Effort estimation framework for migrating OpenShift to Amazon EKS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Effort estimation framework for migrating OpenShift to Amazon EKS

Table of Contents

Effort estimation framework for migrating OpenShift to Amazon EKS

Pratap Kumar Nanda and Pradip kumar Pandey, Amazon Web Services

Overview

Many organizations underestimate the effort required for container platform migrations. Based on observed patterns, teams often need more time and resources than initially planned, which can lead to budget overruns and extended timelines.

This guide provides a structured approach to building accurate effort estimates when migrating from Red Hat OpenShift to Amazon Elastic Kubernetes Service (EKS). We address common complexity areas that impact migration timelines, including:

- Operator dependencies and custom resource definitions
- Security policy translation and implementation
- Container image registry migration
- Network architecture and connectivity requirements

Service availability

Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Applicability

This guidance applies to:

- Organizations migrating containerized workloads from Red Hat OpenShift (versions 4.x) to Amazon EKS
- Environments with 50 or more applications running on OpenShift
- Migrations involving custom operators, security policies, or stateful workloads
- Teams that need structured effort estimation before committing to migration timelines

Target audience

- Platform engineers responsible for Amazon EKS cluster design and configuration
- Migration leads coordinating cross-team execution
- Project managers developing timelines and resource plans
- DevOps engineers reconfiguring CI/CD pipelines
- Site reliability engineers (SREs) establishing observability and operational readiness
- Security engineers implementing IAM and compliance controls

Business challenge

Platform teams frequently face pressure to provide migration estimates early in the planning process, often before completing a thorough technical assessment. These initial estimates can become fixed commitments, creating risk when unforeseen complexities emerge during execution.

Accurate estimation for OpenShift to Amazon EKS migrations requires addressing both technical and organizational factors. Beyond the technical migration work itself, teams must account for several business and planning challenges:

- **Limited historical data** – Organizations often lack reference points from similar container platform migrations
- **Stakeholder alignment** – Multiple teams may have competing priorities and timeline expectations
- **Skills assessment** – Uncertainty around team capabilities and training requirements for AWS-native services
- **Dependency mapping** – Complex application interdependencies that become apparent during discovery
- **Business continuity** – Requirements for minimal downtime and rollback capabilities
- **Technical debt** – Legacy configurations and workarounds that surface during migration planning
- **Contractual drivers** – Timeline constraints from licensing agreements or contract renewals
- **Architectural decisions** – Trade-offs between lift-and-shift approaches versus re-architecting for cloud-native patterns

This guide helps you build estimation frameworks that account for these variables, enabling more accurate planning and stakeholder communication throughout your migration journey.

Targeted business outcomes

Migration estimates directly influence budget approvals, resource allocation, and business planning decisions. Inaccurate estimates can create cascading impacts across the organization, affecting teams beyond the platform engineering group. Any incorrect estimation leads to financial loss for organization .

This guide helps you develop estimation practices that deliver the following business outcomes:

- **Budget accuracy** – Comprehensive upfront estimates reduce mid-project funding requests. Organizations can allocate appropriate resources from the start, avoiding budget revisions that typically range from 30-50% of the original estimate.
- **Timeline predictability** – Reliable schedules enable coordinated planning across the organization. Product teams can plan feature releases, business stakeholders can commit to customer timelines, and operations teams can schedule training and readiness activities.
- **Proactive resource planning** – Early identification of skill gaps and capacity requirements allows you to make strategic decisions about contractor engagement, team training, or AWS managed services. This prevents reactive hiring and enables thoughtful team development.
- **Minimized business disruption** – Estimates that account for comprehensive testing, validation, and rollback procedures reduce production incidents during migration. Applications maintain availability and performance while the underlying platform transitions.
- **Stakeholder trust** – Delivering migrations within committed timeframes and budgets builds organizational confidence in technical planning. This credibility strengthens support for future modernization initiatives and strategic technology investments.

Migration consideration

Accurate migration estimates require a comprehensive understanding of your current environment and target architecture before committing to timelines and resource allocations. Overlooking critical factors during the assessment phase can lead to significant estimate variances as the project progresses.

To develop reliable effort estimates for your OpenShift to Amazon EKS migration, evaluate the following factors:

Start with discovery

The discovery phase establishes the foundation for accurate migration estimates. Insufficient discovery leads to incomplete assessments and estimate variances during execution. Allocate adequate time for this phase based on your environment's complexity.

A comprehensive discovery phase should include:

- **Workload inventory** – Catalog all applications, services, and workloads running on OpenShift
- **Dependency mapping** – Identify application interdependencies, external integrations, and service relationships
- **Security policy review** – Document existing security controls, RBAC configurations, and compliance requirements
- **Storage configurations** – Assess persistent volume usage, storage classes, and data persistence patterns
- **Network topology** – Map network policies, ingress/egress rules, service mesh configurations, and connectivity requirements
- **CI/CD pipeline assessment** – Evaluate build processes, deployment workflows, and automation tooling

Planning consideration: Legacy workloads with limited documentation or infrequent updates typically require extended discovery time. Factor additional effort for environments where application ownership or architectural knowledge is unclear.

