & QA • Engagement with OpenLMIS community, particularly product committee	• Background/education in health, logistics, computer science, IT, informatics etc. • Knowledge of/experience in health and/or supply chain sectors highly recommended	Central

Tier 3	System & Database Administrators Software Engineers	• Troubleshoot and resolve issues requiring code changes • New feature development • Testing & QA • System administration and monitoring • System & server updates and maintenance • Engagement with OpenLMIS community, particularly technical committee and developer forums	• Software engineering background/educations • Experience and familiarity with various languages, tools, and principles: - Java - JavaScript (AngularJS 1) - Docker - Consul - Python - PostgreSQL - Apache Zookeeper - Apache Kafka - REST API - GitHub - Jenkins • Knowledge/experience with PostgreSQL, Jenkins, and hosting/monitoring tools (such as AWS, Azure, AWS RDS, Scalyr etc.) • Knowledge of/experience with Agile development recommended	Central*
Tier 4	OpenLMIS Core team	• OpenLMIS Core improvements and feature development • Respond to support requests from implementation tier 3 support teams • Provide guidance and suggestions for development approaches/design	N/A (Core team includes a variety of roles and personnel)	Remote

*****_Depending on the implementation, software development resources could be remote, though closer to the implementation is recommended._

Recommendations

Further recommendations related to support personnel:

- **Quantity:**
  - The number of support personnel at each level is dependent on the size and scope of an implementation (number of users, sites, programs, etc. being supported). For a national-scale implementation supporting at least one significant program, such as Essential Medicines, a minimum of six staff between tier 2 and 3 is recommended.
  - Quantity of tier 1 support staff will vary based on the number users and number and size of geographic areas. Ideally at least one superuser should be available per municipality/district (or equivalent). There should be sufficient technical officers to ensure routine site-visits and maximize availability and accessibility to superusers and end-users.
- **Placement:**
  - Providing as much – or as many tiers – of support locally as possible is recommended to increase the timeliness, quality, and sustainability of system support.
  - Tier 1 support should be as close to users as possible, to ensure users receive the support and feedback they need; improving user experience and system acceptance.
- **Sustainability:**
  - Providing all or much of the support locally helps make the system more sustainable. Providing upper-tier support locally requires the identification of a partner (or the individuals) and likely significant capacity building – at least with OpenLMIS itself. Involving the local support team for all tiers as early as possible in the implementation will ease this process. Additionally, starting with (or having a significant handover period with) a combined support team, where locally-based developers and system administrators can work with remote/temporarily local teams with significant OpenLMIS knowledge and experience, will help to ensure support can be provided locally.
  - Further, it is common for implementations to have teams dedicated to supporting a specific system (such as OpenLMIS). This specialization can make it difficult to ensure continuity and maintenance of institutional knowledge, as well as manage the variations in support needs (both type and volume of requests). Ideally, the MoH (or implementing organization) could have a general information systems/technical support team (or contractor) responsible for all or several information systems.

3.3 TOOLS

To facilitate the implementation and persistence of the processes identified, it is useful to also implement some basic tools (see Annex 3 for complete list of software, langagues, tools, etc. required or recommended for deployment, development, support, and maintenance of OpenLMIS).

Issue Tracking & Management

In order to support the issue triage, tracking, and management, operational support teams log issues in ticketing systems, such as Atlassian Jira (used by the OpenLMIS global team and OpenLMIS deployment in Malawi). A ticket tracking system such as Jira provides capabilities to track and manage the issues and as well as visibility in to the volume, status, and quality of support. The ticketing system allows issues to easily be reviewed, prioritized, categorized, and updated. Jira and other ticketing systems also offer reporting to track support-related metrics (such as time to resolution, issues created and resolved etc.). Jira also includes a help desk application – Jira Service Desk – that allows issue reporting and communication between support staff and users.

Issue Reporting

As noted previously, users should have several channels for reporting issues, including phone, e-mail, WhatsApp, and/or web-based tools: users who can easily report issues and receive timely feedback will have a better experience. While many of those channels don't require specific tools (just the creation of the dedicated e-mail, phone number, or WhatsApp group), some ticketing systems, such as Jira Service Desk, allow for a link to the support site to be embedded in the application and/or automatically create a ticket from an e-mail, facilitating the reporting and logging of issues, and reducing the workload on the support team. The OpenLMIS Core team uses Jira for issue reporting and management (refer to the Issue Reporting & Escalation Process in section III.1 and additional resources in Annex 2 for more detail).

Additionally, the creation of an issue report forms or guide outlining the basic questions to ask and information to include on an issue report can help ensure as much of the relevant information as possible is gathered in the initial report. While the form or guide can be provided to users, it can also be very useful for tier 1 support staff.

