Primero “Tier 1” DIY Deployment
Resources Allocation and Work Plan

Introduction

Primero is a digital public good designed for social impact. Primero's application code is open source and is freely available to use. However, as with any software, there are costs associated with hosting, development, deployment, operations and maintenance. We are committed to managing Primero sustainably and securely. This is our obligation to our stakeholders and data subjects. We also want to make this transformative tool available to as many organizations and partners as possible. To do this, we've designed a sustainable business model that allows us to be cost effective, predictable and scalable. You can read more here. When you are looking to set up Primero, there are programmatic considerations that must be taken into account including coordination with partners and data security and information sharing processes. Please consult with the in-country coordination body when looking to set up a system for case management.

For those who have opted for using the “Tier 1” DIY Open Source Digital Public Good model, this provides you with access to the open-source repo to set up your own Primero instance. You have unlimited access to the Primero Community and all documentation. To implement Primero successfully, partners must have the required technical capabilities. This includes, but is not limited to, teams capable of setting up servers, security, DNS, configuration, maintaining updates and providing support. Partners must be able to run Ansible scripts available on GitHub to receive new releases and updates. Partners should also have the capacity to monitor and manage updates and keep their stack and infrastructure updated. Tier 1 allows partners to utilize the baseline configurations of CPIMS+ but must manage their own configuration and localization. As this option is fully managed and maintained by the partner, costs for their own human resources and IT costs such as fees for hosting, SSL/TLS, and DNS (costs vary by service) must be covered by the partner.

The open source model for Primero is an implementation model we call “Tier 1”. This model has 2 sites, one “*demo*” and one “*production*”. We use the demo CPIMS+ to manage configurations (forms, roles, reports), to receive updates and security patches, and to test and train. *Demo has no real data on it*. For the production site, the selected data center is managed by a capable technical team who can support the infrastructure set up and

maintenance. All the *real* data is stored on this server. This implies a higher level of accountability and management for the in-country team.

Roles and Responsibilities

In this model, it should be clear that the Primero Team is not responsible for managing or supporting any of the following and partners are responsible for:

1. All infrastructure including setting up servers, hosting, setting up monitoring and security measures, security certificates, data durability/data storage, backups, DNS, and configuration of the demo and production instance
2. Owning, protecting and securing the data
3. Disaster recovery protocols and data breach protocols
4. Installation of the latest supported version to local infrastructure
5. Configuration promotion from in-country-hosted demo to locally hosted production including updates and possibly performing occasional manual data migration
6. Clear process in place for configuration promotion from “demo” to “production” to receive the most up-to-date releases (which include updates and security patches)
7. Primero v2 is a progressive web application (PWA) and we have an identity provider that helps us securely authenticate and manage users which your team would need to implement. If a mobile device management solution is requested for mobile devices, this must be procured and maintained.
8. Checking for service availability; Docker is up and running and containers are up

The set-up of local infrastructure demands ongoing technical support and budgets. Therefore, you must select a technical partner which has demonstrated a very good capacity, strong technical infrastructure skills and understanding of the work for the project sustainability.

Summary of Skills Required for Tier 1

Below is a summary of the key skills required to provide ongoing technical support for Tier 1. You must have a team or select a technical partner which has demonstrated a very good capacity, strong technical infrastructure skills and understanding of the work for the project sustainability.

DevOps

Docker Proficiency

- Containerization Knowledge: Deploying applications using Docker containers for isolation and scalability.
- Lifecycle Management: Handling container deployment, restarting services, and monitoring performance.
- Debugging and Troubleshooting: Identifying and resolving issues within the containerized environment.
- Optimizing Resource Utilization: Managing resource allocation to ensure efficient CPU and memory usage.

SQL Server Management

- Installation and Configuration: Setting up SQL servers and configuring them for optimal performance.
- Database Administration: Performing backups, restorations, and data migrations.
- Security Management: Implementing access controls and encryption to safeguard sensitive data.
- Performance Tuning: Optimizing query performance and indexing strategies.