Categorize workloads by migration complexity

Migration effort varies significantly based on workload characteristics. A stateless API service requires substantially less effort than a stateful application with persistent storage, custom operators, and complex dependencies. Consider the following application categorization factors.

Categorize workloads into complexity tiers to improve estimation accuracy:

- **Simple** – Stateless applications with standard configurations and no custom dependencies. Examples include stateless web services, API gateways, and standard containerized applications.
- **Moderate** – Applications with state management requirements, standard persistent storage needs, and minimal custom tooling. Examples include applications using standard storage classes or common third-party integrations.
- **Complex** – Workloads requiring custom operators, specific security contexts, or extensive external integrations. Examples include applications with custom resource definitions (CRDs), service mesh configurations, or specialized networking requirements.
- **High risk** – Business-critical applications with strict availability requirements, legacy codebases, or undocumented dependencies. These workloads require comprehensive testing, validated rollback procedures, and extended validation periods.

Applying effort multipliers: Assign different effort factors based on complexity tiers. Complex workloads typically require 4-5x the migration effort of simple workloads. High-risk workloads may require additional effort for extended testing, stakeholder coordination, and risk mitigation activities.

Document the complexity tier for each workload during discovery to ensure your overall estimate reflects the actual migration scope.

Gather operational knowledge from application maintainers

Architecture diagrams and technical documentation may not reflect the current state of your environment. Engineers who maintain and troubleshoot applications often possess operational knowledge that isn't captured in documentation. This should be a critical factor when planning for migration.

Conduct structured interviews with application owners, development teams, and operations staff to capture:

- **Known issues and failure patterns** – Recurring problems, error conditions, and operational challenges
- **Existing workarounds** – Temporary solutions or configuration adjustments that maintain functionality
- **Critical dependencies** – Undocumented integrations, shared services, or external system dependencies
- **Performance characteristics** – Resource utilization patterns, scaling behaviors, and capacity requirements
- **Operational procedures** – Manual intervention requirements, restart procedures, or maintenance routines

Allocate time in your project plan for these discovery conversations. The insights gathered help identify migration risks early, preventing timeline impacts from unexpected technical challenges during execution.

Build estimates as per bottom-up approaches

Avoid top-down estimation methods that divide total workloads by available time. This approach doesn't account for workload complexity variations, organizational constraints, or operational realities such as holidays, team availability, and competing priorities.

Use a bottom-up estimation methodology:

1. Estimate by workload category – Calculate effort for each complexity tier based on the assessment framework
2. Include integration work – Account for service connectivity, API integrations, and cross-application dependencies
3. Add testing effort – Include functional testing, performance validation, and user acceptance testing
4. Document migration activities – Allocate time for runbooks, architecture updates, and knowledge transfer
5. Apply contingency buffers – Add reserves for unforeseen technical challenges and scope adjustments

Timeline reconciliation: If the bottom-up estimate exceeds business timeline expectations, address the gap during planning rather than during execution. Options include:

- Phased migration approach with prioritized workload batches
- Additional resource allocation or contractor engagement
- Scope adjustments based on business priority
- Extended timeline with revised milestone commitments

Early alignment between technical estimates and business expectations reduces mid-project disruptions and enables informed decision-making.

Account for supporting activities beyond workload migration

Migration projects require substantial effort beyond workload movement. Include these supporting activities in your estimate to ensure adequate resource allocation.

- Environment setup and configuration – Amazon EKS cluster provisioning, node group configuration, and AWS service integration
- IAM policies and RBAC mapping – Translation of OpenShift security contexts to AWS IAM roles and Kubernetes RBAC policies
- Secrets migration and rotation – Migration to AWS Secrets Manager or Amazon EKS secrets encryption, including credential rotation procedures
- Monitoring and alerting reconfiguration – Implementation of Amazon CloudWatch, AWS X-Ray, or third-party observability tools with appropriate alerting thresholds
- Runbook updates – Documentation of operational procedures, troubleshooting guides, and incident response workflows for the new platform
- Team training and knowledge transfer – Upskilling on Amazon EKS, AWS services, and updated operational procedures
- Stakeholder communication – Regular status reporting, milestone reviews, and executive updates throughout the migration
- Change management processes – Coordination with change advisory boards, approval workflows, and deployment scheduling
- Rollback testing and validation – Development and testing of rollback procedures to ensure business continuity

Explicitly document these activities in your project plan with dedicated effort allocations. Omitting these items from estimates leads to under-resourcing and timeline pressure during execution.

Factor In learning curve

Teams transitioning from OpenShift to Amazon EKS require time to develop proficiency with AWS-native services, EKS operational patterns, and cloud-native tooling. Initial migration velocity will be lower as teams build familiarity with the new platform.

