AWS Well-Architected Framework

Life Sciences Lens

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Life Sciences Lens: AWS Well-Architected Framework

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Abstract and introduction ..... **i**
Lens availability 2
Definitions **3**
Design principles **7**
Scenarios **9**
LS-S01 Research and discovery 9
Use cases 9
LS-S02 Clinical development 10
Use cases 9
Reference architecture 11
LS-S03 Manufacturing and supply chain 13
Use cases 9
Reference architecture 11
LS-S04 Commercial and medical affairs 15
Use cases 9
Reference architecture 11
Data lifecycle **18**
Design phase 18
Data gathering, analysis, publication, and manufacturing phases 18
Operational excellence **20**
Organization 20
LSOPS01-BP01 Regularly identify and classify applicable regulatory frameworks 21
LSOPS02-BP01 Establish a central control framework 23
LSOPS02-BP02 Implement regulatory reviews 26
LSOPS03-BP01 Perform supplier and vendor assessment of each vendor 27
LSOPS03-BP02 Limit available services to improve regulatory adherence 28
LSOPS03-BP03 Establish clear definitions of responsibilities between you, vendors, and users 29
LSOPS04-BP01 Establish IT quality oversight 30
LSOPS04-BP02 Enforce controls in IT tooling and automation 32
LSOPS04-BP03 Incorporate formal risk management into your IT processes 34
Prepare 37
LSOPS05-BP01 Enable a configuration management framework 37
LSOPS06-BP01 Validate data quality during ingestion 40

LSOPS06-BP02 Implement data pipeline testing 41
Operate 43
LSOPS07-BP01 Maintain a controlled multi-account environment 45
LSOPS07-BP02 Isolate GxP data from non-GxP data 47
LSOPS08-BP01 Continuously generate and maintain evidence 48
LSOPS09-BP01 Track data ownership and lineage over the life of the project 51
LSOPS10-BP01 Reproducibility 52
LSOPS10-BP02 Store data in a format that works both for archiving and for active use by retaining related metadata 54
LSOPS11-BP01 Identify clear dataset owners and access history 55
LSOPS12-BP01 Create a controlled semantic layer 56
Evolve 57
LSOPS13-BP01 Store data in an AI/ML-ready formats 58
Security 59
Design principles 59
Identity and access management 60
LSSEC01-BP01 Implement the principle of separation of duties 61
LSSEC01-BP02 Maintain a history of IAM configurations and changes over time 62
LSSEC01-BP03 Set up alerts for IAM configuration changes and perform audits 63
Data privacy 64
LSSEC02-BP01 Determine applicable regulatory frameworks and enforce data privacy requirements by implementing controls 64
Reliability 66
Design principles 66
Foundations 67
LSREL01-BP01 Identify and protect sensitive data elements with auditable classification. 68
LSREL01-BP02 Decouple anonymization logic from core workflows using orchestration and versioning. 71
LSREL02-BP01 Build resilient and highly available research solutions 73
LSREL02-BP02 Maintain continuous data availability and integrity 75
LSREL03-BP01 Digitize and modernize archival of research data 76
LSREL03-BP02 Implement tiered storage and recovery strategies 78
LSREL04-BP01 Map regulatory requirements to reliability controls 79
LSREL04-BP02 Implement risk-based reliability testing for regulated systems 81
LSREL04-BP03 Establish reliability qualification procedures 82

Workload architecture 84
LSREL05-BP01 Design edge buffering and queuing for laboratory instruments during network disruptions 86
LSREL06-BP01 Orchestrate workflows with checkpointing and failure isolation 88
LSREL07-BP01 Implement system-wide data checksums and transfer validation 90
LSREL07-BP02 Build idempotent and reproducible processing pipelines 91
LSREL07-BP03 Use staged validation and data quarantine mechanisms 92
LSREL07-BP04 Track data lineage with lifecycle metadata 93
LSREL08-BP01 Incorporate validated redundancy into architecture design 94
LSREL08-BP02 Design compliance-aware failover workflows 95
LSREL08-BP03 Align architecture priorities with scientific and regulatory context 96
Change nanagement 97
LSREL09-BP01 Create and verify rollback plans 98
LSREL09-BP02 Implement safe deployment strategies (like blue/green or canary) 100
LSREL09-BP03 Verify data integrity and point-in-time recovery 101
LSREL09-BP04 Maintain auditable rollback and recovery records 102
LSREL09-BP05 Implement risk-based change control for validated systems 104
LSREL09-BP06 Automate validation testing for changes 105
LSREL10-BP01 Implement comprehensive reliability testing 106
LSREL10-BP02 Validate end-to-end reliability of regulated workloads 107
LSREL10-BP03 Test data integrity under failure conditions 108
Failure management 110
LSREL11-BP01 Implement monitoring of equipment telemetry to detect anomalies 111
LSREL11-BP02 Apply predictive maintenance using AI models 112
LSREL11-BP03 Plan redundancy for critical laboratory equipment 114
LSREL12-BP01 Implement recovery processes with data integrity verification 115
LSREL12-BP02 Define recovery time objectives based on scientific and business impact ... 116
LSREL12-BP03 Maintain data consistency in distributed research systems 118
LSREL12-BP04 Implement cyber resilience for GxP-regulated backup data 119
LSREL13-BP01 Implement comprehensive monitoring for regulated systems 123
LSREL13-BP02 Monitor data integrity across scientific processing pipelines 124
LSREL13-BP03 Track reliability metrics aligned to regulatory needs 125
Performance efficiency 127
Design principles 127
Architecture selection 128

LSPERF01-BP01 Design and benchmark computing architecture for genomic workloads to optimize cost-performance ratio 130
LSPERF01-BP02 Specialized hardware selection and optimization for genomic and molecular workloads 132
LSPERF01-BP03 Performance optimizations should validate data integrity 133
LSPERF02-BP01 Data-aware storage tiering 134
LSPERF02-BP02 Secure data separation by classification 136
LSPERF02-BP03 Elastic data processing pipelines 137
LSPERF03-BP01 Workload-specific performance analysis 138
LSPERF03-BP02 Environment isolation by workload type 140
LSPERF03-BP03 Tailored service configuration by use case 141
LSPERF04-BP01 Performance consistency through clinical trial lifetime 143
Compute and hardware 144
LSPERF05-BP01 Specialized hardware matching 145
LSPERF05-BP02 Establish a tiered infrastructure strategy 147
LSPERF05-BP03 Implement a comprehensive system optimization strategy 148
LSPERF06-BP01 Run comprehensive, benchmark-driven assessments 149
LSPERF06-BP02 Perform a total cost of ownership analysis 150
LSPERF06-BP03 Perform an environment compatibility validation 151
LSPERF07-BP01 Use a risk-based validation framework 152
LSPERF07-BP02 Design a compliant-by-design infrastructure 153
LSPERF07-BP03 Develop an agile change management process 155
LSPERF08-BP01 Implement holistic system performance monitoring beyond traditional latency metrics 156
LSPERF08-BP02 Track resource utilization with clinical context 157
LSPERF08-BP03 Implement clinical system monitoring with workflow validation and impact analysis 158
Data management 159
LSPERF09-BP01 Evaluate data stores based on regulatory requirements and data governance capabilities 160
LSPERF09-BP02 Optimize query performance and meet diverse data access requirements in your environment 161
LSPERF10-BP01 Implement tiered caching architecture with clinical data classification and lifecycle management 163
LSPERF10-BP02 Establish intelligent cache warming and preloading based on clinical workflow patterns and seasonal variations 164

LSPERF11-BP01 Optimize large dataset storage based on project phases and collaboration needs 165
LSPERF11-BP02 Monitor data usage patterns and automatically adjust storage tiers for cost optimization 166
Network and content delivery 167
LSPERF12-BP01 Implement network segmentation with defense-in-depth controls 169
LSPERF12-BP02 Deploy accelerated encryption technologies with hardware offloading ... 170
LSPERF12-BP03 Optimize data transfer with intelligent traffic management and compression 171
LSPERF13-BP01 Conduct comprehensive site technology assessment and gap analysis 173
LSPERF13-BP02 Implement secure data exchange architecture with regulatory controls .. 174
LSPERF13-BP03 Deploy resilient connectivity with bandwidth optimization for remote locations 175
LSPERF14-BP01 Conduct performance benchmarking across geographic research hubs ... 177
LSPERF14-BP02 Evaluate multi-CDN architectures with intelligent traffic routing capabilities 178
LSPERF14-BP03 Assess edge computing integration for localized processing of research workloads 180
LSPERF15-BP01 Implement application-aware network path optimization and traffic prioritization methods 182
LSPERF15-BP02 Deploy synthetic transaction monitoring and real user monitoring with automated performance optimization 183
LSPERF16-BP01 Deploy intelligent traffic shaping with security-aware bandwidth allocation for different data types 185
LSPERF17-BP01 Measure baseline data transfer performance and evaluate how data sovereignty requirements affect latency 186
LSPERF17-BP02 Implement data classification-based transfer assessment with region-specific regulatory validation 187
Process and culture 189
LSPERF18-BP01 Perform cross-functional performance reviews between IT and scientists 190
LSPERF18-BP02 Document performance standards aligned with validation requirements 191
LSPERF18-BP03 Create integrated dashboards showing both performance and metrics ... 192
LSPERF19-BP01 Implement infrastructure as code for consistent test environments 193
LSPERF19-BP02 Establish comprehensive performance metrics and evidence collection ... 194

LSPERF19-BP03 Schedule regular performance tests with production-representative data loads ... 195
Cost optimization ... 198
Design principles ... 198
Practice Cloud Financial Management ... 198
LSCOST01-BP01 Establish and implement a comprehensive financial governance framework ... 199
LSCOST01-BP02 Analyze and optimize vendor cost structures and economic relationships ... 201
LSCOST01-BP03 Build the right skills and fostering a cost-aware culture ... 202
Cost-effective resources ... 203
LSCOST02-BP01 Implement a strategic approach to AWS credit program utilization ... 204
Manage demand and supply resources ... 205
LSCOST03-BP01 Align infrastructure capacity with R&D development stages ... 205
Optimize over time ... 207
LSCOST04-BP01 Implement tiered storage strategies aligned with data access patterns and retention requirements ... 207
LSCOST04-BP02 Implement data retention and deletion policies aligned with regulatory requirements ... 209
Sustainability ... 211
Design principles ... 212
Research computing optimization ... 213
LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage ... 214
LSSUS01-BP02 Use energy efficient hardware and services ... 217
Sustainability metric tracking and reporting ... 220
LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads ... 220
Data management efficiency in data analytics and data lifecycle ... 223
LSSUS03-BP01 Optimize data management for sustainability in life sciences ... 223
LSSUS03-BP02 Process data closer to source ... 226
Sustainability in manufacturing environments ... 230
LSSUS04-BP01 Continuously improve the monitoring of resource consumption ... 230
LSSUS04-BP02 Use digital twins to optimize resource usage through in silico experimentation ... 233
Sustainability in clinical trials ... 236
LSSUS05-BP01 Design sustainable clinical trial architecture ... 236

Conclusion **240**
Contributors **241**
Document revisions **242**
Notices **243**
AWS Glossary **244**

Life Sciences Lens - AWS Well-Architected Framework

Publication date: **December 30, 2025** (Document revisions)

This paper describes the Life Sciences Lens for the AWS Well-Architected Framework, which enables you to review and improve your cloud-based architectures and better understand the impact of design decisions. We present general design principles and specific best practices aligned to the six pillars of the Well-Architected Framework.

The life sciences industry includes organizations involved in the research, development, manufacture, distribution, and tracking of insights related to drug therapies and related interventions for improving human health and fighting disease.

A key requirement for many life sciences is GxP adherence. GxP refers to the collection of quality guidelines and regulations that verify that systems, facilities, equipment, and processes used in regulated industries are designed, monitored, and controlled to produce consistent, high-quality outcomes while maintaining data integrity and patient safety. The x in GxP represents different contexts where these practices apply throughout the drug development life cycle. Beginning with research and discovery, companies identify drug targets, analyze biological and genomic data, and gather laboratory data to create candidates for new therapies following Good Laboratory Practices (GLP). Clinical trials validate safety and effectiveness with regulatory oversight, following Good Clinical Practices (GCP) with in-depth data analysis and collaboration research and clinical organizations (internal and external).

After approval, Good Manufacturing Practices (GMP) must be followed to manufacture them. Once drug therapies are available for patients, data must be collected and analyzed to understand efficacy of their use in treatments and to evaluate adverse events through the use of real world data (RWD) and real world evidence (RWE).

There are common patterns of regulatory adherence, data strategy and collaboration, and compute requirements throughout this pipeline of developing drug therapies, bringing them to market, and gaining insights into their use and efficacy. The Life Sciences Lens addresses these scenarios and provides guidance for common patterns along with recommendations for services, architectures, and configurations to incorporate into a life sciences workload in an AWS environment.

Lens availability

Custom lenses extend the best practice guidance provided by AWS Well-Architected Tool. AWS WA Tool allows you to create your own custom lenses, or to use lenses created by others that have been shared with you.

To begin reviewing your life sciences workload, download and import the Life Sciences Lens into AWS Well-Architected Tool from the public AWS Well-Architected custom lens GitHub repository.

Definitions

- **21 CFR Part 11:** Regulations by the US Food and Drug Administration (FDA) that specify requirements for GxP systems.
- **ALCOA+:** Framework to verify data integrity based on the following principles:
  - Attributable
  - Legible
  - Contemporaneously recorded, original, or a true copy
  - Accurate
  - Complete
  - Consistent
  - Enduring
  - Available
  - Traceable
- **Bioinformatics:** The application of computational tools and methods to analyze, interpret, and manage biological data, including genomic sequences, protein structures, and molecular interactions.
- **Biomarker:** Measurable biological indicator (such as a gene, protein, or metabolite) that can be objectively measured to assess normal biological processes, disease states, or responses to therapeutic interventions.
- **Clinical Data Interchange Standards Consortium (CDISC):** Organization that develops data standards for the pharmaceutical industry. CDISC standards are used to provide study data for FDA submissions.
- **Change control:** A systematic process for managing modifications to validated systems, verifying that changes are documented, tested, approved, and implemented without compromising system integrity or regulatory adherence.
- **Contract research organization (CRO):** Third-party company that provides research and development services (including those in support of clinical trials) to pharmaceutical, biotechnology, and medical device industries.
- **Data governance:** Framework and implementation of policies to verify data quality, security, privacy, and regulatory adherence throughout the data lifecycle.

- **Data lineage:** Tracking and maintaining documentation related to the flow of data from its origin through its use, transformation, and storage in a regulated system.
- **Data sovereignty:** The concept that data is subject to the laws and governance structures of the country or region where it is collected or stored, which is particularly important for patient data or other data used in clinical trials.
- **Digital Imaging and Communications in Medicine (DICOM):** A standard for handling, storing, printing, and transmitting medical imaging information and related data. For more detail, see Medical Imaging on AWS.
- **Digital twin:** Virtual representation of a physical system or process that enable experimentation, simulation, analysis, and prediction of outcomes.
- **Drug discovery:** Process of identifying and developing new pharmaceutical compounds for treating specific diseases or conditions.
- **Electronic Lab Notebook (ELN):** Digital system that replaces traditional paper laboratory notebooks, enabling researchers to capture, organize, search, and share experimental data, protocols, and observations. ELNs provide audit trails for regulatory adherence, improve data integrity through version control and electronic signatures, and integrate with laboratory systems such as LIMS.
- **FAIR data principles:** Guidelines for managing scientific data to promote data sharing and collaboration in research environments. FAIR stands for findable, accessible, interoperable, and reusable.
- **Fast Healthcare Interoperability Resources (FHIR):** Data standard for storing and exchanging healthcare information electronically and enabling interoperability between healthcare systems.
- **Genomics or omics:** The study and use of genetic sequencing and related data to produce actionable insights which also includes proteomics and transcriptomics.
- **General Data Protection Regulation (GDPR):** European Union regulation governing data protection and privacy. For more detail, see General Data Protection (GDPR) Center.
- **GxP:** Good practice guidelines that improve quality and cover a variety of life sciences workloads, including (but not limited to) Good Laboratory Practices (GLP), Good Clinical Practices (GCP), and Good Manufacturing Practices (GMP).
- **ISA-95:** International standard that defines a framework for integrating enterprise business systems with manufacturing control systems. It establishes communication interfaces between operational layers in industrial environments.
- **Pharmacovigilance:** The science and activities related to detecting, assessing, understanding, and blocking adverse effects or other drug-related problems throughout the lifecycle of

pharmaceutical products. Includes systematic monitoring and evaluation of safety information from healthcare providers and patients, with regulatory requirements for adverse event reporting and risk management.

- **Quality management system (QMS):** System that documents processes, procedures, and responsibilities for achieving quality policies and objectives to meet regulatory requirements in life sciences organizations. For more detail, see resources in AWS Artifact.
- **Health Insurance Portability and Accountability Act (HIPAA):** United States federal law that establishes privacy and security standards for protecting patient data stored or transmitted by regulated entities.
- **Laboratory information management system (LIMS):** Software system that manages laboratory operations, including sample tracking, workflow automation, data management, instrument integration, and quality control for analytical and research laboratories. LIMS systems provide chain of custody documentation and verify data integrity and traceability required for GxP adherence.
- **Molecular dynamics:** Computational simulation methods used to analyze the physical movements and interactions of atoms and molecules over time. For more detail, see High Performance Computing.
- **Picture Archiving and Communication System (PACS):** Medical imaging technology or systems that store, retrieve, manage, distribute, or make available medical images.
- **Protected health information (PHI):** Individually identifiable health information that is stored or transmitted and subject to regulatory requirements, such as HIPAA.
- **Real world data (RWD):** Data collected from real-world sources outside of clinical trials including electronic health records, claims databases, and patient registries. This provides a broader set of data in contrast to that collected in clinical trials which is limited to specific trial cohorts and protocols.
- **Real world evidence (RWE):** Clinical evidence derived from analysis of real-world data (RWD), used to support regulatory decisions and demonstrate the efficacy of treatments or therapies in real-world settings.
- **Semantic layer:** Abstraction layer providing business-friendly definitions and relationships for underlying data, enabling consistent interpretation across different systems and users.
- **Quality Risk Management (QRM):** Systematic approach to identifying, assessing, controlling, communicating, and reviewing risks to pharmaceutical product quality throughout the product lifecycle, supporting science-based decision-making with the primary goal of patient safety.

- **Validation (like frameworks, approaches, and protocols):** Systematic processes to verify that systems, processes, and methods consistently produce results that meet predetermined specifications and quality attributes.

For the latest AWS terminology, see the AWS glossary in the AWS Glossary Reference.

Design principles

The Well-Architected Framework identifies a set of general design principles to facilitate good design in the cloud. In addition, the following design principles should also be considered for designing and operating life sciences workloads.

The development of drug therapies is highly regulated for safety, efficacy, and security of drug therapies. In the United States, the Federal Drug Administration (FDA) mandates a series of Good Practices (GxP) which covers practices related to laboratories, clinical development, and manufacturing. This requires careful management of data integrity throughout the scenarios. Drugs are represented by data including the related research, biological data, genomic data, data gathered through clinical trials, data related to its manufacture, and real world data (RWD) collected after they are used in real world clinical settings. Data collaboration is also a common design principle, as there are several organizations involved including research institutions, governmental regulators, contract research organizations (CROs), and healthcare payors and providers.

- **Implement strong security foundations:** Follow industry best practices as detailed in the AWS Well-Architected Security Pillar whitepaper. Address GxP requirements that affect IT both as a source of requirements for workloads and as a set of controls over how those workloads should be built and operated.
- **Adhere to regulatory frameworks:** Become familiar with regulatory frameworks based on geography and industry best practices such as ISO 27001 and HIPAA. Create a controlled infrastructure with a prescribed account vending process, incorporating a layered model emphasizing infrastructure qualification and tooling verification. Shift from document-based to data-driven processes, prioritizing data integrity to align with FDA requirements (21 CFR Part 11 and ALCOA+). Implement continuous adhrerence through automated audit controls using defined configurations, compliance packs, and regular auditing to create a comprehensive, trustworthy environment that can integrate with a data lake.
- **Develop robust quality management systems:** Effective quality management (QM) and quality risk managment (QRM) is mandatory in the highly-regulated pharmaceutical, biotechnology, and medical device industries. Integrate effective QM and QRM practices into information technology systems to improve product quality, patient safety, and regulatory adherence throughout the product lifecycle.
- **Design comprehensive data management strategies:** Begin by reviewing the Data Analytics Lens. Address complexity based on the type of data created, collected, and stored, and consider

the complete lifecycle of that data in relation to project phases. For geographically distributed operations such as multi-site clinical trials, decentralized research, or global manufacturing, implement distributed file systems that enable edge compute capabilities while maintaining centralized governance. Design architectures that process and validate data locally at clinical sites, manufacturing facilities, or research labs to optimize performance and reduce latency, while synchronizing to central repositories for regulatory adherence, audit, and collaboration. Balance local autonomy with centralized security controls, verifying data integrity and regulatory adherence across edge locations through encrypted data transfer, access controls, and comprehensive audit trails.

- **Enable secure cross-organizational collaboration:** Implement robust governance frameworks, audit trails, access controls, and long-term archival plans for managing data throughout its entire lifecycle. Adopt standardized formats (like CDISC, FHIR, and ISA-95) to promote collaboration between teams and systems. Consider each phase of the project from design through closeout while verifying that data can be securely shared with research institutions, regulators, and contract research organizations (CROs) while adhering to regulations. For more detail, see Data lifecycle.

Scenarios

In this section, we cover key scenarios that are common in many life sciences implementations.

Scenarios

- LS-S01 Research and discovery
- LS-S02 Clinical development
- LS-S03 Manufacturing and supply chain
- LS-S04 Commercial and medical affairs

LS-S01 Research and discovery

Research and discovery (R&D) represents the crucial first phase in life sciences innovation, where organizations work to identify and develop potential drug candidates. This phase involves complex data analysis across multiple domains including drug discovery, bioinformatics, genomics, and laboratory operations. Organizations face challenges in managing and analyzing vast amounts of complex scientific data, implementing efficient computational workflows, and maintaining regulatory adherence while accelerating the pace of discovery. Modern approaches use advanced technologies like AI/ML, high-performance computing, and cloud storage solutions to process and analyze data at scale, while adhering to Good Laboratory Practices (GLP) and Findable, Accessible, Interoperable, and Reusable (FAIR) data principles.

Use cases

- **LS-S01-UC01 Drug discovery:** Drug discovery is a costly and lengthy process. Pharmaceutical companies are actively seeking to accelerate the discovery phase, especially in areas such as understanding of disease mechanisms using multi-modal data analysis and rapidly identifying molecular targets and candidate drug leads. These steps involve collection, standardization, storage, management, and governance of large amounts of complex data along with the availability of specialized compute to generate insights from them. With the advent of specialized AI models, pharmaceutical organizations are seeking to use them to advance their drug discovery timelines by orders of magnitude in a secure, transparent, and ethical manner.
- **LS-S01-UC02 Bioinformatics:** Bioinformatics plays a key role in providing as well as using the necessary tools and databases that enable researchers and clinicians to interpret biological data. Bioinformaticians and data scientists, in most types of organizations, such as clinical diagnostics,

population sequencing, pharmaceutical R&D, and agriculture, have the necessary skills to build and use pipelines that process raw data, such as genomics data, and interpret that data using analytics tools, machine learning models, and statistical analysis. Bioinformatics teams seek to simplify their analysis lifecycle by focusing more on the science and less on infrastructure. Additionally, they constantly seek to optimize the performance, accuracy, and cost of analysis for better scientific and business outcomes.

- **LS-S01-UC03 Genomics:** Genomics and other omics data (like transcriptomics or proteomics) play a crucial role in clinical diagnostics, drug discovery, gene editing, precision medicine, and bio-surveillance. Omics data typically has a large footprint (for example, raw and processed genomics data for single human patient can range from 100 to 200 GB). Based on the type of project, organizations can maintain tens of thousands, sometimes millions of patients worth of data. Such organizations face challenges to manage the cost and governance of this data. They actively seek to simplify data lifecycle, implement FAIR principles within and across organizations, and reduce short and long-term storage and usage costs. Raw omics data needs to be processed using bioinformatics workflows to derive insights from them. Organizations actively look to process these data at scale in the most performant and cost-efficient manner. They are faced with several options to do so, whether running it themselves in a datacenter on-premises, using a vendor that provides a pre-built experience, or building their own infrastructure in the cloud. They seek guidance on the best option for them given their goals, team size, internal skillsets, budget, and existing environmental constraints.
- **LS-S02-UC04 Connected labs:** R&D organizations are seeking to modernize their laboratory infrastructure, aiming to use data to enhance research capabilities. A key focus of this transformation is optimizing laboratory data integration and accessibility. By addressing the current state of siloed data across laboratory instruments, Laboratory Information Management Systems (LIMS), and Electronic Lab Notebooks (ELN), companies seek to enable data integration between on-premises data and cloud analytics while following FAIR data principles. This forward-thinking approach empowers scientists with more comprehensive and readily available data, accelerating insights and innovation. Furthermore, the establishment of robust data governance for unstructured lab data improves the usage of AI/ML and advanced analytics to accelerate drug discovery.

LS-S02 Clinical development

Clinical trials stand as the bridge between scientific discovery and patient care. The landscape of clinical trials has evolved dramatically, moving beyond simple surveys to encompass vast, diverse datasets from multiple sources and decentralized participants. Decentralized clinical trials

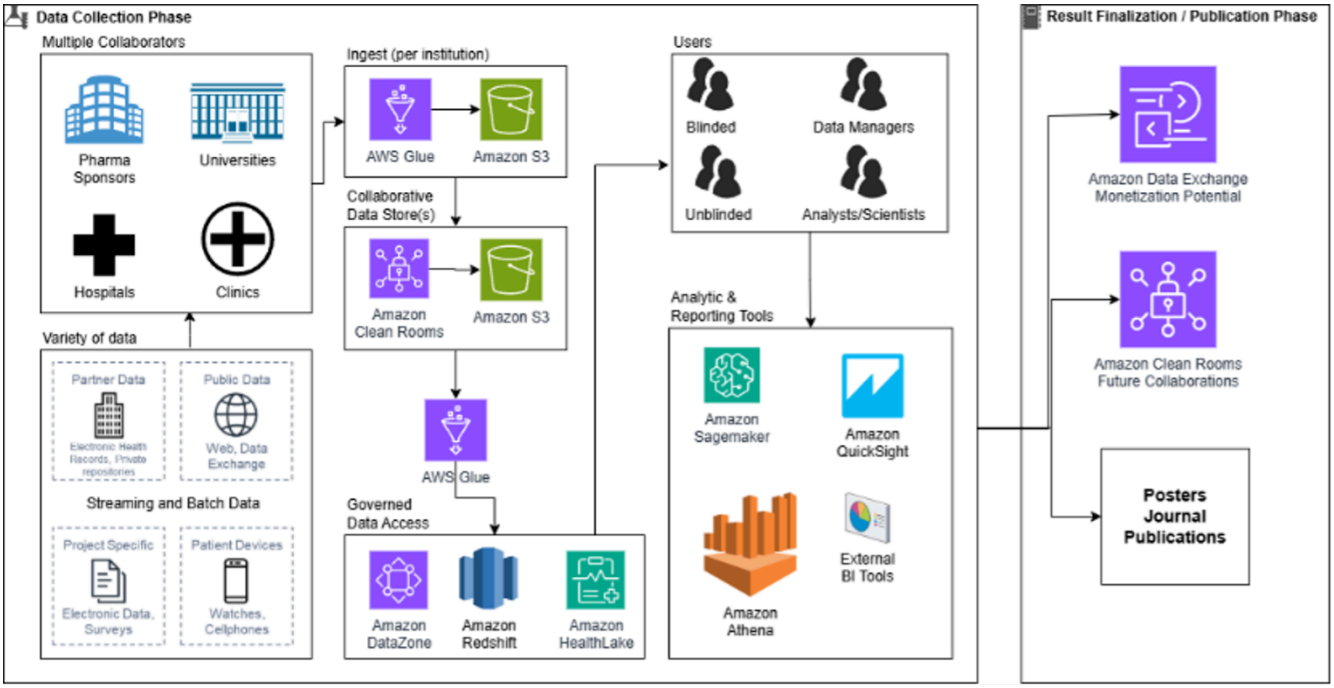
(DCTs) enable patient-centric data collection through remote monitoring via mobile devices, wearables, and telemedicine, allowing participants to contribute from home while maintaining data quality and regulatory adherence. This aims to address critical needs, providing a blueprint for a comprehensive data strategy that accelerates research, enhances collaboration, and ultimately leads to more informed, impactful results in patient healthcare.

Use cases

- **LS-S02-UC01 Data collaboration:** Researchers face the challenge of collecting, managing, and analyzing complex data from various clinics, decentralized trial participants using mobile devices and wearables for remote monitoring, real-world observations, medical records, and public repositories. This shift demands a robust, flexible, and secure data solution that can empower even small technical teams to efficiently gather, process, and use this wealth of information. For multi-institution collaboration researchers need the ability to share identified and de-identified data.
- **LS-S02-UC02 Real time responsive studies:** Researchers need the ability to analyze incoming data upon ingestion, enabling them to identify patterns, adjust study parameters, and explore emerging questions promptly.
- **LS-S02-UC03 Secondary analysis:** Studies are often conducted using data from prior trials. Result sets and artifacts should be prepared to enable these sorts of studies by clearing out personal health information (PHI) and other proprietary data.
- **LS-S02-UC04 Extended duration:** Clinical trials can run for decades. Tooling built for extended durations should be adaptable to change as technology changes but still keep the data consistent, accurate and secure.

Reference architecture

The following reference architecture aims to address the critical needs, providing a blueprint for a comprehensive data strategy that accelerates research, enhances collaboration, and ultimately leads to more informed, impactful results from clinical trials.



1. Data ingestion:
   - Use AWS Glue for both batch and real-time data integration from various sources.
   - Schedule AWS Glue jobs to ingest streaming or batch data using an API based on events or a schedule.
2. Secure collaboration:
   - Use AWS Clean Rooms to build a secure environment for multi-party collaboration using the raw, identified data.
   - AWS Clean Rooms can be used while data is still being gathered to allow for faster insights.
   - Grant collaborating users granular access through query rules allowing for matching without risking PHI exposure.
   - Enable SQL queries on combined datasets without moving or exposing raw data.
   - Store raw data in a single Amazon S3 bucket per partner partitioned by patient to enable permission granting and simple right to forget for individual patients.
3. Transform and explore:
   - Create AWS Glue workflows managing AWS Glue jobs or scripts to transform and prepare the data
   - Insert data into AWS HealthLake for interoperability through the FHIR model.
   - Insert related structured data into Redshift for later complex analysis and internal collaboration.

- Build a DataZone domain to assign the correct permissions allowing engineers, data scientists, product managers, analysts, and business users to access data so that they can discover, use, and collaborate to derive data-driven insights.
- Data scientists, engineers and managers can access data using Quick, Amazon Sagemaker, Aazon Athena, or other business intelligence (BI) tools for internal analysis, exploration, and visualization creation.

4. Results sharing:
   - Use AWS Data Exchange to make the study's findings available to the broader research community.
   - AWS Data Exchange can be used to monetized resulting data.
   - AWS Clean rooms can be used again at the end. Maintain identified copies of data for potential future research questions.
   - Create visualizations, reports, and posters from the same tools used for initial exploration with the finalized, enriched data.

LS-S03 Manufacturing and supply chain

Life sciences manufacturing and supply chain is the critical bridge between scientific discovery and the delivery of life-saving therapies and diagnostics. As companies work to bring innovative treatments to market, they face increasing pressure to modernize their operations and adapt to evolving patient needs. Key challenges include improving operational equipment effectiveness (OEE) in an era of smaller batch production, transitioning from manual spreadsheets to integrated production planning, and scaling quality checks with limited manpower.

Additionally, the rise of personalized medicine, where drug shelf life is often just a few days, demands a more predictable and agile supply chain. Manufacturers also seek holistic visibility across drugs, dosage forms, and production plants, as well as optimized order management to reduce locked-up capital in inventory. AWS has played a pivotal role in enabling digital transformation across the industry, assisting companies to address these challenges with cloud-based solutions that enhance efficiency, scalability, and real-time analytics.

Use cases

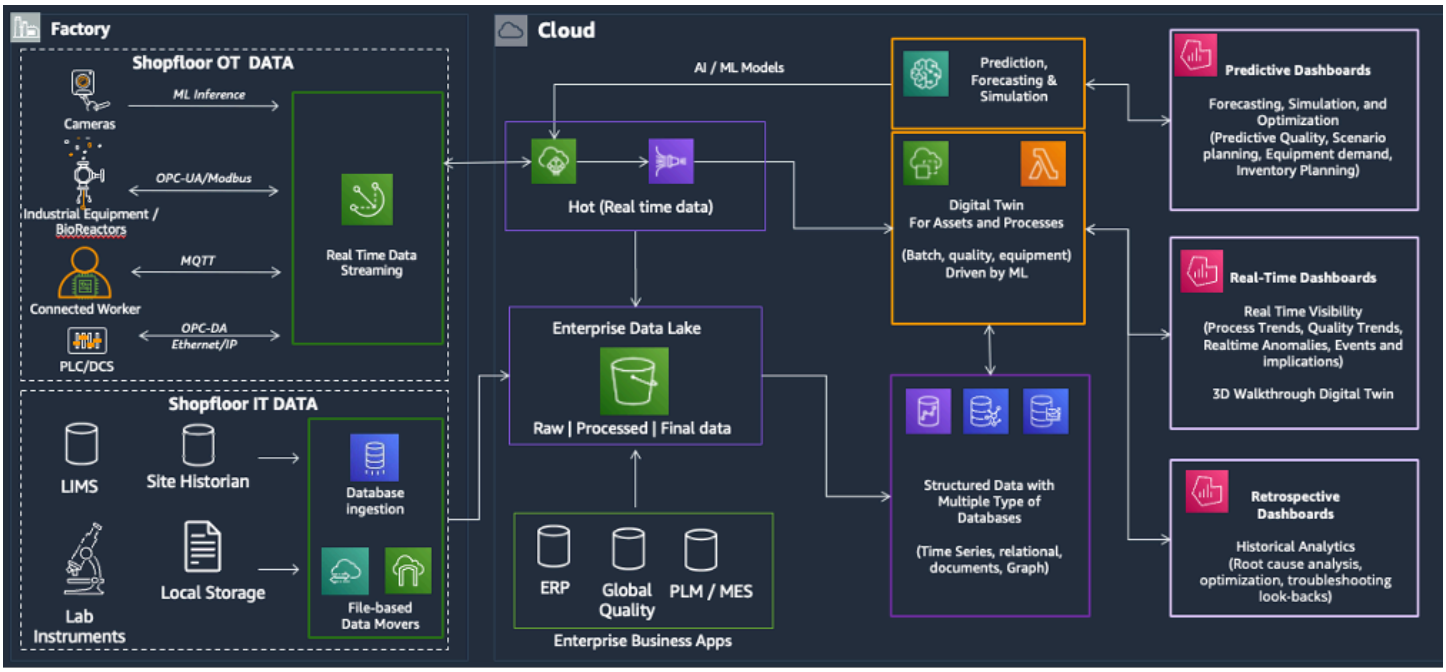
- **LS-S03-UC01 Smart factory for biopharma:** Biopharmaceutical manufacturers are embracing smart factory initiatives to enhance efficiency, quality, and scalability in drug production. These factories use automation, IoT-enabled equipment, and AI-driven analytics to optimize

production processes, reduce downtime, and provide consistent product quality. By integrating real-time monitoring, predictive maintenance, and digital twins, manufacturers can improve operational equipment effectiveness (OEE), streamline batch production, and enhance adherence to regulatory standards. A smart factory approach enables data flow across production lines, accelerating decision-making and enabling a more agile response to evolving patient and market demands.

- **LS-S03-UC02 Digital twin for factory and lab:** Biopharma manufacturers and manufacturing labs need real-time visibility into their operations to optimize efficiency, reduce variability, and enhance quality control. A *digital twin*, a virtual replica of physical assets, processes, and systems, enables organizations to simulate, monitor, and predict outcomes across both factory and lab environments. By integrating IoT sensors, AI-driven analytics, and historical data, digital twins assist to identify bottlenecks, improve process optimization, and enhance predictive maintenance. This technology enables better decision-making, accelerates troubleshooting, and creates a more adaptive and resilient manufacturing and R&D solution.
- **LS-S03-UC03 AI for bioprocess development:** Biopharma companies are using AI to accelerate and optimize bioprocess development, reducing development time for new therapies. AI-driven models analyze complex datasets from experiments, sensor readings, and historical production runs to uncover patterns, predict optimal conditions, and refine bioprocess parameters. This enables researchers to improve yield, enhance product quality, and minimize variability in cell culture, fermentation, and purification processes. By integrating AI into bioprocess development, manufacturers can streamline scale-up efforts, enhance process robustness, and reduce costly trial-and-error experimentation, ultimately driving more efficient and reliable therapeutic production.
- **LS-S03-UC04 Computer vision for quality control:** Providing consistent product quality in life sciences manufacturing is critical, yet traditional quality control (QC) methods are often labor-intensive and prone to human error. Computer vision, powered by AI and machine learning, enables automated, real-time inspection of biopharma production lines, detecting defects, contamination, and deviations in packaging, labeling, and drug formulation. By integrating high-resolution imaging with deep learning models, manufacturers can enhance batch predictability, reduce waste, and improve adherence to stringent regulatory standards. This approach scales QC processes efficiently, even with limited manpower, improving overall manufacturing precision and product safety.
- **LS-S03-UC05 Supply chain forecasting:** Life sciences manufacturers face increasing complexity in managing supply chains, especially with the rise of personalized medicine and just-in-time production. AI-driven supply chain forecasting uses real-time data, historical trends, and external factors such as market demand and regulatory changes to predict material needs,

optimize inventory levels, and minimize disruptions. By improving visibility across suppliers, manufacturing sites, and distribution networks, companies can reduce waste, free up locked capital, and speed up the delivery of critical therapies. This predictive approach enhances resilience, agility, and efficiency in life sciences manufacturing supply chains.

Reference architecture



LS-S04 Commercial and medical affairs

Medical affairs and commercial teams serve distinct yet complementary roles in life sciences companies. The commercial team drives success through sales, marketing, and business development. Medical affairs maintains scientific integrity through clinical support, regulatory adherence, and scientific communications. Medical affairs also serves as a critical bridge between clinical development and commercialization, maintaining scientific integrity and regulatory adherence in commercial activities. By sharing their perspectives and knowledge, these teams can develop a more nuanced view of the landscape, enabling them to make informed decisions about product development, marketing strategies, and customer engagement.

Use cases

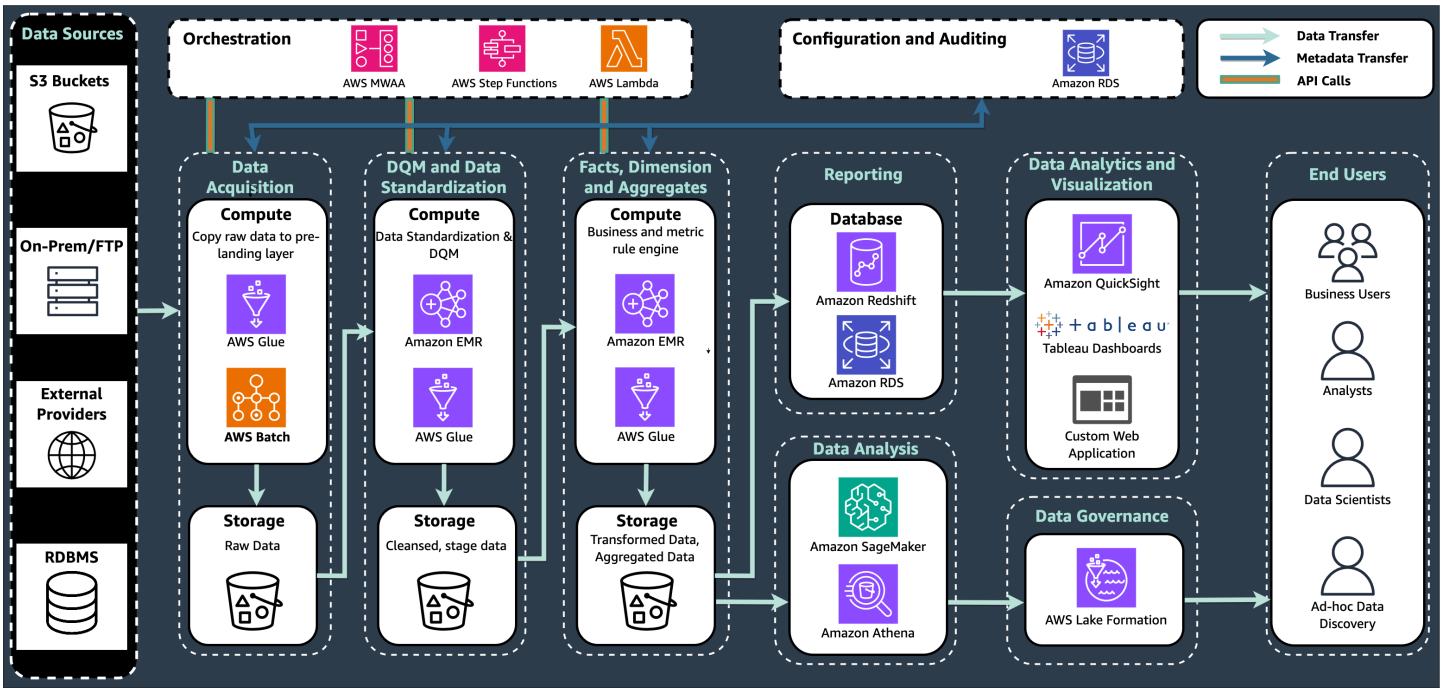
- **LS-S04-UC01 Real world evidence collection and management:** Real world evidence (RWE) has become a critical tool for the life science industry to navigate the changing healthcare landscape

and address the increasing scrutiny on drug pricing and reimbursement. By using diverse data sources such as electronic health records, insurance claims, wearable devices, and social media, RWE provides a more comprehensive view of a drug's performance in real-world settings, beyond the controlled environment of clinical trials. This approach not only demonstrates a drug's effectiveness and safety but also supports pricing strategies and negotiations with payers, potentially leading to better formulary positioning and optimal reimbursement rates in an increasingly value-based healthcare system.