Ubuntu System Administration

- User and Permission Management: Creating and managing system users, groups, and roles.
- System Security: Configuring firewalls, managing SSH keys, and setting up intrusion detection.
- Monitoring and Logging: Implementing tools for real-time system monitoring and log analysis.
- Automating Tasks: Using shell scripting and cron jobs for routine maintenance and updates.

Server Networking and Integration

- Multi-Server communication: Establishing communication between multiple servers for load balancing and fault tolerance.
- Network Configuration: Setting up virtual networks, subnets, and routing.
- Cloud Integration: Deploying services on cloud platforms and managing hybrid environments.

- Scalability and Redundancy: Implementing failover mechanisms and high-availability solutions.
- Backup: Implementing backup strategy to ensure disaster recovery

You can read more about devops in the following resources:

https://learn.microsoft.com/en-us/training/career-paths/devops-engineer?

https://www.udemy.com/course/devops-with-docker-kubernetes-and-azure-devops/?srsltid=AfmBOoqSwhp_rEbE_HWAnxTWwTLZIVhRcCVcAh5sWB9gKjfk3k6rNmt3

Networking

Network Restrictions and Access Control

- Implementing firewall rules and access control lists (ACLs) to manage traffic flow.
- Enforcing IP-based restrictions and geofencing to protect sensitive resources.
- Configuring VLANs and subnets for network segmentation and isolation.

Port Management and Security Rules

- Opening and closing ports as per the application's requirements.
- Monitoring and controlling inbound and outbound traffic to prevent unauthorized access.
- Implementing rate limiting and traffic filtering for DDoS protection.

SSL Certificate Management

- Generating and installing SSL certificates for secure communications.
- Managing certificate renewal and revocation to maintain compliance.
- Ensuring proper encryption protocols for data-in-transit protection.

IP Whitelisting and Blacklisting

- Whitelisting trusted IP addresses for secure access to critical services.
- Blocking suspicious or malicious IP addresses to mitigate potential threats.
- Configuring dynamic IP filtering for adaptive security measures.

VPN Configuration and Management

- Setting up secure VPN tunnels for remote access and data privacy.
- Managing VPN user authentication and access control policies.
- Monitoring VPN usage and performance for seamless connectivity.

You can read more about networking in the following resources:

https://www.simplilearn.com/tutorials/networking-tutorial

Rails, React , Git and CI/CD pipeline

System Debugging and Issue Resolution

- Error Identification and Analysis: Quickly pinpoint root causes of system errors and performance bottlenecks using logging and monitoring tools.
- Effective Troubleshooting: Resolve issues related to server performance, application crashes, and configuration errors without impacting the live environment.
- Proactive Monitoring: Use tools like New Relic, Grafana, or Splunk to monitor system health and performance in real-time.
- Documentation and Reporting: Maintain comprehensive logs and error reports for future reference and compliance audits.

Version Control Management (Git Expertise)

- Repository Management: Understand how to fork repositories for independent feature development and experimentation.
- Branch Management: Create, manage, and merge branches efficiently, following Git best practices (e.g., Git Flow or GitHub Flow).
- Conflict Resolution: Handle merge conflicts and rebase branches to keep codebase clean and conflict-free.
- Commit History Management: Use meaningful commit messages, squash commits when necessary, and maintain a clean commit history for better collaboration.

Seamless System Updates and Deployment

- Pulling and Merging Updates: Retrieve the latest updates from the main repository and merge them with minimal disruption to the application.
- Staging and Testing: Implement a staging environment to test updates before deployment, ensuring compatibility and stability.

- CI/CD Integration: Leverage Continuous Integration/Continuous Deployment pipelines (e.g., Jenkins, GitLab CI, or GitHub Actions) for automated testing and deployment.
- Rollback Mechanism: Establish a robust rollback strategy to revert to previous versions in case of deployment failures.