Apply a learning curve factor to early migration phases:

1. Initial workloads – Estimate 1.5-2x baseline effort for the first batch of migrations as teams establish processes, tooling, and troubleshooting approaches
2. Mid-phase workloads – Reduce the multiplier as teams gain experience with common migration patterns and AWS service integration
3. Later workloads – Approach baseline estimates as teams achieve proficiency with Amazon EKS operations and migration workflows

Planning consideration: Structure your migration waves to place simpler, lower-risk workloads early in the schedule. This allows teams to build competency before addressing complex or business-critical applications. Factor this velocity ramp into your overall timeline rather than assuming consistent productivity throughout the project.

Use t-shirt sizing for early conversations, hours for planning

Use estimation techniques appropriate to the project maturity and decision-making requirements.

Initial assessment phase: T-shirt sizing (Small, Medium, Large, Extra Large) provides directional guidance for early planning and feasibility discussions without requiring detailed analysis. This approach supports initial budget conversations and timeline expectations.

Detailed planning phase: Convert t-shirt sizes to specific effort ranges once the project receives approval and enters active planning. Define clear criteria for each size category to ensure consistent application across workloads.

Example sizing framework:

- Small – 40-80 hours: Stateless workloads with standard configurations

- Medium – 80-160 hours: Applications with moderate complexity and standard dependencies
- Large – 160-320 hours: Complex workloads with custom operators or extensive integrations
- Extra Large – 320+ hours: High-risk, business-critical applications requiring extended validation

Document your sizing definitions and share them with stakeholders to establish common understanding. This ensures alignment between technical teams and business stakeholders throughout the estimation and planning process.

Include contingency reserves

Migration projects encounter unforeseen challenges including late-discovered dependencies, tooling issues, team availability constraints, and technical complexities that emerge during execution.

Include a contingency reserve of 15-25% in your total effort estimate. The appropriate percentage depends on:

- Environment complexity – Higher contingency for environments with limited documentation or legacy workloads
- Team experience – Additional reserve for teams new to Amazon EKS or AWS services
- Risk assessment – Increased contingency for migrations with strict availability requirements or complex dependencies
- Communicating contingency: Frame contingency as risk management rather than estimate padding. This reserve addresses inherent project uncertainties and enables the team to respond to challenges without requiring scope or timeline renegotiation.

Historical data shows that projects without explicit contingency allocation typically experience schedule delays rather than early completion. Building contingency into the initial estimate provides flexibility to manage risks while maintaining stakeholder commitments.

Separate estimation from commitment

Distinguish between estimates and commitments when communicating with stakeholders. An estimate represents your best assessment based on available information at a specific point in time. A commitment is a formal delivery promise with associated accountability.

Document estimate assumptions: include the conditions and constraints that underpin your estimate:

- Team availability and resource allocation
- Completion of discovery and assessment phases
- Stability of production environments during migration
- Access to subject matter experts and application owners
- Availability of required AWS services and tooling
- No major scope changes or additional workload discoveries

Confidence levels: communicate estimate confidence based on information completeness. Early-phase estimates with limited discovery have lower confidence than detailed estimates built from comprehensive assessments.

Assumption tracking: establish a process to revisit estimates when underlying assumptions change. Significant deviations such as resource constraints, scope additions, or environmental discoveries warrant estimate updates and stakeholder communication.

A clear distinction between estimates and commitments enables realistic planning and maintains stakeholder trust throughout the migration lifecycle.

Document your assumptions clearly

All migration estimates rely on underlying assumptions about resources, access, scope, and constraints. Document these assumptions explicitly and include them in project planning materials and stakeholder communications.

Key assumptions to document:

- Team size and availability – Dedicated resource allocation, expected availability percentages, and skill levels
- Environment access timelines – AWS account provisioning, network connectivity, and access to source OpenShift environments
- Third-party support responsiveness – Vendor response times for tooling issues, licensing questions, or technical escalations

- Scope boundaries – Included and excluded workloads, migration approach decisions, and out-of-scope activities
- Testing requirements – Validation depth, performance testing scope, and acceptance criteria
- Maintenance windows – Availability and duration of approved downtime windows for migration activities

Assumption management: include assumptions in executive summaries and project charters rather than relegating them to appendices. When assumptions prove incorrect or conditions change, documented assumptions provide clear rationale for estimate adjustments and timeline revisions.

Transparent assumption documentation supports informed decision-making and facilitates productive conversations when project conditions evolve.

Get input from multiple perspectives

Engage multiple teams in the estimation process to capture diverse perspectives on migration complexity, risks, and effort requirements. Single-source estimates often miss domain-specific considerations that impact project scope and timeline.

Include representatives from:

- Platform engineering – Infrastructure provisioning, cluster configuration, and AWS service integration
- Application development – Code modifications, dependency updates, and application-specific migration requirements
- Security and compliance – IAM policy design, security control implementation, and regulatory requirement validation
- Operations and SRE – Monitoring setup, incident response procedures, and operational readiness activities
- Project management – Timeline coordination, resource allocation, and stakeholder communication planning

Each discipline identifies risks and effort areas specific to their domain. Platform engineers assess infrastructure complexity, while security teams evaluate compliance requirements and policy migration effort. Operations teams identify monitoring and alerting reconfiguration needs that may not be apparent to development teams.