- **LS-S04-UC02 Adverse event detection and reporting:** Life sciences companies have a responsibility to monitor adverse events during clinical trials and post-market approval, as part of their regulatory obligations around pharmacovigilance. These companies receive unstructured adverse event reports in the form of free text and notes during both clinical trials and post-market surveillance. This manual processing approach is resource-intensive and costly. By implementing automation solutions, companies aim to streamline the intake and analysis of adverse event information, significantly reduce human intervention, maintain regulatory adherence, and lower operational costs while improving the accuracy of safety reporting.
- **LS-S04-UC03 Omnichannel personalized content assembly and campaign optimization:** Traditional, uniform marketing struggles to effectively reach and engage modern HCPs and patients. Personalization is no longer optional but necessary. The omnichannel approach represents a unified approach to stakeholder communication that combines multiple channels to deliver customized content effectively. The approach understands healthcare professional preferences, enabling personalized content delivery through a centralized repository of product and clinical information. From the medical affairs perspective, it enables efficient distribution of scientific content and management of key opinion leader relationships, while supporting medical science liaisons in their real-time interactions with healthcare professionals. By fostering collaboration between commercial and medical teams, it improves scientific accuracy in customer-facing materials while using medical insights to inform commercial strategies. This framework assists to identify unmet medical needs, treatment patterns, and emerging scientific questions, continuously refining engagement strategies. This integrated approach streamlines communication, enhances engagement effectiveness, and provides consistent messaging across touchpoints while maintaining adherence to industry regulations.
- **LS-S04-UC04 Regulatory adherence:** Organizations must maintain robust systems for material review, medical information dissemination, and field activities, while setting clear boundaries between medical and commercial functions. Success requires integrated approaches to risk management, including monitoring programs, issue detection, and corrective action plans, supported by appropriate technology solutions and documentation systems. The framework must also adapt to emerging challenges in digital health, social media, and virtual engagement

models. Essential elements include clear policies and procedures, regular training, strong documentation practices, and effective monitoring systems, each of which contributes to a culture of compliance-alignment. This landscape continues to evolve with new regulations, changing business models, and technological advances, requiring organizations to maintain agility in their approaches while consistently adhering to global requirements and industry best practices. The ultimate goal is to balance effective stakeholder engagement and business growth with unwavering commitment to regulatory adherence and patient safety.

Reference architecture



Data lifecycle

Life sciences organizations must carefully orchestrate data management across complex project lifecycles that span years or even decades, from initial research design through clinical trials to post-market surveillance. This section provides guidance for managing life sciences data through each critical phase—design, active data collection and analysis, reuse preparation, and compliant archival. The approach emphasizes regulatory adherence, data integrity, secure collaboration, and cost-effective storage strategies while keeping valuable research data accessible and reusable for future scientific endeavors. By following these lifecycle management practices, organizations can maintain audit readiness, enable cross-organizational collaboration, and meet stringent regulatory requirements throughout the entire data journey.

As with most data analytics workloads, begin with the Data Analytics Lens. Designing a data repository for life sciences related projects adds complexity based on the type of data created, collected, and stored, as well as the lifecycle of that data. The most straightforward way to approach this is to look at the lifecycle of a project.

Design phase

A group of documents will be generated to describe the project and decide how the data will be gathered and who will have access, such as the protocol, manufacture, and distribution plan, data capture forms, consent forms, and infrastructure as code (IaC) scripts. House these documents and the supporting material together in an auditable document management system. Make documents available in an exportable format to facilitate archiving. When the time comes, they can be transferred alongside other data and infrastructure artifacts to deep storage systems like Amazon Glacier.

System build outs should be reproducible. Construct environments using infrastructure as code so that you can create exact test and development replicas as well as take down the entire environment and archive the solution. Immediately rebuild archived solutions on demand if required. For example, you can build AWS systems using AWS CloudFormation Stacks.

Data gathering, analysis, publication, and manufacturing phases

During the active phase of the project, data will be generated from multiple sources, including electronic data capture, electronic health records, Internet of Things (IoT) device logs, and public

records. When deciding how to gather and store the data, keep in mind potential later usage and collaboration with other teams or for use in AI-driven workflows. Implement standardized data models and formats that promote interoperability and long-term accessibility.

Adopting industry standards such as CDISC or FHIR for health record data or ISA-95 for manufacturing records verifies that data can be shared, analyzed, and understood across different systems and by various stakeholders. Consider this standardization from the outset when writing project documents and designing data collection processes. For example, you can store FHIR data in AWS HealthLake to ease interoperability later on by guiding and enforcing the FHIR model.

During this phase, auditability and access controls are critical. Implementing a comprehensive data protection impact assessment (DPIA) assist to identify and mitigate potential risks. It is essential to maintain a clear map of data access, and appropriately de-identify sensitive information when necessary while still allowing for traceability back to individuals when required.

The system should maintain comprehensive audit trails using AWS CloudTrail and Amazon CloudWatch, tracking instances of data access and modifications. The infrastructure must support both PHI-containing and de-identified datasets, with appropriate security controls. Organizations can use AWS Glue for data cataloging and AWS Lake Formation for fine-grained access control, limiting data access to authorized personnel while maintaining the ability to collaborate securely through AWS Clean Rooms externally and Amazon Data Zone for internal traceable data governance.

As data is produced, determine if it is reusable. Can it be used for future clinical exploration? Can it be used for improving manufacturing processes in the future? Identify those data sets and prepare them to be reusable. Export the data into Amazon S3 in a format that contains the metadata and is offered through AWS Data Exchange, such as Iceberg or Parquet.

Once a study has concluded and the papers have been delivered or a manufacturing line has been shut down, there are regulations in different countries mandating retention of the related data. To adhere to these regulations, you can store the artifacts in Amazon Glacier. After the data is safely archived, remove the entire environment that once collected and housed the data. This verifies that no one can later access PHI inappropriately. By building out the AWS environment using AWS CloudFormation stacks, you can delete entire environments (stacks) in a straightforward and efficient way.

Operational excellence

The operational excellence pillar allows life science project designers to focus on managing risks associated with operating workloads in the cloud, satisfying regulatory requirements, and becoming more agile by automating the operation and management of traditionally error-prone manual processes.

This pillar delves deeper into the critical considerations necessary to effectively implement the design principles, providing practical guidance on how to apply these foundational concepts within a life sciences product or project.

Focus areas

- Organization
- Prepare
- Operate
- Evolve

Organization

LSOPS01: How do you identify applicable regulatory frameworks for your workload?

Life sciences workloads must comply with various regulatory frameworks (FDA, EMA, HIPAA, GDPR) depending on application type, data sensitivity, and operational geography. Failure to identify applicable frameworks creates operational risk through regulatory incidents and potential service disruptions.

LSOPS02: How do you implement regulatory oversight throughout your project lifecycle across the identified frameworks?

Regulatory oversight requires structured governance to maintain regulatory adherence as projects evolve and team members change. This includes establishing leadership roles, defining oversight processes, and consistently applying regulatory requirements across each project phase.

LSOPS03: How do you verify that services from external vendors have been approved by your teams to meet regulatory requirements?

Regulatory requirements extend to vendor solutions, requiring a proactive approach to verify regulatory adherence when selecting vendors. When choosing solutions, select only those verified as capable of meeting regulatory requirements by the vendor. Avoid trying to make non-compliant services fit regulatory requirements.

LSOPS04: How do you establish formal quality oversight of IT?

Life sciences regulations emphasize quality management as a key requirement. Establishing IT quality oversight provides the best way to meet IT regulatory requirements while maintaining a separation of responsibilities from pharmaceutical quality systems.

Best practices

- LSOPS01-BP01 Regularly identify and classify applicable regulatory frameworks
- LSOPS02-BP01 Establish a central control framework
- LSOPS02-BP02 Implement regulatory reviews
- LSOPS03-BP01 Perform supplier and vendor assessment of each vendor
- LSOPS03-BP02 Limit available services to improve regulatory adherence
- LSOPS03-BP03 Establish clear definitions of responsibilities between you, vendors, and users
- LSOPS04-BP01 Establish IT quality oversight
- LSOPS04-BP02 Enforce controls in IT tooling and automation
- LSOPS04-BP03 Incorporate formal risk management into your IT processes

LSOPS01-BP01 Regularly identify and classify applicable regulatory frameworks

Conduct a systematic assessment to identify applicable regulatory frameworks based on product type assessment, geographic analysis, data classification, development phase mapping, and risk-

based classification for comprehensive regulatory coverage for your life sciences workloads. Repeat this process as the project and its requirements evolve over time.

Desired outcome: A comprehensive regulatory framework map that clearly identifies applicable regulations, their specific requirements, and how they impact different aspects of your life sciences project. This enables informed decision-making and verifies that you don't overlook regulatory requirements.

Common anti-patterns:

- You assume a one-size-fits-all approach to auditing without accounting for specific regional or other requirements.
- You identify applicable regulatory frameworks as a one-time activity and assume adherence even as the project evolves.
- You assume the applicable regulatory frameworks have been identified without regular review to verify.

Benefits of establishing this best practice:

- Provides clear guidance for project stakeholders on regulatory expectations.
- Identifies required regulatory submissions and approvals.
- Reduces risk of becoming non-compliant over time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

GxP refers to the regulations and guidelines applicable to life sciences organizations that make food and medical products such as drugs, medical devices, and medical software applications. Depending on the location of research and development, governing bodies may include U.S. Food & Drug Administration (FDA), European Medicines Agency (EMA), or International Council for Harmonisation (ICH) Good Clinical Practice guideline (ICH-GCP). In the US, handling of Personal Health Information (PHI) is governed by HIPAA, in the EU, General Data Protection Regulation (GDPR). If you are manufacturing a product, you need to align with Good Manufacturing Practices (GMP).

Review regulations related to your project location and industry. Consider what type of data you will be handling in your application, where it is collected, and where it is stored. Consider if there

are data sovereignty or residency requirements that would determine the Region in which data can be stored or processed.

Implementation steps

1. Identify relevant regulatory frameworks based on the type of project and how data is managed, collected, and stored.
2. Document the frameworks that apply to your project and region. Review the resources available from AWS Compliance Programs to find AWS resources for common regulatory frameworks.
3. Identify requirements that are common across frameworks to identify necessary checks to satisfy applicable controls.

Resources

Related documents:

- Don't Blame Regulators: How Software Excellence Satisfies Compliance
- Life Sciences Compliance in the Cloud
- GxP
- GxP Systems on AWS

Related tools:

- AWS Audit Manager

LSOPS02-BP01 Establish a central control framework

Establishing a central control framework and mapping to applicable frameworks can simplify processes, avoid duplicate effort, and reduce operational overheads. The IT quality team can define the required control objectives, while IT process and tooling experts can determine the best way to implement the controls.

Desired outcome:

- Unified control framework that maps to multiple regulatory requirements.
- Reduced duplication of efforts across different frameworks.

- Streamlined audit processes with centralized evidence collection.

Common anti-patterns:

- You implement separate controls for each framework.
- You lack clear mapping between control objectives and regulatory requirements.
- You create duplicate controls that are common across frameworks.

Benefits of establishing this best practice:

- Streamlined reporting through consolidated evidence collection.
- Reduced audit preparation time and resource requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop a hierarchical structure with clear accountability that bridges the gap between high-level objectives and specific technical implementations. This structure should incorporate risk-based prioritization to focus resources on the most critical controls while maintaining comprehensive coverage of requirements.

Implementation steps

1. Inventory the applicable regulatory frameworks and requirements:

- Use AWS Audit Manager to catalog regulatory requirements across frameworks and collect evidence of auditing.
- Consider AWS Systems Manager Documents for standardized documentation.

2. Create a unified control catalog with clear mapping to regulatory requirements:

- Implement AWS Systems Manager Parameter Store for centralized control definitions.
- Consider Service Catalog for managing approved control implementations that can be deployed to multiple workloads.

3. Establish control ownership and responsibilities across teams:

- Use AWS Resource Access Manager for sharing control resources across teams.
- Consider AWS Organizations for defining organizational control responsibilities.

4. Develop standardized templates for control documentation and evidence collection:

- Store templates in Amazon S3 with appropriate access controls.
- Consider AWS CloudFormation for templating control implementation patterns.

5. Implement continuous monitoring of control effectiveness:

- Configure AWS Config Rules to verify control implementation.
- Consider AWS Security Hub CSPM for aggregating control status.

Resources

Related guides, videos, and documentation:

- Don't Blame Regulators: How Software Excellence Satisfies Compliance

Related examples:

- Conformance Pack Sample Templates for AWS Config
- Streamline compliance management with AWS Config custom rules and conformance packs

Related tools:

- AWS Audit Manager
- AWS Systems Manager
- Service Catalog
- AWS Resource Access Manager
- Amazon S3

- AWS CloudFormation
- AWS Config

LSOPS02-BP02 Implement regulatory reviews

Incorporate and electronically document formal reviews throughout the project lifecycle to improve regulatory adherence. This includes initial framework assessment reviews, design control gates, change management reviews, pre-submission readiness checks, and periodic audits to verify ongoing adherence. Document the decisions and maintain traceability of regulatory requirements implementation throughout each stage.

Desired outcome: A systematic, documented review process that improves regulatory adherence at every stage of the project lifecycle, with clear evidence of requirements verification, issue remediation, and approval decisions that satisfy regulatory expectations.

Common anti-patterns:

- Conducting reviews only at project completion rather than throughout the lifecycle.
- Treating reviews as checkboxes rather than substantive evaluations.

Benefits of establishing this best practice:

- Early identification and remediation of audit issues.
- Reduced regulatory submission risks.
- Streamlined audit processes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establish a formal review framework with clearly defined stages aligned to your development lifecycle. Define specific entry and exit criteria for each review gate, which explicitly addresses regulatory requirements. Implement electronic documentation systems that maintain review histories, decisions, and supporting evidence.

Regulatory reviews should be integrated into your project lifecycle rather than treated as separate activities. Map key decision points in your development process where regulatory reviews provide

maximum value. These typically include initial planning, design completion, pre-implementation, post-implementation, and pre-submission phases.

Create standardized templates and checklists for each review type for consistent evaluation. Involve cross-functional teams including regulatory affairs, quality assurance, technical experts, and business stakeholders to provide comprehensive evaluation from multiple perspectives.

Implementation steps

1. Configure AWS Systems Manager Change Calendar with a DEFAULT_CLOSED entry to block changes during regulatory review windows.
2. Develop AWS Systems Manager Automation documents to automate approved changes based on review procedures and findings.
3. Implement AWS Audit Manager assessments for periodic audit verification.

Resources

Related examples:

- Change Management for Life Sciences
- Automated Evidence Collection for Life Sciences continuous compliance solutions using AWS Audit Manager

Related tools:

- AWS Systems Manager Documents
- AWS Audit Manager

LSOPS03-BP01 Perform supplier and vendor assessment of each vendor

Establish criteria for the selection and evaluation of suppliers, and create a plan for the monitoring and re-evaluation of those suppliers. Assess vendor controls while considering the intended use of the services and possible risks involved to the system.

Desired outcome: Vendors are established as approved IT suppliers of purchased services.

Common anti-patterns:

- Treating AWS as a SaaS provider whose solutions usually directly support GxP processes, and therefore incorrectly assessing the risk of using AWS services for to support GxP workloads.
- Asking questions in the supplier questionnaire that are irrelevant considering the services to be used.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use as much supplier documentation as possible to expedite a supplier assessment of AWS.

Implementation steps

1. Perform a general market assessment to establish AWS position in the market and financial stability.
2. Collected documentary evidence of the suitability of the AWS control framework for supporting GxP workload. Establish an AWS account and download required third-party assessment reports and certifications from AWS Artifact.
3. If there are perceived gaps in the information obtained, contact your account team to complete a supplier assessment questionnaire.
4. With the downloaded documentary evidence and questionnaire, perform an analysis and generate a assessment summary with your conclusions. Retain this in case of inspection.

Resources

Related tools:

- AWS Artifact

LSOPS03-BP02 Limit available services to improve regulatory adherence

Use infrastructure tooling to allow only services that fit into required regulatory frameworks.

Desired outcome: Only approved services will be available for use.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Verify components and services used as available to comply with identified frameworks. Check vendor documentation to confirm that the products you use are approved at the vendor level.

Implementation steps

1. Identify the available services by referring to AWS Compliance Programs.
2. Review audit guides for the available services.
3. Setup an AWS Organization to be able to centrally manage policies and controls.
4. Implement service control policies (SCP) limiting access to only the available services.

Resources

Related guides, videos, and documentation:

- AWS Compliance Programs
- What is AWS Organizations?
- Service control policies (SCPs)

Related tools:

- AWS Organizations
- AWS Identity and Access Management

LSOPS03-BP03 Establish clear definitions of responsibilities between you, vendors, and users

A shared responsibility model to clearly delineate responsibilities between involved parties makes it straightforward to work together.

Desired outcome: Have a structure and expectations in a tool to streamline audits.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Select prepared frameworks when applicable to apply industry standards.

Implementation steps

1. Review the AWS Shared Responsibility Model to identify your areas of responsibility.
2. Access AWS Artifact for audit reports and documentation.
3. Deploy AWS Audit Manager framework to create assessment frameworks aligned with specific life sciences regulations like FDA, EMA, and ICH-GCP requirements.

Resources

Related tools:

- AWS Artifact
- AWS Audit Manager

LSOPS04-BP01 Establish IT quality oversight

It is important to distinguish between IT and product quality. A dedicated IT quality management team (separate from pharma quality) should define control objectives and provide quality oversight. They will act as a liaison to product quality and determine which additional IT controls are required and which IT records can be used as evidence of computer systems assurance.

Desired outcome:

- Clear separation of IT quality and product quality functions with defined responsibilities.
- Comprehensive documentation system providing evidence of IT systems adherence.
- Established processes for qualifying and validating IT systems supporting GxP-regulated activities.

Common anti-patterns:

- Forcing GxP process validation and product quality concepts onto IT, negatively impacting process efficiency, instead of using IT industry best practices and quality controls.

Benefits of establishing this best practice:

- Enhanced adherence to GxP regulations through proper IT system verification.
- Greater process efficiency and delivery cadence.

Level of risk exposed if this best practice is not established: High

Implementation guidance

An IT quality management team must operate independently while maintaining strong collaboration with product quality teams. This team develops frameworks for GxP-relevant system classification, establishes validation strategies, and implements risk-based IT system validation approaches. The quality oversight function focuses on system quality, data integrity and traceability while using AWS services for audit monitoring, documentation control, and audit trails. Success depends on balancing regulatory adherence with operational efficiency through risk-based controls and clear communication channels.

Implementation steps

1. Create a team structure with clear reporting lines, assign qualified personnel, and develop the IT quality manual that defines roles, responsibilities, and core quality processes.
2. Create control objectives and SOPs aligned with GxP requirements, establish validation procedures, and implement a risk assessment framework with defined quality metrics.
3. Implement audit monitoring tools, documentation management systems, and audit trail mechanisms while establishing validated testing environments.
4. Conduct comprehensive GxP and IT quality procedure training, establish escalation protocols, and maintain training records demonstrating staff competency.

Resources

Related documents:

- Quality assurance
- Don't Blame Regulators: How Software Excellence Satisfies Compliance
- What is Code Quality?

Related examples:

- The future of quality assurance: Shift-left testing with QyrusAI and Amazon Bedrock
- Beyond compute: Shifting vulnerability detection left with Amazon Inspector code security capabilities
- Developing an Service Catalog self-managed engine for governance

Related tools:

- AWS Config
- AWS Systems Manager
- Amazon Inspector

LSOPS04-BP02 Enforce controls in IT tooling and automation

Implement quality controls and automation in the standard IT tooling used by the development teams whenever possible to enable continuous validation.

Desired outcome:

- Automated enforcement of quality and regulatory controls within development workflows.
- Reduced manual oversight burden while maintaining regulatory adherence.
- Early detection of regulatory issues during the development process.

Common anti-patterns:

- Relying primarily on manual reviews and approvals for verification.
- Inconsistent application of controls across different teams or environments.
- Adding audit checks only at the end of development cycles.

Benefits of establishing this best practice:

- Increased development velocity through automated audit verification.
- Improved quality through consistent application of controls.
- Greater scalability of processes across growing organizations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Identify key control points in your development and operational processes where automated verification can replace manual checks. Implement infrastructure as code templates with pre-validated configurations, automated testing for regulatory requirements, and continuous

monitoring for drift from approved baselines. These automated controls should generate appropriate evidence for audit purposes while remaining transparent to developers.

Consider an automated approach where you translate regulatory requirements into automated tests and policies. This allows teams to validate adherence continuously throughout the development lifecycle rather than as a separate activity. When properly implemented, automated controls can provide greater assurance than manual processes while reducing the compliance burden on teams.

Implementation steps

1. Identify key control points in development and operational workflows:

- Use AWS Glue Data Quality for data pipelines.

2. Integrate automated testing into CI/CD pipelines:

- Use AWS CodePipeline with validation gates.
- Use AWS CodeBuild for running automated tests.

3. Deploy continuous monitoring solutions for configuration drift:

- Use AWS Config Rules to detect non-compliant resources.
- Use Amazon EventBridge for automated remediation of issues.

Resources

Related documents:

- Compliance and Auditing

Related examples:

- Measure performance of AWS Glue Data Quality for ETL pipelines

Related tools:

- AWS CodePipeline
- AWS CodeBuild
- AWS Config
- Amazon EventBridge
- AWS Glue Data Quality

LSOPS04-BP03 Incorporate formal risk management into your IT processes

Risk assessments should be conducted to identify potential vulnerabilities in electronic systems and data management processes. It is a crucial component of effective quality management. It involves proactively identifying, analyzing, and mitigating potential risks that could impact the quality of products or services. By integrating risk management into the quality management process, organizations can improve their overall quality, reduce costs, and enhance customer satisfaction.

Desired outcome:

- Systematic identification and assessment of IT risks across the organization.
- Risk-based decision making for IT investments and control implementation.
- Documented evidence of risk assessment and treatment for regulatory purposes.

Common anti-patterns:

- Using generic risk templates without tailoring to specific life sciences requirements.
- Lacking formal methodology for risk prioritization and acceptance criteria.

Benefits of establishing this best practice:

- Improved business continuity through proactive risk identification.
- Greater stakeholder confidence in IT system reliability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implement a formal risk management framework that aligns with industry standards such as ICH Q9 Quality Risk Management while using cloud capabilities for automation and real-time monitoring. Cloud-based tools can identify risks across your infrastructure, evaluate control effectiveness, and track mitigation activities. Integrate risk assessments into deployment pipelines, and evaluate new systems and changes before implementation. Centralized risk repositories provide visibility across the organization, enabling consistent risk evaluation and prioritization.

When implementing risk management processes, organizations should balance comprehensive risk coverage with operational efficiency. While thorough risk assessment is essential for regulated systems, excessive documentation can create unnecessary overhead. Focus on identifying and addressing meaningful risks that could impact product quality, patient safety, or regulatory adherence. Verify that your risk management approach accommodates both traditional and cloud-based architectures to provide consistent coverage across hybrid environments.

Implementation steps

1. Define a formal risk management methodology aligned with life sciences requirements:

- Use AWS Audit Manager for creating custom risk assessment frameworks.
- Use AWS Security Hub CSPM for centralized visibility of security risks.

2. Establish risk assessment templates and scoring criteria for IT systems:

- Store templates in AWS Systems Manager Documents for consistent application.
- Consider Service Catalog for standardized risk assessment processes.

3. Conduct baseline risk assessments for GxP-relevant systems:

- Use AWS Config for identifying resource configurations that may pose risks.
- Consider Amazon Inspector for automated vulnerability assessments.
- Use AWS IAM Access Analyzer to provide visibility needed to proactively manage permissions.

4. Implement risk mitigation strategies with clear ownership and timelines:

- Use AWS Systems Manager OpsCenter for tracking risk remediation activities.
- Consider AWS Organizations for implementing preventive controls at scale.

5. Integrate risk reviews into change management processes:

- Implement AWS Systems Manager Change Manager with risk assessment gates.
- Consider AWS CodePipeline for automated risk evaluations during deployments.

6. Establish continuous risk monitoring mechanisms:

- Configure Amazon CloudWatch for monitoring risk indicators.
- Consider AWS Security Hub CSPM for aggregating security findings across services.

Resources

Related tools:

- AWS Audit Manager
- AWS Security Hub CSPM
- AWS Systems Manager
- Service Catalog
- AWS Config
- Amazon Inspector
- AWS Systems Manager OpsCenter
- AWS Organizations
- AWS Systems Manager Change Manager
- AWS CodePipeline
- Amazon CloudWatch
- AWS Service Management Connector
- AWS IAM Access Analyzer

Prepare

LSOPS05: How do you demonstrate control over the environment within which GxP workloads operate?

Configuration management (CM) involves the identification, recording, and reporting of components and relationships. Keep items identified as supporting GxP workloads under a state of control, providing an approved baseline for further evolution and allowing safe restoration of a verified baseline in case of problems.

LSOPS06: How do you implement data quality controls to minimize entry errors?

Data quality is paramount for life science projects. Validating data as it is ingested results in quality output and conclusions.

Best practices

- LSOPS05-BP01 Enable a configuration management framework
- LSOPS06-BP01 Validate data quality during ingestion
- LSOPS06-BP02 Implement data pipeline testing

LSOPS05-BP01 Enable a configuration management framework

Use a management framework to record resource configurations, track changes, identify gaps (like defects and issues) and track resolution of changes.

Desired outcome:

- Automated detection of unauthorized or unplanned configuration changes.
- Centralized visibility into resource dependencies and relationships.

Common anti-patterns:

- Lacking version control for infrastructure configurations.

- Allowing ad-hoc changes without proper documentation or approval.

Benefits of establishing this best practice:

- Enhanced compliance-alignment through comprehensive configuration documentation.
- Improved troubleshooting capabilities with complete change history.
- Faster recovery from incidents through established baseline restoration.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Build a configuration management system that automatically discovers and documents resources supporting GxP workloads. Infrastructure as code enables version control and consistent deployment, while continuous monitoring detects unauthorized changes and configuration drift. These capabilities create a foundation for demonstrating the control required for GxP adherence while reducing manual overhead.

When implementing resource management for regulated environments, balance comprehensive tracking with operational efficiency. Focus on capturing information relevant to regulatory requirements, including configuration parameters affecting validated functionality and dependencies between components, while verifying that your framework accommodates both cloud and on-premises resources.

Implementation steps

1. Implement automated discovery and inventory management for IT resources:

- Deploy AWS Config for continuous resource inventory and configuration tracking.
- Use AWS Systems Manager Inventory for detailed software and configuration information.

2. Establish version-controlled repositories for infrastructure configurations:

- Use AWS CloudFormation for templated infrastructure deployment.

3. Define formal change management processes for resource modifications:

- Implement AWS Systems Manager Change Manager for controlled change processes.
- Consider Service Catalog for standardized resource provisioning.

4. Implement continuous monitoring for configuration drift:

- Configure AWS Config Rules to detect non-compliant configurations.
- Consider Amazon EventBridge for automated remediation workflows.

5. Create comprehensive system baselines with restoration capabilities:

- Use AWS Backup for consistent backup and recovery capabilities.
- Consider AWS Systems Manager Automation for standardized restoration procedures.

6. Document resource relationships and dependencies in a centralized system:

- Use AWS Resource Groups for logical organization of related resources.
- Consider AWS Service Management Connector for integration with existing CMDB systems.

Resources

Related examples:

- Synchronize with CMDB

Related tools:

- AWS Config
- AWS Systems Manager Inventory
- AWS CloudFormation
- AWS Systems Manager Change Manager
- Service Catalog
- Amazon EventBridge

- AWS Backup
- AWS Systems Manager Automation
- AWS Resource Groups
- AWS Service Management Connector

LSOPS06-BP01 Validate data quality during ingestion

Define data quality measurements and implement checks during ingestion.

Desired outcome: Data is clean and ready for analysis upon ingestion.

Benefits of establishing this best practice:

- More trusted results.
- Faster output to conclusions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Build data quality checks into data ingestion pipelines.

Plan to isolate erroneous data and have a way to review it as needed.

Implementation steps

1. Identify expected patterns in received data.
2. Build data quality checks in AWS Glue Data Quality.
3. Incorporate checks in data pipelines.
4. Build processes to catch and isolate erroneous data. Alert personnel to investigate data errors

Resources

Related examples:

- Measure performance of AWS Glue Data Quality for ETL pipelines

Related tools:

- AWS Glue
- AWS Glue Data Quality

LSOPS06-BP02 Implement data pipeline testing

Create data handling tests including user authentication and authorization, data collection, and ingestion into later warehouses. Tests should cover quality in addition to technical requirements.

Desired outcome: Data is reliable and ready for immediate business use. Reports and analytics can be trusted without manual verification.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Cloud technologies allow for comprehensive test environments that can replicate entire system architectures and automate complex test scenarios.

Consider implementing a testing strategy that follows data through its complete lifecycle, from initial entry through the processing stages to final storage. Cloud-based environments can replicate your production architecture, allowing validation of the integration points and data transformations. Automated testing can run scenarios that simulate real-world usage patterns, while data validation tools verify information remains consistent as it moves through different systems.

When implementing testing, focus on critical data paths that impact patient safety, product quality, or regulatory adherence. Consider continuous testing approaches that run core validation scenarios automatically when changes occur, supplemented by more comprehensive testing at key milestones.

Implementation steps

1. Map complete data flows across each system component:

- Use AWS X-Ray to trace data paths across distributed systems.
- Consider AWS AWS Glue Data Catalog for documenting data structures and relationships.

2. Create realistic test environments that replicate production architecture:

- Implement AWS CloudFormation templates for consistent environment deployment.
- Consider Service Catalog for standardized test environment provisioning.

3. Develop test scenarios that follow data through complete processing paths:

- Consider AWS Step Functions for orchestrating complex test workflows.

4. Implement data validation checks at key integration points:

- Use AWS Lambda functions for custom validation logic.
- Consider Amazon EventBridge for coordinating validation across services.

5. Automate testing as part of deployment pipelines:

- Implement AWS CodePipeline for continuous testing integration:
- Consider AWS CodeBuild for running test suites:

6. Monitor data quality metrics throughout test runtime:

- Configure Amazon CloudWatch for tracking data quality indicators.
- Consider AWS Glue DataBrew for data quality validation.

7. Generate comprehensive test reports for regulatory documentation:

- Store test evidence in Amazon S3 with appropriate retention policies.
- Consider AWS Systems Manager Automation for standardized report generation.

Resources

Related guides, videos, and documentation:

- How to test serverless functions and applications
- OPS05-BP02 Test and validate changes

Related tools:

- AWS X-Ray
- AWS Step Functions
- AWS CodePipeline
- AWS Lambda
- Amazon EventBridge
- AWS Glue DataBrew
- Amazon CloudWatch
- AWS Systems Manager

Operate

LSOPS07: How do you isolate data and workloads in a auditable manner?

Life sciences projects often require different groups to work together but maintain clear boundaries between resources. Building an auditable environment makes the separation manageable. Data isolation is a valuable tool for GxP systems to maintain data integrity, improve data security, and adhere to regulatory requirements.

LSOPS08: How do you prepare for audits?

Maintaining audit readiness in life sciences projects is crucial because it blocks costly regulatory violations and potential product recalls that could harm both patients and the company's reputation. Strong audit preparation assists you in adhering to regulations, enables faster responses to agency inquiries, and demonstrates the organization's commitment to product quality and safety standards.

LSOPS09: How are you tracking data lineage and performing data change management?

Part of being audit ready is being able to identify the source and destination of project datasets. As data changes over time, being able to track those changes is critical to attain valid and auditable results at the conclusion of a project.

LSOPS10: Are you prepared for the different phases of system and data lifecycle (like creation, collection, analysis, and archival)?

Collect data in a manner that can later be converted or exported for archival. Verify your infrastructure is reproducible in case of re-engagement requirements.

LSOPS11: How do you maintain data governance when collaborating across organizations?

When collaborating with other institutions, be clear about data ownership and management. Keep a clear log of data lineage and history of access. Implement a clean room technology that limits data sharing to only assigned individuals. The technology should have a complete auditing history to see how the data has moved, changed and who has accessed it.

LSOPS12: How do you implement and maintain semantic data management?

A controlled semantic layer gives everyone working on life sciences projects the same vocabulary and way to organize data, making sure everything matches up correctly. This shared language assists teams work together better and makes it easier to combine and analyze different types of research data.

Best practices

- LSOPS07-BP01 Maintain a controlled multi-account environment
- LSOPS07-BP02 Isolate GxP data from non-GxP data

- LSOPS08-BP01 Continuously generate and maintain evidence
- LSOPS09-BP01 Track data ownership and lineage over the life of the project
- LSOPS10-BP01 Reproducibility
- LSOPS10-BP02 Store data in a format that works both for archiving and for active use by retaining related metadata
- LSOPS11-BP01 Identify clear dataset owners and access history
- LSOPS12-BP01 Create a controlled semantic layer

LSOPS07-BP01 Maintain a controlled multi-account environment

Implement standardized account management using templated environments and automated account provisioning. Establish organizational guardrails through policies and baseline configurations. Maintain centralized control over shared services while providing consistent security across accounts.

Desired outcome: Identifiable resource allocation

Common anti-patterns:

- Combining resources into a single account and attempting to separate it with complex rules.

Benefits of establishing this best practice: Improved security and auditability.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement a structured multi-account strategy that separates workloads based on their regulatory requirements, security needs, and operational characteristics. Centralized governance mechanisms can enforce consistent controls across the accounts while allowing appropriate flexibility for different workload types. Standardized account provisioning verifies that new environments are created with baseline controls already in place, while continuous monitoring verifies ongoing adherence to organizational standards.

When implementing a controlled multi-account environment, balance centralized governance with workload-specific requirements. Establish clear guidelines for account structure and responsibility boundaries, but allow teams appropriate autonomy within those guardrails. Implement preventive controls for critical requirements while using detective controls and remediation for less critical

aspects. This approach maintains appropriate control while enabling innovation and operational efficiency.

Implementation steps

1. Design a multi-account structure aligned with regulatory boundaries:

- Implement AWS Organizations for hierarchical account management.
- Consider AWS Control Tower for standardized account governance.

2. Establish centralized identity and access management:

- Configure AWS IAM Identity Center for centralized user management.
- Consider AWS IAM Roles for fine-grained permission controls.

3. Implement standardized account provisioning processes:

- Use AWS Control Tower Account Factory for consistent account creation.
- Consider Service Catalog for standardized environment provisioning.

4. Deploy preventive guardrails for critical requirements:

- Configure AWS Organizations Service Control Policies (SCPs) for preventive controls.
- Consider AWS Control Tower Guardrails for additional preventive measures.

5. Establish baseline configurations across accounts:

- Use AWS CloudFormation StackSets for multi-account resource deployment.
- Consider AWS Systems Manager for configuration management.

6. Implement centralized monitoring and verification:

- Configure AWS Config Aggregators for multi-account visibility.

- Consider AWS Security Hub CSPM for consolidated security and monitoring.

Resources

Related documents:

- Walkthrough: Automate Account Provisioning in AWS Control Tower by Service Catalog APIs

Related examples:

- AWS Samples: AWS Control Tower Automate Account Creation
- Automate account customization using Account Factory Customization in AWS Control Tower

Related tools:

- AWS Organizations
- AWS Control Tower
- AWS IAM Identity Center
- AWS IAM
- Service Catalog
- AWS CloudFormation StackSets
- AWS Systems Manager
- AWS Config
- AWS Security Hub CSPM
- AWS CloudTrail
- Amazon EventBridge
- AWS Resource Access Manager

LSOPS07-BP02 Isolate GxP data from non-GxP data

Take steps to isolate and segment GxP data from non-GxP data. In conjunction with the recommendations around data discovery and classification, separate GxP data so the organization can implement the necessary technical and administrative controls.

Desired outcome: Demonstrable division between GxP and non-GxP data.

Common anti-patterns:

- Granting access at a workload level grants access to the data, GxP and non-GxP.
- Retaining logs that are adjacent to GxP relevant metadata.
- Including GxP data in logs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Incorporate system separation, including table and row-level access controls.

Implementation steps

1. Foster system separation though architecture design and deployment. Create distinct datastores (like Amazon S3 and Amazon RDS) for GxP data.
2. Implement table and row-level access controls through application logic.
3. Apply AWS Lake Formation rules for consistent control to data sets.
4. Produce evidence of verification of access controls.

Resources

Related tools:

- Amazon RDS
- AWS Lake Formation
- Amazon S3

LSOPS08-BP01 Continuously generate and maintain evidence

Use appropriate IT tooling to generate evidence used for regulatory requirements. Maintain audit trails of the changes, approvals, and validations. Store the data in an immutable store. Generate reports as needed from the data. Enable immediate access to evidence during inspections or audits.

Desired outcome: You have a body of evidence ready for auditors.

Benefits of establishing this best practice: Smoother audit process with less wasted time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Start with a clear understanding of what is required, and build audit readiness practices around those requirements.

Implementation steps

1. Define evidence requirements based on regulatory obligations:

- Map evidence needs to specific regulations like 21 CFR Part 11, GxP, or HIPAA.
- Consider AWS Audit Manager for creating custom evidence collection frameworks.

2. Implement automated evidence collection mechanisms:

- Configure AWS Config to capture resource configurations and changes.
- Enable AWS CloudTrail for comprehensive API activity logging.
- Consider Amazon CloudWatch for operational metrics and logs.

3. Establish centralized, secure evidence repositories:

- Use Amazon S3 with appropriate encryption and access controls.
- Consider S3 Object Lock for immutable evidence storage.
- Implement AWS Backup for evidence protection and retention.

4. Create structured evidence organization with metadata:

- Implement Amazon S3 metadata tagging for evidence classification.
- Consider AWS Glue for cataloging and organizing audit artifacts.

5. Generate reports for audits and inspections:

- Use AWS Audit Manager for automated report generation.
- Consider Quick for audit visualization and reporting.

6. Implement appropriate retention policies for documentation:

- Configure S3 Lifecycle policies for evidence retention.
- Consider AWS Backup for long-term archival of documentation.

Resources

Related documents:

- Guidance for Change Management on AWS
- AWS Systems Manager Change Manager
- Logging AWS Account Management API calls using AWS CloudTrail
- How can I secure the files in my Amazon S3 bucket?

Related examples:

- Streamline and automate compliance monitoring and reporting with AWS Backup Audit Manager
- Leveraging AWS CloudTrail Insights for Proactive API Monitoring and Cost Optimization

Related tools:

- AWS Audit Manager
- AWS Config
- AWS CloudTrail
- Amazon CloudWatch
- Amazon S3
- AWS Backup
- AWS Glue
- Quick

- AWS Systems Manager
- AWS Lambda
- Amazon EventBridge

LSOPS09-BP01 Track data ownership and lineage over the life of the project

Data lineage in life sciences projects involves tracking the origin, movement, and transformation of data throughout its lifecycle. This process is important because it provides a clear history of how data has been collected, processed, and used. Maintaining accurate data lineage improves data integrity, aids in troubleshooting errors, and supports regulatory adherence. It also allows researchers and regulators to understand and trust the data's journey from its source to final analysis, which is critical for validating research findings and meeting stringent industry standards.

Desired outcome: Have a clear, auditable history of data lineage.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use data sharing and management tooling that includes audit history of access and changes and allows for granular permissions.

Implementation steps

1. Set up an Amazon DataZone or Amazon SageMaker AI Unified Studio domain and related projects:

- Create a business domain for life sciences data.
- Configure projects and data catalogs.
- Set up user roles and access controls.

2. Enable automatic lineage tracking:

- Use Amazon DataZone or Amazon SageMaker AI Unified Studio lineage capabilities.

- Connect to data sources (like Amazon S3 or Amazon RDS).
- Use built-in metadata collection.

3. Implement governance:

- Use DataZone or SageMaker AI Unified Studio data sharing workflows.
- Configure approval processes.
- Set up automatic data classification.

Resources

Related documents:

- Data lineage in Amazon DataZone
- Best practice 4.5 – Track data and database changes

Related tools:

- Amazon DataZone
- Amazon SageMaker AI Unified Studio

LSOPS10-BP01 Reproducibility

Build technology products with infrastructure as code so you can rebuild them if needed. In archive documents, store the structure of your products. Store data in a format that is simple to archive, such as Iceberg.

Desired outcome: Maintain a history of changes made in the IT environment and methods to programmatically create environments when needed.

Level of risk exposed if this best practice is not established: Medium

Benefits of establishing this best practice: In addition to being ready to close at project conclusion, this practice eases movement from test to production and verifies that testing is performed on the exact architecture that will be deployed. Additionally, it becomes more

straightforward to reproduce the environment in the event that certain components are needed after project conclusion.

Implementation guidance

Establish a robust infrastructure as code framework using industry-standard tools and practices. This foundation fosters consistent, repeatable deployments while maintaining complete version control and documentation.

Implement comprehensive documentation practices that automatically capture infrastructure configurations, dependencies, and changes. This system should integrate with version control and provide clear rebuild instructions.

Establish structured change control procedures that maintain infrastructure stability while enabling necessary updates. This process should include proper documentation, testing, and approval workflows.

Implementation steps

1. Create an infrastructure as code foundation:

- Establish version-controlled repository for infrastructure code and configurations.
- Implement automated CI/CD pipelines for infrastructure deployment using AWS CodeBuild.
- Create standardized templates for common infrastructure components. Consider AWS CloudFormation and StackSets.