You can read more about Rails, React , Git and CI/CD pipelines in the following resources:
Rails - https://guides.rubyonrails.org/
React - https://react.dev/learn
GIT - https://www.w3schools.com/git/
CI/CD - https://github.blog/developer-skills/github/a-beginners-guide-to-ci-cd-and-automation-on-github/

Single Sign On

Understanding the Identity Provider (IdP)

- Familiarity with Protocols: Master protocols like SAML (Security Assertion Markup Language), OAuth 2.0, and OpenID Connect (OIDC).
- Access Management: Gain administrative access to configure authentication flows and manage user roles and permissions.
- Audit and Logging: Track authentication events and monitor suspicious login activities for security compliance.

You can read more about SSO in the following resources:

https://www.onelogin.com/learn/how-single-sign-on-works

If the above skills and resources are not available, refer to the other business model options for your deployment or reach out to the Primero Team for additional guidance.

Total Cost of Ownership

This resource plan outlines the capacity requirements, budget estimates, and sustainability considerations for implementing Primero Tier 1 (DIY/Open Source). Tier 1 provides complete control and zero licensing fees, but requires full technical self-sufficiency. This analysis separates costs based on whether you have existing capacity/infrastructure versus the need to build from scratch.
Tier 1 means you are fully independent - no UNICEF/Primero support beyond documentation and community forums. Your team must handle everything: setup, security, configuration, updates, operations, and maintenance.

SCENARIO 1: BUILD NEW TEAM + NEW INFRASTRUCTURE

Assumption

- No existing HR capacity → Hire 100% (1.7 FTE)
- No existing infrastructure → Procure 100%
- Starting completely from scratch

ONE-TIME INVESTMENT COSTS
Sample Estimated Costs (Southeast Asia Market Rates)

Item	Description	Cost (USD)	Notes
Training (ELIMINATED)	Self-learning via Primero resources	$0	
Infrastructure Setup			
Production and Demo Server Setup	VM/cloud provisioning	$7,000	Production and Demo
Network & Security Config	Firewall, VPN, bastion	$1,500	Security infrastructure
SSL/TLS Certificates	3-year (2 servers)	$800	Both environments
DNS Registration	Domain (3 years)	$150	Production domain
Backup Infrastructure	Production only	$1,500	Production backup

Security Audit (Reduced)	Infrastructure review	$1,500	Reduced scope, focusing on infrastructure configuration review, not full penetration testing. Primero application has been extensively tested. Find the details here
Software & Tools			
Monitoring Tools Setup	Initial configuration	$500	Setup only
Documentation	System documentation	$800	Runbooks, procedures
Contingency (10%)		$1,175	
TOTAL ONE-TIME		**$13,925**	

ANNUAL RECURRING COSTS

1. Human Resources: Sample Estimated Costs (Cambodia Market Rates)

Role	FTE	Monthly (USD)	Annual (USD)	Justification
DevOps/Infrastructure Engineer	0.6 FTE	$1,440	$17,280	Operations, updates
System Administrator	0.3 FTE	$450	$5,400	Infrastructure monitoring
Database Administrator	0.2 FTE	$360	$4,320	PostgreSQL (2 DBs)
Primero Admin	0.6 FTE	$900	$10,800	Config + support combined
TOTAL HR	**1.7 FTE**	**$3,150**	**$37,800**	

FTE: Full-time equivalent

2. Infrastructure Costs

New Cloud Infrastructure - Azure Southeast Asia

Item	Monthly (USD)	Annual (USD)	Notes
Production Cloud Hosting	$150	$1,800	8GB RAM, 2 CPU, 500GB
Demo Cloud Hosting	$100	$1,200	Smaller

Managed PostgreSQL - Production	$100	$1,200	Recommended
Managed PostgreSQL - Demo	$60	$720	Smaller
Backup - Production Only	$50	$600	Production critical
Backup - Demo Config	$10	$120	Config only
SSL/TLS Annual Renewal	$22	$267	Both certificates
SMTP Service	$25	$300	Email notifications
Monitoring Tools	$40	$480	Both environments
CDN/Data Transfer	$40	$480	Bandwidth costs
TOTAL INFRASTRUCTURE	**$597**	**$7,167**	