Collaborative estimation produces more comprehensive and defensible project plans than isolated assessment efforts.

Review past migrations

Use historical project data and external resources to inform your estimation approach and identify common complexity areas.

Internal retrospectives: If your organization has completed previous platform migrations or major infrastructure projects, review post-project analyses to identify:

- Activities that required more effort than initially estimated
- Timeline variances and their root causes
- Areas where estimates proved accurate or conservative
- Lessons learned that apply to container platform migrations
- External resources: For organizations undertaking their first major migration, gather insights from multiple sources:
- AWS Professional Services and AWS Partners – Leverage migration experience and reference architectures specific to OpenShift to Amazon EKS transitions
- Industry communities – Engage with Cloud Native Computing Foundation (CNCF) communities, AWS user groups, and Kubernetes forums
- Peer organizations – Connect with teams who have completed similar migrations through professional networks or industry events
- AWS case studies and reference implementations – Review documented migration patterns and architectural guidance

External perspectives provide realistic insights into common challenges, effort distribution, and risk areas that may not be apparent during initial planning.

Track actuals vs estimates during execution

Treat estimation as an ongoing process throughout the migration rather than a one-time planning activity. Track actual effort against estimates to identify variances early and adjust remaining project plans accordingly.

Variance tracking: monitor actual migration effort for completed workloads and compare against initial estimates. Significant deviations such as consistent 30% overruns in early migration waves indicate the need to reassess estimates for remaining workloads.

Checkpoint reviews: establish regular review intervals to evaluate:

- Actual versus estimated effort for completed migrations
- Emerging patterns in complexity or technical challenges
- Changes in team velocity or productivity
- New risks or dependencies discovered during execution
- Validity of original assumptions and constraints

Corrective actions: when variance patterns emerge, update estimates for remaining workloads and communicate timeline or resource implications to stakeholders. Early detection and adjustment prevent larger schedule impacts later in the project.

Schedule checkpoint reviews at logical project milestones such as after each migration wave or monthly intervals rather than waiting for project completion to assess estimate accuracy.

Clear communication

Provide effort ranges rather than single-point estimates when uncertainty exists. Ranges reflect the inherent variability in complex migration projects and communicate estimate confidence levels to stakeholders.

Example range format:

"We estimate 800-1,100 hours for this migration phase, with final effort dependent on the complexity discovered during dependency mapping and workload assessment."

Range construction: base your range on:

- Lower bound – Effort required if assumptions hold and no significant complications arise
- Upper bound – Effort required if moderate complexity or dependency issues emerge
- Confidence level – Narrower ranges indicate higher confidence based on information completeness

Range-based estimates set realistic expectations and provide flexibility to accommodate discovery findings without requiring formal estimate revisions. As you gather more information through discovery and early migration phases, you can narrow ranges and increase estimate precision.

Document the factors that influence the range boundaries so stakeholders understand what drives potential variance.

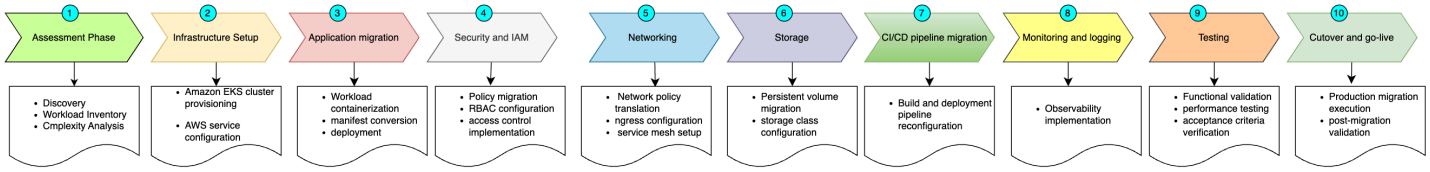
Conclusion

Effective migration estimation focuses on reducing uncertainty, establishing realistic expectations, and maintaining stakeholder alignment throughout the project lifecycle. Comprehensive upfront assessment and planning enable smoother execution and more predictable outcomes.

The estimation practices outlined in this guide thorough discovery, workload complexity assessment, cross-functional collaboration, and continuous validation provide a foundation for successful OpenShift to Amazon EKS migrations. Organizations that invest in structured estimation approaches experience fewer mid-project disruptions, more accurate budget management, and stronger stakeholder confidence in technical delivery.

Migration lifecycle phases

Structure your OpenShift to Amazon EKS migration using a phased approach that aligns with standard software development lifecycle (SDLC) practices. Each phase requires specific activities and effort allocation.