2. Develop a documentation framework:

- Develop comprehensive documentation covering the infrastructure components.
- Create automated documentation generation for code and configurations. Consider Amazon Q.

3. Implement archive management:

- Implement versioned archive system for infrastructure code. Amazon S3 Versioning can be used to retain multiple versions of documents.
- Create automated backup procedures for configuration files using AWS Backup. Establish a clear tagging system for archived components.

Resources

Related documents::

- Best Practices for Tagging AWS Resources

Related examples:

- Generating documentation with Amazon Q Developer

Related tools:

- Amazon Q
- Amazon S3 Versioning
- AWS Backup
- AWS CodeBuild
- AWS CloudFormation (with drift detection)

LSOPS10-BP02 Store data in a format that works both for archiving and for active use by retaining related metadata

Select a data format which is queried while the project is ongoing but archives natively. Iceberg's data format is ideal for life sciences projects because it stores metadata alongside the data. It offers advanced data versioning, time travel capabilities, and schema evolution, essential features for maintaining data lineage and regulatory adherence while handling large-scale scientific datasets that frequently change over time.

Desired outcome: Have a portable dataset that contains the data and metadata in a single package

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Verify that the output of data processing results in portable data formats to allow for simple archiving.

Implementation steps

1. Build standard data pipelines using AWS Glue
2. For deep analysis on structured data use Amazon Redshift.
3. Build a data lake of the data in Amazon S3 in Iceberg format.

Resources

Related documentsn:

- What is Apache Iceberg?

Related examples:

- Build a high-performance quant research platform with Apache Iceberg

Related tools:

- Modernize Data Archiving
- Amazon S3 Versioning

LSOPS11-BP01 Identify clear dataset owners and access history

If you are receiving a dataset, identify who is responsible for data accuracy and governance and who manages the data lineage during collection. Depending on the storage method, who is responsible for securing the data in its different stages?

Desired outcome: Have an automated method of auditing shared data between partners.

Benefits of establishing this best practice: Data can be limited to just what is agreed on. Secure data can be automatically filtered.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Design a collaborative governance framework. Establish unified governance architecture spanning organizational boundaries. This foundation enables consistent data policies while respecting each organization's requirements.

Implementation steps

1. Configure collaborative workflows:
   - Implement data exchange protocols specific to organizational relationships.
   - Establish shared metadata repositories with organization-specific views.
   - Deploy cross-organizational audit mechanisms.
2. Use AWS Clean Rooms to enable granular control and data sharing without moving data or exposing sensitive information.

Resources

Related documents:

- AWS Clean Rooms: Privacy-enhanced collaboration use cases

Related examples:

- Enable data collaboration among public health agencies with AWS Clean Rooms – Part 1
- Automate AWS Clean Rooms querying and dashboard publishing using AWS Step Functions and Quick – Part 2

Related tools:

- AWS Clean Rooms

LSOPS12-BP01 Create a controlled semantic layer

A semantic layer translates complex life sciences data into understandable business terms, acting as a bridge between raw data and the scientists who need to use it.

Desired outcome: Have a consistent, well managed semantic layer to allow the data to be consumed by analysts

Benefits of establishing this best practice: Provides consistency of result definition. Allows for more straightforward communication with auditors.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use a tool to centralize and implement semantic layer management.

Implementation steps

1. Create an Amazon DataZone or Amazon Sagemaker Unified Studio domain.
2. Use tools to build and manage the semantic layer. Maintain controlled terminology, business glossaries, and metadata while verifying that you adhere to regulatory requirements.

Resources

Related tools:

- Amazon DataZone
- Amazon SageMaker AI Unified Studio

Evolve

LSOPS13: Are you able to adapt as technology advances (for example, can you implement AI into your analysis process)?

The rapid advancement of AI and other emerging technologies in life sciences has become a crucial differentiator in efficiency, accuracy, and competitive advantage, making technological adaptability essential for long-term success in the field.

Best practices

- LSOPS13-BP01 Store data in an AI/ML-ready formats

LSOPS13-BP01 Store data in an AI/ML-ready formats

Expectations of what can be done with data are rapidly expanding. Life sciences projects produce rich data. Work to maximize data storage to allow for new technologies such as AI/ML.

Desired outcome: Be able to pivot and adapt as there are new technological demands on existing systems and data.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use data processing that stores data in adaptable, open standard formats. This will allow for the greatest flexibility as technologies progress.

Implementation steps

1. Store data in document formats ingestible by Amazon Sagemaker and Amazon Bedrock.
2. Use AWS Glue to transform data into Iceberg and place in an Amazon S3 data lake.

Resources

Related documents:

- Accelerating Life Sciences Innovation with Agentic AI on AWS
- What is Apache Iceberg?
- Building an End-to-End Data Strategy for Analytics and Generative AI: Insights from AWS and Workhuman

Related examples:

- Use generative AI on AWS for efficient clinical document analysis
- Revolutionizing clinical trials with the power of voice and AI

Related tools:

- Amazon Bedrock
- Amazon SageMaker AI

Security

The security pillar provides AWS recommendations that assist to meet your business and regulatory requirements. The practices in this pillar inform architectures that protect GxP workloads and data, control access, and respond automatically to security events.

Life sciences is a heavily regulated industry to protect public health and patient safety by maintaining the quality, safety, and effectiveness of pharmaceuticals, medical devices, and other products that directly impact health.

Life sciences companies have an obligation to maintain data integrity, which begins with strong security measures. Therefore, it is important that a well-architected security and adherence program is used by life sciences entities when they move workloads to the cloud. A well-architected security program helps organizations design and operate workloads while meeting the necessary regulations, aligning with the proper frameworks, and maintaining data integrity.

However, it is important to note that the AWS Well-Architected core covers security best practices which are industry agnostic. Therefore, there are few additional best practices added by this industry lens.

Focus areas

- Design principles
- Identity and access management
- Data privacy

Design principles

The security pillar of the AWS Well-Architected Framework sets out principles that can assist to strengthen the security of your workload:

- **Implement a strong identity foundation:** Implementing the principle of least privilege is foundational to the security of life sciences workloads. Centralize identity management, and aim to avoid reliance on long-term static credentials.
- **Implement the principle of separation of duties:** Avoid conflicts of interest, abuse, errors and detect control failures that include security breaches, information theft, and circumvention of security controls.

- **Be continually inspection-ready:** Monitor, alert, and audit actions and changes to your environment in real time. Integrate log and metric collection with systems to investigate and remediate issues automatically.
- **Apply security at each layer:** Apply a defense in depth approach with multiple security controls. Security should apply to each layer, from the edge of the network to the application and code.
- **Automate security best practices:** Automated software-based security mechanisms improve your ability to scale more securely, rapidly, and cost-effectively.
- **Encrypt data in transit and at rest:** Classify your data to identify health data and other sensitive data. Use encryption, tokenization, and de-identification to decrease the sensitivity of data, and implement access controls.
- **Keep people away from data:** Use mechanisms and tools to reduce the need for direct access or manual processing of health data, consistent with the principle of least privilege.
- **Prepare for security events:** Prepare for an incident by having incident management and investigation policy and processes that align to your organizational requirements and applicable regulatory frameworks.

Identity and access management

LSSEC01: How do you accommodate separation of duties as part of your identity and access management design?

Separation of duties, as it relates to security, has two primary objectives.

The first objective is the avoidance of conflict of interest, abuse, and errors.

The second objective is the detection of control failures that include security breaches, information theft, and circumvention of security controls.

Separation of duties is also essential for demonstrating that data integrity has been maintained. The FDA, for example, clearly states in its guidance that the system administrator role should only be assigned to personnel who are not responsible for the record content. This separation stops an individual whose role has a direct interest in the results of the decision from having the ability to modify or delete critical data. This protects the integrity of the data and avoids the risk of allegations of tampering.

Best practices

- LSSEC01-BP01 Implement the principle of separation of duties
- LSSEC01-BP02 Maintain a history of IAM configurations and changes over time
- LSSEC01-BP03 Set up alerts for IAM configuration changes and perform audits

LSSEC01-BP01 Implement the principle of separation of duties

For users with increased privileges, it is important to distribute system administration activities, so no one administrator can hide their activities or control an entire system. Separation of duties can mitigate risk on critical tasks by requiring separate requesters and approvers for a task. A common example is the use of an approver during the running of an automation on AWS Systems Manager. This principle can be used to implement numerous tasks including controlling access to your cloud resources. Use roles with limited permissions based on functional needs when increased privileges are not required.

Desired outcome: Administrative tasks related to GxP data stores require approvals.

Common anti-patterns:

- Failure to implement a least privilege administration model.
- Lack of approvals in administration workflows.

Benefits of establishing this best practice: By implementing a least-privilege model and separating duties, appropriate control over access to GxP related resources can be demonstrated.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Determine which administrative tasks may have the potential to impact GxP data integrity. For each of these tasks, separate task approval from task execution.

Configure roles with least privilege to be used for routine functions that do not require administrative permissions.

Implementation steps

1. Introduce approval steps into automated administrative workflows.

2. Set required IAM permissions as needed.

Resources

Related best practices:

- SEC03-BP02 Grant least privilege access

Related documents:

- Run an automation that requires approvals

LSSEC01-BP02 Maintain a history of IAM configurations and changes over time

By logging the IAM policy that was assigned to an IAM user, group, or role, you can determine the permissions that belonged to a user at a specific time. For example, you can view whether a user had permission to modify settings on a specific date in the past.

Desired outcome: A complete history of IAM configurations is maintained and available for review.

Benefits of establishing this best practice: Provide the ability to view the IAM policy that was assigned to an IAM user, group, or role over time.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Customize AWS Config to record configuration changes to IAM global resources in your home Region.

Implementation steps

1. Determine a home AWS Region where you want AWS Config to record and store configuration changes to IAM resources, as the same IAM data is available in different AWS Regions.
2. In your home region, enable recording by AWS Config and enable recording of global resources or select specific IAM resources.
3. Create a service control policy (SCP) to stop recording being turned off.

Resources

Related documents:

- How to Record and Govern Your IAM Resource Configurations Using AWS Config

LSSEC01-BP03 Set up alerts for IAM configuration changes and perform audits

Compliance-related access rules should be automated with alerting or automated risk mitigation actions.

Desired outcome: Ability to mitigate the risk of irregular access configurations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set up alerts for monitoring activities by users with increased privileges.

Perform periodic audits of control effectiveness.

Implementation steps

1. Set up alerts to notify on AWS IAM configuration changes including when an IAM user is created or when conflicting permissions are added to a user or role, such as being able to approve its own requests on a given workflow.
   a. The added notification can be set up using a combination of AWS CloudTrail, Amazon CloudWatch, and Amazon SNS.
2. Automate permissions management and refinement through IAM Access Analyzer with security integration workflows that alert teams to access policy changes. For unused roles, access keys, or passwords, IAM Access Analyzer provides quick links in the console to assist you to delete them. For unused permissions, IAM Access Analyzer reviews your existing policies and recommends a refined policy that is tailored to your access activity.

Data privacy

LSSEC02: How do you determine and enforce data privacy requirements?

Data privacy requirements vary across the globe. For example, in the US, the handling of protected health information (PHI) is governed by HIPAA. In the EU, GDPR, Clinical Trial Regulations, and others may apply.

Best practices

- LSSEC02-BP01 Determine applicable regulatory frameworks and enforce data privacy requirements by implementing controls

LSSEC02-BP01 Determine applicable regulatory frameworks and enforce data privacy requirements by implementing controls

Many life sciences organizations fall under data privacy requirements or regulations which influences data security and architecture, for example where data may be physically located.

Desired outcome: Control objectives identify the locations and conditions where specific data is required to be stored and controls are implemented.

Common anti-patterns:

- Exporting data from shared data sets.
- Manually obfuscating sensitive fields.

Benefits of establishing this best practice: Clarifies requirements as well as control automation and audit.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Begin by reviewing data privacy requirements within applicable regulatory frameworks. To determine applicable regulatory frameworks, start with local regulations and frameworks for the country where your sensitive data is generated, hosted, and processed.

Engage with legal counsel who can assist you to define the scope of the local regulations, as well as additional regulation frameworks that may apply to you.

Update documentation of data residency requirements and control objectives with specific details on which data elements are subject to allowed or disallowed storage and transmission locations. For more detail, see SEC01-BP03 Identify and validate control objectives.

Once the determination of requirements has been made and documented, technical controls can be put in place to enforce them. Choose which geographic regions to include in your environment.

Control objectives should be updated to clearly indicate where data is expected to be located due to data residency requirements. Implement controls to keep data within those Regions.

Implementation steps

1. Update control objectives to address data residency regulatory requirements.
2. Separate workloads that have different data residency requirements.
3. Implement controls that enhance your digital sovereignty governance posture.
4. Tag PHI data as sensitive and grant least privilege access only where required.
5. Restrict access by location of resource.
6. Implement detective controls that notify security operations when resources are found in unauthorized locations.
7. Implement backups to enable recovery from data corruption and data deletion.
8. Update threat models to cover the accidental or malicious storage of data in unauthorized locations.

Resources

Related documents:

- Data Residency with Hybrid Cloud Services Lens - AWS Well-Architected
- Data protection in Amazon DataZone

Related tools:

- AWS DataZone
- Amazon SageMaker AI Unified Studio

Reliability

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle.

Focus areas

- Design principles
- Foundations
- Workload architecture
- Change nanagement
- Failure management

Design principles

- **Preserve data integrity throughout the entire lifecycle:** Implement checksums, validation mechanisms, and lineage tracking at every stage where data is ingested, transferred, processed, or stored. Design idempotent and reproducible pipelines that maintain scientific validity and regulatory adherence even under failure conditions.
- **Build reliability into architecture from the start:** Design for Multi-AZ redundancy, fault isolation, and graceful degradation rather than relying solely on reactive recovery. Incorporate redundancy and failover mechanisms as foundational architectural elements, not afterthoughts.
- **Implement proactive monitoring and predictive maintenance:** Detect reliability issues before they affect operations through comprehensive monitoring of system health, data integrity, and equipment telemetry.
- **Design for regulatory adherence and auditability:** Map regulatory requirements (like FDA, EMA, and ICH) to reliability controls and maintain auditable evidence throughout the workload lifecycle.
- **Enable fault isolation with checkpointing and recovery:** Break complex scientific workflows into modular, independently testable steps with well-defined checkpoints. Preserve intermediate progress so that transient failures don't force complete reruns, reducing cost and maintaining research momentum.
- **Plan for long-term data preservation and recovery:** Implement tiered storage strategies with immutability controls and efficient recovery mechanisms that meet retention requirements

spanning decades. Verify that archived data remains retrievable, verifiable, and usable for reproducibility and regulatory audits.

- **Test reliability under realistic failure scenarios:** Validate both technical resilience and regulatory adherence through comprehensive testing that includes chaos engineering, data integrity verification under failure conditions, and validation of regulated workloads.

Foundations

LSREL01: How do you reliably anonymize, pseudo-anonymize, and re-identify sensitive life sciences data in a compliance-aligned manner?

Life sciences workloads often process sensitive and regulated datasets, such as genomic, biomarker, or clinical trial information. Data anonymization must balance privacy protection with scientific utility and reproducibility. Architectures should incorporate anonymization, masking, and encryption strategies that protect sensitive identifiers while preserving the ability to re-identify data when authorized. Controls must apply anonymization consistently, validate recoverability, and make audit evidence available for compliance-related purposes.

LSREL02: How do you maintain continuity of scientific workflows and data availability during lab system downtime?

Foundational research systems such as Laboratory Information Management Systems (LIMS), Electronic Lab Notebooks (ELN), Electronic Health Records (EHR) systems and scientific data repositories are critical to R&D productivity and data integrity in the life sciences domain. Downtime due to planned maintenance or unplanned outages can disrupt experiments, delay time-sensitive milestones, and compromise reproducibility. Resilient architectures that maintain high availability and robust data accessibility allow scientific workflows to continue while reducing the risk of lost progress or regulatory issues.

LSREL03: How do you provide resilient long-term storage and recovery of research and clinical data?

Life sciences organizations must verify that clinical trial and research data is securely retained and retrievable for extended periods. Retention requirements stem from regulatory, scientific, and business drivers, including reproducibility of research, regulatory audits, and the potential need to revisit datasets long after studies conclude. Reliable long-term storage strategies must address durability, immutability, data integrity, and efficient recovery at scale.

LSREL04: How do you verify your workload meets regulatory requirements for system reliability?

Life sciences workloads often operate under strict regulatory frameworks (FDA, EMA, ICH) that mandate specific reliability requirements. These may include system availability targets, data recovery capabilities, and business continuity provisions. Your reliability strategy should incorporate these regulatory requirements as foundational elements, with documented evidence of regulatory adherence.

Best practices

- LSREL01-BP01 Identify and protect sensitive data elements with auditable classification.
- LSREL01-BP02 Decouple anonymization logic from core workflows using orchestration and versioning.
- LSREL02-BP01 Build resilient and highly available research solutions
- LSREL02-BP02 Maintain continuous data availability and integrity
- LSREL03-BP01 Digitize and modernize archival of research data
- LSREL03-BP02 Implement tiered storage and recovery strategies
- LSREL04-BP01 Map regulatory requirements to reliability controls
- LSREL04-BP02 Implement risk-based reliability testing for regulated systems
- LSREL04-BP03 Establish reliability qualification procedures

LSREL01-BP01 Identify and protect sensitive data elements with auditable classification.

Conduct a systematic analysis to identify sensitive data elements (like PHI, identifiers, and lab results), establish classification standards, and map business processes that generate or consume

them. Define which data should be anonymized, irreversibly de-identified, or remain re-identifiable to support authorized patient re-linking after research.

Desired outcome: The complete dataset will be accessible to a broader group of individuals and organizations. However, access to individual data elements will be restricted to only those with the necessary permissions.

Common anti-patterns:

- Encrypting each data element.
- No or incorrect data classification.
- Incorrect classification of de-identify data element.

Benefits of establishing this best practice:

- Enables broader data sharing and secondary analysis while preserving trust, meeting regulatory requirements.
- Fosters a culture of trust and appropriate data handling.
- Verifies that anonymization procedures correspond effectively with legal requirements and organizational goals through collaborative effort among technical, legal, and governance teams.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Consider Amazon SageMaker AI Unified Studio and AWS Lake Formation for a unified view of data sources and consumers. This combination of services assists you to centrally govern, secure, and globally share data for analytics and machine learning. With AWS Lake Formation, you can enforce fine-grained access control for your data at row, column, and cell level. This centralized access control mechanism reduces the risk of configuration errors and consistently enforces policies across your data access patterns, improving the reliability of your data governance.

Use AWS Key Management Service (KMS) to securely manage and audit encryption keys during the lifecycle of life science data like genomics or bioscience research data. Implement automatic key rotation and use AWS CloudTrail integration for full traceability and recovery assurance. Forward CloudTrail logs to a dedicated account for log integrity and to avoid tampering, supporting reliable audit trails for regulatory verification. This provides a resilient and highly available key

management system that consistently encrypts and decrypts data even during Regional failover or transient service issues.

Store classification manifests (including dataset UUID, sensitive field mappings, classification levels, and SHA-256 digests) in Amazon S3 with S3 Object Lock enabled. This makes your classification records immutable, stopping accidental or malicious deletion and providing a reliable audit trail for compliance-related and recovery scenarios. Configure appropriate retention periods based on regulatory requirements.

Implementation steps

1. Identify and inventory the data sources that generate or capture sensitive data elements, and use AWS AWS Glue Data Catalog as a centralized metadata repository. Use Amazon Macie to automatically discover and classify sensitive data elements, reducing manual classification errors and improving consistency across datasets. Add custom attributes in the Data Catalog to indicate if a data element requires de-identification, irreversible anonymization, or encryption. Automate this discovery and classification process to maintain accuracy as data elements are added, updated, or removed across systems.
2. Make decision which data elements to secure and protect in system of record. Consider using AWS KMS for creating and controlling encryption keys to protect data across systems. Consider using AWS CloudTrail for audit trail what keys are used by who, when, and on which data elements.
3. Consider providing a scalable decryption function or API to reverse data based on role and permissions at data-domain, row, column, or cell level.

Resources

Related best practices:

- Encrypting data at rest and in transit.
- Copies of data with or without sensitive data based on need.
- Removing sensitive data for downstream systems and relying on source systems to make sure data is reverse traceable with primary keys.
- Adding features to source systems to avoid data egress even within the organization.

Related documents:

- Guidance for Data Anonymization on AWS

Related tools:

- Amazon Macie
- Amazon SageMaker AI Unified Studio
- AWS Lake Formation
- AWS Key Management Service
- AWS CloudTrail

LSREL01-BP02 Decouple anonymization logic from core workflows using orchestration and versioning.

Implement data anonymization as a modular, orchestrated workflow with support for AI-based or rule-based anonymization. Store intermediate data in secure, versioned storage to enable rollback, reproducibility, and auditability, verifying that sensitive data handling aligns with life sciences regulatory and research requirements.

Desired outcome: A modular system architecture where anonymization processes operate independently of core application logic, allowing for better scalability, simple maintenance, and improved reliability. The system maintains data lineage and provides rollback capabilities for reproducibility for scientific and regulatory purposes.

Benefits of establishing this best practice:

- Improves system resilience by isolating failures in the anonymization process.
- Enables independent scaling of anonymization resources based on workload.
- Provides audit trail and reproducibility for regulatory adherence.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Decouple anonymization logic from core application workflows by implementing it as an independent, orchestrated process. Use AWS Step Functions to define anonymization workflows as state machines with distinct stages for data ingestion, validation, transformation (rule-based or

AI-based), quality verification, and output generation. This separation allows the anonymization process to scale independently, fail gracefully without impacting core systems, and be updated or rolled back without touching production application code.

For rule-based anonymization (like deterministic masking, tokenization, and generalization), implement transformation logic in AWS Lambda functions that can be versioned and tested independently. For AI-based anonymization (like context-aware redaction and synthetic data generation), deploy models as Amazon SageMaker AI endpoints that can be updated independently of the orchestration layer. Use Amazon EventBridge to trigger workflows based on data arrival events or scheduled intervals, maintaining loose coupling between data producers and anonymization consumers.

Store intermediate transformation manifests in Amazon S3 with versioning enabled to create an immutable audit trail of anonymization operations. Each manifest should capture the complete context of a transformation: input data digest (SHA-256), transformation version identifier, operator ID, timestamp, anonymization parameters, and output data location. Enable S3 Object Lock in compliance mode with retention periods aligned to regulatory requirements. This manifest-based approach enables reproducibility allowing you to re-run historical anonymization with the exact same logic and parameters—and supports rollback by maintaining references to both original and transformed data states.

Implementation steps

1. Create Step Functions state machines for anonymization workflows with Lambda functions for rule-based logic or SageMaker AI endpoints for AI-based models. Configure EventBridge rules to trigger workflows on data arrival or schedule.
2. Store transformation manifests in S3 with versioning and Object Lock enabled. Include input digest, transform version, operator ID, timestamp, and parameters in each manifest.
3. Version anonymization logic in Git repositories and store model artifacts in S3 with versioning enabled. Tag Lambda function versions and SageMaker AI model artifacts with semantic versioning and link to transformation manifests for reproducibility.
4. Configure CloudWatch Logs, CloudTrail, and X-Ray for workflow monitoring. Set up alarms for failures, duration anomalies, and data quality metrics.

Resources

Related best practices:

- Monitoring and observability for data transformation workflows
- Data lineage and provenance tracking for regulatory adherence
- Automated testing and validation of anonymization quality

Related guides, videos, and documentation:

- Guidance for Data Anonymization on AWS
- AWS Step Functions Developer Guide

Related tools:

- AWS Step Functions
- AWS Lambda
- Amazon SageMaker AI
- Amazon S3
- Amazon EventBridge
- Amazon CloudWatch
- AWS CloudTrail

LSREL02-BP01 Build resilient and highly available research solutions

Design research systems using fault-isolated, redundant deployment patterns that allow continuous access during maintenance or localized outages. Use highly available configurations where near-zero downtime is required (for example, ELNs) and active-passive designs with automated failover for systems with less stringent latency or availability needs (for example, LIMS).

Desired outcome:

- Research systems remain available during maintenance or outages.
- Ongoing experiments and data collection continue without interruption.
- Researchers have a consistent experience across sites and geographies.

Common anti-patterns:

- Relying on single-instance deployments for critical systems.

- Maintenance performed without phased rollouts or rollback plans.
- Highly available architectures implemented without routine failover testing.

Benefits of establishing this best practice:

- Protects ongoing experiments from interruption, reducing risk of wasted samples or time.
- Preserves continuity in multi-day or multi-week lab workflows.
- Provides global research teams access to shared tools without productivity loss.
- Enhances regulatory readiness by reducing gaps in system availability records.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When designing research workloads for resiliency, use deployment topologies that minimize the scope of impact of maintenance and outage events. Multi-site or multi–AZ deployments allow for uninterrupted operation when a system component is taken offline. Automated health checks and intelligent traffic routing directs users to healthy endpoints, maintaining continuity during upgrades or failover scenarios. Maintenance windows should be orchestrated to minimize user disruption, and resilience should be validated regularly through simulated failovers or planned game days.

Implementation steps

1. Deploy AWS IoT Greengrass for local buffering of instrument data.
2. Deploy LIMS, EHR and ELN solutions across multiple Availability Zones using Amazon EC2 Auto Scaling for workload redundancy.
3. Place applications behind an Elastic Load Balancer (ALB or NLB) to support highly available deployment configurations, or configure Amazon Route 53 DNS failover with health checks for active-passive failover strategies.
4. Use AWS Systems Manager to coordinate upgrades with minimal disruption. Monitor resilience, health status, and failover events in Amazon CloudWatch, and validate readiness with regular failover exercises.

Resources

Related best practices:

- Change management and controlled rollout strategies
- Incident response playbooks for scientific research systems
- Risk-based classification of R&D applications

LSREL02-BP02 Maintain continuous data availability and integrity

Implement real-time or near-real-time data replication strategies across Availability Zones or AWS Regions to protect against system failures or maintenance interruptions. Use warm standby environments with synchronized datasets to enable quick failover and avoid data loss or research disruption.

Desired outcome:

- Continuous data access during maintenance and outages.
- Minimal risk of data loss or corruption across failure events.
- Trust in the accuracy, completeness, and reproducibility of research datasets.

Common anti-patterns:

- Relying on manual backups without automated validation or recovery testing.
- Replicating data without maintaining consistency or integrity verification.
- Treating replication as optional for intermediate or temporary data sources unless cost/time-effective to reproduce.

Benefits of establishing this best practice:

- Enables uninterrupted access to experimental results and research datasets.
- Reduces risk of losing critical data from unique or costly experiments.
- Supports reproducibility and regulatory adherence (like audit trails and traceability).
- Strengthens collaboration by making shared datasets available globally.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Data availability must be preserved even when system components undergo maintenance or fail unexpectedly. Structured research data, such as LIMS transactions, should use multi-zone replication to maintain availability, while unstructured research datasets should be replicated across independent storage domains. Monitoring replication lag and validating dataset consistency are critical to avoid silent data corruption. Automated recovery validation and regular restore drills build trust that data can be recovered accurately and within defined RPO and RTO objectives.

Implementation steps

1. Configure Amazon RDS Multi-AZ deployments for LIMS databases to achieve transactional durability.
2. Use Amazon S3 Cross-Region Replication (CRR) for uninterrupted access to critical datasets, and protect large-scale file-based research workloads with Amazon FSx for Lustre combined with snapshot policies managed by AWS Backup.
3. Define automated recovery validation policies in AWS Backup for audit tracking.
4. Continuously monitor replication lag and recovery objectives through Amazon CloudWatch, aligning recovery metrics with the needs of research workflows.

Resources

Related best practices:

- Data lifecycle management for research datasets
- Data integrity and reproducibility controls
- Governance documentation for GxP-relevant systems

LSREL03-BP01 Digitize and modernize archival of research data

Replace legacy storage formats such as tapes, optical disks, or local hard drives with digital storage strategies that provide redundancy, immutability, and verifiable integrity. Centralized digital archives keep data accessible, protected from degradation, and adhere to long-term retention requirements.

Desired outcome:

- Research and clinical data remains intact and accessible for extended periods.
- Immutable archives block tampering or accidental deletion.
- Metadata supports efficient search and retrieval for audits or scientific review.

Common anti-patterns:

- Relying on physical tapes or local servers with no periodic validation.
- Storing critical data without metadata, making retrieval inefficient.
- Keeping data in non-standard formats that hinder long-term usability.

Benefits of establishing this best practice:

- Improved durability and accessibility compared to physical archives.
- Reduced operational costs and manual overhead of physical storage.
- Simplified adherence to regulatory audits through traceable, verifiable archives.
- Supports reproducibility by keeping datasets intact and usable over decades.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Long-term storage strategies should prioritize immutability and regulatory alignment. Establish centralized digital archives with policies that enforce retention and stop premature deletion. Periodically validate archived datasets with integrity checks and maintain metadata to verify that datasets are searchable and contextualized. Migrating legacy media into durable repositories reduces the risk of data degradation and provides consistent access across global research teams.

Implementation steps

1. Digitize and migrate legacy media into Amazon S3 with Object Lock to enforce immutability.
2. Use Amazon Glacier Vault Lock to apply retention rules.
3. Migrate large historical datasets efficiently with AWS DataSync.
4. Schedule automated integrity checks with Amazon S3 Inventory and log validation results into AWS Audit Manager for audit tracking.
5. Maintain searchable metadata indexes using Amazon DynamoDB or Amazon OpenSearch Service to enable rapid dataset retrieval during audits or scientific reviews.

Resources

Related best practices:

- Data integrity validation for regulatory adherence
- Metadata management and data cataloging
- Lifecycle management policies for long-term research data

LSREL03-BP02 Implement tiered storage and recovery strategies

Adopt a tiered approach to storage and retrieval, balancing cost, durability, and recovery speed. Critical research datasets should be stored in durable, retrievable formats, while less frequently accessed archives can be stored in lower-cost, long-term archival tiers.

Desired outcome:

- Cost-effective storage of large volumes of research data.
- Ability to retrieve datasets reliably within regulatory or operational timelines.
- Long-term durability and reproducibility for clinical trial and research data.

Common anti-patterns:

- Treating each dataset equally, leading to unnecessary costs or slow recovery.
- Archiving data without defining retrieval timelines or compliance-related needs.
- Failing to test retrieval procedures, risking failed restores during audits.

Benefits of establishing this best practice:

- Reduces storage costs without sacrificing durability.
- Provides scalable recovery aligned with business and compliance-related needs.
- Improved preparation for regulatory audits or re-analysis of research and clinical data.
- Supports sustainable scaling of data management as volumes grow exponentially.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Classify datasets by criticality, regulatory requirements, and access frequency. Assign storage tiers accordingly, with active datasets kept on performant storage and archival datasets transitioned to lower-cost tiers. Lifecycle automation reduces human error and improves adherence to retention schedules. Recovery workflows must be tested periodically to confirm that timelines can be met during regulatory audits or scientific reviews.

Implementation steps

1. Store frequently accessed datasets in Amazon S3 Standard, then transition infrequently accessed data to Amazon Glacier Flexible Retrieval or enable Amazon S3 Intelligent-Tiering.
2. Archive rarely accessed datasets with Amazon Glacier Deep Archive for the lowest-cost, long-term retention.
3. Use S3 Lifecycle Policies to automate transitions across tiers.
4. Implement recovery workflows using AWS Step Functions to orchestrate retrieval steps, and align with regulatory timelines and audit needs.

Resources

Related best practices:

- Business continuity planning for research data
- Backup validation and recovery testing
- Risk-based classification of clinical and research datasets

LSREL04-BP01 Map regulatory requirements to reliability controls

Create a comprehensive mapping between applicable regulatory requirements (like GxP, 21 CFR Part 11, and GDPR) and your reliability controls. Document how each control satisfies specific regulatory requirements and maintain this mapping as regulations evolve. For example, if regulations require system availability of 99.9% for critical applications, implement and document the architecture decisions, monitoring systems, and recovery procedures that support this requirement.

Desired outcome: A clear, documented traceability matrix between regulatory requirements and implemented reliability controls that demonstrates adherence and guides architecture decisions.

This mapping serves as evidence during audits and inspections while verifying that reliability measures directly address regulatory obligations.

Common anti-patterns:

- Implementing reliability controls without considering regulatory requirements.
- Failing to document how reliability controls satisfy specific regulations.
- Not updating the mapping when regulations or system architecture changes.
- Focusing only on technical controls without considering procedural and documentation requirements.

Benefits of establishing this best practice:

- Provides clear evidence of regulatory adherence for audits and inspections.
- Aligns reliability investments with regulatory priorities.
- Facilitates impact assessment when regulations change.
- Supports risk-based decision making for reliability investments.

Implementation guidance

Use AWS Config to track configuration changes that might affect adherence to regulatory requirements.

Consider implementing AWS Audit Manager to continuously audit your AWS usage to simplify risk assessment and adherence with regulations.

Implement tagging strategies to identify resources subject to specific regulatory requirements.

Use AWS Systems Manager documents to standardize and automate checks.

Implementation steps

1. Identify the applicable regulatory requirements related to system reliability (like FDA, EMA, and ICH).
2. Create a traceability matrix documenting each requirement and corresponding control.
3. Use AWS Config to establish configuration rules that enforce regulatory requirements.

4. Implement AWS CloudWatch alarms to monitor adherence to availability requirements.
5. Document architecture decisions that support regulatory requirements using AWS Well-Architected Tool.
6. Establish a review process to update the mapping when regulations or systems change.

Resources

Related best practices:

- Implement comprehensive monitoring for regulated systems
- Establish reliability qualification procedures
- Design for fault isolation and graceful degradation

LSREL04-BP02 Implement risk-based reliability testing for regulated systems

Develop a risk-based approach to reliability testing that prioritizes critical system components based on patient safety, data integrity, and regulatory impact. For high-risk components in GxP systems, implement more rigorous testing including fault injection, recovery testing, and performance under stress. Document test results as part of your evidence package.

Desired outcome: A comprehensive testing program that verifies reliability requirements are met, with testing depth proportional to risk. The testing approach provides documented evidence that systems can maintain reliability under various conditions, supporting both regulatory adherence and operational confidence.

Common anti-patterns:

- Applying the same level of testing to each component regardless of risk.
- Focusing only on functional testing while neglecting reliability aspects.
- Not documenting test procedures and results for regulatory evidence.
- Testing only in development environments without validating production configurations.

Benefits of establishing this best practice:

- Focuses testing resources on components with the highest risk assessment.

- Provides documented evidence of reliability for regulatory submissions.
- Identifies potential failures before they impact operations.
- Builds confidence in system resilience under adverse conditions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use AWS Fault Injection Service to safely test system resilience through controlled experiments.

Consider implementing AWS Resilience Hub to assess and improve application resilience.

Implement chaos engineering principles using AWS Fault Injection Service for GxP-critical systems.

Use AWS CloudWatch Synthetics to create canaries that continuously verify critical paths.

Implementation steps

1. Perform a risk assessment to categorize system components by regulatory impact.
2. Define testing protocols with depth and frequency based on risk categories.
3. Implement automated testing using Amazon CloudWatch Synthetics for continuous verification.
4. Use AWS Fault Injection Service to test recovery mechanisms for high-risk components.
5. Document test procedures, results, and remediation actions in a format suitable for regulatory review.
6. Establish a regular cadence for reviewing and updating testing protocols.

LSREL04-BP03 Establish reliability qualification procedures

Create formal verification procedures for reliability aspects of your system that align with regulatory expectations. Include installation qualification (IQ), operational qualification (OQ), and performance qualification (PQ) elements that verify reliability features like high availability, disaster recovery, and data backup and restore capabilities. Maintain these qualification records as part of your regulatory documentation.

Desired outcome: A formal, documented qualification process that verifies reliability features work as intended and meet regulatory requirements. The qualification procedures provide evidence that

the system can reliably perform its intended functions under expected conditions and recover from failures, satisfying both technical reliability needs and regulatory requirements.

Common anti-patterns:

- Focusing qualification only on functional aspects while neglecting reliability features.
- Not including disaster recovery and failover testing in qualification procedures.
- Qualifying systems only at initial deployment without re-qualification after significant changes.
- Documenting only successful test results without capturing and addressing failures.
- Separating reliability qualification from the overall validation process.
- Using manual, non-repeatable qualification procedures that can't be consistently executed.

Benefits of establishing this best practice:

- Provides documented evidence of reliability capabilities for regulatory adherence.
- Thoroughly verifies reliability features before production use.
- Establishes baseline performance for monitoring and continuous improvement.
- Reduces risk of reliability-related findings during inspections.
- Builds confidence in the system's ability to maintain data integrity during failures.
- Creates a foundation for continuous adherence as systems evolve.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use AWS Systems Manager documents to create standardized qualification procedures.

Consider AWS Backup for testing and validating backup and recovery procedures.

Implement infrastructure as code using AWS CloudFormation for consistent, repeatable infrastructure deployment.

Use AWS Config to verify that production environments match qualified configurations.

Consider implementing AWS Resilience Hub to assess application resilience against defined resilience policies.

Use AWS Fault Injection Service to validate recovery procedures in a controlled manner.

Implementation steps

1. Define qualification protocols for reliability features (like HA, DR, or backup and restore) based on risk assessment.
2. Create IQ procedures to verify infrastructure components are properly installed and configured:

- Validate AWS service configurations match design specifications.
- Verify networking components, security controls, and monitoring systems.

3. Develop OQ procedures to test reliability features under normal conditions:

- Test automatic failover between availability zones.
- Verify data replication mechanisms function correctly.
- Validate backup processes complete successfully.

4. Establish PQ procedures to verify performance under expected load and stress conditions:

- Test system recovery after simulated failures.
- Verify data integrity is maintained during recovery operations.
- Validate system meets defined recovery time and point objectives.

5. Document qualification results with evidence of successful testing.
6. Implement change control procedures to maintain qualified state and determine when re-qualification is required.

Workload architecture

LSREL05: How do you design for resilience when network connectivity between on-premises lab equipment and cloud resources is disrupted?

Life Sciences workloads often depend on continuous data collection from on-premises laboratory equipment and clinical devices. Network issues can result in data loss, invalidate experiments, and compromise scientific integrity. Designing for resilience requires edge-based strategies to buffer data, maintain operations during outages, and verify data completeness when connectivity is restored.

LSREL06: How do you design workflows for fault isolation and graceful degradation to avoid full reruns?

Life sciences organizations commonly run complex workflows such as genomics pipelines, image analysis, and biomarker discovery, which may span hours or days. These workflows consist of multiple chained tasks using orchestration engines. If one task fails, the entire workflow should not be forced to restart unless scientifically necessary. Designing for fault isolation, checkpointing, and graceful degradation contains failures, preserves intermediate progress, and verifies that your research studies remain reproducible and efficient.

LSREL07: How do you design life sciences workloads to preserve data integrity across the entire data lifecycle?

Maintaining data integrity in life sciences workloads requires a holistic approach spanning ingestion, processing, transfers, transformations, and storage. Workloads must verify data fidelity at each stage, detect corruption or anomalies early, and demonstrate that data has not been altered inappropriately. Holistic data integrity also provides confidence in reproducibility, regulatory adherence, and scientific validity, verifying that derived insights are trustworthy.

LSREL08: How do you architect life sciences workloads for high availability while preserving controls?

Designing reliable life sciences workloads requires proactive planning and architectural choices that balance availability and regulatory adherence. Unlike reactive recovery or operational continuity, this involves building availability into the foundation of the workload. Architecture decisions should demonstrate that redundancy, failover, and resilience mechanisms have been designed,

validated, and documented up front, so that regulatory adherence and scientific integrity are preserved even under failure conditions.

Best practices

- LSREL05-BP01 Design edge buffering and queuing for laboratory instruments during network disruptions
- LSREL06-BP01 Orchestrate workflows with checkpointing and failure isolation
- LSREL07-BP01 Implement system-wide data checksums and transfer validation
- LSREL07-BP02 Build idempotent and reproducible processing pipelines
- LSREL07-BP03 Use staged validation and data quarantine mechanisms
- LSREL07-BP04 Track data lineage with lifecycle metadata
- LSREL08-BP01 Incorporate validated redundancy into architecture design
- LSREL08-BP02 Design compliance-aware failover workflows
- LSREL08-BP03 Align architecture priorities with scientific and regulatory context

LSREL05-BP01 Design edge buffering and queuing for laboratory instruments during network disruptions

Implement local data buffering, intelligent queuing, and automatic retry mechanisms at the edge to verify that data from laboratory instruments and clinical devices is never lost during network connectivity issues. Edge infrastructure should provide sufficient storage capacity, prioritize critical data streams, and automatically resume transmission with integrity verification when connectivity is restored.

Desired outcome:

- Avoid gaps in data collection from laboratory instruments or clinical devices in the event of network disruption.
- Data integrity is verified before and after transmission to detect corruption.
- Experiments and clinical workflows remain valid despite temporary connectivity issues.

Common anti-patterns:

- Relying solely on instrument internal storage without additional edge buffering capacity.

- Missing integrity verification, allowing corrupted data to propagate to cloud storage.
- Lack of monitoring for buffer utilization, leading to unexpected data loss when capacity is exceeded.
- Manual intervention required to resume data transmission after network recovery.

Benefits of establishing this best practice:

- Avoids invalidation of long-running experiments due to transient network issues.
- Reduces risk of losing irreplaceable clinical or research data from one-time procedures.
- Maintains scientific integrity by capturing complete and accurate data.
- Provides operators with visibility and control during network disruptions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Design edge buffering based on instrument data generation rates, expected network outage durations, and data criticality. Select edge compute infrastructure that provides sufficient local storage capacity and supports automatic retry mechanisms with exponential backoff. Implement continuous monitoring of buffer utilization with alerts before capacity thresholds are reached, giving operators time to take corrective action. Validate buffered data for integrity before and after transmission to detect corruption during network disruptions.