Resources

In addition to the recommended tools, the development of various resources is recommended to support both end-users and the support teams:

- **User guide & training materials**
    - Users should have access to a full user guide detailing the functionality and processes supported by OpenLMIS
    - Additional training materials, such as demo videos or presentations, can also be useful resources for users
- **Quick reference guide(s)**
    - Short 1-2-page informational guides providing FAQs and basic workflow instructions, ideally different versions targeted at different user roles (a store room manager versus a program manager etc.)
- **Troubleshooting guide(s)**
    - The implementation should provide Tier 1 support staff with a troubleshooting guide that outlines common issues and basic steps that should be taken to resolve/investigate them prior to escalating the issue.
- **Technical documentation**
    - Implementers and, subsequently, Tier 2 and 3 support should create and maintain technical documentation for the implementation's specific infrastructure, conventions, and extensions (customizations) made to the OpenLMIS Core product. Technical documentation is vital to ensuring continuity of support.

4. TRAINING

To effectively support OpenLMIS, the support personnel require the relevant training and knowledge. Table 6 outlines the OpenLMIS training and/or knowledge required for each tier of support.

Table 6. Support Personnel Training Needs

Support Tier	Required Training

Tier 1: Superuser	• Superuser training – basic user training with additional information including:
	  ○ Basic troubleshooting (password resets, browser updates, etc.)
	  ○ Issue reporting

Tier 1: End User Support	• In depth training/knowledge of:
	  ○ System features and workflows
	  ○ Configuration (existing set up)
	  ○ Basic troubleshooting
	  ○ Issue reporting process and tools (Jira, Service Desk, etc.)
	  ○ Support processes & responsibilities
	  ○ Conducting user trainings

Tier 2: Technical Support Officers	• Advanced training/knowledge of:
	  ○ Implementation features and workflows
	  ○ OpenLMIS Core features and standards
	  ○ Reference data configuration requirements & processes
	  ○ Advanced troubleshooting
	  ○ Issue reporting process and tools (Jira, Service Desk, etc.)
	  ○ Support processes & responsibilities (including release management process)
	  ○ Conducting user trainings
	  ○ OpenLMIS Community engagement

Tier 3: System & Database Administrators / Software Engineers	• Advanced training/knowledge of:
	  ○ OpenLMIS architecture, features, standards, & conventions, in particular:
	  ▪ Micro-services
	  ▪ Extension (customization)
	  ▪ Contributing to Core
	  ▪ Relevant tools, languages, and libraries (see Annex 2, 3, & 4 for detail)
	  ○ Reference data configuration requirements & processes (required data element, API calls)
	  ○ Issue reporting process and tools (Jira, Service Desk, etc.)
	  ○ Support processes & responsibilities
	  ○ System monitoring & maintenance
	  ○ Database administration
	  ○ OpenLMIS Community engagement

ANNEX 1. HOSTING SPECIFICATIONS

The OpenLMIS Core team has outlined their recommendations for **deployment topology** on the OpenLMIS' **ReadtheDocs** site. It focuses on specifications for AWS and is reproduced below with annotations to note the actual sizes in non-AWS terms.

In AWS (a region close to your users) this would be:

- A DNS provider for your domain name (e.g. test.openlmis.org). This could be Route 53, however currently OpenLMIS deployments do not utilize this service.
- An SSL certificate from AWS Certificate Manager
- An ELB that can route to/from the OpenLMIS instance and serve the ACM SSL certificate (this becomes more useful when running out of Elastic IPs)
- An EC2 Instance (m4.large - 2vCPU, 8GiB memory, 30GB EBS store)*****
- An RDS Instance (db.t2 class instances are used below because most OpenLMIS deployments have long periods of inactivity where the t2 class' credits are able to accumulate. Choose another class if your deployment will be actively used for more than half the day):
    - For local development, QA, and small private demos: use Ref Distro's included database or a db.t2.micro****** (though you'll need to increase the max_connections parameter to >150)
    - For CD, public demos, UAT, and small production: db.t2.medium*******
    - For medium and larger production instances: db.t2.large******** and up based on need:
        - When reports are frequently run
        - When the number of Products (full and non-full supply) > 500
        - When the number of Requisitions (historical and planned for next 2 processing periods) > 100,000
- a VPC for your EC2 and RDS instances, with appropriate security group - SSH, HTTP, HTTPs, Postgres (limit source to Security Group) at minimum.
- Amazon SES with either the domain (w/DKIM) verified or a specific from-address

***** *A "2vCPU" refers to a dual core 2.3 GHz CPU; "30GB EBS store" refers to a 30GB hard drive*

****** *A "db.t2.micro" is a single-core processor with 1 GB of RAM*

******* *A "db.t2.medium" is a dual-core processor with 4 GB of RAM*

******** *A "db.t2.large" is a dual-core processor with 8 GB of RAM*

For reference, the OpenLMIS Malawi deployment specifications are included below:

OpenLMIS-Malawi Implementation Overview:

- National deployment
    - 700+ facilities (national hospitals through health centers) across 3 regions & 28

districts
    - 600+ users
- 6 programs ~ 2000 products overall
    - Essential Medicines - 268 full supply products, ~1600 non-full supply
    - Malaria - 16 full supply products, 0 non-full supply
    - HIV – 28 full supply products
    - Reproductive Health – 12 full supply products
    - Tuberculosis – 24 full supply products, 20 non-full supply
    - Nutrition – 5 full supply products
- Malawi's deployment includes the following services:
    - Custom-UI
    - Reference data
    - Authentication
    - Notification
    - Fulfilment
    - Requisitions
    - Reporting (embedded Jasper reporting framework
- 5 years of historic data migrated