The migration lifecycle includes the following phases:

1. Assessment phase – Discovery, workload inventory, and complexity analysis
2. Infrastructure setup – Amazon EKS cluster provisioning and AWS service configuration
3. Application migration – Workload containerization, manifest conversion, and deployment
4. Security and AWS Identity and Access Management (Amazon IAM) – Policy migration, Role-Based Access Control (RBAC) configuration, and access control implementation
5. Networking – Network policy translation, ingress configuration, and service mesh setup
6. Storage – Persistent volume migration and storage class configuration
7. CI/CD pipeline migration – Build and deployment pipeline reconfiguration for Amazon EKS
8. Monitoring and logging – Observability implementation with Amazon CloudWatch and related services
9. Testing – Functional validation, performance testing, and acceptance criteria verification
10. Cutover and go-live – Production migration execution and post-migration validation

Per-workload effort breakdown

The following sections provide detailed effort estimation guidance for each migration phase on a per-workload basis.

Create a spreadsheet to track effort estimates for each of the following phases.

Assessment and discovery

The assessment phase establishes the foundation for accurate migration planning. Use the following templates to capture essential information during discovery activities. Complete documentation enables informed effort estimation and risk identification.

Task	Base Hours	Your Input	Calculated (Base hr * Number of resources)
Count total applications	4	No of apps	___ hrs
Document namespaces	2/namespace	No of ns	___ hrs
List deployments/pods	1/app	No of apps	___ hrs
Catalog ConfigMaps	0.5/map	No of maps	___ hrs
Catalog Secrets	0.5/secret	No of secrets	___ hrs
Document PVCs	1/pvc	No of pvcs	___ hrs
Network policies audit	2/policy	No of policies	___ hrs
Routes/Ingress mapping	1/route	No of routes	___ hrs
ImageStreams inventory	0.5/stream	No of streams	___ hrs
BuildConfigs analysis	2/config	No of configs	___ hrs
total			___ hrs

Dependency mapping

Task	Hours	Complexity Factor	Notes
Inter-service dependencies	4 per 10 apps	Low=1x, Med=1.5x, High=2.5x	
External service integrations	3 per integration		
Database connections	4 per database		
Message queue connections	3 per queue		
Third-party API integrations	2 per API		
Shared storage dependencies	4 per share		

Infrastructure build-out

In some organizations, platform teams manage Amazon EKS cluster provisioning and infrastructure configuration independently from application migration activities. If your platform team handles infrastructure setup outside the migration project scope, exclude the cluster setup estimates provided in this section from your overall effort calculation.

EKS cluster setup(optional)

Component	Simple	Standard	Production	Total Hours (Number * Hr)
Amazon Virtual Private Cloud (Amazon VPC) design	8	16	32	

Amazon EKS cluster creation	4	8	16
Node groups config	4	12	24
Amazon IAM roles/policies	8	24	48
Security groups	4	12	24
Elastic Load balancer setup	4	8	16
DNS configura tion	2	4	8
AWS Certificate Management	4	8	16
Cluster Autoscaler	4	8	16
Karpenter (if used)	0	12	24

Supporting infrastructure setup

The following estimates apply when the listed infrastructure components are required for your migration. Include these effort allocations in your overall project estimate based on your specific infrastructure requirements.

Component	Hours	Applicable?	Total Hours(if Y then calculate Hr)
Amazon Elastic Container Registry	8	Y/N	

(Amazon ECR) Repository setup		
AWS Transit gateway (hybrid)	24	Y/N
Direct connect config	40	Y/N
VPN configuration	16	Y/N
PrivateLink setup	12 per endpoint	Y/N
Amazon S3 Buckets (logs/backups)	4	Y/N
AWS KMS Key setup	8	Y/N
AWS Secrets manager setup	8	Y/N
AWS Systems Manager (SSM) Parameter store migration	4 + (0.25 × params)	Y/N

Application migration

Per-application effort matrix

The following estimates correspond to the workload complexity tiers (t-shirt sizes) established during the discovery phase. Apply these effort ranges based on each application's assigned complexity category.

App Type	Count	Hours/App	Total Hours
Stateless - Simple (no deps)	___	8	
Stateless - Standard	___	16	

Stateless - Complex (many deps)	___	32
Stateful - Simple	___	24
Stateful - Standard	___	48
Stateful - Complex	___	80
Legacy/Monolith	___	120
Custom Operators	___	60

OpenShift-specific conversions

Item	Count	Hours Each	Total
Route → Ingress conversion	___	2	
DeploymentConfig → Deployment	___	3	
BuildConfig → Pipeline	___	8	
ImageStream → ECR migration	___	2	
SCC → PSP/PSS migration	___	6	
OpenShift Templates → Helm	___	12	
oc commands → kubectl scripts	___	4	

OAuth config migration	1	16

Storage migration

Persistent volume migration

Storage migration is a critical prerequisite for stateful application migrations. Complete storage class configuration, persistent volume provisioning, and data migration activities before migrating applications that depend on persistent storage.