Implementation steps

1. Deploy edge compute infrastructure (such as AWS IoT Greengrass, AWS Outposts, or local servers) to provide local compute, storage, and data management capabilities. Configure local buffering with retention policies based on data priority and available storage capacity.
2. Configure data transfer mechanisms with built-in retry and exponential backoff strategies. Use services like AWS DataSync for large file transfers with automatic integrity verification, or implement custom retry logic for smaller payloads.
3. Monitor edge storage utilization using Amazon CloudWatch metrics and create alarms for capacity thresholds to alert operators before buffers reach maximum capacity.
4. For disconnected environments, consider AWS Snowball Edge Edge devices with local S3-compatible storage and automatic sync when connectivity is restored.

Resources

Related best practices:

- LSREL07-BP01 Implement system-wide data checksums and transfer validation
- LSREL11-BP01 Implement monitoring of equipment telemetry to detect anomalies

Related tools:

- AWS IoT Greengrass
- AWS IoT Core
- AWS DataSync
- AWS Snowball Edge Edge
- AWS Outposts

LSREL06-BP01 Orchestrate workflows with checkpointing and failure isolation

Implement workflow orchestration with built-in checkpoints, retries, and error isolation. By persisting intermediate results and enabling targeted retries, failed tasks can be reprocessed independently without requiring full pipeline reruns. Event-driven reprocessing further improves efficiency, allowing workflows to resume from the last successful step.

Desired outcome:

- Multi-step workflows can continue execution despite partial failures.
- Failed tasks are retried automatically or isolated for reprocessing.
- Long-running pipelines are resilient and cost-efficient with research reproducibility requirements.

Common anti-patterns:

- Monolithic pipelines without checkpointing, forcing a complete rerun if one step fails.
- Lack of retry or backoff strategies for transient errors, leading to wasted compute cycles.
- Hardcoding dependencies between tasks so that downstream steps fail on minor upstream issues.

- No visibility into workflow state, making recovery manual and error-prone.

Benefits of establishing this best practice:

- Reduces wasted compute and accelerates scientific turnaround time.
- Improves reproducibility and auditability by maintaining step-level logs and artifacts.
- Lowers cost by reprocessing only failed tasks instead of entire workflows.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Break workflows into modular, independently testable steps with well-defined inputs and outputs. Introduce checkpoints by persisting intermediate artifacts in durable storage, allowing workflows to restart from the last good state. Implement retry and backoff strategies for transient errors to avoid unnecessary failures.

Event-driven or rule-based triggers can restart only failed steps, enabling faster recovery. Build in bservability from the start, with centralized logs, metrics, and traces for each step to support debugging and auditing.

Implementation steps

1. Use AWS Step Functions to orchestrate pipelines with retry and error handling policies.
2. Execute step workloads on AWS Batch or Amazon ECS or Amazon EKS for scalable and isolated compute.
3. For genomics workloads, use AWS HealthOmics Workflows to run CWL and WDL pipelines with built-in support for checkpointing.
4. Persist intermediate results in Amazon S3 for recovery and reproducibility.
5. Capture centralized logs and metrics in Amazon CloudWatch for workflow observability.
6. Use Amazon EventBridge to automatically trigger recovery workflows for failed steps without restarting the full pipeline.

Resources

Related best practices:

- Long-term storage and reliable recovery of trial data
- Improve observability and operational insights in pipelines

LSREL07-BP01 Implement system-wide data checksums and transfer validation

Incorporate integrity verification at every transfer and transformation stage. Use cryptographic checksums (like SHA-256 or MD5) or ETags to validate files when moved into or across the cloud. Use managed services like AWS DataSync or Amazon S3 replication that perform integrity checks automatically. For domain-specific use cases (like genomics and imaging), add plausibility checks to detect biologically inconsistent results that may indicate processing corruption.

Desired outcome: Data fidelity is preserved during ingestion, transfer, and transformation, with verifiable proof that data has not been altered.

Common anti-patterns:

- Moving data without checksum or hash verification.
- Relying solely on application logs to detect corruption.
- Using manual copy processes without automated validation.

Benefits of establishing this best practice:

- Reduces risk of silent corruption during high-volume genomic or imaging transfers.
- Increases trust in research outputs by deriving results validated input data.
- Provides auditors and regulators with evidence of data integrity safeguards.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Integrate checksums and validation at every stage where data is moved or transformed. Cryptographic checksums (such as SHA-256) or S3 ETags provide automated validation during transfers into cloud storage. Managed services like AWS DataSync, S3 Replication, and S3 Transfer Acceleration perform integrity checks by default, reducing operational burden. For scientific pipelines, augment checksum validation with domain-specific checks, such as detecting biologically implausible variants in genomic data or corrupt slices in imaging data.

Implementation steps

1. Data should be ingested into Amazon S3 where ETag-based validation confirms file integrity.
2. Use AWS DataSync for on-premises to cloud transfers, verifying that validation occurs automatically.
3. Configure S3 Replication with replication metrics enabled to verify data consistency across buckets.
4. For sensitive research workloads, embed integrity checks directly into processing pipelines (for example, validating SHA-256 digests before and after transformation).

LSREL07-BP02 Build idempotent and reproducible processing pipelines

Design processing systems that produce consistent results regardless of retries or partial failures. Incorporate unique identifiers, transaction logs, and state tracking to verify that retries don't duplicate or corrupt data. In research and clinical analysis, reproducibility of results under repeated execution is a cornerstone of scientific and regulatory assurance.

Desired outcome: Processing pipelines can be retried without causing duplication or corruption, maintaining reproducibility of results under repeated execution.

Common anti-patterns:

- Designing pipelines that overwrite intermediate results without safeguards.
- Using non-unique identifiers for jobs, leading to duplication of processed data.
- Failing to persist state, making retries non-deterministic.

Benefits of establishing this best practice:

- Maintains scientific reproducibility by producing consistent outputs from the same inputs.
- Reduces wasted compute cycles by allowing safe retries after transient errors.
- Increases confidence in regulatory submissions where reproducibility is scrutinized.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Idempotency should be a foundational principle in workflow design. Each job execution must generate unique identifiers tied to input datasets and persist transaction logs or checkpoints. State tracking should record which inputs have been successfully processed, enabling retries without duplication. For long-running genomic analysis, idempotency verifies that a failed job can be retried without corrupting downstream results.

Implementation steps

1. Processing pipelines should use AWS Step Functions or AWS Batch to track execution state and verify that retries are idempotent.
2. Use Amazon DynamoDB or Amazon RDS to persist transaction logs and job state.
3. Store intermediate artifacts in Amazon S3 with unique identifiers as S3 prefixes (for example, prefixing with dataset UUIDs) to maintain reproducibility.
4. Implement retry policies with exponential backoff in AWS Step Functions, so transient failures don't result in duplicate or corrupted processing.

LSREL07-BP03 Use staged validation and data quarantine mechanisms

Introduce controlled validation stages where data is quarantined before being processed further. Apply automated rules for schema conformity, metadata completeness, and domain-specific plausibility (for example, genomic variants outside biological ranges). Only data that passes validation and adherence data standards (such as CDISC or FHIR) proceeds. Isolate data that fails for human review, with full audit trails to support regulatory adherence.

Desired outcome: Only data that meets predefined quality and plausibility criteria enters production pipelines, while problematic data is isolated for review.

Common anti-patterns:

- Directly processing incoming data without validation.
- Overlooking schema, metadata, or biological plausibility checks.
- Mixing invalid data with production datasets, causing downstream corruption.

Benefits of establishing this best practice:

- Stops invalid datasets from contaminating results.

- Accelerates regulatory reviews by providing traceable validation evidence.
- Improves scientific trust by verifying that only high-quality data enters analysis.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Data pipelines should incorporate a quarantine stage where incoming data is validated before processing. Automated rules must check for schema adherence, metadata completeness, and instrument-specific error patterns. Apply domain-specific rules, such as filtering out biologically implausible values in clinical data. Flag and isolate quarantined data, with human review workflows for resolution.

Implementation steps

1. Ingest new datasets into a quarantine Amazon S3 bucket with tagging enabled to indicate that data is pending validation.
2. Use AWS Glue or AWS Lambda to validate schema, metadata, and plausibility.
3. Store failed validation results in Amazon DynamoDB or OpenSearch for investigation, and configure notifications through Amazon SNS to alert data stewards.
4. Only validated data should be transitioned into production pipelines using S3 lifecycle policies or Step Functions orchestration.

LSREL07-BP04 Track data lineage with lifecycle metadata

Assign metadata tags (for example, raw, filtered, processed, and analyzed) at each lifecycle stage, so data state is visible. This enables reproducibility, auditing, and debugging when results need to be traced back to raw inputs. Use catalogs and governance tools to track lineage across storage and processing layers.

Desired outcome: Data state is transparent across ingestion, processing, and analysis, with lineage records supporting reproducibility and audits.

Common anti-patterns:

- Failing to label datasets by processing stage.
- Storing derived data without linkage to raw inputs.
- Using inconsistent or unstructured metadata practices.

Benefits of establishing this best practice:

- Enables reproducibility by tracing results back to raw data.
- Simplifies compliance-related audits by showing how data was transformed.
- Reduces troubleshooting time by quickly identifying the source of anomalies.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Every dataset should be tagged with lifecycle metadata reflecting its stage: raw, filtered, processed, or analyzed. Lineage metadata should include transformation steps, software versions, and parameter settings. These lineage records must be centralized in a catalog or metadata repository to provide transparency across the workload.

Implementation steps

1. Store datasets in Amazon S3 with metadata tags to reflect their lifecycle stage.
2. Use AWS AWS Glue Data Catalog to maintain a centralized record of lineage and transformations.
3. Capture transformation metadata during pipeline execution using AWS Step Functions or AWS Lambda and store results in Amazon DynamoDB or Amazon OpenSearch Service.
4. Include metadata in evidence packages for GxP-regulated workloads.

LSREL08-BP01 Incorporate validated redundancy into architecture design

During the design phase, plan for redundancy across fault-isolated zones and validate that each component meets regulatory requirements. Qualification of secondary systems should not be an afterthought but part of the original architecture plan. Document design decisions to show how redundancy supports both technical reliability and regulatory obligations.

Desired outcome: Workloads remain available during component failures, and redundant systems are equally validated for adherence.

Common anti-patterns:

- Treating secondary systems as non-validated backups rather than qualified components.

- Adding redundancy late in design, creating gaps.
- Documenting only the primary system in validation records.

Benefits of establishing this best practice: Improves confidence in availability without compromising regulatory expectations. Provides reproducible architectural evidence for audits.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When planning workload architecture, design for multi-zone redundancy from the start. Your validation protocols should explicitly include redundant components, not just primaries. For each architecture diagram and system description, demonstrate that redundant paths preserve the same adherence controls, including data capture and security safeguards. Qualification evidence should reflect the entire redundant system architecture as validated.

Implementation steps

1. Define multi-AZ or multi-Region designs in your architecture blueprints and validate them in qualification testing.
2. Document redundancy plans in your system design specifications and standard operating procedures.
3. Use AWS services such as Amazon RDS Multi-AZ, Amazon S3 cross-region replication, or Elastic Load Balancing across multiple Availability Zones, verifying that validation records include both primary and redundant configurations.

LSREL08-BP02 Design compliance-aware failover workflows

Architect failover mechanisms that explicitly maintain critical functions such as authentication, audit logging, and data validation. During the planning phase, map how these functions transition across components and zones, and include them in system qualification protocols.

Desired outcome: Failover processes preserve your regulatory state and do not bypass required controls.

Common anti-patterns:

- Assuming failover preserves compliance-aligned workflows without testing.

- Not qualifying the secondary environment.
- Logging or audit trails that break during transition.

Benefits of establishing this best practice: Maintains trust with regulators and reduces deviations during outages.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Plan failover mechanisms alongside your core architecture, treating them as integral rather than auxiliary processes. Map how functions such as authentication, authorization, encryption, and audit logging persist across failover scenarios. Include these workflows in risk assessments and validation test plans to verify that they behave consistently under failover conditions.

Implementation steps

1. Document failover workflows in architecture specifications and validate them with test evidence.
2. Use AWS managed services to design automated failover with Amazon Route 53 health checks, Amazon RDS Multi-AZ failover, or Auto Scaling group replacements.
3. Keep AWS CloudTrail, AWS Config, and Amazon CloudWatch monitoring active during failover, and include these checks in validation evidence to demonstrate persistence.

LSREL08-BP03 Align architecture priorities with scientific and regulatory context

Early in the design cycle, evaluate the scientific and regulatory priorities of each workload. In patient-facing clinical workloads, design for continuous availability with reconciliation workflows. In regulated manufacturing, prioritize data integrity in the architecture, even if this limits availability. Planning with these priorities in mind avoids trade-off surprises later.

Desired outcome: Availability trade-offs are consciously designed to meet workload-specific requirements.

Common anti-patterns:

- Treating each workload with a uniform HA design.
- Prioritizing availability at the cost of data integrity in regulated processes.

- Failing to document the rationale for architectural trade-offs.

Benefits of establishing this best practice: Aligns architectures with the unique regulatory and scientific priorities of each workload, and improves transparency in audits.

Implementation guidance

Architectural planning should explicitly evaluate the workload's scientific and regulatory context before finalizing availability and failover mechanisms. For example, clinical trial systems may tolerate some reconciliation steps post-recovery if availability is critical, while batch manufacturing systems may require uncompromising data integrity even if it reduces availability. Document these trade-offs, rationale, and validation approach as part of the design package.

Implementation steps

1. Capture availability and data integrity requirements in system requirements specifications and risk assessments.
2. Map RTO and RPO targets to these requirements.
3. On AWS, implement availability features with services like Amazon Aurora Multi-AZ or Amazon S3 Versioning, paired with validation steps for data integrity.
4. Store architecture trade-off documentation and validation outcomes in a controlled repository for audit readiness.

Change nanagement

LSREL09: How do you design, validate, rollback, and recover from failed changes so life sciences workloads remain reliable and compliant?

Reliable rollback and recovery procedures verify that failed changes do not compromise the availability, continuity, or data integrity of life sciences research workloads. Rollbacks must be predictable, validated, and automated where possible so that operations can resume quickly. In GxP-regulated environments, these same procedures also support regulatory requirements by providing auditable recovery evidence. Effective change management must also assess risks of, validate, and document changes so the system remains in a controlled and compliant state throughout its lifecycle.

LSREL10: How do you test reliability in life sciences workloads to maintain adherence and resilience?

Reliability testing for life sciences workloads must validate both technical resilience (availability, fault tolerance, performance under stress) and regulatory adherence (GxP, data integrity). A structured testing strategy verifies that workloads perform predictably under normal and adverse conditions, failures are anticipated and contained, and evidence is generated as part of system validation.

Best practices

- LSREL09-BP01 Create and verify rollback plans
- LSREL09-BP02 Implement safe deployment strategies (like blue/green or canary)
- LSREL09-BP03 Verify data integrity and point-in-time recovery
- LSREL09-BP04 Maintain auditable rollback and recovery records
- LSREL09-BP05 Implement risk-based change control for validated systems
- LSREL09-BP06 Automate validation testing for changes
- LSREL10-BP01 Implement comprehensive reliability testing
- LSREL10-BP02 Validate end-to-end reliability of regulated workloads
- LSREL10-BP03 Test data integrity under failure conditions

LSREL09-BP01 Create and verify rollback plans

For every release or approved change, maintain a documented rollback plan that is versioned with the release artifacts. Validate rollback procedures in non-production environments and include rollback steps in the change control package and test evidence. Define roles, approvals, and communications for execution and escalation.

Desired outcome: Rollback processes are predictable, validated, and repeatable so that failed changes do not compromise operations, regulatory adherence, or data integrity.

Common anti-patterns:

- Relying on undocumented or manual rollback steps.
- Skipping rollback testing due to time constraints.

- Treating rollback as optional rather than mandatory in GxP environments.

Benefits of establishing this best practice:

- Reduces risk of prolonged downtime that can invalidate experiments or delay clinical timelines.
- Preserves integrity of regulated datasets and audit trails.
- Demonstrates predictable change control to auditors and stakeholders.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Rollback plans should be created alongside release plans and treated as first-class artifacts in change control. Plans must include the exact steps to revert application code, configuration, database schema, and dependent infrastructure changes. Validation of rollback plans in staging or pre-production environments should exercise the entire procedure (including approvals and notifications) so that time-to-rollback and behavioral impacts are well understood.

Where human steps are required, provide clear runbooks and defined approvers. Where feasible, automate rollback actions to reduce human error. Record rollback test evidence, and attach the evidence to the validation package for auditability.

Implementation steps

1. Create pipelines and rollback actions in AWS CodePipeline and codify deployment artifacts with AWS CloudFormation templates so infrastructure changes are reversible.
2. Use AWS CodeDeploy or CloudFormation change-sets with pre-defined rollback behavior.
3. Implement standardized rollback runbooks using AWS Systems Manager Automation so operators can run approved, automated steps.
4. Store validated rollback artifacts and test evidence in AWS CodeCommit or Amazon S3, and include change-control metadata for traceability.

Resources

Related best practices:

- Continuity of workflows and data availability during downtime

- Fault isolation and graceful degradation in workflows
- Automated validation in deployments

LSREL09-BP02 Implement safe deployment strategies (like blue/green or canary)

Adopt deployment patterns that minimize blast radius and allow rapid diversion to the last known good state. Phased rollouts provide early detection of issues and enable quick rollback without impacting users or downstream regulatory workflows.

Desired outcome: Failed releases can be quickly rolled back or diverted with minimal disruption to end users and regulatory processes.

Common anti-patterns:

- Performing deployments at once without a rollback path.
- No traffic segmentation or user impact analysis during rollouts.
- Failing to gather metrics and health signals during phased deployments.

Benefits of establishing this best practice:

- Limits impact to a subset of experiments or studies rather than the entire user base and workflows.
- Enables rapid return to a validated state, protecting study timelines.
- Improves stakeholder confidence in release safety and stability.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Choose a rollout strategy based on risk and verification needs: blue/green for fast switchovers to fully validated environments, canary for incremental exposure and metric observation, or feature-flag driven rollouts for business-level segmentation. Instrument deployments with health metrics, business KPIs, and signals so an automated or manual rollback decision can be made quickly. Validate the traffic-shift and rollback procedures during staging exercises and include them in release approvals.

Implementation steps

1. Use blue/green deployment patterns available in AWS Elastic Beanstalk or by provisioning parallel stacks using AWS CloudFormation and switching traffic with Amazon Route 53 or load balancer reconfiguration.
2. Use Amazon ECS or Amazon EKS with traffic shifting configured (for example, using AWS App Mesh or AWS CodeDeploy integration) to implement canary releases.
3. Implement automated traffic shifting and rollback policies in AWS CodeDeploy so that failing canaries automatically trigger rollbacks or traffic diversion.

Resources

Related best practices:

- Continuity of workflows and data availability during downtime
- Resilient environment provisioning and lifecycle management
- Automated validation in deployments

LSREL09-BP03 Verify data integrity and point-in-time recovery

Protect application and data state so that you can restore to a precise point-in-time prior to a failed change. Backups must be consistent, validated, and aligned with organizational RTO and RPO targets. Include configuration and metadata in backups so restored environments are audit-complete.

Desired outcome: Data and application state can be restored to the exact state prior to the failed change, preserving adherence and operational continuity.

Common anti-patterns:

- Infrequent or unplanned backups without restore validation.
- Omitting metadata or configuration in backup sets.
- Not aligning backup frequency to defined RTO and RPO needs of regulated systems.

Benefits of establishing this best practice:

- Avoids loss of experiment data, patient records, or audit evidence during failed changes.

- Shortens recovery time, preserving study timelines and sample integrity.
- Demonstrates recoverability to auditors and stakeholders.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define RTO and RPO aligned to the criticality of datasets and regulated processes. Implement automated, policy-driven backups that include application-level snapshots, database backups (including transaction logs for point-in-time recovery), and configuration exports. Regularly perform restore drills and document restoration time and fidelity as part of validation evidence. Store backups in tamper-evident, redundant storage and that retention policies meet regulatory retention requirements.

Implementation steps

1. Implement policy-driven backups using AWS Backup for supported services.
2. Enable automated backups and snapshots for databases (for example, automated backups for Amazon RDS and snapshot schedules for Amazon EC2 or Amazon EBS).
3. Configure point-in-time recovery for services that support it, such as enabling PITR for Amazon DynamoDB.
4. Store and catalog backup artifacts in Amazon S3 with appropriate lifecycle and retention rules, and use AWS Backup's recovery testing features to validate restores.

Resources

Related best practices:

- Long-term storage and reliable recovery of trial data
- Observability and monitoring of pipelines
- Automated validation in deployments

LSREL09-BP04 Maintain auditable rollback and recovery records

Capture and retain immutable, timestamped records of every rollback and recovery action, including approvals, deviations, timing, and outcomes. Audit and governance artifacts are required evidence for GxP adherence and should be integrated into change control and incident reporting.

Desired outcome: Every rollback or recovery is traceable and auditable with internal SOPs and external GxP requirements.

Common anti-patterns:

- Treating rollback as a purely technical process without traceability.
- Failing to log deviations or decisions during recovery.
- Storing audit records in a non-durable or unsearchable formats.

Benefits of establishing this best practice:

- Demonstrates regulatory adherence and governance over change activities.
- Enables post-event analysis and corrective actions to avoid recurrence.
- Supports transparent reporting to sponsors, regulators, and QA teams.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Integrate audit capture into every phase of change and rollback: record the release checksum, approvals, automated actions taken, timestamps for each step, deviations, and post-rollback verification results. Use immutable storage and tamper-evident logs and include audit evidence as part of the change record. Verify that your log retention policy meets regulatory retention policies and that search and indexing capabilities support timely retrieval for inspections.

Implementation steps

1. Enable AWS CloudTrail to capture API activity and operational actions related to deployment and rollback.
2. Use AWS Config to record configuration state and change history for infrastructure resources.
3. Store immutable audit records in Amazon S3 with S3 Object Lock enabled to enforce retention and immutability.
4. Index and make records discoverable with Amazon OpenSearch Service or a governance catalog for rapid response to audit requests.

Resources

Related best practices:

- Security and logging for research environments
- Resilient environment provisioning and lifecycle management
- Automated validation in deployments

LSREL09-BP05 Implement risk-based change control for validated systems

Establish a structured, risk-based change control process that categorizes changes according to their potential impact on product quality, patient safety, and data integrity. Apply proportional testing and validation depending on the change classification. High-risk changes to GxP systems should undergo rigorous verification and dual review, while low-risk changes may be handled with streamlined procedures. Track, approve, and document changes, and update validation evidence to confirm the system remains in a validated state after the change.

Desired outcome: Changes are assessed, approved, tested, and documented based on risk so that systems remain reliable and validated.

Common anti-patterns:

- Treating each change identically.
- Making undocumented infrastructure changes.
- Failing to demonstrate that the validated state was preserved.

Benefits of establishing this best practice: Maintains continuity of validated research systems, reduces audit findings, and minimizes the chance of change-related downtime or data issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Risk-based change control should be embedded in governance processes. Each proposed change should be assessed for potential impact on regulated workloads. Testing scope and depth must scale with risk, and approval workflows should reflect that classification. Documentation should be

continuously updated so there is a clear link between requirements, specifications, executed tests, and evidence that the validated state was maintained.

Implementation steps

1. Use AWS Config to track infrastructure changes and maintain a configuration history.
2. Codify infrastructure in AWS CloudFormation with change sets for controlled, reversible changes.
3. Store approvals, validation results, and associated artifacts in Amazon S3 with Object Lock for immutability.
4. Use AWS Audit Manager to map evidence against regulatory controls and demonstrate ongoing validation.

LSREL09-BP06 Automate validation testing for changes

Develop automated functional and regulatory validation test suites that run after system changes to verify critical functionality, data integrity, and regulatory controls. Include tests for audit trails, access control, and business functionality. These tests should also reassess resiliency factors like recovery time objectives (RTO) and recovery point objectives (RPO) to verify that changes do not degrade reliability targets. Test results should be retained as evidence in the validation package.

Desired outcome: Automated validation verifies that changes do not compromise functionality, data integrity, or resiliency, and provides evidence of continued adherence.

Common anti-patterns:

- Relying only on manual testing.
- Skipping regression validation for minor changes.
- Not verifying resiliency requirements after updates.

Benefits of establishing this best practice: Reduces downtime risk, improves reproducibility, and increases regulator confidence by demonstrating that every change is validated and reliable.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Validation test suites should be version-controlled and updated as systems evolve. Automating them improves consistency and reduces the burden on QA teams. These tests must be integrated into the deployment pipeline so that every change produces verifiable evidence of continued system validation. Results should be reviewed, approved, and archived as part of the change control record.

Implementation steps

1. Integrate automated validation tests into AWS CodePipeline so they run after deployments.
2. Execute functional tests with containerized workloads on Amazon ECS/EKS or serverless tests with AWS Lambda.
3. Capture logs and results in Amazon CloudWatch Logs and archive them to Amazon S3 for long-term retention.
4. Use AWS Audit Manager to generate audit evidence linking test execution to change approvals.

LSREL10-BP01 Implement comprehensive reliability testing

Develop structured protocols that test reliability aspects including load performance, failover, recovery, and long-term stability. For regulated systems, include reliability tests in the validation package with traceability to user and regulatory requirements. Use controlled chaos engineering experiments to validate resilience by safely injecting faults and failures into your applications while preserving data integrity and adherence.

Desired outcome:

- Reliability tests are systematic, repeatable, and documented.
- Systems demonstrate resilience to failure injection and load stress.
- Test evidence supports regulatory validation requirements.

Common anti-patterns:

- Treating reliability tests as optional instead of mandatory.
- Running only idealized tests without simulating failures.
- Lack of traceability between reliability test cases and regulatory or user requirements.

Benefits of establishing this best practice:

- Improves predictability of experiments and workloads under stress.
- Builds regulator and auditor confidence in system resilience.
- Reduces costly delays by uncovering reliability gaps early.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define reliability acceptance criteria in line with user and regulatory requirements.

Incorporate resilience tests into validation and qualification protocols.

Perform controlled chaos experiments to uncover weaknesses safely.

Automate reliability testing where possible for repeatability.

Implementation steps

1. Run fault-injection experiments with AWS Fault Injection Service (FIS).
2. Use AWS CodePipeline to integrate reliability tests into CI/CD.
3. Capture test logs in Amazon CloudWatch Logs and archive evidence in Amazon S3 Object Lock for regulatory adherence.
4. Document test execution and results in AWS Audit Manager.

LSREL10-BP02 Validate end-to-end reliability of regulated workloads

Conduct end-to-end reliability validation exercises that test not just recovery but the overall system's ability to maintain adherence and functionality during adverse events. These tests should validate high availability, monitoring alerts, automated failover, and controls in production-like scenarios.

Desired outcome:

- Holistic system reliability validated under real-world conditions.
- Compliance-aligned functions (audit trails, access controls, data integrity) remain intact during adverse events.

- Test outcomes provide documented evidence for regulatory audits.

Common anti-patterns:

- Focusing only on technical recovery while ignoring controls.
- Testing components in isolation but never validating the end-to-end system.
- No evidence of reliability testing available for auditors.

Benefits of establishing this best practice:

- Demonstrates system resilience across full workflows, not just components.
- Strengthens audit readiness with comprehensive reliability evidence.
- Reduces operational risk by validating reliability before real incidents occur.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Plan integrated reliability tests across application and data layers.

Simulate production-like workloads during validation.

Validate not only failover and monitoring but also regulatory controls.

Store test artifacts centrally with immutable retention.

Implementation steps

1. Use AWS Elastic Load Balancing with Auto Scaling to validate failover.
2. Run simulated production workloads using AWS Batch or Amazon ECS.
3. Trigger alerts through Amazon CloudWatch Alarms and verify monitoring in AWS Config.
4. Archive reliability validation evidence in Amazon Glacier for long-term retention.

LSREL10-BP03 Test data integrity under failure conditions

Design and execute specific tests to validate that data integrity is preserved during system failures, network outages, or recovery processes. For workloads involving trial data, manufacturing records,

or patient datasets, enforce transactional boundaries, roll back partial updates safely, and block corrupted data from entering downstream systems.

Desired outcome:

- Data integrity maintained during failures and recovery.
- Tests confirm correct handling of partial transactions and error scenarios.
- Evidence demonstrates adherence to data integrity requirements.

Common anti-patterns:

- Relying on functional tests without failure/integrity validation.
- No rollback or compensation testing for partial failures.
- Assuming backups restore consistent datasets without testing.

Benefits of establishing this best practice:

- Protects against corrupted datasets invalidating scientific outcomes.
- Builds trust in data reproducibility and traceability for regulators.
- Avoids costly reruns of experiments or trials due to data loss.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Incorporate integrity validation into test plans for recovery and failover.

Test for both transient disruptions and long-term outages.

Use domain-specific integrity checks for genomic, clinical, or manufacturing datasets.

Implementation steps

1. Enable Amazon RDS point-in-time recovery and validate consistency after restore.
2. Run AWS Glue Data Quality jobs to verify schema and record consistency.
3. Store validation reports in Amazon S3 with Object Lock for immutability.
4. Integrate validation results into reports with AWS Audit Manager.

Failure management

LSREL11: How do you detect and manage laboratory equipment failures for continuity of research operations?

Unexpected equipment failures in laboratories such as with sequencers, chromatography systems, or imaging devices can result in data loss, disrupted experiments, compromised reproducibility, and non-adherence to regulatory requirements. By adopting telemetry-driven monitoring, predictive maintenance, and redundancy planning, life sciences organizations can detect issues early, proactively mitigate failures, and maintain business continuity for critical research operations.

LSREL12: How do you design workloads to recover from disruptions while maintaining life sciences data integrity?

Recovery from infrastructure or service disruptions in life sciences workloads must prioritize both restoration of services and preservation of data integrity. Research and clinical systems must keep data complete, consistent, and verifiable during and after recovery. Recovery strategies should therefore include mechanisms to validate integrity, align recovery times with scientific/business impact, and verify that distributed systems reconcile correctly across environments and partners.

LSREL13: How do you monitor workloads to detect reliability issues before they affect life sciences operations?

Proactive monitoring is essential for maintaining reliability in life sciences workloads. Monitoring strategies should detect early warning signs of potential issues that could affect system availability, performance, or data integrity. By addressing these signs before they impact operations, organizations protect critical research processes, maintain reproducibility, and uphold regulatory obligations.

Best practices

- LSREL11-BP01 Implement monitoring of equipment telemetry to detect anomalies

- LSREL11-BP02 Apply predictive maintenance using AI models
- LSREL11-BP03 Plan redundancy for critical laboratory equipment
- LSREL12-BP01 Implement recovery processes with data integrity verification
- LSREL12-BP02 Define recovery time objectives based on scientific and business impact
- LSREL12-BP03 Maintain data consistency in distributed research systems
- LSREL12-BP04 Implement cyber resilience for GxP-regulated backup data
- LSREL13-BP01 Implement comprehensive monitoring for regulated systems
- LSREL13-BP02 Monitor data integrity across scientific processing pipelines
- LSREL13-BP03 Track reliability metrics aligned to regulatory needs

LSREL11-BP01 Implement monitoring of equipment telemetry to detect anomalies

Capture equipment telemetry such as temperature, vibration, cycle counts, and error codes in real time to identify anomalies early. A consistent telemetry pipeline enables proactive monitoring, alerts, and traceability of equipment performance across research facilities.

Desired outcome:

- Continuous monitoring of lab equipment to detect anomalies early.
- Reduced risk of unplanned downtime through proactive alerts.
- Complete telemetry records available for troubleshooting and audits.

Common anti-patterns:

- Not collecting or collecting telemetry data inconsistently.
- Storing telemetry without time-stamping or contextual metadata.
- Failing to establish thresholds or alerts on critical parameters.

Benefits of establishing this best practice:

- Improves reliability of experiments through early detection of issues.
- Enables root cause analysis and reproducibility through equipment performance records.
- Supports regulatory adherence by providing traceable operational logs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A resilient telemetry strategy requires secure, standardized pipelines for collection and long-term storage. Verify that your data includes time stamps and contextual metadata to support traceability. Integrate alerts with incident response processes for timely remediation. Preserve telemetry records for audits and long-term performance analysis.

Implementation steps

1. Deploy AWS IoT Greengrass to capture telemetry locally and preprocess sensitive data at the lab site.
2. Stream telemetry securely into AWS IoT Core for ingestion.
3. Store data in Amazon Timestream for time-series analysis or Amazon S3 for long-term archival.
4. Configure anomaly detection with Amazon CloudWatch Alarms and notify research operations teams through Amazon SNS.
5. Provide researchers with performance dashboards using Quick for visualization.

Resources

Related best practices:

- Incident detection and alerting
- Data integrity and traceability for regulated environments
- Root cause analysis frameworks

LSREL11-BP02 Apply predictive maintenance using AI models

Use AI/ML models to analyze telemetry, usage logs, and maintenance records for predicting potential equipment failures. Integrating predictive insights with laboratory management systems reduces downtime, optimize calibration schedules, and extend equipment life.

Desired outcome:

- Anticipation of failures before they occur, reducing experiment disruption.
- Optimized maintenance schedules that balance reliability with operational efficiency.

- Integration of predictive insights into research workflows and logs.

Common anti-patterns:

- Relying solely on reactive maintenance after equipment fails.
- Collecting data but not training or updating predictive models.
- Not integrating predictive insights with LIMS or quality systems, leading to disconnected records.

Benefits of establishing this best practice:

- Reduces equipment failure rates and downtime.
- Extends equipment life cycles and optimized resource utilization.
- Enhances regulatory adherence through integrated maintenance and calibration logs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Predictive maintenance requires high-quality, centralized datasets including logs, calibration records, and telemetry. Periodically validate AI models to maintain trustworthiness in regulated environments. Integrating outputs into research systems verifies that predictions drive actionable workflows, rather than remaining siloed in technical teams.

Implementation steps

1. Ingest telemetry into Amazon CloudWatch and usage logs into Amazon S3.
2. Train ML models in Amazon SageMaker AI using historical performance datasets.
3. For turnkey options, deploy Amazon Lookout for Equipment to analyze telemetry streams.
4. Integrate predictive alerts into Amazon EventBridge to trigger workflows or incident responses.
5. Store maintenance logs in Amazon RDS or integrate directly with LIMS databases for traceability.

Resources

Related best practices:

- AI/ML lifecycle management in regulated environments

- Integration of IT and OT (Operational Technology) systems
- GxP-aligned system validation for ML-driven processes

LSREL11-BP03 Plan redundancy for critical laboratory equipment

Design redundancy strategies for critical laboratory equipment by maintaining hot spares, parallelized runs, or vendor service agreements. Effective redundancy maintains continuity of operations even during failures of high-value or high-throughput instruments.

Desired outcome:

- Continuous operation of critical research workflows during equipment failures.
- Reduced delays in experiments and studies by having spare capacity.
- Documented continuity plans for audits and inspections.

Common anti-patterns:

- Treating equipment equally without prioritizing critical instruments.
- Failing to budget or plan for spare capacity in core systems.
- Assuming vendor maintenance SLAs are sufficient for continuity of research operations.

Benefits of establishing this best practice:

- Reduces operational risk by maintaining continuity during unexpected failures.
- Increases confidence in meeting research timelines and commitments.
- Demonstrates resilience and preparedness to regulators and auditors.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Start by classifying lab equipment by criticality and throughput to identify which instruments require redundancy. Define strategies such as hot spares, workload sharing, or vendor backup agreements. Maintain documented runbooks for failover to redundant equipment, and periodically simulate outages to test readiness.

Implementation steps

1. Track redundancy configurations using AWS Config for reporting.
2. Store redundancy plans, SLAs, and vendor contracts in Amazon S3, and enable fast search with Amazon OpenSearch Service.
3. Orchestrate escalation procedures using Amazon SNS and AWS Lambda when failures are detected.
4. Record continuity outcomes in Amazon DynamoDB for audit traceability.
5. Periodically simulate infrastructure and application-level failures with AWS Fault Injection Service (FIS) to validate the continuity of data capture, workflow orchestration, and failover processes supporting laboratory equipment.
6. For physical instruments, conduct tabletop or vendor-led failure simulations to keep redundancy plans practical.

Resources

Related best practices:

- Business continuity and disaster recovery planning
- Risk-based classification of lab assets
- Vendor management and SLA governance

LSREL12-BP01 Implement recovery processes with data integrity verification

Design recovery procedures with explicit steps to verify data integrity once recovery operations complete. For regulated life sciences workloads, include checksums, reconciliation processes, or dual-control verification. Verification evidence should be documented as part of disaster recovery and audit processes.

Desired outcome:

- Recovered datasets are validated for completeness and accuracy.
- Data integrity verification is automated where possible.
- Audit evidence of verification steps is retained for compliance-related purposes.

Common anti-patterns:

- Recovery plans restore services without validating data correctness.
- Assuming backups are correct without performing validation checks.
- No retention of verification evidence for audit purposes.

Benefits of establishing this best practice:

- Avoids propagation of corrupted or incomplete data into research workflows.
- Provides regulators confidence that scientific data is accurate after recovery.
- Reduces risk of invalid conclusions or repeated experiments.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define recovery procedures that integrate integrity checks, including checksum validation, record counts, and cross-system reconciliation. Automate as much as possible to reduce manual errors, and document the results. For GxP workloads, require dual sign-offs for verification evidence. Retain logs and reports as part of change and recovery records.

Implementation steps

1. After recovery, run checksum verification jobs with AWS Lambda across restored datasets in Amazon S3.
2. Use AWS Glue or AWS DataBrew to validate schema and record counts.
3. Store verification logs in Amazon S3 with Object Lock for immutability.
4. Integrate verification results into audit tracking using AWS Audit Manager.

LSREL12-BP02 Define recovery time objectives based on scientific and business impact

Recovery time objectives (RTOs) should be established through analysis of the scientific and business impact of downtime. For clinical trial systems, delays may affect patient safety or data collection. For manufacturing or analytical systems, downtime may disrupt schedules, supply chains, or data reproducibility. RTOs must be documented and justified in continuity planning.

Desired outcome:

- Recovery time targets reflect business and scientific criticality.
- Downtime risks are balanced against cost of recovery capabilities.
- RTOs are documented and approved as part of continuity planning.

Common anti-patterns:

- Arbitrary RTOs set without input from science or operations stakeholders.
- Uniform recovery targets across systems.
- Failure to periodically review RTOs as workloads evolve.

Benefits of establishing this best practice:

- Verifies that critical research and clinical activities resume within acceptable windows.
- Avoids over-investment in low-priority systems while safeguarding high-value workloads.
- Builds alignment between IT, R&D, QA, and business stakeholders.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Conduct impact analyses for each workload, identifying scientific, regulatory, and operational consequences of downtime. Define system-specific RTOs based on these findings and align them with acceptable risk tolerances. Document RTOs in continuity and disaster recovery plans, and review them regularly to keep them appropriate as workloads scale.

Implementation steps

1. Use AWS Well-Architected Resilience Hub to document RTO targets for workloads.
2. Configure Amazon RDS Multi-AZ for critical databases to meet short RTOs, while less critical workloads can use Amazon Glacier for slower recovery.
3. Run recovery drills using AWS Elastic Disaster Recovery (DRS) to validate RTO adherence.
4. Record results in AWS Audit Manager for tracking.

LSREL12-BP03 Maintain data consistency in distributed research systems

Distributed life sciences workloads (for example, spanning multiple sites, cloud Regions, or CRO integrations) require mechanisms to maintain data consistency during partial failures and recovery. This includes distributed transactions, compensating actions, and reconciliation processes to improve accuracy and completeness across system components.

Desired outcome:

- Data remains accurate and consistent across distributed components after recovery.
- Conflicts or anomalies are detected and resolved automatically where possible.
- Reconciliation evidence is preserved for audit and reproducibility.

Common anti-patterns:

- No reconciliation of data between distributed systems after recovery.
- Assuming eventual consistency will resolve discrepancies without validation.
- Ignoring data mismatches introduced during failover or partial recovery.

Level of risk exposed if this best practice is not established: High

Implementation guidance

For distributed systems, design recovery processes that reconcile states across components. This may involve compensating transactions, replaying messages, or performing checksum-based reconciliation. Where eventual consistency is used, implement monitoring and exception handling to identify unreconciled discrepancies. Document reconciliation processes in DR runbooks.

Implementation steps

1. Use Amazon DynamoDB global tables or Amazon Aurora Global Database to maintain multi-region consistency.
2. For asynchronous pipelines, implement reconciliation jobs using AWS Step Functions and AWS Lambda to compare datasets across Regions.
3. Capture anomalies in Amazon CloudWatch Logs and route issues into incident workflows through Amazon EventBridge.

4. Retain reconciliation evidence in Amazon S3 for compliance-related purposes.

LSREL12-BP04 Implement cyber resilience for GxP-regulated backup data

Implement cyber resilience strategies for your backup data to protect against ransomware and other cyber threats, with specific considerations for GxP-regulated systems. Cyber resilience goes beyond traditional backup approaches by keeping backups immutable, isolated, and recoverable even during sophisticated cyber attacks while maintaining data integrity in accordance with ALCOA + principles.

Desired outcome: A robust backup strategy that protects GxP-regulated data from cyber threats, which recovers clean, unaltered data following an attack while maintaining regulatory adherence and data integrity.

Common anti-patterns:

- Relying solely on encryption without implementing immutability, leaving GxP data backups vulnerable to deletion.
- Using the same security domain for production and backup systems, allowing credential compromise to affect both environments.
- Assuming backups are valid without regular validation testing, potentially discovering issues only during actual recovery.
- Allowing single-person authorization for critical recovery operations, reducing segregation of duties required for regulated systems.
- Failing to document backup immutability controls as part of the quality management system.