3. Support & Maintenance

Item	Monthly (USD)	Annual (USD)	Notes
External Support (ELIMINATED)	$0	$0	Assumption: not required or can be covered by contingency
Security Audit	$83	$1,000	Primero receives ongoing security updates from UNICEF, find the details here If required for compliance: $1,000/year (reduced scope)
Ongoing Training (ELIMINATED)	$0	$0	Assumption: the team is self-sufficient with the Primero resource and community
Software Licenses/Updates	$50	$600	Ubuntu Pro, tools
TOTAL SUPPORT	**$133**	**$1,600**	

Contingency (10%): $4,657

TOTAL COST OF OWNERSHIP

Category	Year 1	Annual (Year 2+)	5-Year Total

NEW CASH COSTS	$65,149	$51,224	$270,045
Opportunity Costs	$0	$0	$0
TRUE TOTAL COST	**$65,149**	**$51,224**	**$270,045**

SCENARIO 2: PARTIAL RESOURCES AVAILABLE

Assumptions

- 50% HR available (0.85 FTE existing) → Hire 50% (0.85 FTE new)
- 50% infrastructure available → Procure 50%
- Can leverage existing systems and team

SCENARIO 3: 100% EXISTING RESOURCE (CAPACITY + INFRASTRUCTURE)

Assumptions

- 100% HR available (1.7 FTE existing with spare capacity) → No new hiring
- 100% infrastructure available → incremental costs can also be absorbed
- Existing team absorbs Primero workload

TOTAL COST OF OWNERSHIP - COMPLETE TIER COMPARISON

Tier/Scenario	New Investment (5yr)	Opportunity Cost (5yr)	TRUE Total Cost (5yr)
Tier 1 - Scenario 1	$270,045	$0	$270,045
Tier 1 - Scenario 2	$135,023	$135,023	$270,045
Tier 1 - Scenario 3	$0	$270,045	$270,045
Tier 4 - Scenario 1	$239,663	$0	$239,663
Tier 4 - Scenario 2	$119,832	$119,832	$239,663
Tier 4 - Scenario 3	$0	$239,663	$239,663
Tier 3 – without hiring Primero Admin	$42,250	$36,000	$78,250
Tier 3 – Hire Primero Admin	$78,250	$0	$78,250

- Tier 1 TOC 5 Years = $270,045 (Year 1: $13,925 setup + $51,224 HR&Infra, Year 2-5: $51,224 HR&Infra)
- Tier 4 TOC 5 Years = $239,663 (Year 1: $6,963 setup + $34,780 HR&Infra + $25,000 Primero fee), Year 2-5: $34,780 HR&Infra + $8,450 Primero fee)

	Tier 1	Tier 4	Tier 3
5-Year Cost	$270,045	$239,663	$78,250
FTE Required	1.7	1.15	0.5
Production Management	You	You	Primero Team
Demo Management	You	Primero Team	Primero Team
Release management	You	Primero Team	Primero Team
Professional Support	Community only	4hr/wk, 2hr/mo	4hr/wk, 2hr/mo
Configuration Tools	Manual	Automated	Automated
Training	Self-learning	Included	Included

Tier 3 (Saas):
Lowest TRUE cost: $78,250 vs $239K-270K for alternatives (3x less)
Minimum FTE: 0.5 vs 1.15-1.7 for alternatives (70% less)
Lowest risk: Primero support included
Predictable budget: Fixed $8,450/year
Focus on mission: Program delivery, not IT infrastructure

Comparing the local hosting options: Tier 4 and Tier 1
Tier 4 needs 32% less FTE (1.15 vs 1.7)
Tier 4 needs less infrastructure (production only)
Tier 4 Net benefit: $30,383 less compared with Tier 1