Storage Type	Volume Count	Size (GB)	Hours formula	Total hours
Block (gp2/gp3)	___	___	2 + (0.01 × GB) per vol	
Amazon EFS/ NFS shared	___	___	8 + (0.02 × GB) per vol	
Database (Amazon RDS migration)	___	___	16 + (0.05 × GB) per db	
Object (S3)	___	___	4 + (0.001 × GB)	

Storage class mapping

Task	Hours
Analyze current storage classes	4
Design Amazon EKS storage class mapping	8
Amazon EBS CSI driver setup	4
Amazon EFS CSI driver setup (if needed)	8

Test storage performance	16
Data validation procedures	8

Security and compliance

Security and compliance requirements are critical considerations for every organization. Carefully evaluate security controls, access policies, and regulatory requirements during migration planning.

Recommended approach: Engage security and compliance teams early in the planning process. Their involvement ensures that IAM policies, network security controls, encryption requirements, and audit logging configurations align with organizational security standards and regulatory obligations.

Identity & access

Task	Base	Complexity	Total Hours
Role-Based Access Control (RBAC)mapping analysis	16		
Role/ClusterRole conversion	2 per role		
Service account migration	1 per SA		
IAM Roles for Service Accounts (IRSA) configuration	4 per app		
OpenID Connect (OIDC) provider setup	8		
AD/LDAP integration	24-40		
SSO reconfiguration	16-32		

Security Policies

Task	Hours
Security Context Constraints (SCC) → Pod Security standards mapping	16
Network policy conversion	2 per policy
Open Policy Agent (OPA)/Gatekeeper policies	4 per policy
Image scanning setup (Amazon ECR)	8
Secrets encryption config	8
Audit logging setup	12
Compliance documentation	24-40

CI/CD pipeline migration

Most organizations have existing CI/CD pipelines that require reconfiguration to deploy applications to Amazon EKS. The following estimates cover integration effort for common CI/CD platforms and deployment workflow updates.

Refer to the following table to estimate effort based on your tooling.

Pipeline Conversion

Source pipeline type	Count	Target	Hours Each	Total
OpenShift pipelines	___	Same Openshift	8	
OpenShift pipelines	___	AWS CodePipeline	16	

Jenkins (on OpenShift)	___	Jenkins (EKS)	12
Jenkins (on OpenShift)	___	AWS CodePipeline	24
Custom buildConfigs	___	CodeBuild	16
GitLab CI	___	GitLab CI (reconfig)	6
GitHub Actions	___	GitHub Actions (reconfig)	4

Registry migration

Task	Hours
Amazon ECR setup & organization	8
Image migration script development	16
Image migration execution	2 + (0.1 × image count)
Image signing setup	12
Pull secret configuration	4
Lifecycle policies	4

Observability stack

Organizations approach observability implementation in different ways. Some have existing monitoring stacks that require integration with Amazon EKS, while others need to establish observability infrastructure from the ground up. Use the following effort estimates based on your organization's current observability maturity.

Monitoring

Current → Target	Hours
Prometheus (keep) → Prometheus (EKS)	24
Prometheus → Amazon Managed Prometheus	32
Prometheus → CloudWatch Container Insights	40
Grafana (keep) → Grafana (EKS)	16
Grafana → Amazon Managed Grafana	24
Dashboard migration	2 per dashboard
Alert rules migration	1 per rule
Custom metrics validation	4 per app

Logging

Current → Target	Hours
EFK → EFK on Amazon EKS	24
EFK → Amazon OpenSearch	32
EFK → CloudWatch Logs	24
Fluentd/Fluent Bit config	16
Log parsing rules migration	8
Retention policy config	4

Implementing comprehensive observability requires effort beyond basic infrastructure setup. The estimates below reflect the work required to configure monitoring dashboards, centralized logging, and alerting systems for your migrated applications.

Testing and validation

Testing effort

Test Type	Base Hours	Per App Factor	Your Hours
Unit test validation	8	+1 per app	
Integration testing	24	+2 per app	
Performance baseline	16	+4 per critical app	
Performance validation	24	+4 per critical app	
Security scanning	16	+1 per app	
DR/Failover testing	24	+4 per critical app	
User Acceptance Testing (UAT)coordination	40	fixed	
Regression testing	16	+2 per app	

Documentation and training

Documentation is an essential component of migration activities. Comprehensive documentation supports operational readiness, knowledge transfer, and long-term platform maintainability.

Deliverable	Hours
Architecture documentation	24-40
Runbook creation	4 per app
Incident response procedures	16
Admin training (per person)	8 people

Developer training (per person)	4 people
Knowledge transfer sessions	16

Project management and coordination

Activity	Formula
Project management	15% of total technical hours
Stakeholder meetings	4 hrs/week weeks
Change management	8% of total technical hours
Risk management	4% of total technical hours
Vendor coordination	8 hrs/week weeks

Executive summary

Use the following summary table to consolidate effort estimates from each migration phase. This provides a comprehensive view of total project effort based on your specific requirements and workload complexity.