Benefits of establishing this best practice:

- Enhanced protection against ransomware and other cyber threats that specifically target backup infrastructure.
- Maintained data integrity for GxP-regulated information during recovery.
- Improved confidence in recovery capabilities during security incidents.
- Enhanced adherence to regulatory requirements for data protection and recovery.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Traditional backup approaches for GxP systems focus on data availability and integrity but may not adequately address sophisticated cyber threats that specifically target backup infrastructure. Life sciences organizations must implement cyber-resilient backup strategies that maintain regulatory adherence while protecting against evolving threats.

Implement a comprehensive cyber resilience strategy for your GxP-regulated backups that includes four fundamental pillars:

- **Immutability:** Stop backup data from being altered or deleted during its retention period, maintaining the ALCOA+ principle of originality
- **Logical isolation:** Separate backup storage from production environments with distinct security controls to avoid compromise of both systems
- **Integrity validation:** Regularly verify backup data remains uncorrupted and can be successfully restored, improving ongoing adherence to data integrity requirements
- **Access controls:** Implement multi-party approval for critical recovery operations to maintain appropriate segregation of duties for regulated systems

For GxP-regulated systems, document your cyber-resilient backup strategy within your quality management system, including validation of backup immutability controls and recovery processes. This documentation should address how your backup strategy adheres data integrity requirements and should be sufficient to demonstrate regulatory adherence during inspections or audits.

Implementation steps

1. Assess your GxP backup strategy for cyber resilience gaps:

- Evaluate current backup solutions for immutability capabilities and potential vulnerabilities.
- Identify potential attack vectors that could compromise both production and backup systems.
- Determine appropriate retention periods based on data criticality, regulatory requirements, and threat models.
- Document recovery time objectives (RTOs) and recovery point objectives (RPOs) for cyber recovery scenarios.
- Classify backup data based on GxP relevance and criticality.

2. Implement immutable backup storage for GxP data:

- Configure AWS Backup Vault Lock or S3 Object Lock for GxP data with appropriate retention periods.
- Define immutability settings based on data classification and regulatory requirements.
- Implement technical controls to avoid override of immutability settings.
- Document immutability configurations in your data protection policies and Quality Management System.
- Test immutability by attempting to delete or modify protected backups.

3. Establish logical isolation for GxP backup storage:

- Create dedicated AWS accounts for GxP backup storage with separate administrative controls.
- Implement AWS Backup logically air-gapped vault for critical GxP systems requiring enhanced protection.
- Configure strict cross-account access controls using IAM and service control policies (SCPs).
- Establish network isolation between production and backup environments.
- Implement separate authentication mechanisms for backup administration.
- Document isolation controls in your Quality Management System.

4. Implement multi-party approval for GxP system recovery:

- Configure AWS Backup multi-party approval workflows for GxP systems.
- Define approver roles and responsibilities with appropriate separation of duties.
- Document escalation procedures for emergency scenarios requiring expedited recovery.
- Implement comprehensive audit trails for approval actions.
- Align your approval workflows with your quality management system requirements.
- Regularly test approval workflows to verify effectiveness during actual incidents.

5. Validate GxP backup integrity and recovery processes:

- Implement AWS Backup restore testing with automated validation of restored resources.
- Schedule regular recovery exercises in isolated environments for critical GxP systems.
- Document validation procedures and success criteria for different resource types.
- Test recovery from various cyber attack scenarios including ransomware and data corruption.
- Validate data integrity after restoration to improve consistency and completeness.
- Maintain validation documentation as part of your quality management system.

6. Monitor and audit GxP backup protection:

- Configure AWS CloudTrail logging for backup and recovery operations.
- Implement Amazon CloudWatch alarms for unauthorized access attempts or policy violations.
- Regularly review backup protection controls through automated checks.
- Conduct periodic security assessments of backup infrastructure.
- Maintain comprehensive documentation of cyber resilience controls for audits and regulatory inspections.

Resources

Related best practices:

- LSREL13-BP01
- LSOPS03-BP01

Related documents:

- GxP Systems on AWS
- Building cyber resiliency with AWS Backup logically air-gapped vault
- Validate recovery readiness with AWS Backup restore testing
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework

Related examples:

- Building cyber resiliency with AWS Backup logically air-gapped vault

- Validate recovery readiness with AWS Backup restore testing
- Improve recovery resilience with AWS Backup support for Multi-party approval

Related tools:

- AWS Backup
- AWS Backup Vault Lock
- Amazon S3 Object Lock
- AWS CloudTrail
- Amazon CloudWatch
- AWS IAM

LSREL13-BP01 Implement comprehensive monitoring for regulated systems

Establish monitoring that spans infrastructure, applications, data integrity, and audit controls. For GxP systems, verify that your monitoring covers validation-critical parameters identified in risk assessments, so that regulated workloads can demonstrate continuous oversight.

Desired outcome:

- Holistic visibility into workload health and reliability.
- Early detection of anomalies across infrastructure and applications.
- Assurance that monitoring captures validation-critical parameters in GxP systems.

Common anti-patterns:

- Monitoring only infrastructure without application or data-level coverage.
- Relying on reactive alerts instead of proactive anomaly detection.
- Lack of defined monitoring scope for regulated workloads.

Benefits of establishing this best practice:

- Enables quick response before failures impact experiments or studies.
- Provides audit-ready evidence of system oversight for regulators.

- Improves researcher trust in system stability and availability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Design monitoring across layers, including infrastructure (compute, storage, network), application (latency, errors, throughput), and data integrity. Incorporate health checks aligned with business priorities. Define thresholds for alerting and automate incident response workflows. Retain monitoring records in adherence to retention and audit requirements.

Implementation steps

1. Instrument workloads with Amazon CloudWatch metrics, alarms, and dashboards.
2. Capture logs centrally in Amazon CloudWatch Logs.
3. Use AWS X-Ray for distributed tracing of microservices.
4. Monitor configuration drift with AWS Config and events with AWS Security Hub CSPM.
5. Store monitoring evidence in Amazon S3 for regulatory audits.

LSREL13-BP02 Monitor data integrity across scientific processing pipelines

Implement monitoring specifically for data integrity across research pipelines. Track errors such as checksum mismatches, validation failures, or audit trail gaps. For scientific domains like genomics or clinical processing, add plausibility checks to detect biologically impossible or outlier results that may signal workflow errors.

Desired outcome:

- Early detection of data corruption or processing errors.
- Continuous assurance of data accuracy, completeness, and traceability.
- Adherence to data integrity expectations for regulated research.

Common anti-patterns:

- Only validating data at ingestion or final outputs, not during processing.
- Ignoring intermediate pipeline results when monitoring for errors.

- Not capturing or preserving logs for data validation failures.

Benefits of establishing this best practice:

- Protects reproducibility by keeping datasets accurate across processing steps.
- Reduces wasted compute from propagating corrupted or invalid data.
- Strengthens audit confidence through consistent integrity checks.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Incorporate integrity checks throughout pipelines. Validate the following during this process:

- File checksums
- Schema conformance
- Expected data patterns

Retain metadata and logs for traceability. Define biologically relevant plausibility thresholds (for example, variant frequencies or image metrics) to detect anomalies early. Integrate alerts with workflow orchestration so that failed steps are isolated.

Implementation steps

1. Validate file integrity with Amazon S3 ETag checks or checksum verification jobs in AWS Lambda.
2. Store audit trails in Amazon DynamoDB or Amazon S3.
3. Use AWS Glue DataBrew or AWS Data Quality rules for schema and value checks.
4. Implement plausibility validations in genomics workflows through AWS HealthOmics Workflows.
5. Route alerts through Amazon EventBridge into incident response pipelines.

LSREL13-BP03 Track reliability metrics aligned to regulatory needs

Define and monitor reliability metrics that align with both operational resilience and regulatory requirements. For GxP-regulated systems, include system availability, backup/restore success, recovery test completion, and data integrity checks. Retain metric history for audit evidence.

Desired outcome:

- Clear visibility into workload reliability health.
- Metrics aligned with both business SLAs and regulatory expectations.
- Historical reporting available for audits and inspections.

Common anti-patterns:

- Collecting technical metrics without mapping to regulatory requirements.
- Not retaining monitoring data for required regulatory periods.
- No baselines to measure whether reliability is improving or degrading.

Benefits of establishing this best practice:

- Provides measurable evidence of system reliability for regulators and auditors.
- Builds trust with researchers and clinical teams in system performance.
- Supports proactive investment in reliability improvements based on trends.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Define metrics in collaboration with QA, governance, and IT teams. Track technical indicators (uptime, error rates, and RTO and RPO adherence) alongside compliance-related ones (backup success, recovery validation, and audit trail completeness). Retain reliability data in tamper-evident storage for required retention periods. Review metrics periodically to drive improvement.

Implementation steps

1. Collect uptime and error metrics using Amazon CloudWatch and log retention policies.
2. Monitor backup success using AWS Backup Audit Manager.
3. Track recovery validation evidence in AWS Audit Manager.
4. Store metric histories in Amazon S3 with Object Lock for immutability.
5. Build dashboards using Quick for regulators and QA stakeholders.

Performance efficiency

The performance efficiency pillar focuses on using computing resources efficiently to meet system requirements and maintaining that efficiency as demand changes and technologies evolve. In life sciences, this includes optimizing for both high-performance computing workloads and time-sensitive clinical applications while maintaining regulatory adherence.

Focus areas

- Design principles
- Architecture selection
- Compute and hardware
- Data management
- Network and content delivery
- Process and culture

Design principles

- **Match compute resources to workload characteristics:** Select specialized hardware and architectures based on specific life sciences workload requirements—high-memory instances for genomic analysis, GPU acceleration for molecular modeling, and consistent low-latency compute for clinical applications. Benchmark and validate performance against scientific and regulatory requirements.
- **Design for variable and unpredictable workloads:** Implement auto-scaling, containerized workflows, and serverless architectures that dynamically adapt to the highly variable patterns of genomic sequencing, molecular modeling, and research cycles. Avoid over-provisioning by matching resource consumption to actual demand while maintaining performance during peak loads.
- **Balance performance optimization with regulatory adherence:** Apply risk-based validation frameworks that provide full qualification for high-risk clinical systems while using streamlined approaches for research environments. Integrate automated testing and continuous validation to maintain adherence without creating bottlenecks that slow innovation.
- **Implement intelligent data management strategies:** Deploy tiered storage architectures that balance high-performance access for active research with cost-efficient archival for long-term retention. Use data classification, lifecycle policies, and caching strategies to optimize both

performance and cost across diverse dataset types from basic records to terabyte-scale scientific data.

- **Optimize network performance for large-scale data movement:** Design network architectures that support secure, high-throughput transmission of large datasets while maintaining encryption controls. Implement intelligent traffic management, compression, and content delivery solutions for global research collaboration and multi-site clinical trials.
- **Foster cross-functional performance culture:** Establish collaborative reviews between IT and scientific teams that align technical metrics with scientific outcomes. Create integrated monitoring dashboards showing both system performance and metrics, enabling data-driven optimization decisions that preserve scientific rigor and patient safety.
- **Continuously monitor and optimize based on real-world usage:** Track comprehensive performance metrics including system latency, resource utilization, and clinical workflow impact. Use monitoring data to identify bottlenecks, validate performance against SLAs, and make evidence-based improvements to both research and clinical systems.

Architecture selection

LSPERF01: How do you select architectural patterns that accommodate the computing needs of genomic sequencing and molecular modeling?

When selecting architectures for genomic sequencing and molecular modeling, prioritize scalable compute patterns that efficiently handle variable workloads. Evaluate high-performance computing options with high memory-to-CPU ratios and GPU acceleration to optimize performance while maintaining data integrity for your scientific workflows.

LSPERF02: How do you manage diverse datasets ranging from basic records to terabyte-scale scientific datasets including sensitive data?

Implement tiered storage strategies using hot to cold storage tiers for cost-efficient management of large scientific datasets while keeping sensitive patient records in encrypted database instances. Deploy auto scaling data processing pipelines that efficiently handle both high-velocity instrument data and carefully controlled clinical information flows.

LSPERF03: How do you prioritize optimization between research computing and clinical application components in your architecture?

Analyze workload characteristics to identify optimization targets. Deploy high-performance computing resources for research pipelines requiring massive parallel processing, while prioritizing availability and consistent performance for clinical applications. Use dedicated and isolated environments with tailored service configurations to meet distinct research and clinical requirements.

LSPERF04: How do you approach performance considerations in your designs for long-term clinical trials and longitudinal studies?

For long-term clinical trials, design architectures that maintain consistent performance over extended timeframes. Implement data lifecycle policies that optimize storage costs while preserving query performance. Deploy versioned infrastructure-as-code to create reproducible environments that can be efficiently recreated years later for regulatory adherence.

Best practices

- LSPERF01-BP01 Design and benchmark computing architecture for genomic workloads to optimize cost-performance ratio
- LSPERF01-BP02 Specialized hardware selection and optimization for genomic and molecular workloads
- LSPERF01-BP03 Performance optimizations should validate data integrity
- LSPERF02-BP01 Data-aware storage tiering
- LSPERF02-BP02 Secure data separation by classification
- LSPERF02-BP03 Elastic data processing pipelines
- LSPERF03-BP01 Workload-specific performance analysis
- LSPERF03-BP02 Environment isolation by workload type
- LSPERF03-BP03 Tailored service configuration by use case
- LSPERF04-BP01 Performance consistency through clinical trial lifetime

LSPERF01-BP01 Design and benchmark computing architecture for genomic workloads to optimize cost-performance ratio

Focus on designing architectures that can dynamically scale to accommodate the highly variable workload patterns inherent in genomic sequencing and molecular modeling. Implement auto scaling capabilities, containerized workflows, and serverless processing pipelines that can efficiently handle both massive batch processing jobs and intermittent analysis requests without over-provisioning resources.

Desired outcome: A scalable, cost-efficient, and cloud-based architectural framework that provides optimal performance at every stage of genomic data processing and satisfies high-performance and variable computing needs.

Benefits of establishing this best practice:

- Control costs by matching resource consumption to actual demand.
- Accelerate time-to-insight by removing processing queues and bottlenecks.
- Enable researchers to run analyses without capacity planning.
- Support the growing volume of genomic data generated by next-generation sequencing technologies.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Design your architecture to dynamically scale with the variable workload patterns common in genomic sequencing and molecular modeling. These workloads often exhibit unpredictable patterns, with intense computational demands during certain phases followed by periods of minimal activity.

Configure Amazon EC2 Auto Scaling groups with appropriate scaling policies based on CPU utilization, memory usage, or custom metrics specific to your genomic workloads. Implement predictive scaling when possible, especially for recurring workload patterns in research cycles. Consider using Spot Instances with a fallback mechanism to optimize costs during large-scale genomic data processing.

Package genomic analysis tools and dependencies in containers using Amazon ECS or Amazon EKS to provide consistent runtime environments. Design containers to be stateless, storing

intermediate results in Amazon S3 or other durable storage. Use Amazon ECR to manage container images with version control for reproducible research.

Build event-driven architectures using AWS Lambda for sequence alignment, variant calling, and other discrete processing steps. Use AWS Step Functions to orchestrate complex genomic workflows, handling retries and error conditions automatically. Implement AWS Batch for compute-intensive tasks that exceed Lambda's execution limits.

Position compute resources close to data stores to minimize transfer times of large genomic datasets. Implement data partitioning strategies that align with your processing patterns. Consider using AWS HealthOmics for specialized genomic workloads, which provides purpose-built infrastructure that automatically provisions and scales the underlying infrastructure.

Implementation steps

1. Deploy containerized genomic workflows on Amazon ECS with Fargate.
2. Implement AWS Auto Scaling for variable sequencing workloads.
3. Use AWS Batch for cost-efficient molecular modeling jobs.
4. Create AWS Step Functions for serverless processing pipelines.
5. Use Amazon SageMaker AI for on-demand ML inference scaling.
6. Use AWS HealthOmics for storing genomics data and sequence stores.

Resources

Related guides, videos, and documentation:

- Work with EC2 Fleet

Related tools:

- Amazon Elastic Container Service
- AWS Fargate)
- AWS Auto Scaling
- AWS Batch
- AWS Step Functions
- Amazon Sagemaker

- AWS HealthOmics

LSPERF01-BP02 Specialized hardware selection and optimization for genomic and molecular workloads

Strategically select and optimize specialized computing resources based on specific computational requirements. Use high memory-to-CPU ratio instances for genome assembly, GPU-accelerated instances for molecular dynamics simulations, and FPGA solutions for specialized algorithms so that each workload component runs on its most performance efficient infrastructure.

Desired outcome: A performance-optimized hardware architecture that aligns computing resources with specific genomic and molecular workload requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When designing your computational architecture, carefully analyze your workload requirements and match them with appropriate specialized computing resources to optimize both performance and cost-efficiency. Begin by profiling your computational tasks to understand their specific memory, CPU, GPU, and I/O usage patterns. For memory-intensive operations such as genome assembly or large dataset processing, select high memory-to-CPU ratio instances like Amazon EC2 R6g or X2g instances that provide the necessary memory capacity without overprovisioning compute resources. When dealing with highly parallel workloads such as molecular dynamics simulations or machine learning training, use GPU-accelerated instances like Amazon EC2 P4d or G5 instances to dramatically improve processing speed. For specialized algorithms that benefit from custom hardware acceleration, consider FPGA-based solutions like Amazon EC2 Finstance family that allows for tailored hardware configurations.

Implement a continuous evaluation process to monitor resource utilization and performance metrics, adjusting instance types as workload characteristics evolve.

Use containerization to maintain workload portability across different compute resources, and implement workload-specific auto scaling policies rather than relying solely on generic CPU utilization metrics.

Implementation steps

1. Deploy Amazon EC2 High Memory instances for genome assembly.

2. Use EC2 P4d instances with NVIDIA A100 GPUs for simulations.
3. Implement AWS Inferentia for cost-effective ML inference.
4. Configure Amazon EC2 F2 instances for FPGA-accelerated tasks.
5. Use AWS Batch to route workloads to optimal instances.

Resources

Related tools:

- Amazon EC2 instance types
- AWS Batch

LSPERF01-BP03 Performance optimizations should validate data integrity

Maintain rigorous data integrity controls while pursuing performance optimizations. Implement checksums, audit trails, and validation steps within high-performance workflows, which assists to avoid compromising scientific accuracy and reproducibility by performance-focused architectural decisions, particularly for regulated or clinically-relevant research.

Desired outcome: Implement a balanced system architecture that maintains rigorous data integrity controls while achieving optimal performance, verifying that scientific accuracy and reproducibility for regulated and clinically relevant research.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Working backward from your research regulatory requirements, balance performance with data integrity to achieve scientific accuracy and regulatory adherence.

Implement validation at each processing stage using cryptographic checksums (MD5/SHA-256) to verify data remains unaltered during transfers, storing these checksums for future validation.

Establish comprehensive audit trails documenting transformations with timestamps, parameters, and identity information—essential for regulated research.

Design your architecture with strategically positioned automated validation steps that detect anomalies without manual intervention, maximizing integrity assurance while minimizing performance impact.

For high-throughput workflows, deploy parallel validation processes that run concurrently rather than sequentially.

Use Amazon S3 Object Lock for immutable storage, AWS CloudTrail for audit logging, and AWS Config for monitoring.

With regulated data, incorporate digital signatures and version control to maintain unalterable provenance records, improving both speed and trustworthiness throughout your research workflows.

Implementation steps

1. Implement AWS KMS for encryption of sensitive research data.
2. Configure Amazon S3 checksums for data integrity validation.
3. Deploy AWS CloudTrail for comprehensive audit logging.
4. Use Amazon EventBridge to monitor validation workflows.
5. Implement AWS Config for adherence to research standards.

Resources

Related tools:

- AWS KMS
- Amazon S3
- AWS CloudTrail
- Amazon EventBridge
- AWS Config

LSPERF02-BP01 Data-aware storage tiering

Implement intelligent storage tiering strategies that align storage performance characteristics with data access patterns. Place frequently accessed reference data on high-performance tiers,

move aging research data to cost-optimized tiers, and archive completed study data to deep storage, while maintaining appropriate encryption and access controls based on data sensitivity classification.

Desired outcome: Implement a comprehensive, data-driven storage management system that automatically places data on the most appropriate storage tier based on access patterns, age, and sensitivity classification.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Working backward from your research data needs, design an intelligent storage architecture that aligns with actual access patterns to optimize both performance and cost.

Start by classifying data based on access frequency and sensitivity. Deploy high-performance options like Amazon EFS with provisioned throughput or Amazon FSx for Lustre for frequently accessed data requiring low-latency retrieval.

As access patterns decrease, implement automated lifecycle policies that transition data to cost-optimized tiers such as S3 Standard-Infrequent Access or S3 Intelligent-Tiering.

For rarely accessed audit data, use S3 Glacier Deep Archive to minimize costs while maintaining accessibility.

Maintain consistent encryption and access controls across each tier, with appropriate keys and permissions based on sensitivity classification. Use S3 Analytics to continuously monitor access patterns and refine tiering policies, while implementing object tagging to preserve context across tiers.

For domain-specific optimization, consider purpose-built solutions like AWS HealthOmics, AWS HealthImaging, and AWS HealthLake.

Implementation steps

1. Configure S3 Intelligent-Tiering for scientific dataset storage.
2. Store sensitive data in encrypted Amazon RDS instances.
3. Implement Amazon S3 Lifecycle policies for data archival.
4. Implement AWS Lambda for auto scaling data processing.

5. Consider purpose-built storage such as HealthLake, HealthOmics, and HealthImaging.

LSPERF02-BP02 Secure data separation by classification

Establish clear boundaries between different data types based on sensitivity and regulatory requirements. Maintain sensitive patient records in encrypted, highly-available database instances with comprehensive audit capabilities, while allowing broader access to de-identified research datasets through separate storage mechanisms with appropriate controls for scientific collaboration.

Desired outcome: Establish a comprehensive data separation framework that clearly separates different data types based on sensitivity levels and regulatory requirements, providing appropriate storage, access controls, and availability for each category.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Design your data architecture with clearly defined boundaries that separate information based on sensitivity levels and regulatory requirements. Implement a comprehensive data classification system that identifies and tags data according to sensitivity, regulatory scope, and access requirements. For sensitive patient records and protected health information (PHI), use encrypted, highly-available database services like Amazon RDS with multi-AZ deployments and encryption at rest using AWS KMS customer-managed keys.

Enable comprehensive audit logging through services like AWS CloudTrail and Amazon RDS Enhanced Monitoring to track access and modifications to sensitive data, improving adherence to regulations like HIPAA or GDPR.

For de-identified research datasets intended for broader scientific collaboration, establish separate storage mechanisms using services like Amazon S3 with appropriate bucket policies and access controls that facilitate controlled sharing while blocking unauthorized access. Implement robust de-identification processes that follow established standards like HIPAA Safe Harbor or Expert Determination methods before moving data to these collaborative environments.

Use AWS Identity and Access Management (IAM) with attribute-based access control (ABAC) to create fine-grained permissions based on data classification tags, verifying that users can only access data appropriate to their role and research needs. Consider implementing additional

safeguards like VPC endpoints and network isolation to provide network-level separation between sensitive and de-identified data environments.

Implementation steps

1. Implement AWS Organizations for multi-account data separation.
2. Configure S3 bucket policies for sensitivity-based isolation.
3. Use AWS IAM for fine-grained data access controls.
4. Deploy AWS Lake Formation for centralized data governance.
5. Implement AWS KMS with Customer Managed keys for each data category.

LSPERF02-BP03 Elastic data processing pipelines

Design data processing architectures that automatically scale to accommodate both predictable and unexpected processing demands. Implement event-driven pipelines that efficiently handle high-velocity instrument data streams while maintaining strict controls over clinical data flows, providing consistent performance during peak research periods without compromising security or regulatory requirements.

Desired outcome: Implement an adaptive data processing architecture that automatically scales to meet varying workload demands while maintaining strict security and audit controls for each data type, particularly clinical information.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To effectively manage high-velocity data processing, implement event-driven architectures that respond to data events in real-time, enabling efficient handling of instrument data streams. Maintain separate processing flows for different data types, applying appropriate controls to clinical data while optimizing research workloads.

Use managed AWS services like AWS Lambda, Amazon SQS, and Amazon Kinesis that automatically scale to match processing needs without manual intervention. Implement throttling mechanisms to protect downstream systems from being overwhelmed during peak research periods. Set up comprehensive monitoring to track performance and automatically adjust resources based on observed patterns, while maintaining regulatory guardrails to verify that automatic scaling doesn't compromise security requirements.

Regularly conduct load testing to verify that your architecture can scale as expected under stress, validating the system's ability to handle varying data volumes while maintaining performance and regulatory standards.

Implementation steps

1. Deploy Amazon Kinesis for high-velocity data streaming.
2. Implement AWS Lambda for event-driven data processing and auto scaling data processing.
3. Use AWS Step Functions to orchestrate workflows.
4. Configure Amazon MSK for reliable clinical data pipelines.
5. Implement AWS Auto Scaling for predictable research peaks.

Resources

Related tools:

- Amazon Kinesis
- AWS Lambda
- AWS Step Functions
- Amazon MSK (Managed Streaming for Apache Kafka)
- AWS Auto Scaling

LSPERF03-BP01 Workload-specific performance analysis

Conduct systematic analysis of workload characteristics to identify distinct performance requirements and optimization opportunities. Profile computational patterns, data access behaviors, and resource utilization across different workflow stages to guide targeted optimization efforts, allocating resources on empirical performance needs rather than assumptions.

Desired outcome: Implement systematic workload profiling to identify optimization opportunities and guide resource allocation based on empirical performance data.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To optimize genomics or life sciences workloads effectively, begin with systematic analysis using AWS observability tools. Deploy Amazon CloudWatch with custom metrics and dashboards to establish baseline performance across your architecture. Configure detailed monitoring through CloudWatch Container Insights for containerized workloads or CloudWatch agent for EC2 instances to capture CPU, memory, disk, and network utilization patterns.

Profile computational patterns by implementing AWS X-Ray tracing to understand request flows and component interactions throughout your application stack. For ML-based workloads, use Amazon SageMaker AI Profiler to analyze model training and inference performance characteristics. These tools help identify computational bottlenecks and guide decisions about instance types, model optimization techniques, or architectural changes that could improve performance.

Analyze data access behaviors using CloudWatch metrics for services like Amazon S3, Amazon DynamoDB, and Amazon RDS. Implement S3 Storage Lens to gain visibility into object storage patterns and optimize data placement strategies. For database workloads, use RDS Performance Insights or DynamoDB CloudWatch metrics to identify query patterns that might benefit from indexing or caching strategies with Amazon ElastiCache.

Map resource utilization across different workflow stages by correlating metrics from multiple sources in CloudWatch dashboards. This correlation helps identify how resource requirements fluctuate throughout your workload lifecycle and where targeted optimizations would deliver the greatest impact. Use AWS Cost Explorer, cost allocation tags, and CloudWatch dashboards together to understand the cost implications of different resource allocation strategies.

Implement a data-driven optimization approach using AWS Compute Optimizer for right-sizing recommendations and AWS Well-Architected Tool to evaluate your architecture against best practices. Test optimization hypotheses using A/B testing methodologies with AWS AppConfig before full deployment. Document findings and optimization decisions in AWS Systems Manager documents to build organizational knowledge around performance optimization practices specific to your Genomics or Lifesciences workloads.

Implementation steps

1. Deploy CloudWatch dashboards for workload monitoring.
2. Use SageMaker AI Profiler to analyze model performance.

3. Implement X-Ray tracing for request flow analysis.
4. Create S3 Storage Lens dashboards for data patterns.
5. Optimize with AWS Compute Optimizer recommendations.

LSPERF03-BP02 Environment isolation by workload type

Implement clear separation between research and clinical environments based on their fundamentally different requirements. Establish a dedicated infrastructure for computationally-intensive research pipelines with burst capacity for parallel processing, while maintaining separate, highly-available environments for clinical applications where consistency and reliability are paramount.

Desired outcome: Create distinct infrastructure environments optimized for the unique requirements of research computing and clinical applications.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing clear separation between research and clinical environments is essential for organizations working with generative AI in healthcare and life sciences. Begin by establishing distinct AWS accounts for each environment using AWS Organizations with service control policies (SCPs) that enforce appropriate guardrails. This separation creates natural boundaries for access controls, resource allocation, and regulatory requirements while still enabling cross-account data sharing when necessary through services like AWS RAM.

For research environments, prioritize flexibility and computational power by implementing Amazon SageMaker AI with its comprehensive ML development capabilities. Configure auto scaling compute resources using Amazon EC2 Auto Scaling groups with GPU-accelerated instances like P4d or G5g to support computationally intensive workloads. Implement AWS Batch for efficiently managing parallel processing jobs with spot instances to optimize costs during model training and experimentation phases. This approach provides researchers with the burst capacity needed for iterative development while maintaining cost efficiency.

For clinical environments, focus on high availability and consistency by deploying infrastructure across multiple Availability Zones using AWS CloudFormation or AWS CDK with immutable patterns. Implement Amazon RDS multi-AZ deployments for database reliability and Amazon

ElastiCache for consistent performance. Configure detailed monitoring with Amazon CloudWatch and AWS X-Ray for predictable performance characteristics critical for clinical applications. Implement AWS Config rules to enforce configuration adherence with regulatory requirements like HIPAA or GxP.

Establish controlled pathways for promoting validated models from research to clinical environments using AWS CodePipeline with approval gates and validation tests. Store model artifacts in Amazon S3 with versioning enabled and implement AWS Lambda functions to validate model metadata before clinical deployment. Use Service Catalog to create standardized, pre-approved deployment patterns that clinical teams can use without compromising governance requirements.

Implementation steps

1. Create account separation using AWS Organizations.
2. Deploy research workloads on SageMaker AI with GPU instances.
3. Build clinical systems with Multi-AZ RDS deployments.
4. Implement CodePipeline for controlled model promotion.
5. Configure CloudWatch dashboards for environment monitoring.

LSPERF03-BP03 Tailored service configuration by use case

Customize infrastructure and service configurations to align with the specific requirements of each workload category. Fine-tune compute, storage, and networking parameters for research environments to maximize throughput and processing efficiency, while configuring clinical environments with emphasis on consistent performance, redundancy, and predictable behavior under each condition.

Desired outcome: Implement tailored infrastructure configurations that precisely match the unique requirements of research and clinical workloads.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Customizing infrastructure and service configurations for different workload categories is essential for optimizing both performance and cost efficiency in generative AI deployments. Begin by

conducting a detailed workload assessment using AWS Well-Architected Tool with the Generative AI Lens to identify the specific requirements and characteristics of each workload category. This assessment establishes clear performance targets, reliability requirements, and cost constraints that will guide your configuration decisions.

For research environments focused on model development and experimentation, prioritize compute flexibility and processing efficiency. Configure Amazon EC2 instances with the latest generation of accelerators like AWS Trainium for training workloads. Implement Amazon FSx for Lustre configured with high throughput capabilities to support efficient data processing during training. Use Amazon SageMaker AI with managed spot training to reduce costs for non-time-sensitive workloads while maintaining the ability to scale compute resources during intensive experimentation phases.

For clinical or production environments where consistent performance is critical, implement multi-AZ deployments across infrastructure components using AWS CloudFormation or AWS CDK. Configure Amazon RDS with multi-AZ deployments and provisioned IOPS for consistent database performance. Implement Amazon ElastiCache with reserved nodes to provide stable, predictable caching performance. Deploy inference endpoints using Amazon SageMaker AI with auto scaling configured based on predictable traffic patterns rather than reactive scaling, providing consistent latency even during traffic variations.

Tailor networking configurations to each environment's specific needs using advanced VPC features. For research environments, implement AWS Transit Gateway with increased bandwidth allocations to support large data transfers. For clinical environments, implement AWS Global Accelerator to provide consistent network performance and AWS Shield Advanced for enhanced protection against availability-impacting events.

Implement comprehensive monitoring tailored to each environment using Amazon CloudWatch with custom metrics and dashboards. For research environments, focus monitoring on resource utilization and throughput metrics. For clinical environments, prioritize monitoring of latency percentiles, error rates, and availability metrics with automated alerting through Amazon SNS when performance deviates from established baselines.

Implementation steps

1. Assess workloads using AWS Well-Architected Tool.
2. Deploy research models on SageMaker AI with spot instances.
3. Configure clinical systems with Multi-AZ architecture.

4. Implement FSx for Lustre for high-throughput processing.
5. Create CloudWatch dashboards for environment monitoring.

LSPERF04-BP01 Performance consistency through clinical trial lifetime

Design architectures with long-term performance stability as a foundational principle, improving the consistency of system behavior across the multi-year or multi-decade lifespan of clinical trials. Implement forward-compatible data models, establish performance baselines with comprehensive monitoring, and create governance processes for managing technology transitions without disrupting ongoing studies or compromising data integrity.

Desired outcome: Establish an enduring architectural framework that maintains consistent performance and data integrity across multi-year or multi-decade clinical trial lifecycles.

Level of risk exposed if this best practice is not established: High

Implementation guidance

For long-term clinical trials, design your architectures with a focus on maintaining consistent performance throughout extended timeframes. Begin by implementing comprehensive data lifecycle policies that strategically balance storage costs with query performance requirements. As data ages, consider transitioning it through storage tiers—from high-performance storage for active analysis to more cost-effective archival solutions for data that requires less frequent access but must remain retrievable for auditing purposes.

Deploy infrastructure components using versioned infrastructure as code (IaC) templates. This approach keeps your environments reproducible even years later when regulatory audits may require you to demonstrate the exact computational conditions under which analyses were performed. Document dependencies thoroughly, including specific library versions, container images, and configuration parameters. Consider creating immutable snapshots of complete environments at critical milestones.

Implement automated testing frameworks that can validate the consistency of results across environment recreations. This verifies that your infrastructure remains capable of reproducing the same analytical outcomes over time, which is essential for scientific validity and regulatory adherence. Additionally, establish clear governance processes for managing changes to the environment, which assists you in properly tracking, approving, and validating modifications against baseline performance metrics.

Implementation steps

1. Deploy AWS CloudFormation templates for reproducible trials.
2. Store clinical data in Amazon S3 with intelligent tiering.
3. Monitor performance with Amazon CloudWatch custom dashboards.
4. Implement AWS Config rules for governance audits.
5. Document system changes in AWS Systems Manager.

Compute and hardware

LSPERF05: How do you choose compute for life science workloads like molecular modeling, genomics, bioinformatics, and AI drug discovery?

Explore strategies for matching specialized life sciences workloads with optimal computational resources to enhance performance and efficiency across molecular modeling, genomics, and AI-driven research applications.

LSPERF06: What process do you follow to evaluate specialized hardware accelerators for molecular dynamics simulations or genomic analysis?

Implement a systematic approach to evaluating specialized hardware accelerators for optimal performance in molecular dynamics simulations and genomic analysis.

LSPERF07: How do you balance high-performance computing needs with GxP requirements?

Life sciences organizations face the challenge of balancing computational power with GxP adherence. Explore strategies for creating validated computing environments that meet both scientific needs and regulatory requirements. Topics include infrastructure design, validation approaches, data integrity, and change management that enables innovation within frameworks. Learn practical methods for optimizing performance while successfully navigating GxP regulations in life sciences applications.

LSPERF08: What metrics do you monitor to verify if compute resources meet the performan ce demands of time-sensitive clinical applications?

Key performance metrics and monitoring protocols for verifying that compute resources consistently meet the demands of time-critical clinical applications. This discussion covers essential technical indicators, threshold management, and proactive monitoring strategies specifically tailored for clinical environments where application performance directly impacts patient care, diagnostic accuracy, and treatment delivery.

Best practices

- LSPERF05-BP01 Specialized hardware matching
- LSPERF05-BP02 Establish a tiered infrastructure strategy
- LSPERF05-BP03 Implement a comprehensive system optimization strategy
- LSPERF06-BP01 Run comprehensive, benchmark-driven assessments
- LSPERF06-BP02 Perform a total cost of ownership analysis
- LSPERF06-BP03 Perform an environment compatibility validation
- LSPERF07-BP01 Use a risk-based validation framework
- LSPERF07-BP02 Design a compliant-by-design infrastructure
- LSPERF07-BP03 Develop an agile change management process
- LSPERF08-BP01 Implement holistic system performance monitoring beyond traditional latency metrics
- LSPERF08-BP02 Track resource utilization with clinical context
- LSPERF08-BP03 Implement clinical system monitoring with workflow validation and impact analysis

LSPERF05-BP01 Specialized hardware matching

Deploy purpose-built compute configurations optimized for specific workload types. Use GPUs for molecular dynamics and AI drug discovery, high-memory systems (4-8GB RAM per core) for genomics assembly, and high-IOPS storage architectures for sequencing data processing.

Desired outcome: Implement specialized computing environments (GPU-accelerated, high-memory, and high-IOPS storage) that enhance performance and cost-efficiency across molecular dynamics, AI drug discovery, genomics assembly, and sequencing data processing workloads.

Common anti-patterns:

- Deploying GPU clusters without sufficient memory bandwidth, creating processing bottlenecks.
- Provisioning high-memory systems with inadequate I/O capabilities, causing data starvation.
- Selecting hardware solely for compute power while ignoring interconnect speeds critical for HPC workloads.
- Standardizing on single OS configurations incompatible with specialized bioinformatics tools.
- Underestimating storage IOPS requirements for high-throughput sequencing pipelines.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Match purpose-built compute configurations to specific life science workload types. This provides maximum computational throughput while avoiding overprovisioning expensive resources.

Accelerate critical healthcare advancements with purpose-built computational infrastructure. Strategic hardware deployment removes processing bottlenecks, enabling researchers and clinicians to achieve breakthrough insights and advance patient care more rapidly.

Select hardware that can adapt to evolving research requirements. Life sciences workloads evolve rapidly, requiring infrastructure that can scale and adapt to new computational methods.

Provide appropriate resource allocation for cross-functional teams. Consistent compute environments provide scientific reproducibility and facilitate collaboration between researchers, clinicians, and data scientists.

Implementation steps

1. Profile and benchmark workloads to establish baselines and identify consumption patterns.
2. Categorize workloads by resource needs and document performance requirements.
3. Deploy specialized hardware with GPUs, high-memory systems, and high-IOPS storage.
4. Monitor utilization, measure metrics, and review hardware effectiveness regularly.
5. Establish standards, approval processes, and refresh strategies for hardware.

LSPERF05-BP02 Establish a tiered infrastructure strategy

Implement a layered approach organizing systems by criticality, optimizing resources, enhancing resilience, and aligning technology investments with business priorities for more effective infrastructure management.

Desired outcome: A resilient, cost-effective infrastructure with optimized resource allocation that aligns with business criticality, enhances scalability, and provides appropriate protection levels for different system tiers.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establish a cohesive cloud-native architecture. This unified approach creates a seamless computational fabric that uses the strengths of cloud environments while appearing as a single resource pool.

Design a strategic resource distribution across cloud services and regions. This approach provides reliable core services while enabling elasticity to handle variable computational demands cost-effectively.

Create unified data access patterns within the cloud environment. Effective data management reduces duplicated storage costs and provides computational tasks with access required datasets regardless of execution location.

Implement workflow tools that can intelligently distribute tasks across the entire resource pool. Smart orchestration verifies that workloads run in the optimal environment based on current conditions and requirements.

Develop comprehensive management practices for the cloud environment. Unified operations assist the cloud architecture to function efficiently and meet governance requirements across deployment contexts.

Implementation steps

1. Assess workloads by analyzing requirements and identifying cloud-suitable applications.
2. Implement cloud bursting with automatic scaling and efficient data synchronization.
3. Deploy data management with consistent access and automated staging mechanisms.
4. Configure orchestration tools with decision rules and monitoring feedback loops.

5. Create unified governance with standardized monitoring, security, and cost tracking.

LSPERF05-BP03 Implement a comprehensive system optimization strategy

Establish comprehensive workload profiling, benchmarking, and monitoring practices covering each computational aspect (compute, memory, storage, and network), enabling data-driven decisions for resource allocation and identifying bottlenecks before they impact research timelines.

Desired outcome: Implement comprehensive workload monitoring across compute, memory, storage, and network resources to enable data-driven optimization, proactively identify bottlenecks, and provide efficient resource allocation for timely research delivery.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop a systematic approach to measure each aspect of computational workload performance. Comprehensive profiling creates visibility across the entire resource spectrum and provides baseline metrics for future comparisons.

Execute controlled performance testing to quantify resource requirements accurately. Structured benchmarking provides objective data for making architecture decisions and validating vendor performance claims.

Implement real-time performance tracking to detect emerging issues before they impact research. Proactive monitoring blockes unexpected delays in critical research pipelines through early identification of resource constraints.

Base infrastructure decisions on quantified performance metrics rather than assumptions. Data-driven allocation deploys resources where they will have the greatest impact on research outcomes.

Build organizational capabilities around performance optimization and bottleneck identification. A performance-aware culture continuously improves computational efficiency across research activities.

Implementation steps

1. Establish multi-dimensional profiling tools with standardized metrics for applications.
2. Implement benchmark protocols with standardized testing and comparative analysis.

3. Configure monitoring with alerts, trend analysis, and resource visibility dashboards.
4. Establish allocation processes with impact-based priorities and forecasting models.
5. Build capabilities through staff training and cross-team optimization sharing.

LSPERF06-BP01 Run comprehensive, benchmark-driven assessments

Execute standardized application-specific benchmarks using representative datasets across multiple hardware solutions (GPUs, FPGAs, ASICs), measuring not only raw performance but also performance-per-watt, scaling efficiency, and cost-effectiveness with your actual production workloads in molecular dynamics (GROMACS, NAMD, AMBER) and genomics (BWA-MEM, GATK, Minimap2).

Desired outcome: Conduct comprehensive benchmarks of scientific applications across diverse hardware solutions to measure performance, energy efficiency, scaling, and cost-effectiveness for optimal computational research infrastructure selection.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish Standardized Testing Methodology: Develop rigorously controlled benchmark protocols specific to life sciences applications. Standardized methodologies facilitate consistent, reproducible results that can be compared across different hardware architectures and over time.