Malawi's current server configuration:

OpenLMIS Application Server

- CPU: Dual-core 2.4 GHz Intel Xeon® E5-2676 v3 (Haswell)
- RAM: 8 GB
- Hard drive capacity: 86 GB, currently using about 54 GB

Database Server

- CPU: Dual-core 2.5GHz Intel Xeon
- RAM: 8 GB
- Hard drive capacity: 100 GB, currently using about 27 GB

Malawi does not currently use the new reporting and data warehousing and visualization tools deployed with OpenLMIS v3.4 (to be fully integrated with OpenLMIS with the 3.5 release).

ANNEX 2. ADDITIONAL RESOURCES

The OpenLMIS Core team has significant documentation available to support implementors in planning and pursuing deployments of OpenLMIS. The OpenLMIS Core Confluence page, ReadtheDocs site, and Implementor Toolkit (on www.openlmis.org) house these resources. For reference, key documents are noted below.

- OpenLMIS v3 architechture: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/51019809/Architecture+Overview+Version+3)
    - Description and detailed information on OpenLMIS v3 architecture
- Technical set up guide: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/115681666/OpenLMIS+Technical+Set up+Guide)
    - Provides an overview of the techincal set up required for an OpenLMIS deployement for technical impolementation managers or developers preparing for an implementation
- Key developer links: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/138018819/Essentials+Toolbox+Index+of+most+important+link+for+the+devs)
    - OpenLMIS has extensive documentation, this page contains a list of links to the most critical pages for developers working on OpenLMIS
- Developer Guide: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/26574890/Developer+Guide)
    - Contains all developer resources
- New Developer Onboarding (https://openlmis.atlassian.net/wiki/spaces/OP/pages/392233014/New+Developer+Onboarding+with+OpenLMIS)
    - Guide and resources for developers new to OpenLMIS to become familiar with the product, code, community, processes, and tools used for OpenLMIS development
- OpenLMIS Configuration Guide: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/112138794/Configuration+Guide+for+Implementer+Administrator)
- OpenLMIS Implementor Toolkit - Maintain: (http://openlmis.org/4-maintain/)
    - Includes downloadable templates for:
        - Support plan
        - Release deployment plan
        - Release deployment scheudle
- OpenLMIS Core release management process: (http://docs.openlmis.org/en/latest/conventions/versioningReleasing.html)
- Contributing to OpenLMIS: (http://docs.openlmis.org/en/latest/contribute/contributionGuide.html)
    - Includes guidelines to reporting a bug to the Core team, links to Core tools and the developer forum, and localizing language files

    - OpenLMIS Transifex projects & UI Extension Guide
- OpenLMIS Jira: (https://openlmis.atlassian.net/projects/OLMIS/issues/)
    - OpenLMIS Core backlog and issue reporting
- OpenLMIS Frameworks & Libraries (https://openlmis.atlassian.net/wiki/spaces/OP/pages/56918061/Frameworks+Libraries)
    - List of OpenLMIS' relevant frameworks and libraries
- OpenLMIS Software Development Methodology: (https://openlmis.atlassian.net/wiki/spaces/OP/pages/57409539/Software+Development+Methodology)
    - Description and definitons of Agile software development methodolgy and tools used by OpenLMIS Core team

ANNEX 3. OPENLMIS SOFTWARE TOOLS & LANGUAGES

OpenLMIS v3 uses or recommends a number of software, tools, languages, and frameworks for development, deployment, and support. A list of these and their use is provided in Table 7 (see additional resources in Annex 2 for details).

Table 7. OpenLMIS Software Tools & Languages

Tool, Language, or Framework	Use
Development	
Apache Kafka	Stream processing
Apache Zookeeper & Consul	Distributed service management and synchronization
Authentication/SSO	Spring Security: B. AUTH; OAuth
Docker	Deployment packaging (Docker Hub & Docker Compose)
GitHub	SCM tool
Jasper Engine	Reporting
Java	Primary language
Java Resource Bundle & Transifex	Localization and il8n
JavaScript (AngularJS 1)	UI framework
Jenkins	Countinious delivery and integration
Junit, Mockito, PowerMock	Integrated testing
PostgreSQL	Database
Python, Shell, Ruby	Additional languages
REST API	UI separation
SLF4J to Syslogd with Docker Engine	Logging
SMTP & Spring Integration Channels	Notifications
Style-Guide, CheckStyle, Sonar	Static Analysis
Swagger	API Documentation
Other (recommended)	

ReadTheDocs	Documentation
Atlassian Confluence	Collaborative community documentation
Atlassian Jira	Ticket and issue tracking and management
Agile Software Development & Principles for Digital development	Software development methodology
Scalyr	System monitoring and alerts
Google Analytics	Application use monitoring