Technical hours summary

Section	Hours
1. Assessment & discovery	
2. Infrastructure build-out	
3. Application migration	
4. Storage migration	
5. Security & compliance	

6. CI/CD pipeline migration	
7. Observability stack	
8. Testing & validation	
9. Documentation & training	
Technical subtotal	
Adjustment factor (section 3.3)	×1
Adjusted technical total	

Add below Project Management effort as part of the migration effort.

Category	Calculation	Hours
Technical (adjusted)	from above	
Project Management (15%)		
Meetings		
Change Management (8%)		
Risk Buffer (10-20%)		
Grand Total		

COST ESTIMATION (**Optional**)

Organizations often require cost projections for migration activities to support budget planning and approval processes. Use the following template to translate effort estimates into financial projections based on your team composition and resource rates.

Role	Rate/Hr	Hours	Cost
Solution architect	$___		

Platform engineer	$___
DevOps engineer	$___
Security engineer	$___
QA engineer	$___
Project manager	$___
Total labor	

Additional Costs

Item	Monthly	Months	Total
Parallel environment	$___	___	
Migration tools	$___	___	
Training/certification			
Contingency (15%)			

Notes and assumptions

1. Estimates assume team has basic Kubernetes experience
2. Add 30-50% if team is new to EKS
3. Add 20% if strict compliance requirements (PCI, HIPAA)
4. Subtract 20% if using Infrastructure as Code already
5. All hours are person-hours, not elapsed time
6. Does not include AWS infrastructure costs
7. Assumes standard business hours availability

Estimation example

The following worked example applies the estimation framework from the Migration lifecycle phases section to a specific scenario:

1 OpenShift cluster with 100+ applications migrating to Amazon EKS. Use this as a reference template — *adjust values based on your environment*.

Executive summary

OpenShift to AWS EKS Migration Scope: 1 Cluster | 100+ Applications

Parameter	Details
Source Platform	OpenShift (1 Cluster)
Target Platform	AWS EKS
Application Count	100+ Applications
Base Effort Range:	2,800 - 3,400 Hours
Estimated Duration	16 - 20 Weeks
Recommended Team Size	6 - 8 Members

> **Note**
>
> The base effort range of 2,800–3,400 hours represents core migration activities. The grand total of approximately 4,255 hours includes 20% contingency reserve and 15% project management overhead applied to the base effort.

Refer to the following estimation breakdown.

Assumptions and prerequisites

The effort estimates provided in this guide are based on the following baseline assumptions. Deviations from these conditions will impact actual migration effort and timelines.

Technical environment:

1. Applications are containerized and running on OpenShift 4.x
2. Target AWS account is provisioned with required IAM permissions and service quotas
3. Network connectivity is established between source environments and AWS (for hybrid or on-premises migrations)
4. Migration approach is lift-and-shift with platform-specific adjustments; no major application refactoring required

Organizational readiness:

1. Business stakeholders are available for validation and sign-off activities
2. Team possesses foundational Kubernetes knowledge with limited Amazon EKS operational experience
3. Standard business hours are available for migration work, with flexibility for off-hours cutover windows as needed

If your environment differs from these assumptions, such as requiring application modernization, operating on OpenShift 3.x, or having teams with no Kubernetes experience, adjust effort estimates accordingly using the complexity multipliers and learning curve factors described in earlier sections.

Application complexity breakdown

Based on a typical enterprise distribution for 100+ applications:

Complexity Tier	Count (Estimated)	Effort Per App	Total Hours
Simple	40 apps	8-12 hours	320-480

Moderate	35 apps	16-24 hours	560-840
Complex	20 apps	32-48 hours	640-960
High Risk / Critical	8 apps	56-72 hours	448-576

Complexity Tier Definitions

Simple Applications

- Stateless services
- Standard Kubernetes manifests
- No custom operators
- Minimal external dependencies
- Examples: Static APIs, utility services, batch jobs

Moderate Applications

- Some stateful components
- ConfigMaps and Secrets dependencies
- Standard persistent volume requirements
- 2-3 external integrations
- Examples: Backend services with database connections, queue consumers

Complex Applications

- Custom operators or CRDs
- Advanced networking requirements
- Multiple persistent volumes
- Heavy integration dependencies
- Custom security contexts
- Examples: Data processing pipelines, middleware services

High Risk / Critical Applications

- Business critical with strict SLAs
- Zero or near-zero downtime requirements
- Complex state management
- Undocumented legacy configurations
- Requires extensive rollback planning
- Examples: Payment services, core APIs, customer-facing applications

Phase 1: Discovery and assessment

Considering migration scope of 1 cluster | 100+ applications. The following table represents the discovery and assesment effort.