TIER 1: RISKS & MITIGATION MEASURES

Risk	Impact	Probability	Mitigation

Staff turnover/knowledge loss	High	High	Cross-training, comprehensive documentation, knowledge transfer protocols
Infrastructure failure	High	Low-Medium	Regular backups, disaster recovery plan, redundant systems
Security vulnerabilities	Critical	Medium	Regular patching, security audits, firewall rules, access controls
Version compatibility issues	Medium	Medium	Test upgrades in staging environment, maintain backup before updates
Scenario 3 specific			
Resource prioritization conflicts	High	High	Clear SLAs, escalation paths, clear agreements
Other project dominates resources	High	Medium-High	Guaranteed minimum allocation, regular reviews
Infrastructure changes impact Primero	Medium	Medium	Change management process, Primero representation in decisions, testing protocols
Insufficient dedicated focus on Primero	High	Medium-High	Dedicated Primero administrator, separate documentation, quarterly reviews
Dependency risk if other project ends	High	Low-Medium	Transition plan, gradual independence roadmap, identified dedicated resources
Technical debt from shared architecture	Medium	Medium	Regular technical reviews, Primero-specific requirements documented, refactoring budget

KEY SUCCESS FACTORS FOR TIER 1

Technical Success Factors:

1. Strong Linux/DevOps skills - Non-negotiable for Tier 1
2. Ansible proficiency - Must be comfortable running and customizing playbooks
3. Staging environment - ALWAYS test before applying to production
4. Comprehensive backups - Automated, tested monthly, offsite
5. Monitoring and alerting - Know about problems before users report them
6. Security hardening - SSH keys, firewalls, regular audits
7. Version control - All configuration in Git repositories
8. Keep Skills Current - Primero evolves - Ruby on Rails, PostgreSQL, Ubuntu versions,

Organizational Success Factors:

1. Realistic expectations - Tier 1 requires ongoing effort
2. Adequate staffing - Don't understaff based on optimistic estimates
3. Multi-year commitment - Don't start if funding uncertain beyond Year 1
4. Executive support - Leadership understands and supports technical needs
5. Documentation culture - Everything documented
6. Continuous learning - Staff development budget and time allocation
7. Community engagement - Active in Primero community, learn from others, monitor GitHub for issues/updates, contribute back when possible

User Adoption Success Factors:

1. User-centered configuration - Design for actual workflows, not ideal workflows
2. Comprehensive training - Not one-time, but ongoing support
3. Change management - Prepare users for new system, address concerns
4. Feedback loops - Regular user input, continuous improvement
5. Champion network - Super users in each team to support peers
6. Responsive support - Users get help when they need it

WHEN TIER 1 FAILS - WARNING SIGNS

🚩**Red Flags That Indicate Problems:**

Staff spending >60% time on maintenance vs improvements
Repeated missed updates - falling behind on security patches
Frequent unplanned downtime - users can't access system
Key person left and no one can maintain - tribal knowledge loss
Security incidents or near-misses - unauthorized access attempts
Backup failures - backups not running or not tested
Performance degradation - system slow, users complaining
Staff burnout - team overwhelmed, considering leaving
Configuration errors - frequent mistakes breaking functionality
Update failures - Ansible playbooks not working, system broken

If You See These Signs - Take Action!

Don't suffer in silence - If Tier 1 isn't working, it's better to switch tiers than to risk data loss, security breaches, or program failure.

Getting Started with the Deployment Process (Technical)

If the above skills and resources are available and you are ready to deploy the system you can follow this section outlines the standardized Tier 1 deployment process.

The demo and production environments use an Ansible-based deployment procedure. The process relies on updating the central inventory file within the Primero repository and executing Ansible commands to perform controlled rollouts.

To complete the Tier 1 deployment you will need 3 servers:

1. Chef server – A ubuntu server which will contain the Primero repository and passwordless sudo access to both demo and production servers. The primero repo will also include Ansible files and configurations.
2. Demo server – A ubuntu server which will host the demo application.
3. Production server – A ubuntu server which will host the production application.

As a best practice, all deployments must be performed on the demo environment first, validated thoroughly, and only then applied to production.