Evaluate Comprehensive Performance Metrics: Measure multiple dimensions of hardware performance beyond raw speed. Comprehensive metrics provide a holistic view of hardware value, including energy efficiency, cost factors, and scaling characteristics.

Compare Across Hardware Accelerator Types: Conduct benchmarks across diverse specialized computing architectures. Systematic comparison across acceleration technologies reveals which hardware best matches specific application characteristics.

Validate with Production Workloads: Use actual production datasets and workflows for benchmark validation. Real-world validations make sure benchmark results translate to meaningful improvements in day-to-day research operations.

Create Decision Support Framework: Develop a structured approach to translate benchmark data into procurement decisions. A formal framework validates objective evaluation of hardware options based on quantified performance metrics aligned with research priorities.

Implementation steps

1. Define benchmark frameworks with standardized tests for molecular dynamics and genomics.
2. Implement measurements for performance-per-watt metrics and total cost calculations.
3. Execute testing across GPU, FPGA, and ASIC solutions for bioinformatics.
4. Perform validation using actual simulations to verify benchmark predictions.
5. Establish hardware selection using scoring matrices and ROI models.

LSPERF06-BP02 Perform a total cost of ownership analysis

Calculate complete TCO incorporating hardware acquisition costs, software licensing, power consumption, cooling requirements, physical footprint, specialized expertise needs, and expected hardware lifespan, while quantifying research productivity improvements to determine true value beyond initial purchase price.

Desired outcome: Develop comprehensive TCO analysis including direct and indirect costs while quantifying research productivity gains to reveal true value and inform strategic technology investments beyond purchase price alone.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Develop financial models that capture direct and indirect infrastructure expenses. Complete cost visibility blocks unexpected budget impacts and enables accurate comparison between technology options.

Evaluate costs across the entire useful lifespan of research infrastructure. Lifecycle analysis reveals the true long-term financial impact of technology decisions beyond initial acquisition costs.

Measure how infrastructure investments translate to research output improvements. Productivity quantification enables value-based decisions by connecting technology costs to scientific outcomes.

Integrate cost and benefit data to calculate true return on infrastructure investment. Holistic value analysis balances financial considerations with research advancement to optimize technology investment decisions.

Monitor ongoing value delivery from infrastructure investments after deployment. Continuous tracking verifies that technologies deliver expected benefits and informs future procurement decisions with actual performance data.

Implementation steps

1. Document costs including direct expenses, operational costs, and staffing requirements.
2. Develop financial models with depreciation schedules and projected maintenance costs.
3. Establish research metrics with productivity indicators and improvement forecasts.
4. Create value assessments with cost-per-output calculations and comparison frameworks.
5. Implement value tracking with ongoing measurement and periodic TCO reassessment.

LSPERF06-BP03 Perform an environment compatibility validation

Evaluate hardware accelerators within your existing computational infrastructure, verifying integration with workload managers, container technologies, data transfer mechanisms, and storage systems while assessing software support maturity, driver stability, and vendor commitment to the scientific computing community.

Desired outcome: Identify hardware accelerators compatible with current systems, providing support for workloads, containers, and data needs while evaluating software maturity and vendor reliability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Assess how accelerators interact with existing infrastructure components. Comprehensive compatibility testing blocks deployment delays and verifies that accelerators function properly within the broader technology environment.

Assess the development status of libraries and tools supporting the accelerator. A mature software solution verifies that researchers can effectively utilize accelerator capabilities without extensive custom development.

Determine the reliability of driver implementations and vendor support responsiveness. Stable operation is critical for production research environments where downtime directly impacts scientific progress.

Evaluate the accelerator provider's dedication to research and scientific applications. Vendor commitment continues the development of features relevant to computational life sciences.

Identify potential challenges and mitigations across technical, operational, and strategic dimensions. Comprehensive risk assessment provides for informed decision-making that accounts for both opportunities and potential complications.

Implementation steps

1. Conduct testing to validate compatibility with existing systems and technologies.
2. Analyze software support through libraries, compiler capabilities, and documentation.
3. Evaluate readiness through load testing and peer institution consultation.
4. Assess vendor alignment by reviewing roadmaps and life sciences investments.
5. Develop risk framework with technical evaluations and contingency plans.

LSPERF07-BP01 Use a risk-based validation framework

Use risk-based validation with full validation (Installation, Operation, and Performance Qualification) for high-risk systems affecting patient safety, streamlined methods for lower-risk environments. Focus resources where critical and use automated testing to maintain regulatory adherence without slowing innovation. Revalidate based on risk assessment, not after every change.

Desired outcome: Achieve regulatory adherence while optimizing resource allocation through a balanced validation approach that maintains patient safety in critical systems yet maintains operational efficiency. Enable innovation in research environments through appropriate validation levels, resulting in fewer bottlenecks and greater productivity without compromising quality or standards.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop a framework for classifying computational systems based on GxP impact and patient safety considerations. Risk-based stratification maintains an appropriate level of validation intensity and blocks over-validation of research systems.

Design validation approaches proportional to the established risk categories. Tiered protocols verify that critical systems receive comprehensive validation while avoiding unnecessary documentation burden for research-focused systems.

Implement ongoing verification practices integrated with development and operational workflows. Continuous validation approaches maintain regulatory adherence while supporting agility and innovation in computational environments.

Establish criteria for when systems require revalidation based on change impact rather than arbitrary rules. Intelligent triggers block unnecessary revalidation while providing proper scrutiny of significant changes.

Maintain regulatory adherence while preserving computational agility for research innovation. A balanced approach blocks validation processes from becoming bottlenecks while maintaining patient safety and data integrity in regulated contexts.

Implementation steps

1. Create risk classification framework with documented regulatory justification.
2. Develop validation strategies appropriate to system risk levels.
3. Deploy automated testing integrated with development pipelines.
4. Create change assessment framework with risk-based review process.
5. Optimize validation efficiency while maintaining regulatory adherence.

LSPERF07-BP02 Design a compliant-by-design infrastructure

Design a computing architecture that addresses both performance and regulatory requirements through technical controls built directly into the infrastructure. Implement segregated computing environments with appropriate data controls between GxP and non-GxP workloads, deploy immutable infrastructure techniques that enhance both security and compliance, and use containerization with validated base images to accelerate deployment while maintaining regulatory integrity. Establish infrastructure templates with pre-validated components, automated audit trail generation, and built-in data integrity mechanisms that minimize the performance overhead typically associated with compliance retrofitting, while verifying that computational workloads execute in appropriately validated environments based on their regulatory classification.

Desired outcome: Create a computing infrastructure that balances performance and adherence through built-in controls, proper isolation, and automated validation, enabling efficient scientific work without regulatory burden.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Design distinct computing environments with appropriate boundaries for different regulatory contexts. Isolation allows tailored controls while avoiding regulatory requirements from limiting innovation in research-focused areas.

Use infrastructure as code and immutable deployment patterns that enhance both security and regulatory adherence. Immutable approaches block configuration drift while creating inherently more auditable and consistent environments.

Create a library of pre-validated infrastructure components and patterns. Pre-validated components accelerate deployment while maintaining compliance-aligned assurance through already-verified building blocks.

Integrate mechanisms directly into the infrastructure rather than as external processes. Embedded controls reduce overhead while blocking continuous compliance-aligned assurance without manual intervention.

Design infrastructure that maintains computational efficiency while meeting regulatory requirements. Performance-preserving approaches stop regulatory controls from becoming computational bottlenecks.

Implementation steps

1. Design segregated domains with controlled data transfer between boundaries.
2. Implement immutable deployment with versioned infrastructure processes.
3. Build validated component library with pre-approved templates.
4. Deploy automated built-in audit trails and verification.
5. Optimize balance between controls and performance requirements.

LSPERF07-BP03 Develop an agile change management process

Develop a specialized change management process that enables rapid iteration for scientific computing while maintaining GxP adherence through a combination of procedural and technical safeguards. Implement digital solutions for documentation-as-code that creates machine-readable artifacts alongside software development, establish predefined patterns for common computational workflows, and deploy automated monitoring tools that provide real-time verification without manual intervention. Create cross-functional teams combining computational scientists, IT specialists and experts who collaboratively define fit-for-purpose validation approaches that protect data integrity and regulatory requirements while enabling the performance optimization and innovation velocity required by modern life sciences research.

Desired outcome: Establish efficient change management that enables fast scientific progress while maintaining adherence through automated documentation, predefined patterns, and cross-functional collaboration.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Design change processes specifically optimized for computational science workflows. Specialized frameworks balance requirements with the need for scientific agility and rapid iteration.

Integrate documentation directly into development workflows using automated approaches. Documentation-as-code creates machine-readable artifacts that evolve alongside computational systems.

Create standardized approaches for common scientific computing challenges. Pre-defined patterns accelerate implementation while improving the consistency of regulatory and GxP approaches across research platforms.

Implement automated monitoring that continuously validates the state of systems. Real-time verification replaces periodic manual audits with constant assurance of compliant operations.

Establish collaborative teams that combine scientific, technical, and compliance-aligned expertise. Integrated teams develop balanced approaches that satisfy stakeholders while optimizing for research outcomes.

Implementation steps

1. Create streamlined change management tailored for research computing needs.

2. Deploy automated documentation with version-controlled artifacts.
3. Develop reusable patterns for common computational processes.
4. Implement monitoring tools with alerts for potential issues.
5. Create collaborative governance between scientists, IT, and auditing teams.

LSPERF08-BP01 Implement holistic system performance monitoring beyond traditional latency metrics

Implement a multi-layered latency monitoring system tracking the entire clinical workflow times, not just system metrics. Capture 95th and 99th percentiles. Set medical SLAs (stroke imaging <3min, labs <30sec). Use distributed tracing across HL7/DICOM/FHIR interfaces to identify bottlenecks.

Level of risk exposed if this best practice is not established: High

Desired outcome: Clinically-relevant performance visibility enabling faster detection of latency issues before they impact patient care, with clear attribution of delays across interconnected systems and measurable adherence to workflow-specific medical time requirements.

Implementation guidance

Implement holistic performance tracking across complete clinical workflows. Comprehensive monitoring reveals the true patient impact of system performance beyond isolated technical metrics.

Define performance indicators that directly relate to patient care implications. Clinically-aligned metrics validate monitoring focuses on measurements that matter for healthcare outcomes.

Implement Medically-Relevant Service Level Agreements: Develop performance targets derived from actual clinical requirements. Medical SLAs make sure technical performance goals align with true healthcare operational needs.

Deploy Healthcare-Specific Instrumentation: Implement monitoring technology optimized for healthcare data exchange standards. Specialized instrumentation provides visibility into healthcare-unique data flows that generic tools may miss.

Establish Clinical Impact Alerting System: Create notification mechanisms based on patient care thresholds. Clinically-focused alerting facilitates rapid response to performance issues that could affect medical decision making.

Implementation steps

1. Deploy workflow monitoring across entire clinical process with persistent identifiers.
2. Define performance indicators with statistical tracking and baseline measurements.
3. Create performance standards with thresholds for critical workflows.
4. Implement monitoring for healthcare-specific protocols and interfaces.
5. Deploy progressive alerting for transactions approaching clinical thresholds.

LSPERF08-BP02 Track resource utilization with clinical context

Track compute metrics tied to clinical workflows and patient volumes for predictive capacity planning. Monitor specialized clinical accelerators like GPUs for imaging. Analyze resource usage patterns by department and procedure to optimize allocation while providing emergency capacity. Implement context-aware anomaly detection that distinguishes normal clinical activity spikes from true issues, with alerts weighted by clinical importance.

Desired outcome: Implement comprehensive clinical-aware resource monitoring that provides predictive insights, optimizes capacity allocation based on workflow patterns, and delivers intelligent alerting prioritized by patient care impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establish relationships between infrastructure performance and specific healthcare workflows. Correlation analysis reveals how technical resource consumption directly relates to patient care activities.

Implement specialized tracking for purpose-built healthcare computing hardware. Targeted monitoring optimizes utilization of clinical-specific accelerators critical for advanced medical applications.

Develop detailed understanding of resource utilization across different healthcare dimensions. Multidimensional analysis enables optimized resource allocation aligned with actual clinical operations.

Implement intelligent monitoring that understands expected clinical activity patterns. Context-aware anomaly detection blocks false alarms while verifying that real issues receive prompt attention.

Develop notification systems that prioritize based on patient care impact. Clinical weighting provides attention to the most critical healthcare services when resource constraints occur.

Implementation steps

1. Implement contextual monitoring with clinical workflow identifiers.
2. Deploy specialized hardware tracking for imaging and genomic workflows.
3. Create resource consumption models by department and procedure type.
4. Configure pattern recognition for normal clinical activity variations.
5. Implement priority alerting based on clinical criticality and patient impact.

LSPERF08-BP03 Implement clinical system monitoring with workflow validation and impact analysis

Implement synthetic transactions simulating clinical workflows while monitoring data integrity throughout your infrastructure. Regularly test recovery capabilities against clinical RTOs. Develop dashboards that translate technical metrics into patient impact metrics with proper prioritization and regulatory adherence. Document historical performance data to demonstrate system reliability and support continuous improvement efforts.

Desired outcome: Establish a monitoring framework with synthetic clinical workflow testing, recovery validation, patient-impact dashboards, and historical performance tracking to improve system reliability and regulatory adherence.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Deploy automated testing that simulates real healthcare user activities. Synthetic transactions verify functionality from a clinical perspective rather than just confirming technical availability.

Deploy specialized tracking for healthcare data as it moves through systems. Data pipeline monitoring improves clinical information integrity beyond basic system performance metrics.

Implement regular testing of failover systems for patient-critical applications to verify that they consistently meet established recovery time objectives (RTOs) and recovery point objectives (RPOs). Deploy automated recovery verification processes that validate systems will function as

expected during actual incidents affecting clinical care, confirming that recovery objectives are not just theoretically defined but practically achievable under real-world conditions

Develop dashboards that translate technical issues into clinical care implications. Patient impact visualization assists with appropriate response prioritization based on actual healthcare effects.

Establish comprehensive historical performance tracking that satisfies healthcare regulations. Compliance-focused data retention verifies that your monitoring data fulfills regulatory requirements for clinical systems.

Implementation steps

1. Configure healthcare simulations mimicking clinical workflows and calculations.
2. Implement verification for clinical data completeness and integrity.
3. Deploy scheduled failover testing aligned with clinical requirements.
4. Create dashboards showing patient impact and diagnostic delays.
5. Deploy secure archives for performance metrics and reporting.

Data management

LSPERF09: What processes do you have in place to select and implement multi-purpose data stores for clinical and research data management?

Implementation processes and criteria for specialized data stores that effectively manage both structured clinical data and unstructured research information while maintaining performance optimization and regulatory requirements.

LSPERF10: What strategies do you employ for caching frequently accessed clinical reference data?

Methodologies for implementing effective data caching systems optimizing access to the frequently used clinical reference data.

LSPERF11: How will you balance high-performance data access with cost-efficient storage in your data tiering and lifecycle policies?

Strategies for balancing performance and cost through optimized data tiering and lifecycle policies across active and archival datasets.

Best practices

- LSPERF09-BP01 Evaluate data stores based on regulatory requirements and data governance capabilities
- LSPERF09-BP02 Optimize query performance and meet diverse data access requirements in your environment
- LSPERF10-BP01 Implement tiered caching architecture with clinical data classification and lifecycle management
- LSPERF10-BP02 Establish intelligent cache warming and preloading based on clinical workflow patterns and seasonal variations
- LSPERF11-BP01 Optimize large dataset storage based on project phases and collaboration needs
- LSPERF11-BP02 Monitor data usage patterns and automatically adjust storage tiers for cost optimization

LSPERF09-BP01 Evaluate data stores based on regulatory requirements and data governance capabilities

Select data stores that provide features for healthcare regulations (HIPAA, FDA 21 CFR Part 11, GxP) including audit trails, data integrity controls, and encryption at rest and in transit. Assess the data store's ability to implement role-based access controls, data lineage tracking, and automated retention policies. For clinical data, prioritize solutions offering validated environments and regulatory documentation, while verifying that research data stores can accommodate less restrictive access patterns for collaboration and discovery workflows.

Desired outcome: Implement balanced security and governance framework that enforces robust controls while maintaining research flexibility, automates processes with minimal manual intervention, and provides comprehensive audit documentation across critical systems.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish a comprehensive evaluation system for assessing data store audit capabilities. This foundation aligns with healthcare regulations while maintaining required security and governance features.

Implement robust role-based access management across different data stores. This framework should support varied access patterns while maintaining strict regulatory requirements for clinical data.

Deploy comprehensive security measures including encryption and audit mechanisms. This system should preserve data integrity while supporting both clinical and research workflows.

Design integrated governance framework that spans different data stores. This assists with consistent policy enforcement while accommodating different workflow requirements.

Establish systematic validation processes for clinical data environments. This framework should maintain regulatory adherence while enabling efficient research collaboration.

Implementation steps

1. Deploy regulatory process mapping and automated monitoring systems.
2. Implement end-to-end encryption and comprehensive audit trail mechanisms.
3. Establish role-based access control with automated review procedures.
4. Create data lineage tracking with visualization capabilities.
5. Deploy automated policy enforcement and documentation systems.
6. Implement continuous security monitoring and response protocols.

Resources

- AWS HealthLake
- AWS HIPPA

LSPERF09-BP02 Optimize query performance and meet diverse data access requirements in your environment

Analyze your organization's specific query patterns. Select OLTP-optimized stores for transactional clinical applications requiring low-latency point queries, and columnar stores (like Amazon

Redshift) for analytical workloads involving large-scale clinical trial analysis. For unstructured research data, evaluate stores based on throughput requirements for genomic processing, search capabilities for literature and imaging data, and integration with machine learning pipelines. Consider hybrid approaches where query engines can span multiple storage types to avoid data duplication.

Desired outcome: Implement high-performance multi-store architecture that maintains consistent query response times across storage types, enables seamless cross-store integration with minimal latency, scales linearly under peak loads, and optimizes resource utilization through automated workload management.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive system for analyzing and categorizing query patterns across different workloads. This foundation enables optimal storage selection while assisting to meet performance requirements for various use cases.

Design storage architecture that aligns specific data stores with workload characteristics. This framework should optimize performance for both transactional and analytical requirements while minimizing redundancy.

Implement monitoring and optimization mechanisms across different storage types. This creates consistency in performance levels while maintaining efficiency for varied access patterns.

Deploy unified query capabilities across multiple storage systems. This framework should enable seamless data access while optimizing for specific workload requirements.

Design storage solutions that accommodate growth in both data volume and query complexity. This improves long-term performance sustainability while maintaining cost efficiency

Implementation steps

1. Deploy comprehensive workload monitoring and classification systems.
2. Implement multi-storage architecture with OLTP and columnar capabilities.
3. Create automated performance optimization and testing procedures.
4. Deploy unified query management with cross-store integration.
5. Establish automated scaling and capacity planning mechanisms.

6. Implement resource monitoring and growth management protocols.

LSPERF10-BP01 Implement tiered caching architecture with clinical data classification and lifecycle management

Deploy multi-tier caching that categorizes clinical reference data by access frequency and criticality (memory-based caching for ultra-high frequency data like active drug formularies and emergency protocols, and distributed caching for diagnostic codes and lab reference ranges). Establish automated lifecycle management with real-time updates for safety-critical information such as drug recalls and adverse event alerts. Use scheduled refresh cycles for stable reference data like anatomical classifications and billing codes based on clinical data volatility patterns.

Desired outcome: Implement intelligent caching architecture that maintains optimal clinical data access through multi-tier classification, propagates critical updates while managing reference data cycles, automates lifecycle management across cache tiers, and delivers reliable safety-critical data through verified priority-based mechanisms.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish multi-level caching framework that aligns with clinical data criticality and access patterns. This foundation enables optimal performance for different data categories while providing for appropriate refresh cycles.

Implement comprehensive classification mechanisms for different types of clinical reference data. This framework should categorize data based on access frequency, criticality, and volatility requirements.

Deploy automated lifecycle management processes for different data categories. This system should handle real-time updates for critical data while managing scheduled refreshes for stable reference data.

Design specialized handling for high-priority clinical updates and alerts. This propagates critical information while maintaining system stability.

Implement intelligent cache distribution and management across tiers. This framework should optimize resource utilization while maintaining required access speeds for different data categories.

Implementation stepsa>

1. Deploy multi-tiered caching with memory-based and distributed systems.
2. Implement automated data classification and access pattern monitoring.
3. Create lifecycle management with version control and refresh cycles.
4. Establish real-time critical update system with verification procedures.
5. Deploy comprehensive cache performance tracking and optimization.
6. Implement automated alerts and regular efficiency reviews.

LSPERF10-BP02 Establish intelligent cache warming and preloading based on clinical workflow patterns and seasonal variations

Develop predictive cache warming strategies that analyze historical clinical access patterns to preload relevant reference data before peak usage periods. Implement workflow-aware preloading for scheduled procedures, seasonal health trends, and active clinical trial requirements.

Desired outcome: Implement a predictive cache management system that uses ML-driven historical analysis for proactive data availability, adapts to workflow patterns and seasonal demands, optimizes resource allocation based on usage predictions, and provides consistent performance through automated adjustments during varying demand periods.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement intelligent cache warming mechanisms based on historical usage analysis. This foundation enables proactive data availability while optimizing system performance during peak periods.

Establish comprehensive monitoring of clinical access patterns and workflows. This framework should identify and respond to regular usage patterns while adapting to seasonal variations.

Deploy cache management systems that balance resource utilization with access speed requirements. This provides optimal performance during high-demand periods while maintaining efficient resource usage.

Design adaptive caching strategies that account for predictable variations in clinical workflows. This system should automatically adjust to known seasonal patterns and usage trends.

Implement dynamic resource management that optimizes cache allocation based on predicted demand. This improves the efficiency of caching resources while maintaining performance levels.

Implementation stepsa>

1. Deploy ML-powered workflow monitoring and pattern analysis.
2. Implement AI-driven predictive cache warming with automated scheduling.
3. Create adaptive workflow-aware caching with pattern detection.
4. Establish intelligent seasonal adjustment mechanisms.
5. Deploy real-time cache performance optimization with AI.
6. Implement automated efficiency monitoring and recommendations.

LSPERF11-BP01 Optimize large dataset storage based on project phases and collaboration needs

Design tiering strategies for large data files that consider different project stages and team access requirements. Store active project data in high-performance storage during periods of heavy use and analysis. Move completed project data to intermediate storage tiers when active work finishes but data may still be needed for reference. Archive older datasets to cost-effective deep storage while maintaining the ability to retrieve them when needed for future projects or comparisons.

Desired outcome: Implement intelligent storage tiering architecture that maintains optimal data access through automated placement strategies, improves cost-efficient lifecycle management across performance and archival tiers, enables seamless collaboration with appropriate access speeds, and delivers reliable data retrieval with predictable performance and clear visibility.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Design comprehensive storage architecture that aligns with project lifecycle phases. This foundation enables efficient data management across different stages while optimizing both performance and cost.

Implement systematic approaches for identifying and managing data requirements through different project phases. This fosters appropriate storage allocation based on actual project needs and team collaboration patterns.

Deploy intelligent storage tiering mechanisms that automatically adjust based on project stages. This framework should balance performance requirements with cost-effective storage solutions.

Establish storage access patterns that support team collaboration needs across project phases. Appropriate storage access patterns provide efficient data availability while maintaining appropriate performance levels.

Implement robust archival processes that maintain data accessibility while optimizing storage costs. This system should include clear retrieval mechanisms for future reference needs.

Implementation stepsa>

1. Deploy automated project phase tracking with classification systems.
2. Implement multi-tiered storage with performance optimization.
3. Create comprehensive access pattern monitoring and analysis.
4. Establish automated archival management with metadata tracking.
5. Deploy cross-tier performance monitoring and optimization.
6. Implement automated alerts and review procedures.

LSPERF11-BP02 Monitor data usage patterns and automatically adjust storage tiers for cost optimization

Track how often different datasets are accessed to make smart decisions about where to store them. Set up automatic monitoring that identifies when data usage patterns change and triggers moves to appropriate storage tiers.

Desired outcome: Implement dynamic storage optimization system that continuously monitors access patterns for automated tier placement, provides for cost-efficient distribution through real-time analytics-driven management, maintains predictable costs through intelligent policy enforcement, and provides comprehensive visibility for data-driven storage decisions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive monitoring systems that track data access patterns and frequencies. This foundation enables intelligent decision-making for storage optimization while maintaining necessary access performance.

Implement dynamic storage tiering mechanisms that automatically adjust based on usage patterns. This system should optimize costs while providing appropriate access speeds for different data types.

Deploy cost-aware storage management systems that balance performance requirements with budget constraints. This framework should automatically adjust storage allocation based on defined policies and actual usage.

Create robust monitoring capabilities that track access speeds and service levels across storage tiers. This maintains the required performance levels for critical data while enabling cost optimization.

Develop flexible policy management systems that govern storage tiering decisions. This framework should adapt to changing business needs while maintaining cost efficiency.

Implementation steps

1. Deploy AI-enhanced usage monitoring with real-time pattern analysis.
2. Implement intelligent storage tiers with automated data movement.
3. Create comprehensive cost management with predictive modeling.
4. Establish AI-driven performance optimization and anomaly detection.
5. Deploy automated policy enforcement with monitoring.
6. Implement dynamic benchmarking and continuous improvement systems.

Network and content delivery

LSPERF12: How do you design network architectures to support secure transmission of large datasets while optimizing throughput?

Design principles for secure, high-throughput network architectures optimized for transmitting large datasets such as images, scientific data, and other bandwidth-intensive content.

LSPERF13: What process do you follow to select appropriate networking features for multi-site clinical trials?

A structured approach for selecting networking capabilities that effectively support distributed multi-site clinical trials.

LSPERF14: How do you evaluate content delivery solutions to improve performance of global research collaboration?

Evaluation framework for selecting content delivery solutions that enhance performance of worldwide research collaboration.

LSPERF15: What methods do you use to monitor and optimize network latency for real-time clinical applications?

Techniques for monitoring and minimizing network latency to support reliable real-time clinical application performance.

LSPERF16: How do you balance network throughput requirements for large-scale data transfers with security considerations?

Strategies for optimizing network throughput for large-scale data transfers while maintaining robust security protocols.

LSPERF17: How do you evaluate the performance impact of cross-region data transfers while maintaining data sovereignty requirements?

Methodology for assessing cross-region data transfer performance while improving adherence to data sovereignty regulations.

Best practices

- LSPERF12-BP01 Implement network segmentation with defense-in-depth controls
- LSPERF12-BP02 Deploy accelerated encryption technologies with hardware offloading

- LSPERF12-BP03 Optimize data transfer with intelligent traffic management and compression
- LSPERF13-BP01 Conduct comprehensive site technology assessment and gap analysis
- LSPERF13-BP02 Implement secure data exchange architecture with regulatory controls
- LSPERF13-BP03 Deploy resilient connectivity with bandwidth optimization for remote locations
- LSPERF14-BP01 Conduct performance benchmarking across geographic research hubs
- LSPERF14-BP02 Evaluate multi-CDN architectures with intelligent traffic routing capabilities
- LSPERF14-BP03 Assess edge computing integration for localized processing of research workloads
- LSPERF15-BP01 Implement application-aware network path optimization and traffic prioritization methods
- LSPERF15-BP02 Deploy synthetic transaction monitoring and real user monitoring with automated performance optimization
- LSPERF16-BP01 Deploy intelligent traffic shaping with security-aware bandwidth allocation for different data types
- LSPERF17-BP01 Measure baseline data transfer performance and evaluate how data sovereignty requirements affect latency
- LSPERF17-BP02 Implement data classification-based transfer assessment with region-specific regulatory validation

LSPERF12-BP01 Implement network segmentation with defense-in-depth controls

Design the network with clearly defined security zones separated by firewalls, with sensitive data repositories isolated from general network traffic. Employ a layered security approach (defense in depth) with multiple security controls at each boundary. Use VLANs, subnets, and micro-segmentation to isolate data flows based on security levels and functional requirements. This architecture limits potential attack surface and contains breaches should they occur while maintaining efficient routing paths for authorized large data transfers.

Desired outcome: You have a segmented network architecture that provides multiple layers of security controls, supports regulatory requirements, and enables secure collaboration across research teams. This approach protects sensitive life sciences data while maintaining operational efficiency and regulatory alignment.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Network segmentation is fundamental for life sciences organizations handling sensitive data like clinical trials, genomic data, or patient information. By implementing multiple security layers, you create isolation between different data sensitivity levels and reduce the potential attack surface.

This approach aligns with key regulatory requirements including HIPAA, GxP, and GDPR. Segmentation enables clear audit boundaries and demonstrates regulatory adherence by maintaining separate environments for development, validation, and production systems.

Defense in depth controls allow for granular access management and simplified auditing. This architecture supports the isolation of regulated workloads while enabling secure collaboration between research teams, clinical partners, and third-party vendors.

Implementation steps

1. Create separate VPCs for production, development, and test environments with tiered subnets.
2. Deploy AWS Transit Gateway for centralized connectivity between environments.
3. Implement AWS Network Firewall and AWS PrivateLink for secure service access.
4. Configure security groups and Network ACLs following least-privilege principles.
5. Deploy AWS WAF and AWS Shield for application security and DDoS protection.
6. Enable VPC Flow Logs and AWS GuardDuty for comprehensive security monitoring.
7. Establish AWS Security Hub CSPM for centralized audit and security posture management.
8. Configure AWS Site-to-Site VPN for secure remote connectivity to on-premises resources.

LSPERF12-BP02 Deploy accelerated encryption technologies with hardware offloading

Implement high-performance encryption solutions that don't compromise throughput when securing large datasets in transit. Use hardware-accelerated encryption (specialized NICs, encryption accelerator cards) and efficient protocols like TLS 1.3 with optimized cipher suites. Consider authenticated encryption with associated data (AEAD) ciphers for simultaneous confidentiality and integrity validation. For extremely large datasets, evaluate selective encryption approaches that prioritize sensitive components while optimizing overall transfer speeds.

Desired outcome: You have a hardware-accelerated encryption system that provides data security while maintaining high performance for life sciences workflows. This enables efficient processing of

sensitive research data with automated scaling, regulatory adherence, and consistent performance during peak workloads.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Hardware-accelerated encryption is critical for life sciences workflows involving large-scale data processing, such as genomic sequencing or medical imaging. This approach improves security without creating bottlenecks in research pipelines.

Encryption technologies must meet specific regulatory standards for data protection while supporting high-throughput operations. Hardware offloading enables consistent performance for encrypted data transfers while maintaining regulatory adherence.

The solution scales automatically with workload demands, providing consistent performance during peak processing periods such as batch analysis or multi-site clinical trials. This maintains security without compromising research timelines.

Implementation steps

1. Deploy EC2 instances with NVIDIA T4 or AWS Inferentia chips for hardware-accelerated encryption.
2. Configure TLS 1.3 with AES-GCM cipher suites and AWS CloudHSM for key management.
3. Implement AWS Nitro Enclaves for instance-level encryption offloading.
4. Enable AWS Global Accelerator for optimized encrypted data routing.
5. Set up AWS Certificate Manager for automated TLS certificate lifecycle management.
6. Configure enhanced networking with Elastic Network Adapters for encryption performance.
7. Establish Amazon CloudWatch dashboards to monitor encryption performance metrics.
8. Implement automated key rotation and comprehensive encryption audit logging.

LSPERF12-BP03 Optimize data transfer with intelligent traffic management and compression

Implement quality of service (QoS) mechanisms to prioritize critical data transfers while avoiding bandwidth saturation. Deploy WAN optimizers and SD-WAN solutions that can intelligently route traffic across multiple paths based on real-time conditions. Use lossless compression

algorithms appropriate to your data types before transmission to reduce bandwidth requirements. Implement enhanced TCP congestion control algorithms (like bottleneck bandwidth and round-trip propagation time (BBR)) and parallel data transfer technologies to maximize throughput across high-latency networks without overwhelming network resources.

Desired outcome: You have an optimized network infrastructure that intelligently prioritizes research traffic, efficiently routes data across global sites, and uses advanced compression techniques. This enables faster and more reliable data transfers while maintaining the integrity of sensitive life sciences workloads.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement QoS mechanisms to provide bandwidth priority to critical research data and time-sensitive clinical trial information. This avoids network congestion while maintaining performance for essential workflows and regulatory adherence requirements.

Deploy SD-WAN solutions with real-time path selection capabilities to optimize data movement across global research sites. WAN optimization improves network resource utilization efficiency while maintaining data integrity for sensitive life sciences workloads.

Use lossless compression algorithms specifically designed for life sciences data types. This improves data integrity for genomic sequences, clinical trials, and medical imaging while reducing bandwidth consumption and transfer times.

Implement advanced TCP congestion control with BBR to maximize network efficiency. Configure parallel data transfer capabilities to optimize throughput for large-scale research data sets across high-latency global networks.

Balance network resources through intelligent traffic shaping and bandwidth allocation. Monitor network utilization patterns to avoid saturation while providing consistent performance for critical research operations.

Implementation steps

1. Deploy AWS Global Accelerator for optimized global routing and traffic management.
2. Configure Amazon Route 53 with health checks and failover routing policies for high availability.
3. Implement Amazon CloudFront with optimized cache behaviors for life sciences content distribution.

4. Enable Amazon S3 Transfer Acceleration for expedited large genomic dataset transfers.
5. Configure QoS policies with traffic prioritization for different data types (like genomic, clinical, and imaging).
6. Deploy Application Load Balancers with content-based routing for distributed data transfers.
7. Implement data-specific compression algorithms for genomic, clinical trial, and medical imaging data.
8. Establish compression verification checks and performance monitoring dashboards.

LSPERF13-BP01 Conduct comprehensive site technology assessment and gap analysis

Begin with a thorough evaluation of the existing technological infrastructure at each participating clinical site. Document connectivity capabilities, bandwidth availability, network reliability metrics, and adherence to healthcare standards. Identify critical gaps through a standardized assessment protocol that evaluates both technical capabilities and regulatory adherence. Create a site-specific technology readiness scorecard that informs network design and implementation requirements. This foundational assessment tailors networking solutions to accommodate variability across sites while meeting study requirements.

Desired outcome: You have a comprehensive understanding of your network infrastructure across trial sites with detailed visibility into traffic patterns and bandwidth utilization. This enables informed decision-making for network optimization and provides reliable connectivity for critical trial operations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Document and assess network infrastructure at each trial site, including available bandwidth, current utilization patterns, and existing bottlenecks. Evaluate various connectivity options such as fiber, broadband, and cellular backup solutions. Create comprehensive site profiles detailing redundancy requirements and failover capabilities for critical locations for continuous trial operations.

Analyze expected data volumes and traffic patterns for each trial site, focusing on peak usage periods during different trial phases. Document requirements for real-time data transmission,

considering critical trial activities and patient monitoring needs. Create detailed mapping of data backup and synchronization requirements to maintain data integrity across each location.

Implementation steps

1. Use AWS Systems Manager to maintain site inventory and deploy Amazon CloudWatch agent for bandwidth monitoring.
2. Deploy AWS Network Manager for connectivity assessments and configure VPC Flow Logs for data volume analysis.
3. Create a CloudWatch dashboard for comprehensive traffic visualization and monitoring.
4. Implement automated alerts for bandwidth thresholds and network performance anomalies.
5. Establish regular review cycles for network utilization patterns and optimization opportunities.
6. Document network topology and performance baselines to support capacity planning.

LSPERF13-BP02 Implement secure data exchange architecture with regulatory controls

Design a secure network architecture with data integrity and cross-border regulatory adherence as priorities. Encrypt patient data protection with robust key management. Implement standardized secure protocols for data exchange that work across different connectivity levels while meeting HIPAA, GDPR, and local healthcare regulations. Set up data residency controls aligned with regional health information requirements. Maintain comprehensive audit trails for monitoring and quick incident response.

Desired outcome: You have a secure data management system that enforces encryption standards, meets regional regulatory requirements, and provides comprehensive audit capabilities. This system protects sensitive trial data while maintaining operational efficiency and regulatory adherence.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive data security by defining classification levels for different types of trial information. Implement robust encryption standards protecting data both at rest and in transit across trial locations. Develop key management procedures and access control policies that align with trial security requirements while providing for operational efficiency.

Create detailed documentation of healthcare regulations specific to each Region where trials operate. Map HIPAA requirements directly to network architecture and security controls. Identify and implement data residency requirements for each jurisdiction, and verify that you have proper data storage and transmission paths. Establish comprehensive audit trail mechanisms that meet regulatory standards.

Deploy comprehensive logging systems capturing relevant security and operational events. Define critical monitoring parameters and alert thresholds based on trial requirements and regulatory needs. Establish regular review procedures to evaluate security posture and adhere to regulatory requirements.

Implementation steps

1. Implement comprehensive encryption with AWS KMS for key management, AWS Certificate Manager for SSL/TLS, and AWS Secrets Manager for credential protection.
2. Deploy tools including AWS Config for monitoring, AWS Control Tower for governance, and AWS Organizations for centralized policy management.
3. Secure data transfers using AWS Transfer Family for file exchanges, AWS PrivateLink for service communication, and AWS Direct Connect for dedicated connectivity.
4. Establish robust security operations with AWS Systems Manager for automated responses, AWS Backup for data recovery, and AWS Shield for DDoS protection.
5. Configure centralized logging and monitoring to detect security anomalies and violations.
6. Conduct regular security assessments and implement automated remediation for identified vulnerabilities.
7. Document security controls and maintain evidence of adherence to regulatory requirements.

LSPERF13-BP03 Deploy resilient connectivity with bandwidth optimization for remote locations

Set up multiple connectivity options based on each location's geographic limitations and infrastructure. Use smart traffic management to prioritize essential trial data transmission. Design systems that work offline and sync automatically when connection resumes, maintaining data integrity. Deploy backup options like satellite, cellular, or mesh networks for locations with unreliable connections. Use WAN optimization with compression and caching to minimize bandwidth usage, especially in areas with limited connectivity.

Desired outcome: You have a resilient network infrastructure with redundant connectivity, intelligent traffic prioritization, and robust offline capabilities. This improves the continuity of your trial operations through network disruptions while maintaining optimal performance for critical data transfers and applications.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Deploy robust primary connections with reliable backup options to provide continuous network availability. Implement automated failover mechanisms that trigger when primary connections fail. Set up comprehensive health monitoring systems to track connection status and performance metrics, enabling proactive identification of potential connectivity issues across trial locations.

Define and implement traffic prioritization based on critical data flows essential for trial operations. Create quality of service (QoS) policies that provide necessary bandwidth allocation to high-priority trial data. Establish bandwidth rules that optimize network resource utilization while maintaining performance for critical applications and data transfers.

Develop local caching mechanisms to maintain operations during connectivity interruptions. Implement robust data synchronization protocols that maintain data consistency when connectivity is restored. Create clear conflict resolution procedures to handle data conflicts that may arise during offline operations, maintaining data integrity across trial sites.

Implement advanced compression techniques to maximize available bandwidth efficiency. Configure strategic caching mechanisms to reduce redundant data transfers across the network. Establish comprehensive monitoring systems to track optimization effectiveness and identify areas for performance improvement.

Implementation steps

1. Establish resilient connectivity with AWS Direct Connect as primary path, AWS Site-to-Site VPN for backup, and Amazon Route 53 for automated DNS failover.
2. Optimize network architecture using AWS Transit Gateway for centralized traffic management, AWS Global Accelerator for performance, and VPC endpoints for secure service access.
3. Implement content delivery and security with Amazon CloudFront for edge distribution and AWS WAF for comprehensive traffic filtering and threat protection.
4. Configure automated monitoring and alerting for connectivity status, latency metrics, and security events across network components.

5. Document network topology with primary and backup paths, including recovery procedures and escalation protocols.
6. Conduct regular failover testing and performance optimization to provide business continuity and an optimal user experience.

LSPERF14-BP01 Conduct performance benchmarking across geographic research hubs

Develop comprehensive performance testing across research hubs to establish baselines and find regional bottlenecks. Monitor key metrics like latency, throughput, packet loss, and jitter under simulated research workloads. Create geographic heat maps showing areas needing improved delivery capabilities. Test using actual research data types including genomic files, high resolution images, and complex models. Set up ongoing performance monitoring to track improvements and detect new issues as global research collaboration patterns change.

Desired outcome: You have a comprehensive performance testing and monitoring system that provides real-time visibility into research network performance across geographic locations. This enables proactive optimization of data transfers and provides consistent application performance for global research collaboration.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Define key performance indicators including latency, throughput, and response times. Create a framework specifying testing frequencies and thresholds for research data types and collaboration scenarios. Deploy test endpoints across research hubs and implement automated testing scripts for continuous evaluation of network performance, data transfers, and application responsiveness.

Document region-specific performance metrics, including latency patterns, bandwidth utilization, and inter-hub connectivity. Create heat maps highlighting bottlenecks and areas needing optimization. Establish monitoring systems with alerting thresholds to quickly identify and address performance issues across the research network.

Design testing scenarios reflecting real research workflows, including genomic data transfers and collaboration sessions. Implement testing for file operations and data synchronization. Focus on measuring application performance across research tools to provide a consistent user experience globally.

Deploy monitoring dashboards for real-time performance visibility. Configure automated alerts for performance issues and system health. Maintain regular review cycles including weekly assessments and monthly trend analysis for continuous improvement.

Implementation steps

1. Implement comprehensive performance testing with Amazon CloudWatch Synthetics for synthetic monitoring, AWS X-Ray for distributed tracing, and CloudWatch metrics for detailed performance data collection.
2. Optimize geographic performance using AWS Global Accelerator for intelligent routing, Amazon CloudFront regional caches for content delivery, and Amazon Route 53 geolocation routing for regional traffic management.
3. Accelerate research data transfers with Amazon S3 Transfer Acceleration for global uploads, DataSync for automated transfers, and enhanced networking for optimized throughput.
4. Establish proactive monitoring using CloudWatch dashboards for visualization, Amazon EventBridge for automated alerting, and Amazon RDS Performance Insights for deep analysis of performance bottlenecks.
5. Configure performance baselines and thresholds to automatically detect and respond to anomalies across global infrastructure.
6. Document performance requirements and implement regular optimization reviews to continuously improve user experience.