Activity	Effort (Hours)
Current state infrastructure documentation	40
Application inventory and categorization	60
Dependency mapping (internal and external)	80
Security policy and RBAC review	40
Storage and persistent volume assessment	32
Network topology analysis	40
CI/CD pipeline inventory	24
Compliance and regulatory requirement gathering	24
Stakeholder interviews (app owners, ops teams)	40
Risk assessment and mitigation planning	32
Discovery report and sign-off	16

Phase 2: Target environment setup

Considering migration scope of 1 cluster | 100+ applications. The following table represents the environment setup effort.

Activity	Effort (Hours)
AWS account structure and organization setup	24
Amazon VPC design and implementation	40
Amazon EKS cluster provisioning (Terraform/ CloudFormation)	56
Node group configuration and scaling policies	32
AWS IAM roles and policies setup	48
Networking — subnets, security groups, Network Access Control Lists (NACLs)	40
Elastic load balancer and ingress controller setup	32
DNS configuration and Amazon Route53 integration	16
Container registry setup (ECR)	16
Secrets management (AWS Secrets Manager / external-secrets)	32
Logging infrastructure (Amazon CloudWatch / FluentBit)	40
Monitoring stack (Prometheus, Grafana or CloudWatch Container Insights)	48
Service mesh setup (if required)	40

Backup and disaster recovery configuration	32
Environment validation and testing	24

Phase 3: CI/CD pipeline migration

Considering migration scope of 1 cluster | 100+ applications. The following table represents the CI/CD setup effort.

Activity	Effort (Hours)
Pipeline inventory and analysis	16
Build pipeline modifications for Amazon ECR	40
Deployment pipeline updates for Amazon EKS	56
GitOps setup (ArgoCD / Flux) if applicable	48
Pipeline testing and validation	40
Rollback mechanism configuration	24
Documentation and runbook updates	16

Phase 4: Application migration execution

Considering migration scope of 1 cluster | 100+ applications. The following table represents the application migration execution effort.

Activity	Effort (Hours)
Manifest conversion (OpenShift to vanilla K8s)	160
Image migration to Amazon ECR	80
ConfigMaps and Secrets migration	60

Persistent volume migration and data transfer	120
Service and ingress reconfiguration	80
Role-Based Access Control (RBAC) and security context adjustments	60
Application deployment to Amazon EKS	200
Integration reconnection and validation	120
Smoke testing per application	150
Performance baseline validation	80
Issue resolution and troubleshooting	160

Phase 5: Testing and validation

Considering migration scope of 1 cluster | 100+ applications. The following represent the testing and validation effort.

Activity	Effort (Hours)
Functional testing	120
Integration testing	80
Performance and load testing	60
Security scanning and validation	48
Disaster recovery testing	40
UAT coordination and support	60
Defect resolution	80
Sign-off documentation	16

Phase 6: Cutover and go-live

Considering migration scope of 1 cluster | 100+ applications. The following table represents the cutover and go-live effort.

Activity	Effort (Hours)
Cutover planning and runbook preparation	32
Stakeholder communication and scheduling	16
DNS cutover and traffic shifting	24
Final data synchronization (stateful apps)	40
Go-live monitoring and war room support	48
Immediate issue resolution	40
Rollback execution (if needed)	24
Go-live sign-off	8

Phase 7: Post-migration support and stabilization

Considering migration scope of 1 cluster | 100+ applications. The following table represents the post-migration suppoet and stabilization effort.

Activity	Effort (Hours)
Hypercare support (2 weeks)	160
Performance tuning and optimization	40
Cost optimization review	24
Documentation finalization	40
Knowledge transfer to operations team	48

Lessons learned and retrospective	16
Old environment decommissioning plan	24

Total effort summary

Considering migration scope of 1 cluster | 100+ applications. The following table represents the total effort summary.

Phase	Hours (Range)
Discovery and assessment	428
Target environment setup	520
CI/CD pipeline migration	240
Application migration execution	1,270
Testing and validation	504
Cutover and go-live	232
Post-migration support	352
Subtotal	**3,546**
Contingency buffer (20%)	709
Grand total (base effort + 20% contingency + project management overhead)	**4,255 Hours**

Key dependencies and risks

Consider the following dependencies that might delay migration.

Dependency	Impact if delayed

AWS account access and permissions	Blocks environment setup
Application owner availability	Delays dependency mapping
Network connectivity establishment	Blocks integration testing
Security team sign-offs	Blocks go-live
Change advisory board approvals	Delays cutover scheduling

Best practices

General best practices:

- Work backwards from workload requirements to design well-architected clusters
- Maintain parity between production and non-production environments
- Create Architectural Decision Records (ADRs) for key decisions
- Separate one-way door decisions (irreversible) from two-way door decisions (reversible)
- Add 15% for project management overhead and 10-20% risk buffer
- Ensure Kubernetes version compatibility (Amazon EKS doesn't support alpha features)

Related resources

1. What is Red Hat OpenShift Service on AWS

   Overview of ROSA architecture and capabilities
2. Amazon EKS Best Practices Guide

   Operational best practices for running workloads on Amazon EKS

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Initial publication	—	July 22, 2026