1. Deployment Workflow

The deployment workflow consists of four core stages:

1. Backup and preparation
2. Updating deployment configuration
3. Executing Ansible deployment commands
4. Post-deployment validation

Each stage is mandatory and must be completed in sequence.

2. Detailed Deployment Steps

There are two sets of processes for deployment:

1. **Initial deployment:** First step is initial deployment. This is the same process for both the demo and production environments
2. **Upgrading:** This is the process needed when you need to update the the already deployed environment to the latest version

Initial Deployment

Step 1 — Setup chef and target server

Before initiating any deployment you need to set up the chef and the target servers. The target server is either production or demo server.

- Chef Server – This ubuntu server does not require a high configuration setup. A 4-8 gb ram with 50-100 GB storage is enough.
  The ubuntu version can be the latest. It should have passwordless and sudo access to demo and production.
- Target Server – This is the same for both demo and production. The target server should be of high configuration specially production. It should contain the latest ubuntu version.

Step 2 — Configure ansible inventory file

The inventory file is in the ansible folder inside the Primero repo. Set up all the data needed in the inventory file. You can set up multiple server details in a single inventory file hence you can have the demo and production details in the same inventory file.

```
---

all:

  hosts:

        demo-hostname:

        ansible_user: 'ubuntu'

        primero_host: 'primero.example.com'

        primero_tag: 'latest'


        production-hostname:

        ansible_user: 'ubuntu'

        primero_host: 'primero.example.com'

        primero_tag: 'latest'
```

For the production environment you need to set up the secrets and environmental variables too. Secrets need to be saved in secrets.yml inside the same ansible folder.

Step 3 — Run Ansible Deployment Commands

Once the configuration has been reviewed and confirmed, execute the appropriate Ansible command, specifying the target environment.

You need to activate the python venv in order to run ansible using following commands:-

```
$ cd ansible
$ bin/activate
```

After that, inside the python environment run following commands:-

```
ansible-playbook bootstrap.yml -i inventory/inventory.yml -l target-hostname

ansible-playbook application-primero.yml --tags "local-env" -e @secrets.yml -i inventory/inventory.yml -l target-hostname

ansible-playbook application-primero.yml --tags configure,start -i inventory/inventory.yml -l target-hostname

ansible-playbook certbot.yml -i inventory/inventory.yml -l target-hostname
```

You might encounter some errors when deploying, try to fix those and keep running only that command which returned the error.
Once all the commands have been successfully executed the setup is complete.
You can then run db:seed and other migrations inside the target machine docker.

Upgrading

Step 1 — Perform a Complete Backup

Before initiating any upgrade:

- Ensure you have a working, restorable backup of the environment you are upgrading.
- Do not delete the backup until you have fully confirmed that the upgrade has been completed successfully.
- Perform a new backup every time before starting a deployment.

This step mitigates risk and ensures system recoverability in the event of deployment issues.

Step 2 — Update Inventory Configuration

In the 'inventory.yml' file located within the Ansible folder of the Primero repository:

1. Update the following parameters with the target version:
   - primero_tag
   - primero_repo_branch

2. When upgrading:
   - Update the Demo section for Demo deployments.
   - Update the Production section for Production deployments.

These values determine the exact Primero version that will be deployed.

Step 3 — Run Ansible Deployment Commands

Once the configuration has been reviewed and confirmed, execute the appropriate Ansible command, specifying the target environment.

command:
ansible-playbook application-primero.yml --tags configure,start -i inventory/inventory.yml -l target-hostname

If errors occur, review the logs, resolve issues, and re-run until deployment completes successfully.

Step 4 — Validate the Deployment

After successful completion of the Ansible execution:

1. Confirm the system is now running the intended upgrade version.
2. Perform a full functional validation, including existing and new features.
3. Verify critical workflows, user access, and data integrity.

Only after thorough assessment should the deployment be considered complete.

Summary

The Tier 1 Primero deployment process relies on a controlled, repeatable Ansible-based approach. Following the above workflow—backup, configuration update, deployment execution, and validation—ensures safe and consistent upgrades across demo and production environments.