LSPERF14-BP02 Evaluate multi-CDN architectures with intelligent traffic routing capabilities

Evaluate content delivery network (CDN) providers focusing on their performance in academic and research regions. Create a dynamic multi-CDN system that routes content based on real-time performance, content type, and destination. Test features like origin shields, cache optimization for scientific data, and API configuration. Assess providers' capability to handle specialized research needs including HD microscopy streaming, large dataset syncing, and real-time collaborative visualization. Consider combining commercial CDNs with research networks like Internet2 or GÉANT for optimal performance.

Desired outcome: You have a multi-CDN architecture optimized for research content delivery, with intelligent traffic routing and comprehensive performance monitoring. This enables efficient

distribution of specialized research data while maintaining high performance across academic networks and research locations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Evaluate and select CDN providers based on their performance in academic and research regions. Analyze capabilities for handling specialized research content types and assess integration with research networks. Create comprehensive provider profiles documenting performance metrics, features, and integration capabilities with academic networks like Internet2 or GÉANT.

Design dynamic routing mechanisms that direct content based on real-time performance metrics, content type, and destination requirements. Implement routing policies that consider factors like latency, throughput, and regional performance patterns. Establish automated failover mechanisms for continuous content availability across multiple providers.

Configure CDN settings specifically for research and academic content types, including cache optimization for scientific datasets and streaming configurations for HD microscopy. Implement specialized handling for large dataset synchronization and real-time collaborative visualization tools. Create content-specific delivery rules that optimize performance for different research workflows.

Deploy comprehensive monitoring systems to track CDN performance across academic regions and research networks. Establish real-time performance dashboards and automated alerting systems for quick issue identification. Implement regular performance analysis cycles to continuously optimize content delivery across the multi-CDN architecture.

Implementation steps

1. Deploy Amazon CloudFront as primary CDN with origin shields and cache optimization for research data, while implementing additional CDN providers with Internet2 connections and configuring origin failover for continuous availability.
2. Set up Amazon Route 53 traffic flow policies for intelligent performance-based routing, deploy health checks for endpoint monitoring across academic regions, and implement latency-based routing to optimize delivery paths.
3. Configure specialized cache policies for scientific datasets, API configurations for efficient research application requests, and streaming optimizations to support HD microscopy and real-time visualization needs.

4. Establish comprehensive monitoring with performance tracking across academic regions, cost analysis for usage pattern optimization, and customized dashboards for content delivery efficiency visibility.
5. Implement automated testing to regularly validate CDN performance across different research data types and geographic locations.
6. Document CDN architecture with failover procedures and optimization strategies for different content categories and research applications.
7. Conduct quarterly performance reviews to identify improvement opportunities and adjust configurations based on evolving research needs.

LSPERF14-BP03 Assess edge computing integration for localized processing of research workloads

Assess edge computing solutions that enable research processing near data sources and users. Review edge systems' compatibility with research computing frameworks while improving version consistency. Test edge-based collaboration tools including shared notebooks, visualization systems, and model training. Compare performance and resource efficiency between edge and centralized processing. Evaluate edge solutions supporting secure multi-tenant environments where researchers from different institutions can share computing resources with proper data isolation.

Desired outcome: You have a secure edge computing environment that enables localized processing of research workloads with effective collaboration tools, optimized performance, and strict multi-tenant isolation. This allows researchers from different institutions to efficiently share computing resources while maintaining data security and low-latency access.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Assess edge computing systems for compatibility with research workflows and computing frameworks. Document system capabilities including processing power, memory requirements, and storage options. Create comprehensive evaluation criteria focusing on version management, framework support, and integration capabilities with existing research infrastructure.

Evaluate edge-based collaboration tools and their effectiveness in supporting research workflows. Test implementation of shared notebooks, real-time visualization systems, and distributed model

training capabilities. Document performance metrics and user experience factors across different edge locations and research scenarios.

Conduct systematic comparison of edge versus centralized processing for common research workloads. Measure resource utilization, processing times, and data transfer requirements across different deployment models. Create performance baselines and optimization strategies for various research computing scenarios.

Design secure multi-tenant environments that enable resource sharing while maintaining strict data isolation. Implement access controls and monitoring systems that foster secure collaboration across institutions. Establish clear security protocols and frameworks for edge deployments.

Implementation steps

1. Deploy comprehensive edge infrastructure with AWS Outposts for local processing, AWS Wavelength for ultra-low latency applications, and AWS Local Zones to reduce latency for research tools and visualization.
2. Establish collaborative research solutions using Amazon SageMaker AI for distributed model training, AWS IoT Greengrass for edge device management, and container services for consistent application deployment across locations.
3. Implement robust monitoring and optimization with Amazon CloudWatch for edge performance tracking, AWS Systems Manager for infrastructure management, and AWS X-Ray for distributed application tracing and analysis.
4. Secure edge environments through AWS IAM for granular access control, AWS Security Hub CSPM for centralized security posture management, and AWS KMS for comprehensive data encryption and key management.
5. Document edge architecture with connectivity patterns, data synchronization procedures, and failover mechanisms for research continuity.
6. Conduct regular performance testing to validate latency requirements and optimize resource allocation for evolving research workloads.
7. Establish governance framework for edge deployments including monitoring and automated security controls.

LSPERF15-BP01 Implement application-aware network path optimization and traffic prioritization methods

Analyze network paths to find optimal routes for clinical data and implement dynamic routing for time-sensitive medical information. Use deep packet inspection to identify and prioritize clinical traffic, which verifies that critical systems like patient monitors get priority bandwidth through quality of service (QoS) controls. Set up network segmentation using VLANs to separate clinical from administrative traffic, maintaining guaranteed bandwidth and latency limits for clinical applications.

Desired outcome: You have an intelligent network infrastructure that automatically optimizes paths for clinical data while prioritizing the handling of critical medical traffic through effective segmentation and QoS controls.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establish comprehensive monitoring of network paths for clinical data flows. Implement automated systems to identify optimal routes and provides priority handling to time-sensitive medical information through intelligent path selection.

Deploy deep packet inspection mechanisms to identify clinical traffic patterns. Create QoS policies that guarantee bandwidth for critical medical systems and patient monitoring applications while maintaining required performance levels.

Design and implement VLAN architecture that separates clinical and administrative traffic. Establish clear bandwidth allocation policies that provide guaranteed resources to clinical applications without interference from other network traffic.

Implementation steps

1. Optimize network paths by deploying AWS Transit Gateway for centralized routing, AWS Global Accelerator for performance enhancement, and Amazon Route 53 for intelligent health-aware DNS routing.
2. Implement comprehensive traffic management using AWS Network Firewall for classification, AWS Transit Gateway for QoS enforcement, and VPC endpoints to prioritize access to critical services.

3. Establish strong network segmentation with separate VPCs for clinical and administrative traffic, AWS PrivateLink for secure service connectivity, and VPC Flow Logs for detailed traffic monitoring.
4. Deploy end-to-end performance monitoring with Amazon CloudWatch dashboards for network metrics visualization, configured alarms for threshold violations, and AWS X-Ray for distributed application tracing.
5. Document network architecture with traffic flow patterns, segmentation boundaries, and performance baselines for different application categories.
6. Conduct regular network assessments to identify optimization opportunities and validate segmentation controls.
7. Implement automated remediation for common network performance issues and security policy violations.

LSPERF15-BP02 Deploy synthetic transaction monitoring and real user monitoring with automated performance optimization

Implement synthetic monitoring methods that continuously simulate critical clinical workflows such as record retrieval, patient monitoring data transmission, and medication order processing to measure end-to-end latency and automatically trigger optimization actions when thresholds are exceeded. Use real user monitoring (RUM) techniques to capture actual network performance experienced by clinicians and implement adaptive optimization that adjusts network configurations based on real-time performance data, such as switching to alternate network paths or increasing bandwidth allocation for degraded connections.

Desired outcome: You have an integrated monitoring system combining synthetic testing and real user monitoring that automatically detects and responds to performance issues in clinical workflows. This enables proactive optimization of network performance and provides reliable operation of critical healthcare applications.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Establish comprehensive synthetic monitoring for critical clinical workflows, including automated tests for record retrieval and patient data transmission. Configure monitoring scenarios that simulate real clinical activities with defined performance thresholds. Implement automated

response mechanisms that trigger optimization actions when performance degrades below acceptable levels.

Deploy RUM solutions to capture actual performance metrics experienced by clinical users across different locations and devices. Set up data collection for key performance indicators including page load times, API response times, and network latency. Create performance baselines that reflect real-world usage patterns in clinical environments.

Design automated optimization systems that respond to both synthetic and RUM data. Implement intelligent routing algorithms that can automatically adjust network paths based on performance metrics. Establish bandwidth allocation rules that dynamically adapt to changing network conditions and clinical priorities.

Create comprehensive alerting mechanisms that identify performance issues before they impact clinical operations. Implement automated remediation procedures for common performance issues. Establish escalation paths for situations requiring manual intervention.

Implementation steps

1. Establish comprehensive synthetic monitoring by deploying Amazon CloudWatch Synthetics with canaries simulating clinical workflows, AWS X-Ray traces for performance tracking, and custom metrics to measure healthcare-specific application performance.
2. Implement real user monitoring (RUM) with CloudWatch RUM for real-time clinical application data collection, performance monitoring agents on user devices, and client-side monitoring to track user experience metrics.
3. Deploy performance optimization tools including AWS Global Accelerator for automatic path optimization, Application Auto Scaling to adjust resources based on demand, and AWS Transit Gateway for intelligent network traffic routing.
4. Create integrated monitoring and alerting with CloudWatch dashboards visualizing synthetic and RUM metrics, Amazon EventBridge rules automating responses to performance issues, and Amazon SNS notifications providing rapid response to critical alerts.
5. Document performance baselines specific to clinical workflows and establish escalation procedures for different severity levels of performance degradation.
6. Implement regular performance reviews to identify optimization opportunities and validate monitoring coverage across critical healthcare applications.
7. Configure automated remediation for common performance issues to minimize impact on clinical operations.

LSPERF16-BP01 Deploy intelligent traffic shaping with security-aware bandwidth allocation for different data types

Develop traffic classification systems to identify life sciences data transfers and apply security and throughput policies based on sensitivity and size. Set up dynamic bandwidth allocation that prioritizes large research data transfers during off-peak times while maintaining security measures. Implement compression and de-duplication before encryption to reduce transmission volume and improve efficiency without compromising security.

Desired outcome: You have an intelligent traffic management system that automatically classifies and optimizes life sciences data transfers while maintaining security controls. This enables efficient use of network resources without compromising data protection requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive classification systems that automatically identify and categorize life sciences data types based on sensitivity levels and transfer requirements. Create policies that map security controls and bandwidth allocation rules to different data classifications.

Design intelligent bandwidth allocation systems that adapt to network conditions and usage patterns. Implement automated scheduling for large data transfers during off-peak periods while maintaining required security controls.

Configure compression and de-duplication mechanisms that operate before encryption to maximize transfer efficiency. Implement monitoring systems to verify that optimization doesn't impact data integrity or security requirements.

Implementation steps

1. Implement comprehensive data classification with AWS Network Firewall for traffic categorization, Application Load Balancer for content-based routing, and AWS WAF rules for precise traffic identification.
2. Deploy bandwidth management tools including AWS Transit Gateway for centralized traffic control, VPC traffic mirroring for detailed analysis, and AWS Global Accelerator for optimized routing across networks.
3. Establish multi-layered security with AWS Shield for DDoS protection, AWS WAF rate-based rules to block abuse, and security groups for granular traffic control between resources.

4. Configure detailed monitoring and analytics using Amazon CloudWatch for bandwidth metrics, VPC Flow Logs for traffic pattern analysis, and Amazon GuardDuty for automated threat detection and response.
5. Document traffic management policies with classification criteria, bandwidth allocation priorities, and security enforcement points.
6. Implement automated alerting for unusual traffic patterns, bandwidth threshold violations, and potential security issues.
7. Conduct regular traffic analysis to optimize bandwidth allocation and security rules based on evolving usage patterns.

LSPERF17-BP01 Measure baseline data transfer performance and evaluate how data sovereignty requirements affect latency

Establish performance baselines by measuring transfer metrics for data routes and compare against direct paths to understand impact. Set up ongoing monitoring of transfers to track performance and find optimization opportunities within regulatory limits. Conduct regular testing using synthetic data to verify that routes meet both performance needs and regulatory requirements.

Desired outcome: You have a comprehensive monitoring and optimization system that provides baseline metrics, validation, and performance analysis for cross-region data transfers. This enables efficient management of data transfers while maintaining adherence to regional sovereignty requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive baseline metrics for data transfers across routes, including latency, throughput, and reliability measurements. Document performance characteristics of different transfer paths while maintaining adherence to data sovereignty requirements. Create comparative analysis frameworks to evaluate the impact of audit controls on transfer performance.

Implement continuous monitoring of approved data transfer routes to track performance trends and identify potential bottlenecks. Set up measurement systems that capture both performance metrics and regulatory adherence. Establish regular assessment cycles to evaluate transfer efficiency within regulatory constraints.

Design and implement synthetic testing procedures that simulate real-world data transfers while maintaining regulatory requirements. Create test scenarios that reflect various data types and transfer patterns common in production environments. Develop testing protocols that validate both performance standards and regulatory adherence.

Establish systematic processes for identifying optimization opportunities within adherence boundaries. Create performance improvement strategies that maintain required data sovereignty controls. Implement regular review cycles to assess and enhance transfer efficiency.

Implementation steps

1. Establish comprehensive performance measurement with Amazon CloudWatch for transfer monitoring across routes, VPC Flow Logs for detailed traffic pattern analysis, and AWS Transit Gateway Network Manager for cross-region connectivity visibility.
2. Implement robust monitoring using AWS Config for continuous sovereignty rule verification, AWS CloudTrail for complete audit logging of data transfers, and AWS Security Hub CSPM for centralized status monitoring.
3. Deploy automated testing infrastructure with Amazon CloudWatch Synthetics for transfer route validation, AWS X-Ray for detailed request path tracing across Regions, and Amazon Route 53 health checks for continuous endpoint availability monitoring.
4. Optimize data transfer capabilities using AWS Global Accelerator for path optimization, AWS Transfer Family for secure managed file transfers, and AWS Direct Connect for dedicated high-performance connections supporting critical transfers.
5. Document data sovereignty requirements with approved transfer routes, audit controls, and verification procedures for each data classification.
6. Establish regular reviews with automated reporting on transfer patterns, policy adherence, and performance metrics across regional boundaries.
7. Implement automated remediation for common violations and performance issues.

LSPERF17-BP02 Implement data classification-based transfer assessment with region-specific regulatory validation

Design test frameworks simulating real clinical and research data transfers while following data sovereignty routing rules. Test transfer performance under varying network conditions and data volumes while maintaining regional adherence. Use modeling to predict transfer times and resource needs for different sovereignty scenarios, informing cross-region data sharing decisions.

Desired outcome: You have an automated system for managing cross-Region data transfers that improves adherence to regional requirements while optimizing performance. This enables efficient data sharing across global research locations based on both performance metrics and regulatory requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Establish comprehensive data classification systems that align with regional regulatory requirements and clinical data sensitivity levels. Create clear mapping between data types and their corresponding transfer requirements. Implement automated classification tools that properly handle different data categories across Regions.

Design and execute testing frameworks that evaluate transfer performance while maintaining adherence to regional requirements. Create test scenarios using representative data volumes and types common in clinical research. Implement monitoring systems that track both performance metrics and regulatory adherence during transfers.

Develop region-specific regulatory validation processes that verify adherence to local data sovereignty requirements. Establish automated checks that validate transfer paths and data handling procedures. Create comprehensive audit trails that demonstrate adherence throughout the transfer lifecycle.

Implement modeling systems that analyze historical transfer data to predict performance under various scenarios. Create simulation tools that help evaluate different transfer strategies while maintaining adherence. Establish regular review cycles to refine and improve prediction accuracy.

Implementation steps

1. Implement automated data classification with Macie for sensitive data discovery, S3 Object Tags for appropriate classification labeling, and IAM policies to enforce classification-based access controls.
2. Establish comprehensive performance monitoring using Amazon CloudWatch metrics for cross-region transfer tracking, VPC Flow Logs for detailed traffic analysis, and Transit Gateway for visibility into cross-region transfer routes.
3. Deploy robust tools including AWS Config for regional requirement validation, AWS CloudTrail for detailed transfer audit logging, and AWS Security Hub CSPM for centralized status monitoring.

4. Create advanced analysis capabilities with Quick dashboards for transfer performance visualization, Amazon EventBridge for automated event responses, and AWS Systems Manager for coordinated cross-region operations management.
5. Document data classification standards with handling requirements for each sensitivity level and approved transfer patterns between regions.
6. Implement regular audit reporting with metrics on classification accuracy, transfer policy adherence, and regional status.
7. Establish automated remediation workflows for common classification and issues.

Process and culture

LSPERF18: How do you foster a culture that values performance efficiency without compromising scientific rigor or patient safety?

Establish cross-functional performance reviews where IT and scientists collaborate on optimization goals. Create documented performance standards that align with validation requirements. Implement monitoring dashboards showing both system performance and metrics, empowering teams to make data-driven improvements without compromising scientific integrity or patient safety.

LSPERF19: How do you load test and benchmark your applications and workflows?

Implement automated load testing using IaaC to deploy consistent test environments. Benchmark scientific workflows against predefined performance metrics while capturing validation evidence. Schedule regular performance tests with simulated research and clinical data loads to identify bottlenecks before they impact production operations.

Best practices

- LSPERF18-BP01 Perform cross-functional performance reviews between IT and scientists
- LSPERF18-BP02 Document performance standards aligned with validation requirements
- LSPERF18-BP03 Create integrated dashboards showing both performance and metrics
- LSPERF19-BP01 Implement infrastructure as code for consistent test environments

- LSPERF19-BP02 Establish comprehensive performance metrics and evidence collection
- LSPERF19-BP03 Schedule regular performance tests with production-representative data loads

LSPERF18-BP01 Perform cross-functional performance reviews between IT and scientists

IT professionals and scientists conduct joint performance reviews to evaluate system effectiveness. These teams set specific goals that align technical capabilities with scientific needs. For example, they might target a 99.9% system uptime while verifying that data accuracy meets research requirements.

Desired outcome: Achieving specific uptime and performance goals through joint performance reviews and aligned technical-scientific goal setting.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To effectively implement collaborative performance reviews, bring together IT professionals and scientists in structured evaluation sessions that assess system effectiveness from both technical and scientific perspectives. These cross-functional teams should establish specific, measurable goals that harmonize technical capabilities with scientific requirements, such as maintaining 99.9% system uptime while simultaneously verifying that data accuracy meets rigorous research standards.

Create detailed documentation of these performance standards, clearly articulating how technical metrics connect to validation requirements and scientific outcomes.

Develop comprehensive monitoring dashboards that visualize both technical performance indicators (latency, throughput, resource utilization) and scientific metrics (data quality, validation status, experimental reproducibility). This integrated view enables teams to identify optimization opportunities that don't compromise scientific integrity.

Implement regular review cycles where technical and scientific stakeholders jointly analyze performance data, discuss potential improvements, and make data-driven decisions about system adjustments.

Implementation steps

1. Schedule reviews using AWS Systems Manager OpsCenter items.

2. Track service-level agreements (SLAs) with Amazon CloudWatch dashboards and custom metrics.
3. Document goals in AWS Well-Architected Tool workloads.
4. Implement Amazon SageMaker AI Model Monitor for accuracy checks.
5. Use Service Catalog to enforce approved configurations.

LSPERF18-BP02 Document performance standards aligned with validation requirements

Metrics establish clear standards to assess regulatory adherence. Our approach balances system performance with validation requirements. Analytics-driven monitoring provides visibility into status and system performance, enabling teams to make decisions based on measurements rather than assumptions. This monitoring approach assits organizations to maintain regulatory adherence while improving efficiency, creating a framework where performance and validation work together.

Desired outcome: Implement analytics-driven monitoring that balances system performance with validation requirements, enabling measurement-based decision-making to simultaneously maintain regulatory adherence and improve operational efficiency.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Effective implementation of monitoring requires establishing clear, measurable standards that precisely define regulatory adherence expectations for generative AI systems. Organizations should develop a balanced approach that simultaneously evaluates technical system performance and validation requirements, recognizing that these aspects are complementary rather than competing priorities.

Deploy analytics-driven monitoring solutions that provide comprehensive visibility into both status and operational metrics through integrated dashboards accessible to stakeholders. This dual-perspective monitoring empowers cross-functional teams to make evidence-based decisions grounded in actual measurements rather than assumptions, creating a data-driven culture around audit management.

The monitoring framework should include automated alerts for audit deviations, regular reporting cycles, and trend analysis capabilities to identify potential issues before they become critical.

By implementing this integrated approach, organizations can maintain strict regulatory adherence while continuously improving system efficiency, establishing an operational environment where performance optimization and validation requirements work in harmony rather than opposition

Implementation steps

1. Configure AWS Config rules to automate monitoring.
2. Implement AWS Security Hub CSPM for centralized reporting.
3. Deploy Amazon CloudWatch metrics for performance and validation tracking.
4. Use Amazon OpenSearch Service for advanced data analytics.
5. Create Amazon EventBridge rules to alert on deviations.

LSPERF18-BP03 Create integrated dashboards showing both performance and metrics

The integrated dashboards combine performance and metrics into clear visualizations. Teams gain complete system visibility through real-time data, which enables direct decision-making and reduces process delays. The system enhances operations while maintaining scientific integrity and patient safety standards. Teams use this data to optimize systems while meeting medical and regulatory requirements.

Desired outcome: Deploy integrated dashboards with real-time visualization of performance and metrics, providing complete system visibility that enables immediate decision-making, reducing process delays, and allows teams to optimize operations while maintaining scientific integrity and patient safety standards.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To effectively implement integrated monitoring systems, develop comprehensive dashboards that seamlessly combine technical performance metrics with audit indicators in clear, actionable visualizations. These unified interfaces should present real-time data streams that provide complete system visibility to both technical and scientific stakeholders, reducing information silos that typically delay decision-making processes.

Configure the dashboards to display critical performance parameters (such as response times, throughput, and resource utilization) alongside relevant metrics (including data quality indicators,

validation status, and regulatory adherence scores). This integrated approach enables cross-functional teams to make informed, data-driven decisions without process delays that often occur when metrics are segregated across different systems.

Implement automated alerting mechanisms that notify appropriate personnel when metrics approach predefined thresholds, allowing proactive intervention before issues impact operations. The monitoring system should maintain comprehensive audit trails that document both performance optimizations and regulatory adherence, supporting regulatory requirements while enabling continuous improvement.

Implementation steps

1. Deploy Amazon CloudWatch dashboards with custom generative AI performance widgets.
2. Integrate AWS Audit Manager metrics for visualization.
3. Implement Amazon EventBridge for real-time alerts on threshold violations.
4. Use Quick for interactive data exploration and analysis.
5. Configure AWS Systems Manager for automated remediation workflows.

LSPERF19-BP01 Implement infrastructure as code for consistent test environments

Use infrastructure as code (IaC) to deploy consistent, repeatable test environments that accurately mirror production. By codifying your infrastructure, you reduce configuration drift between test and production environments, This approach enables teams to rapidly provision test environments on demand, run comprehensive load tests against scientific workflows, and tear down resources when testing concludes.

Desired outcome: Implement infrastructure as code (IaC) to deploy consistent, repeatable test environments that mirror production, reducing configuration drift and enabling on-demand provisioning for comprehensive scientific workflow load testing with efficient resource management.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing IaC provides a foundation for creating consistent, repeatable test environments that precisely mirror production configurations for generative AI systems. By codifying infrastructure

specifications, organizations can reduce configuration drift between testing and production environments and verify that performance evaluations and validation tests accurately reflect real-world conditions. This approach enables development and scientific teams to rapidly provision complete test environments on demand through automated processes, significantly reducing setup time and reducing manual configuration errors.

Teams can execute comprehensive load tests against scientific workflows in these environments, gathering performance metrics that reliably predict production behavior while maintaining strict validation controls. The ephemeral nature of IaC-provisioned environments allows organizations to tear down resources immediately after testing concludes, optimizing cost efficiency without sacrificing testing rigor.

Implement version control for infrastructure code to maintain historical records of environment configurations, supporting audit requirements and enabling rollback capabilities when needed.

Implementation steps

1. Use AWS CloudFormation to codify your generative AI infrastructure stack.
2. Implement AWS Config to detect and avoid configuration drift.
3. Deploy Service Catalog to standardize environment provisioning.
4. Integrate Distributed Load Testing on AWS for scientific workflow validation.
5. Configure Amazon EventBridge to automate resource cleanup after testing.

LSPERF19-BP02 Establish comprehensive performance metrics and evidence collection

Establish measurable performance metrics specific to scientific workflows before load testing. Capture traditional indicators alongside domain-specific measurements for research and clinical data processing. Implement automated evidence collection to create audit trails, inform optimization decisions, and verify system reliability. Centralize test results to enable trend analysis and detect regressions across cycles.

Desired outcome: Implement a comprehensive scientific workflow performance measurement framework that establishes metrics before testing, captures domain-specific indicators, automates evidence collection, and centralizes results for trend analysis and regression detection.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Effective performance testing for generative AI systems requires establishing comprehensive, measurable metrics specifically tailored to scientific workflows before initiating load testing activities. Organizations should develop a balanced measurement framework that captures traditional technical indicators (such as latency, throughput, and resource utilization) alongside domain-specific scientific measurements that reflect research quality and clinical data processing integrity.

Implement automated evidence collection mechanisms that systematically document test results, creating detailed audit trails that satisfy regulatory requirements while simultaneously providing valuable data for optimization decisions. These automated systems should capture performance data at regular intervals during testing, correlating system behaviors with specific workloads to identify optimization opportunities and potential bottlenecks.

Centralize test results in a structured repository that enables sophisticated trend analysis across multiple test cycles, facilitating the early detection of performance regressions that might impact scientific outcomes.

Configure dashboards to visualize both point-in-time performance and longitudinal trends, making complex performance data accessible to both technical and scientific stakeholders.

Implementation steps

1. Define metrics in Amazon CloudWatch using custom dimensions for scientific workflows.
2. Deploy AWS X-Ray traces to capture both technical and domain-specific indicators.
3. Implement AWS Audit Manager for automated evidence collection.
4. Use Amazon S3 and Amazon Athena to centralize and query performance test results.
5. Create Quick dashboards for trend analysis across test cycles.

LSPERF19-BP03 Schedule regular performance tests with production-representative data loads

Institute a cadence of regular performance tests using simulated research and clinical data that accurately represents production workloads in terms of volume, variety, and velocity. These scheduled tests should be integrated into your development pipeline to identify performance bottlenecks early, before they impact production operations. Vary test scenarios to account

for both typical daily operations and peak usage patterns, such as end-of-month processing or seasonal research activities. Complement scheduled tests with on-demand testing triggered by significant system changes. This proactive approach to performance validation verifies that scientific and clinical users consistently experience responsive systems that meet their demanding computational requirements.

Desired outcome: By implementing a comprehensive performance testing framework, scientific and clinical users experience consistently responsive systems that meet their computational requirements, with zero production incidents caused by performance bottlenecks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing effective performance testing for generative AI systems requires establishing a consistent cadence of regular evaluations using carefully constructed datasets that accurately represent production workloads. Organizations should develop simulated research and clinical data that mirrors actual usage patterns in terms of volume, variety, and velocity characteristics, verifying that test results provide meaningful insights about real-world performance. Integrate these scheduled performance tests directly into development pipelines as automated gates, enabling early identification of potential bottlenecks before they can impact production operations or scientific outcomes.

Design a comprehensive testing strategy that incorporates diverse scenarios reflecting both typical daily operations and anticipated peak usage patterns, such as end-of-month processing surges or seasonal research activities that might stress system resources. Complement the regular testing schedule with targeted, on-demand evaluations triggered by significant system changes, including infrastructure updates, algorithm modifications, or data pipeline adjustments.

This proactive, multi-faceted approach to performance validation creates a robust quality framework so that scientific and clinical users consistently experience responsive systems capable of meeting their demanding computational requirements.

Implementation steps

1. Deploy AWS DevOps tools like AWS CodePipeline with Amazon CloudWatch Synthetics for automated testing cycles.
2. Use Amazon SageMaker AI Experiments to track model performance across different test scenarios.

3. Implement AWS Step Functions to orchestrate complex testing workflows with varying loads.
4. Configure Amazon EventBridge to trigger on-demand tests when detecting significant system changes.
5. Use Quick dashboards to visualize performance trends across testing cycles.

Cost optimization

The life sciences industry faces unique cost optimization challenges. Rapidly evolving research and development, strict regulatory requirements, and high-performance computing needs create a demand for dynamic cost optimization across diverse workloads. Efficient cost management is critical to enable innovation and deliver life-changing treatments to patients.

Design principles

- **Monitor usage patterns:** Life sciences workloads often have unpredictable usage spikes during research phases or clinical trials. Continuously monitor usage and costs at the application and workload level to scale resources efficiently.
- **Align costs to business value:** Prioritize cost optimization for workloads directly driving research breakthroughs or enabling mission-critical applications. Optimize lower-value workloads over time.
- **Implement flexible capacity:** Use a mix of On-Demand, Savings Plans, Reserved Instances, and Spot Instances to balance performance, availability, and cost for compute-intensive research and development workloads.
- **Centralize cost visibility:** Implement Cloud Financial Management to provide organization-wide cost transparency and accountability, aligning IT spend to business priorities.

Practice Cloud Financial Management

LSCOST01: How does your life sciences team balance cloud costs, regulatory adherence, and research needs through financial controls?

Cloud Financial Management (CFM) allows finance, product, technology, and business organizations to manage, optimize, and plan costs as they grow their usage and scale on AWS. The primary goal of CFM is to allow customers to achieve their business outcomes in the most cost-efficient manner and accelerate economic and business value creation while finding the right balance between agility and control.

Cloud Financial Management with AWS offers a set of capabilities to manage, optimize, and plan for cloud costs while maintaining business agility. CFM is paramount not only to effectively manage cost, but also to verify that investments are driving expected business outcomes. The four areas of the Cloud Financial Management Framework in the AWS Cloud are *see*, *save*, *plan*, and *run*. Each of these areas has a set of activities and capabilities.

Following the best practices in CFM is essential for managing costs in your life sciences workloads. These CFM best practices assist to establish cost transparency to control your resources and plan your spend to optimize your return on investments.

CFM is a strategic framework for managing cloud financial operations in life sciences, focusing on balancing rigorous regulatory adherence, high-performance computational requirements, and cost optimization. This approach integrates financial controls, research infrastructure needs, and cloud economic principles to enable scientifically innovative and fiscally responsible cloud computing strategies.

Best practices

- LSCOST01-BP01 Establish and implement a comprehensive financial governance framework
- LSCOST01-BP02 Analyze and optimize vendor cost structures and economic relationships
- LSCOST01-BP03 Build the right skills and fostering a cost-aware culture

LSCOST01-BP01 Establish and implement a comprehensive financial governance framework

A financial governance framework in life sciences cloud management establishes the structure, policies, and processes for making financial decisions related to cloud usage. It verifies that cloud spending aligns with research priorities, regulatory requirements, and overall business objectives.

Desired outcome: A mature, organization-wide financial governance environment that provides complete visibility and control over cloud spending across each life sciences research initiative, enabling predictable budget management, automated cost optimization, and strategic alignment between cloud investments and research outcomes while adhering to regulatory and organizational financial policies.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Build a Cloud Center of Excellence (CCoE) with representatives from research, IT, finance, and security.

Develop a cloud cost allocation model that ties expenses to specific research projects or therapeutic areas.

Implement tagging policies and service control policies (SCPs) to enforce tagging strategies across your organization.

Establish approval workflows for high-cost cloud resources with different thresholds for various research stages. For example, consider using AWS Budgets for cost management and Service Catalog for deployment of pre-approved services.

Implementation steps

1. Form the Cloud Center of Excellence (CCoE):
    - Identify key stakeholders from your organization's departments.
    - Define roles and responsibilities for each CCoE member.
    - Schedule regular CCoE meetings to oversee the framework implementation.
2. Develop the cloud cost allocation model:
    - Map out existing research projects and therapeutic areas.
    - Create a cost structure that links cloud expenses to specific initiatives.
    - Design a reporting system to track and allocate costs accurately.
3. Implement tagging policies:
    - Define a comprehensive tagging strategy for cloud resources.
    - Create service control policies (SCPs) to enforce the tagging rules.
    - Configure automated tagging where possible to improve consistency.
4. Establish approval workflows:
    - Define thresholds for high-cost cloud resources at different research stages
    - Set up AWS Budgets for cost management:
        - Create budget alerts for various spending thresholds.
        - Configure notifications for key stakeholders when budgets are approaching limits.
    - Implement Service Catalog:

- Create a portfolio of pre-approved services and resources.
- Set up governance and access controls for the Service Catalog.

5. Continuous improvement:
   - Stay informed about new cloud services and cost optimization strategies.
   - Regularly update the framework to incorporate best practices and new technologies.

LSCOST01-BP02 Analyze and optimize vendor cost structures and economic relationships

Vendor economics focuses on optimizing the costs associated with various cloud vendors, software providers, and third-party services used in life sciences research. It aims to maximize value while maintaining the necessary tools and services for effective research.

Desired outcome: An optimized vendor solution that delivers maximum value through strategic multicloud partnerships, volume-based pricing agreements, and streamlined vendor management processes, resulting in reduced overall cloud and software costs while maintaining access to research tools and services.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Create a vendor management system that tracks cloud-related contracts, renewals, and usage.

Implement a multicloud strategy to use AWS services and negotiate better rates.

Develop a standardized process for evaluating and onboarding new cloud services or research tools.

Establish volume-based discounts with key software providers used across multiple research projects.

Implementation steps

1. Establish a vendor management system:
   - Select and implement a vendor management solution.
   - Document contract terms, renewal dates, and pricing structures.
   - Set up automated alerts for contract renewals and reviews.

2. Implement a multicloud strategy:
    - Assess current cloud service providers and their offerings.
    - Develop migration plans for workload distribution.
    - Create policies for selecting cloud services across providers.
3. Standardize your vendor evaluation process:
    - Create a formal evaluation checklist for new services.
    - Define approval workflows for new vendor onboarding.
    - Establish security and regulatory requirements.
4. Negotiate volume-based agreements:
    - Identify high-usage software and services.
    - Analyze usage patterns across research projects.
    - Develop negotiation strategies for bulk pricing.
5. Implement continuous improvement:
    - Schedule regular vendor performance reviews.
    - Monitor market trends and new offerings.
    - Update vendor management strategies.

LSCOST01-BP03 Build the right skills and fostering a cost-aware culture

Building the right skills and fostering a cost-aware culture is crucial for effective cloud financial management in the life sciences industry. This involves educating teams about the financial implications of their cloud usage and empowering them to make cost-effective decisions.

Desired outcome: A cost-conscious organizational culture where team members possess the knowledge and skills to make financially informed cloud decisions, supported by decentralized cost management advocates that drive continuous cost optimization without compromising research quality or regulatory requirements.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Develop focused training programs for employees that foster a cost-aware culture and teach optimization techniques for compliance-based workloads.

Implement a gamification system that rewards research teams for cost-effective cloud usage while maintaining research quality (for example, AWS GameDay).

Create a program where individuals from different departments are trained as cloud cost management advocates to establish decentralized Cloud Financial Management (CFM) capabilities across the organization.

Implementation steps

1. Develop training programs:
   - Create role-specific training modules.
   - Design hands-on workshops for cloud cost optimization.
   - Develop compliance-focused training materials.
2. Implement a gamification system:
   - Design reward mechanisms for cost-effective cloud usage.
   - Set up AWS GameDay framework.
   - Define metrics for measuring success.
3. Create a cloud cost management advocacy program:
   - Identify potential advocates across departments.
   - Develop advocate training curriculum.
   - Define advocate responsibilities.
4. Foster collaborative learning:
   - Organize cost optimization workshops.
   - Create cross-functional teams.
   - Schedule knowledge-sharing sessions.

Cost-effective resources

LSCOST02: How effectively are you using AWS credit and investment programs?

AWS offers various credit programs to optimize costs and accelerate cloud adoption. Using these programs can lead to significant savings and enhanced cloud capabilities.

Best practices

- LSCOST02-BP01 Implement a strategic approach to AWS credit program utilization

LSCOST02-BP01 Implement a strategic approach to AWS credit program utilization

Maximize financial benefits by systematically tracking, allocating, and using available AWS credits. Credits assist to support innovation, migration, and strategic technology initiatives. This practice fosters cross-organizational visibility and efficient resource utilization.

Desired outcome: A comprehensive AWS credit optimization program that maximizes financial value through strategic credit allocation, proactive governance, and cross-organizational visibility. This results in accelerated innovation initiatives, reduced overall cloud costs, and enhanced return on AWS investments while blocking credit expiration and waste.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Develop a comprehensive strategy for AWS credit utilization that encompasses identification, management, and optimization of the available credit programs. This involves establishing centralized governance with clear accountability, creating transparent processes for credit allocation and sharing across teams, and maintaining proactive engagement with AWS to maximize program benefits.

Success requires balancing strategic priorities with tactical performance. Verify that credits support both immediate cost optimization needs and long-term innovation goals while maintaining visibility through effective tracking and reporting mechanisms.

Implementation steps

1. Audit existing credits:
   a. Document the current AWS credits across your organization, including types, amounts, expiration dates, and utilization rates.
   b. Identify unused credits and immediate optimization opportunities.
2. Establish credit governance:
   a. Designate a credit management team and create centralized tracking using AWS Cost Explorer.

   b. Set up automated alerts for expiration dates and implement standardized allocation policies.
3. Deploy allocation framework:
   a. Create criteria for credit requests based on business impact and innovation potential.
   b. Implement tagging strategies and cross-departmental sharing processes to maximize utilization.
4. Monitor and optimize:
   a. Set up real-time tracking with AWS Cost and Usage Reports and regular review cycles.
   b. Create dashboards for stakeholders and establish key performance indicators (KPIs) to measure credit utilization efficiency.

Manage demand and supply resources

LSCOST03: How do you optimize R&D infrastructure costs based on the stage of product development and scale of operations?

Life sciences R&D infrastructure needs vary significantly across different development stages, from early discovery through clinical trials to commercialization. Each stage has unique computational, storage, and processing requirements. Without proper alignment between infrastructure and development stage, organizations risk over-provisioning resources or hindering research progress. Consider how infrastructure can be optimized while maintaining regulatory adherence and data integrity across discovery, pre-clinical, clinical, and commercial stages.

Best practices

- LSCOST03-BP01 Align infrastructure capacity with R&D development stages

LSCOST03-BP01 Align infrastructure capacity with R&D development stages

Design infrastructure that can adapt to changing requirements across different R&D phases while maintaining regulatory adherence and research integrity.

Desired outcome: An adaptive infrastructure framework that efficiently matches computational resources to R&D stage requirements, minimizing costs through automated scaling while improving regulatory adherence and maintaining research velocity across the development phases.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To effectively optimize research and development (R&D) infrastructure costs across development stages, establish a clear framework for identifying and categorizing different R&D phases and their specific infrastructure requirements.

Create a resource mapping matrix that defines computational, storage, and processing needs for each phase. Implement automated scaling solutions that adjust infrastructure based on research phase triggers and actual usage patterns.

Establish monitoring and metrics to track resource utilization and costs across different stages. Verify that regulatory requirements are built into the infrastructure design from the start, with appropriate validation and documentation processes.

Implementation steps

1. Document current infrastructure usage patterns across R&D phases and map regulatory requirements to each stage. Define key performance indicators and cost metrics, and establish baseline costs for each R&D phase. This comprehensive assessment provides a foundation for optimization efforts and maintains regulatory standards throughout the process.
2. Create flexible templates for different R&D phase configurations, and design automated scaling triggers based on phase transitions. Implement robust monitoring and alerting systems, and develop cost allocation mechanisms. This design phase focuses on building a responsive and efficient infrastructure that can adapt to changing research needs while maintaining cost visibility.
3. Deploy the phase-specific infrastructure templates and configure automated scaling policies. Implement resource tagging for detailed cost tracking and set up governance controls. This phase puts the design into action, verifying that the infrastructure can efficiently support various R&D stages while maintaining regulatory adherence.
4. Continuously monitor resource utilization and costs, regularly reviewing and adjusting scaling thresholds. Use AWS right-sizing technologies such as AWS Compute Optimizer, AWS Cost Optimization Hub, and AWS Trusted Advisor to validate scaling policy implementations across

each phase. Conduct periodic regulatory validations and pursue ongoing optimization based on usage patterns. These tools provide data-driven recommendations to optimize resource utilization, validate scaling configurations, and identify cost-saving opportunities. This ongoing process keeps the infrastructure efficient, cost-effective, and compatible as research needs evolve over time.

Optimize over time

LSCOST04: How do you optimize research data lifecycle costs while maintaining regulatory requirements over time?

Life sciences research generates massive amounts of data with varying retention requirements and access patterns. Without proper optimization, organizations risk high storage costs for rarely accessed data while still needing to maintain adherence. Organizations need to balance cost optimization with regulatory requirements for data retention, accessibility, and integrity throughout the research lifecycle.

Best practices

- LSCOST04-BP01 Implement tiered storage strategies aligned with data access patterns and retention requirements
- LSCOST04-BP02 Implement data retention and deletion policies aligned with regulatory requirements

LSCOST04-BP01 Implement tiered storage strategies aligned with data access patterns and retention requirements

Design and implement a tiered storage architecture that matches storage costs to data value and access patterns while verifying adherence to retention requirements. Classify research data based on access frequency, regulatory requirements, and business value to determine appropriate storage tiers. Move infrequently accessed data to lower-cost storage while maintaining required accessibility and integrity controls.

Desired outcome: Tiered storage architecture that optimizes data lifecycle costs through strategic placement of research data across storage tiers based on access patterns and requirements,

verifying regulatory adherence while minimizing storage expenses and maintaining seamless data accessibility throughout the research lifecycle.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Tiered storage strategies optimize costs while maintaining adherence in life sciences data management. Categorize research data based on access patterns, regulatory requirements, and business value to determine appropriate storage tiers. Match storage solutions to data requirements, from frequently accessed data needing high performance to archival data requiring long-term retention. Consider regulations (GxP, HIPAA, GDPR) when designing storage tiers. Implement necessary controls for security, encryption, and audit trails. Automate data movement between tiers based on defined policies while verifying that data remains accessible and protected throughout its lifecycle.

Implementation steps

1. Conduct a comprehensive inventory of research data. Classify each dataset based on its type, access frequency, value, and regulatory requirements.
2. Evaluate available storage options and create 3-4 tiers based on performance needs and cost considerations. Document clear policies for each tier, including what type of data belongs in each.
3. Create policies for moving data between tiers, and define triggers for these transitions. Verify that each policy complies with relevant regulatory requirements.
4. Configure storage classes in your cloud environment and set up appropriate access controls and encryption. Implement automated lifecycle policies to manage data movement between tiers.
5. Create procedures for initial data classification and storage. Establish processes for data retrieval and define clear roles and responsibilities for managing the tiered storage system.
6. Thoroughly test data movement between tiers and validate that access controls and encryption are working as intended. Perform mock audits to check adherence to regulatory requirements.
7. Train researchers and IT staff on the new storage strategy. Create comprehensive documentation of the tiered storage architecture, policies, and user guides for accessing data in different tiers.
8. Implement monitoring for storage usage and costs. Regularly review access patterns and adjust tiering policies as needed to optimize performance and cost-efficiency.

9. Perform quarterly reviews of storage usage and costs. Annually reassess the overall tiered storage strategy and update policies based on changing research needs and regulatory requirements.

LSCOST04-BP02 Implement data retention and deletion policies aligned with regulatory requirements

Develop and enforce comprehensive data retention and deletion policies that comply with industry regulations while minimizing unnecessary storage costs. These policies should clearly define how long different types of research data must be retained, how it should be stored, and when it can be safely deleted or archived.

Desired outcome: A comprehensive, automated data retention and deletion framework that improves regulatory adherence while minimizing storage costs through intelligent lifecycle management, enabling safe data disposal when appropriate and maintaining complete audit trails for data lifecycle decisions across research phases.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Organizations should first analyze their data usage patterns and regulatory requirements across different research stages. Implement automated tools for monitoring data access patterns and costs. Develop clear policies for data classification and retention requirements. Create automated workflows for moving data between storage tiers based on age and usage while maintaining audit trails. Regularly review and optimize storage costs while verifying that regulatory requirements are met.

Implementation steps

1. Analyze current data storage costs and usage patterns across research phases.
   a. Document regulatory requirements for different data types and research stages.
   b. Define metrics for measuring storage optimization success and establish baseline costs.
   c. Create data classification policies that balance research needs with cost optimization opportunities.
2. Create comprehensive data lifecycle policies that define storage tiers, retention periods, and movement criteria.

   a. Develop automation rules for data movement between storage tiers based on age and access patterns.
   b. Establish regulatory documentation requirements for data lifecycle changes.
   c. Define processes for data retrieval and restoration when needed.
3. Deploy automated tools for monitoring data usage and implementing lifecycle policies.
   a. Configure storage tiering based on defined policies and regulatory requirements.
   b. Implement automated documentation and audit trail creation for each data movement.
   c. Set up cost monitoring and optimization tools with appropriate alerting.
4. Regularly review storage costs and usage patterns to identify optimization opportunities.
   a. Adjust automation rules based on changing research needs and cost patterns.
   b. Conduct periodic audits to verify that requirements are met.
   c. Update policies and procedures based on new regulatory requirements or research needs.

Sustainability

Sustainability reporting is increasingly recognized as both important and strategically beneficial for life sciences companies. While it's not yet a global legal mandate, the trend is moving toward mandatory Environmental, Social, and Governance (ESG) and sustainability disclosure, with voluntary reporting already strongly encouraged by investors and customers.

In the European Union, sustainability reporting has become mandatory for many life sciences companies under the Corporate Sustainability Reporting Directive (CSRD).

The CSRD requires in-scope companies, including large life sciences and biotech firms, to prepare detailed sustainability reports covering:

- Environmental impact (including climate change, resource use, pollution, and biodiversity)
- Social factors (employee rights, diversity, and community engagement)
- Governance (business ethics and risk management)
- Supply chain transparency
- Double materiality (both the impact of sustainability matters on the company and the company's impact on society and the environment)

Companies must report in accordance with the European Sustainability Reporting Standards (ESRS), and the requirements are being phased in, with additional sector-specific standards expected by 2026. The sustainability pillar focuses on minimizing the environmental impact of running life sciences workloads in the cloud while, at the same time, meeting critical regulatory requirements, data integrity and retention needs as well as the specific scientific computing demands.

Life sciences organizations face unique sustainability challenges due to industry specifics such as:

- Compute-intensive processes for research, genomic analysis, and molecular modeling
- Long-term data retention requirements for regulatory adherence
- High-performance computing needs for drug discovery and clinical analysis
- Large-scale data collection and processing from laboratory instruments
- Global collaboration requirements across research sites and clinical trials

Focus areas

- Design principles
- Research computing optimization
- Sustainability metric tracking and reporting
- Data management efficiency in data analytics and data lifecycle
- Sustainability in manufacturing environments
- Sustainability in clinical trials

Design principles

When it comes to sustainability, there are a number of design principles, many shared with other pillars, that support impactful change:

- **Optimize research computing resources:** Research computing in life sciences often requires significant computational power for tasks like molecular modeling and genomic analysis. Organizations should implement practices that balance research throughput with resource efficiency. This involves designing workloads that maximize the use of computing resources during active research while scaling down during quiet periods. When implementing research pipelines, consider scheduling mechanisms that batch similar computations together and optimize the use of specialized hardware accelerators. The goal is to maintain or improve scientific output while reducing the overall resource footprint.
- **Implement intelligent data analytics and data lifecycle management for research:** Life sciences organizations face a double challenge of processing vast amounts of real-world data for insights while also maintaining extensive data archives for regulatory adherence. For real-world evidence analysis, organizations must efficiently process large-scale, diverse datasets from multiple sources including electronic health records, claims data, and patient registries. This requires optimizing analytical workflows and storage patterns to handle high-volume data processing while minimizing resource usage. Simultaneously, organizations must maintain data for extended periods to meet GxP requirements, necessitating efficient long-term storage strategies. Design data architectures that balance these needs by intelligently managing both active analytical workloads and long-term storage. For analytical workloads, implement efficient processing patterns that minimize unnecessary data movement and optimize compute resource usage. For compliance-aligned data storage, create tiered architectures that move data through appropriate storage classes based on access patterns while maintaining regulatory accessibility. This requires careful consideration of data classification, retention policies, and access patterns unique to life sciences workflows.

- **Maximize laboratory resource efficiency:** Modern laboratories generate massive amounts of data from various instruments and systems. Design architectures that optimize the integration between laboratory instruments and cloud resources, minimizing unnecessary data transfer and storage. Consider implementing edge computing where appropriate to process and filter data closer to the source. Laboratory automation systems should be designed with sustainability in mind, improving the efficiencyt of compute resources while maintaining the required level of control and monitoring.
- **Build sustainable clinical trial infrastructure:** Clinical trials involve complex data collection and processing pipelines across multiple sites and systems. Design these systems to efficiently collect and process data, avoiding redundant storage and unnecessary data movement. Implement monitoring systems that provide oversight while minimizing resource usage. Consider how to optimize multi-site trial infrastructure through regional processing and efficient data aggregation patterns, especially for decentralized trials that may generate significant amounts of remote monitoring data.

Resources

- Healthy people, healthy planet: Supporting sustainability in healthcare with the cloud

Research computing optimization

LSSUS01: How do you optimize the scheduling and execution of research computing pipelines?

Life sciences research involves intensive computational workloads that can lead to significant energy consumption and resource waste if not properly managed. Organizations need to balance the demands of complex research computations with sustainable resource usage, providing maximum scientific output while minimizing environmental impact.

Best practices

- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage
- LSSUS01-BP02 Use energy efficient hardware and services

LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage

Design research computing workloads to optimize resource utilization and minimize energy consumption through strategic workload separation and queue management. Implement compute environments that align with specific computational demands to maximize hardware efficiency and reduce idle time. Use multi-queue strategies that match workload characteristics to appropriate compute resources, enabling concurrent processing of diverse research tasks while minimizing environmental impact.

Desired outcome: Achieve optimal resource utilization across research computing infrastructure while reducing energy consumption and operational costs. Implement workload-specific compute environments that maximize hardware efficiency for different types of life sciences analyses.

Common anti-patterns:

- You run computational workloads on the same compute environment regardless of resource requirements.
- You don't implement queue management strategies to optimize resource allocation.
- You use oversized compute instances for lightweight computational tasks.
- You don't use spot instances for fault-tolerant research workloads.
- You run compute resources continuously without considering actual utilization patterns.
- You don't separate CPU-intensive, GPU-intensive, and memory-intensive workloads.

Benefits of establishing this best practice:

- Reduce energy consumption and carbon footprint of research computing operations.
- Lower operational costs through optimized resource utilization and spot instance usage.
- Enable concurrent processing of diverse research workloads without resource conflicts.
- Achieve better cost predictability through workload-specific resource allocation.
- Support regulatory requirements for sustainable research practices.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Research computing workloads in life sciences vary significantly in their resource requirements, from CPU-intensive sequence alignment to GPU-accelerated molecular dynamics simulations. Implementing workload-specific compute environments verifies that each type of analysis runs on optimally configured resources, reducing both energy consumption and costs. This approach is particularly important for life sciences organizations that must balance research productivity with sustainability goals while maintaining adherence to regulatory requirements.

Consider the computational characteristics of your research workloads when designing compute environments. Genomics pipelines typically require high CPU and memory resources, while protein structure prediction benefits from GPU acceleration. By separating these workloads into dedicated environments, you can achieve better resource utilization and reduce energy waste from oversized or underutilized compute instances.

Implementation steps

1. Analyze and categorize your research computing workloads by resource requirements:
   - Profile CPU, memory, and GPU utilization patterns for different analysis types.
   - Use AWS Compute Optimizer to identify optimal instance types for each workload category.
   - Consider AWS Batch for orchestrating containerized research workloads.
2. Design compute environments aligned with workload characteristics:
   - Create CPU-optimized environments for sequence alignment and variant calling.
   - Configure GPU-optimized environments for molecular dynamics and deep learning.
   - Set up memory-optimized environments for genome assembly and phylogenetic analysis.
   - Consider AWS ParallelCluster for HPC workload management.
3. Implement multi-queue strategy for efficient resource allocation:
   - Configure high-priority queues for time-sensitive analyses using on-demand instances.
   - Set up spot instance queues for fault-tolerant workloads like Monte Carlo simulations.
   - Use Amazon EC2 Spot Fleet for cost-effective processing of batch workloads.
   - Implement AWS Batch job queues with different compute environments.
4. Enable automated scaling and resource optimization:
   - Configure auto scaling policies based on queue depth and resource utilization.
   - Use AWS Auto Scaling to automatically adjust compute capacity.
   - Implement scheduled scaling for predictable workload patterns.

- Monitor resource utilization with Amazon CloudWatch metrics.

5. Establish monitoring and optimization processes:
    - Track energy efficiency metrics using normalized compute hours per analysis.
    - Monitor cost and utilization patterns with AWS Cost Explorer.
    - Set up alerts for underutilized resources using Amazon CloudWatch alarms.
    - Regularly review and optimize compute environment configurations.

Resources

Related best practices:

- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads
- LSSUS03-BP01 Optimize Data Management for Sustainability in Life Sciences

Related documents:

- Accelerating Life Sciences Research with HPC on AWS
- Navigating GPU Challenges: Cost Optimizing AI Workloads on AWS

Related videos:

- AWS re:Invent 2023 - HPC on AWS for semiconductors and healthcare life sciences (CMP214)
- AWS Summit 2023 - Building sustainable research computing on AWS

Related examples:

- HPC Workloads on AWS
- AWS Batch for Life Sciences
- Genomics Workflows on AWS

Related tools:

- AWS Batch
- AWS ParallelCluster

- Amazon EC2
- AWS Compute Optimizer
- Amazon EC2 Spot Fleet

LSSUS01-BP02 Use energy efficient hardware and services

Select energy-efficient hardware architectures and managed services that optimize power consumption while maintaining research computing performance. Prioritize modern processor architectures and cloud services that incorporate built-in sustainability optimizations to reduce the environmental impact of computational workloads. Implement hardware and service selection strategies that balance performance requirements with energy efficiency goals.

Desired outcome: Achieve significant energy consumption reduction in research computing operations while maintaining or improving computational performance. Implement hardware and service architectures that provide optimal performance-per-watt ratios for life sciences workloads.

Common anti-patterns:

- You default to traditional x86 architectures without evaluating energy-efficient alternatives.
- You don't consider managed services that provide built-in sustainability optimizations.
- You don't evaluate the total cost of ownership including energy consumption.
- You run custom infrastructure instead of using optimized managed services.
- You don't monitor and measure energy efficiency metrics across your compute infrastructure.

Benefits of establishing this best practice:

- Reduce energy consumption by up to 60% through modern processor architectures like AWS Graviton.
- Lower operational costs through improved performance-per-watt ratios.
- Improve research productivity with automatically optimized resource allocation.
- Support organizational sustainability goals and regulatory adherence requirements.
- Benefit from continuous service improvements and optimizations provided by managed services.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Energy-efficient hardware selection is crucial for life sciences organizations seeking to reduce their environmental impact while maintaining research computing capabilities. Modern processor architectures like AWS Graviton processors offer significant energy efficiency improvements over traditional x86 architectures, particularly for compute-intensive workloads common in genomics, proteomics, and molecular modeling. These efficiency gains compound over time, making hardware selection a critical sustainability decision.

Managed services provide additional sustainability benefits by incorporating built-in optimizations that individual organizations would struggle to implement independently. Services like AWS HealthOmics, AWS HealthLake, and AWS Batch continuously optimize resource utilization, automatically scale based on demand, and use the latest energy-efficient infrastructure improvements. This approach allows research teams to focus on scientific outcomes while benefiting from ongoing sustainability improvements.

Implementation steps

1. Evaluate and migrate to energy-efficient processor architectures:
   - Assess workload compatibility with AWS Graviton processors for ARM-based computing.
   - Use Amazon EC2 M6g, C6g, and R6g instances for general-purpose, compute-optimized, and memory-optimized workloads.
   - Benchmark performance and energy consumption for representative research workloads.
   - Consider AWS Graviton3 processors for the latest efficiency improvements.
2. Use managed services with built-in sustainability optimizations:
   - Migrate genomics workflows to AWS HealthOmics for automated resource optimization.
   - Use AWS HealthLake for healthcare data processing with built-in efficiency features.
   - Implement AWS Batch for containerized workloads with automatic scaling and optimization.
   - Consider Amazon SageMaker AI for machine learning workloads with energy-efficient training.
3. Optimize service configurations for energy efficiency:
   - Enable AWS Auto Scaling to automatically adjust capacity based on demand.
   - Configure AWS Lambda for event-driven processing to minimize idle resource consumption.
   - Use Amazon ECS with AWS Fargate for serverless container execution.
   - Implement AWS Step Functions for efficient workflow orchestration.
4. Monitor and measure energy efficiency improvements:

- Track performance-per-watt metrics using Amazon CloudWatch custom metrics.
- Monitor cost savings and energy reduction.
- Implement AWS Config rules to adhere to energy efficiency policies.

5. Establish continuous optimization processes:
   - Regularly review AWS service updates for new energy efficiency features.
   - Conduct periodic assessments of hardware and service selection decisions.
   - Set up automated alerts for suboptimal resource utilization patterns.
   - Create feedback loops to incorporate energy efficiency learnings into future architecture decisions.

Resources

Related best practices:

- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage
- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads

Related documents:

- Improving Sustainability and Price Performance Using AWS Graviton–Based Instances with Pinterest

Related videos:

- How to optimize workloads for sustainability using AWS Graviton-based EC2 instances

Related examples:

- AWS Graviton Getting Started
- AWS HealthOmics Workflows

Sustainability metric tracking and reporting

LSSUS02: How do you monitor and report the sustainability metrics for your research workloads?

Measuring and tracking the environmental impact of research operations presents unique challenges in life sciences due to the diverse nature of workloads and the need to maintain scientific validity. Organizations need effective ways to quantify and report their sustainability efforts while aligning these metrics with both research outcomes and regulatory requirements.

Best practices

- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads

LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads

Establish comprehensive tagging strategies and normalized metrics to track and measure sustainability performance across research workloads. Implement proxy metrics that correlate resource consumption with research outputs to enable meaningful comparisons and optimization opportunities. Create sustainability KPIs that combine resource utilization data with research milestones to drive continuous improvement in energy efficiency and environmental impact.

Desired outcome: Establish measurable sustainability metrics that enable data-driven optimization of research workloads and provide clear visibility into environmental impact across different research activities and architectural patterns.

Common anti-patterns:

- You don't implement consistent tagging strategies across research workloads and resources.
- You measure absolute resource consumption without normalizing for research outputs.
- You don't establish baseline metrics to track sustainability improvements over time.
- You collect metrics but don't integrate them into research workflow planning and optimization.
- You don't correlate resource consumption with actual research deliverables and milestones.
- You rely solely on cost metrics without considering environmental impact measurements.

- You don't establish regular review cycles to assess and act on sustainability metrics.

Benefits of establishing this best practice:

- Enable data-driven decisions for optimizing research workload sustainability.
- Provide clear visibility into environmental impact across different research activities.
- Support regulatory adherence and sustainability reporting requirements.
- Align research operations with organizational sustainability goals and commitments.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Sustainability metrics for research workloads require a strategic approach that balances measurement granularity with practical implementation. Life sciences organizations must track resource consumption in ways that correlate with research outputs, enabling meaningful comparisons across different types of analyses and computational approaches. This is particularly important given the diverse nature of life sciences computing, from high-throughput genomics processing to complex molecular modeling simulations.

Proxy metrics serve as practical indicators of sustainability performance when direct energy measurements are not feasible. By normalizing resource consumption against research outputs (such as vCPU hours per genome processed or GPU hours per molecular structure analyzed), organizations can identify optimization opportunities and track improvements over time. Integration with the AWS Customer Carbon Footprint Tool provides additional context by correlating proxy metrics with actual carbon footprint data.

Implementation steps

1. Establish comprehensive tagging strategy for research workloads:
    - Implement consistent tags for research type (genomics, proteomics, drug discovery).
    - Tag resources by project phase (discovery, validation, production).
    - Include architectural pattern tags (batch processing, interactive analysis, real-time).
    - Use AWS Resource Groups and Tag Editor for centralized tag management.
2. Define and implement normalized sustainability metrics:
    - Calculate vCPU hours per genome processed for genomics workloads.

- Track GPU hours per molecular structure analyzed for computational chemistry.
- Measure storage efficiency through data processed per TB stored.
- Use Amazon CloudWatch custom metrics to track normalized performance indicators.

3. Create sustainability KPIs aligned with research outputs:
   - Combine resource utilization metrics with research milestone achievements.
   - Track energy efficiency improvements over time using baseline comparisons.
   - Implement cost-per-research-output metrics for comprehensive sustainability assessment.
   - Use AWS Cost and Usage Reports for detailed resource consumption analysis.
4. Integrate metrics with AWS Customer Carbon Footprint Tool:
   - Correlate proxy metrics with account and Region-level carbon footprint data.
   - Establish regular reporting cycles for sustainability performance.
   - Create dashboards that combine resource efficiency with environmental impact.
   - Use Quick for sustainability reporting and visualization.
5. Establish continuous monitoring and optimization processes:
   - Integrate sustainability metrics into research workflow planning.
   - Conduct regular sustainability reviews with research teams.
   - Create feedback loops to incorporate sustainability learnings into future research planning.

Resources

Related best practices:

- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage
- LSSUS01-BP02 Leverage energy efficient hardware and services
- LSSUS03-BP01 Optimize Data Management for Sustainability in Life Sciences

Related documents:

- AWS Customer Carbon Footprint Tool

Related examples:

- AWS Solutions for Sustainability

Related tools:

- AWS Customer Carbon Footprint Tool

Data management efficiency in data analytics and data lifecycle

LSSUS03: How do you maximize efficiency in processing large-scale life sciences data?

Life sciences organizations must process and store massive amounts of data while meeting strict regulatory requirements for retention and accessibility. This creates a significant sustainability challenge in balancing the need for efficient data processing with regulatory requirements and long-term storage needs.

Best practices

- LSSUS03-BP01 Optimize data management for sustainability in life sciences
- LSSUS03-BP02 Process data closer to source

LSSUS03-BP01 Optimize data management for sustainability in life sciences

Implement data management practices that reduce redundant storage, optimize processing efficiency, and minimize data movement while maintaining regulatory requirements. Establish centralized data catalogs and automated lifecycle policies to optimize your storage tier utilization based on access patterns. Design data architectures that balance accessibility requirements with energy efficiency goals.

Desired outcome: Achieve significant reduction in storage footprint and energy consumption through optimized data lifecycle management, deduplication strategies, and intelligent storage tiering while maintaining adherence to life sciences regulatory requirements.

Common anti-patterns:

- You store data in the same storage tier regardless of access patterns or lifecycle requirements.
- You don't implement deduplication strategies for redundant research datasets.

- You move large datasets frequently between regions or storage systems without considering energy impact.
- You don't establish data retention policies aligned with regulatory requirements and business needs.
- You maintain multiple copies of reference datasets across different research projects.
- You don't use compression techniques for archival and infrequently accessed data.
- You don't monitor storage utilization and costs to identify optimization opportunities.

Benefits of establishing this best practice:

- Reduce storage costs through intelligent lifecycle management and tiering.
- Lower energy consumption of storage systems by using optimal storage classes.
- Improve data accessibility through centralized cataloging and metadata management.
- Improve regulatory adherence through automated retention and archival policies.
- Reduce data transfer costs and energy consumption through strategic data placement.
- Enable better research collaboration through shared, deduplicated reference datasets.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Life sciences organizations generate and consume vast amounts of data across research, clinical, and manufacturing operations. Effective data management for sustainability requires understanding data access patterns, regulatory retention requirements, and the energy implications of different storage approaches. The key is implementing automated policies that transition data through appropriate storage tiers while maintaining accessibility for research and regulatory needs.

Centralized data catalogs play a crucial role in reducing redundancy and improving data discoverability. By implementing services like AWS Lake Formation for research data or AWS HealthLake for healthcare data, organizations can remove duplicate datasets, improve data governance, and reduce overall storage requirements. This approach is particularly important for reference datasets, genomic databases, and clinical trial data that may be accessed across multiple research projects.

Implementation steps

1. Establish centralized data catalogs and governance:
   - Centralize the management of data by implementing services such as AWS Lake Formation for research data cataloging and governance.
   - Use data lakes such as AWS HealthLake for healthcare and clinical data management.
   - Create standardized metadata schemas for different data types.
   - Establish data ownership and access control policies.
2. Implement intelligent storage lifecycle management:
   - Configure lifecycle policies to automatically transition data between storage classes.
   - Use Amazon S3 Standard for active research data requiring frequent access.
   - Implement dynamic tiering for data with changing or unknown access patterns.
   - Archive audit and reference data to durable, infrequent access storage such as Amazon Glacier or S3 Glacier Deep Archive.
3. Deploy deduplication and compression strategies:
   - Implement data deduplication for reference datasets and genomic databases.
   - Use compression algorithms appropriate for different data types (genomic, imaging, clinical).
   - Use service compression features for automated optimization.
   - Create shared reference datasets to remove redundant storage across projects.
4. Optimize data placement and movement:
   - Analyze data access patterns.
   - Implement regional data placement strategies to minimize transfer costs.
   - Use transfer services such as AWS DataSync for efficient data transfer and synchronization.
   - Consider AWS Storage Gateway for hybrid storage optimization.
5. Monitor and continuously optimize storage efficiency:
   - Use AWS Cost Explorer to track storage costs and utilization patterns.
   - Implement Amazon CloudWatch metrics for storage efficiency monitoring.
   - Set up automated alerts for storage anomalies and optimization opportunities.
   - Conduct regular reviews of data lifecycle policies and storage class effectiveness.

Resources

Related best practices:

- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage
- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads

Related documents:

- Sustainability Pillar - AWS Well-Architected Framework

Related videos:

- AWS re:Invent 2024 - Build and optimize a data lake on Amazon S3 (STG323)
- AWS re:Invent 2024 - Data-driven sustainability with AWS (SUS201)
- Securely Store, Transform, Query and Analyze Health Data with AWS HealthLake

Related examples:

- Guidance for Data Lakes on AWS

Related tools:

- AWS Lake Formation
- AWS HealthLake
- Amazon S3 Intelligent-Tiering
- Amazon Glacier
- AWS DataSync

LSSUS03-BP02 Process data closer to source

Optimize data processing locations to minimize network usage and reduce energy consumption associated with data movement. Implement edge computing and hybrid architectures that process large datasets near their generation points, particularly for bandwidth-intensive applications like genomic sequencing and imaging workflows. Use managed services that provide optimized resource utilization and automatic scaling to reduce infrastructure overhead.

Desired outcome: Significantly reduce network bandwidth usage and associated energy consumption by processing data at optimal locations relative to data sources, while maintaining processing performance and regulatory requirements.

Common anti-patterns:

- You transfer large datasets to centralized processing locations without considering network and energy costs.
- You don't evaluate edge computing options for bandwidth-intensive research applications.
- You process data in regions distant from data generation points without justification.
- You don't consider data sovereignty and regulatory requirements when choosing processing locations.
- You transfer raw data for processing instead of implementing preprocessing at the edge.

Benefits of establishing this best practice:

- Reduce network bandwidth costs and energy consumption for large dataset processing.
- Improve processing performance by reducing network latency for data-intensive operations.
- Lower infrastructure costs through optimized resource utilization and managed service adoption.
- Enhance data security and regulatory adherence by minimizing data movement across network boundaries.
- Enable real-time processing capabilities for time-sensitive research applications.
- Support hybrid and multi-cloud architectures that optimize for both performance and sustainability.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Life sciences research generates massive datasets that traditionally require significant network resources to transfer to centralized processing locations. This approach is particularly inefficient for applications like genomic sequencing, cryo-electron microscopy, and high-resolution imaging where raw data volumes can reach terabytes per experiment. Processing data closer to its source reduces both network energy consumption and processing latency while often improving overall system performance.

Edge computing and hybrid architectures become essential when dealing with specialized equipment that generates large amounts of data continuously. For example, cryo-EM facilities, genomic sequencers, and imaging systems can benefit significantly from local preprocessing that reduces data volumes before cloud transfer. AWS services like AWS Outposts enable on-premises processing with cloud tools, while managed services provide automatic optimization without requiring dedicated infrastructure management.

Implementation steps

1. Analyze data generation patterns and processing requirements:
    - Map data sources and their typical output volumes and processing needs.
    - Identify bandwidth-intensive workflows that would benefit from edge processing.
    - Assess regulatory requirements for data processing locations.
    - Use and application discovery service to understand current data flow patterns.
2. Implement edge computing solutions for high-volume data sources:
    - Deploy edge compute solution such as AWS Outposts for on-premises processing of large genomic datasets.
    - Use AWS Snow Family devices for data processing in remote or bandwidth-constrained locations.
    - Implement IoT edge processing of sensor and instrument data.
    - Consider AWS Wavelength for ultra-low latency processing requirements.
3. Use managed services for optimized data processing:
    - Use services such as Amazon Kinesis Data Streams for real-time data processing and analytics.
    - Implement AWS Transfer Family for optimized file transfer and processing workflows.
    - Deploy AWS Batch at edge locations for containerized processing workloads.
    - Use Amazon SageMaker AI Edge for machine learning inference at data sources.
4. Optimize data preprocessing and filtering at source locations:
    - Implement data compression and filtering before cloud transfer.
    - Use AWS Lambda@Edge for lightweight data processing and transformation.
    - Deploy containerized preprocessing pipelines using Amazon ECS on AWS Outposts.
    - Implement quality control and data validation at source locations.
5. Monitor and optimize data movement and processing efficiency:

- Track data transfer volumes and costs using FinOps tools such as Cloud Intelligence Dashboards.
- Monitor processing performance and resource utilization with Amazon CloudWatch.
- Use AWS X-Ray to trace data processing workflows and identify optimization opportunities.
- Implement automated alerts for unusual data transfer patterns or processing inefficiencies.

Resources

Related best practices:

- LSSUS03-BP01 Optimize Data Management for Sustainability in Life Sciences
- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage
- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads

Related documents:

- Sustainability Pillar - AWS Well-Architected Framework
- AWS Outposts Documentation
- AWS Snow Family Documentation
- Amazon Kinesis Data Streams Documentation
- AWS Transfer Family Documentation
- Optimizing data transfers for high throughput life science instruments using AWS DataSync
- Cloud at the Edge for Healthcare and Life Sciences
- Genomics data and transfer storage use cases

Related videos:

- AWS re:Invent 2024 - AWS wherever you need it: From the cloud to the edge (HYB201)

Related examples:

- Guidance for Optimizing Data Architecture for Sustainability on AWS
- Building a GPU-enabled CryoEM workflow on AWS

Related tools:

- AWS Outposts
- AWS Snow Family
- AWS IoT Greengrass
- AWS Wavelength
- Amazon Kinesis Data Streams
- AWS Transfer Family
- AWS Lambda@Edge
- Amazon SageMaker AI Edge
- AWS Application Discovery Service
- AWS X-Ray

Sustainability in manufacturing environments

LSSUS04: How do you track resource usage in manufacturing environments?

Life sciences manufacturing environments consume substantial resources across complex processes, from bioprocessing to packaging. Organizations need effective ways to monitor and optimize resource consumption while maintaining product quality and GMP requirements, making resource tracking and optimization particularly challenging.

Best practices

- LSSUS04-BP01 Continuously improve the monitoring of resource consumption
- LSSUS04-BP02 Use digital twins to optimize resource usage through in silico experimentation

LSSUS04-BP01 Continuously improve the monitoring of resource consumption

Implement comprehensive monitoring systems that track resource consumption across manufacturing and laboratory equipment to enable data-driven sustainability improvements. Deploy IoT sensors and real-time monitoring solutions that provide visibility into energy

usage, material consumption, and operational efficiency patterns. Establish predictive analytics capabilities that identify optimization opportunities and support proactive maintenance strategies.

Desired outcome: Achieve comprehensive visibility into resource consumption patterns across manufacturing operations, enabling data-driven decisions that reduce energy usage, minimize waste, and optimize equipment efficiency while maintaining production quality and regulatory adherence.

Common anti-patterns:

- You don't monitor resource consumption at the equipment level, relying only on facility-wide measurements.
- You use fixed sampling rates for your equipment regardless of operational patterns.
- You don't implement predictive maintenance based on resource consumption patterns.
- You don't correlate resource consumption with production output and quality metrics.
- You don't establish baseline measurements to track sustainability improvements over time.

Benefits of establishing this best practice:

- Reduce energy consumption through equipment optimization and predictive maintenance.
- Minimize material waste and improve resource utilization efficiency.
- Enable predictive maintenance that reduces equipment downtime and extends asset lifecycles.
- Support regulatory adherence and sustainability reporting requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Manufacturing environments in life sciences require sophisticated monitoring approaches due to the critical nature of production processes and strict regulatory requirements. Equipment such as bioreactors, chromatography systems, and analytical instruments consume significant resources while requiring precise operational conditions. Implementing comprehensive monitoring enables organizations to optimize resource consumption without compromising product quality or regulatory adherence.

Real-time monitoring with IoT sensors provides granular visibility into equipment performance and resource consumption patterns. This data enables predictive analytics that can identify

inefficiencies, predict maintenance needs, and optimize operational parameters for sustainability. The key is implementing monitoring systems that balance data collection granularity with processing efficiency, using adaptive sampling rates that respond to operational conditions and baseline activity levels.

Implementation steps

1. Deploy IoT sensors for comprehensive equipment monitoring:
    - Install IoT sensors on critical manufacturing equipment (bioreactors, HVAC systems, analytical instruments).
    - Monitor energy consumption, water usage, compressed air consumption, and waste generation.
    - Use tools like AWS IoT Device Management for centralized sensor configuration and management.
    - Implement solutions such as AWS IoT Greengrass for edge processing and local data aggregation.
2. Establish real-time monitoring and data collection systems:
    - Use Amazon Kinesis Data Streams for real-time data ingestion from manufacturing equipment.
    - Implement AWS IoT Analytics for processing and analyzing manufacturing data.
    - Store time-series data in Amazon Timestream for efficient querying and analysis.
    - Create real-time dashboards using Quick for operational visibility.
3. Implement adaptive sampling and data efficiency strategies:
    - Configure dynamic sampling rates based on equipment operational states.
    - Use AWS IoT Rules Engine to filter and route data based on operational conditions.
    - Implement data compression and aggregation at the edge to reduce network usage.
    - Establish baseline activity monitoring to optimize data collection frequency.
4. Deploy predictive analytics for maintenance and optimization:
    - Use Amazon SageMaker AI to build predictive models for equipment maintenance.
    - Implement anomaly detection using Amazon Lookout for Equipment.
    - Create predictive analytics for resource consumption optimization.
    - Use AWS Lambda for automated responses to monitoring alerts and anomalies.

Resources

Related best practices:

- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads
- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage

Related documents:

- AWS IoT Core Documentation
- AWS IoT Analytics Documentation

LSSUS04-BP02 Use digital twins to optimize resource usage through in silico experimentation

Implement digital twin technologies to create virtual representations of manufacturing processes that enable in silico experimentation and optimization without consuming physical resources. Use these virtual environments to test different operational scenarios, optimize process parameters, and minimize resource consumption before implementing changes in physical systems. Use simulation capabilities to reduce the need for physical experiments while improving process efficiency and sustainability outcomes.

Desired outcome: Significantly reduce physical experimentation requirements and resource consumption by using digital twins for process optimization, while improving manufacturing efficiency and reducing time-to-market for process improvements.

Common anti-patterns:

- You rely solely on physical experiments for process optimization without considering digital simulation alternatives.
- You implement process changes without first testing them in virtual environments.
- You don't use historical data to improve digital twin accuracy and predictive capabilities.
- You don't validate digital twin predictions against real-world outcomes to improve model accuracy.

Benefits of establishing this best practice:

- Reduce physical experimentation costs and resource consumption.
- Accelerate process optimization cycles and reduce time-to-market for improvements.
- Minimize material waste and energy consumption during process development.
- Enable safe testing of extreme operational scenarios without risk to physical equipment.
- Improve process understanding and predictive capabilities for better decision-making.
- Support regulatory submissions with comprehensive simulation data and analysis.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Digital twins in life sciences manufacturing provide unprecedented opportunities to optimize processes while minimizing resource consumption and environmental impact. These virtual representations enable process development teams to explore optimization scenarios that would be costly, time-consuming, or potentially risky to test in physical systems. For example, chromatography process optimization can involve testing hundreds of parameter combinations virtually before implementing the most promising approaches in actual equipment.

The effectiveness of digital twins depends on the quality of underlying data and models. Life sciences manufacturing processes often involve complex biochemical interactions that require sophisticated modeling approaches. However, the investment in creating accurate digital twins pays dividends through reduced physical experimentation, faster optimization cycles, and improved process understanding. Integration with real-time monitoring data keeps digital twins accurate and provides valuable insights throughout the manufacturing lifecycle.

Implementation steps

1. Identify and prioritize manufacturing processes for digital twin development:
    - Assess processes with high resource consumption or frequent optimization needs.
    - Prioritize critical processes like chromatography, fermentation, and purification.
    - Evaluate data availability and modeling complexity for each process.
    - Use AWS IoT TwinMaker to create digital representations of manufacturing equipment.
2. Develop comprehensive digital twin models:
    - Create physics-based models using AWS SimSpace Weaver for complex process simulations.
    - Integrate historical process data using Amazon S3 and AWS Glue for data preparation.

- Use Amazon SageMaker AI to build machine learning models that enhance digital twin accuracy.
- Implement real-time data integration using AWS IoT Core and Amazon Kinesis.

3. Establish simulated experimentation capabilities:
   - Create simulation environments for testing different operational scenarios.
   - Implement parameter optimization algorithms using Amazon SageMaker AI.
   - Use AWS Batch for running large-scale simulation experiments.
   - Develop automated experiment design and execution workflows using AWS Step Functions.
4. Integrate digital twin insights into manufacturing operations:
   - Create dashboards using Quick for visualizing simulation results.
   - Implement automated recommendations based on digital twin optimization results.
   - Use AWS Lambda for real-time process adjustments based on digital twin predictions.
   - Establish feedback loops to continuously improve digital twin accuracy.
5. Validate and continuously improve digital twin performance:
   - Compare digital twin predictions with actual manufacturing outcomes.
   - Implement continuous learning capabilities using Amazon SageMaker AI.
   - Establish regular model updates and validation cycles.

Resources

Related best practices:

- LSSUS04-BP01 Continuously improve the monitoring of resource consumption
- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads
- LSSUS01-BP01 Design high-performance computing workloads to minimize energy usage

Related documents:

- AWS IoT TwinMaker Documentation
- AWS SimSpace Weaver Documentation
- Amazon SageMaker AI Documentation
- AWS Step Functions Documentation

Related examples:

- AWS IoT TwinMaker Samples

Related tools:

- AWS IoT TwinMaker

Sustainability in clinical trials

LSSUS05: How do you implement sustainable practices in clinical trials?

Clinical trials traditionally involve resource-intensive processes including patient travel, physical documentation, and data movement across multiple sites. Organizations need to balance the implementation of sustainable practices with maintaining trial integrity, patient safety, and regulatory adherence while reducing environmental impact.

Best practices

- LSSUS05-BP01 Design sustainable clinical trial architecture

LSSUS05-BP01 Design sustainable clinical trial architecture

Implement decentralized clinical trial architectures that minimize data movement and reduce environmental impact through regional data processing strategies and edge computing solutions. Design clinical trial systems that process data closer to trial populations, reducing network requirements while maintaining regulatory adherence and data integrity. Use standardized data management services and smart archiving strategies to optimize resource utilization throughout the clinical trial lifecycle.

Desired outcome: Achieve significant reduction in data transfer requirements and environmental impact of clinical trials while improving data accessibility, patient experience, and operational efficiency through decentralized architectures and edge computing solutions.

Common anti-patterns:

- You centralize your clinical trial data processing without considering regional or edge processing alternatives.
- You don't implement data filtering and aggregation at collection points, transferring raw data.
- You use proprietary data formats instead of standardized formats like FHIR for clinical data.
- You store your clinical trial data in the same storage tier regardless of access patterns.
- You don't implement automated data archiving strategies aligned with regulatory retention requirements.
- You don't consider patient travel reduction opportunities through remote monitoring and virtual visits.

Benefits of establishing this best practice:

- Reduce data transfer costs and energy consumption through edge processing and regional architectures.
- Improve patient experience and reduce travel-related emissions through decentralized trial designs.
- Lower operational costs through optimized data management and automated archiving strategies.
- Enhance data accessibility and analysis capabilities through standardized data formats.
- Support regulatory adherence through automated retention and archiving policies.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Clinical trials generate vast amounts of data from diverse sources including electronic health records, patient-reported outcomes, wearable devices, and medical imaging. Traditional centralized approaches require significant data movement and processing resources, contributing to environmental impact while potentially creating bottlenecks in data analysis. Decentralized clinical trial architectures address these challenges by processing data closer to its source and implementing intelligent data management strategies.

The shift toward decentralized clinical trials is driven by both sustainability goals and improved patient outcomes. By implementing edge computing solutions for patient monitoring and regional data processing strategies, organizations can significantly reduce data transfer requirements while maintaining the high standards of data quality and regulatory adherence required in clinical

research. This approach also enables more inclusive trial designs by reducing patient travel requirements and supporting remote participation.

Implementation steps

1. Implement decentralized data collection and processing architecture:
   - Deploy AWS IoT Greengrass at clinical sites for local data filtering and aggregation.
   - Implement AWS Direct Connect for secure, dedicated connectivity between clinical sites.
   - Configure AWS PrivateLink for secure communication between decentralized components.
2. Establish edge computing solutions for patient monitoring:
   - Deploy AWS IoT Core for device connectivity and data ingestion from wearable devices.
   - Use AWS IoT Greengrass for local processing of vital signs and patient-reported outcomes.
   - Implement Amazon Kinesis Data Streams for real-time data processing at the edge.
   - Configure AWS Lambda@Edge for lightweight data transformation and filtering.
3. Implement standardized clinical data management:
   - Use AWS HealthLake for FHIR-based clinical data management and standardization.
   - Implement Amazon S3 with intelligent tiering for clinical trial data storage.
   - Use AWS Glue for data integration and transformation across different clinical data sources.
   - Configure Amazon Redshift for clinical trial data analytics and reporting.
4. Deploy smart data archiving and lifecycle management:
   - Implement Amazon S3 lifecycle policies for automated data archiving based on regulatory requirements.
   - Use Amazon Glacier for long-term archival of completed trial data.
   - Configure AWS Config for monitoring and automated policy enforcement.
   - Implement AWS CloudTrail for comprehensive audit trails of data access and modifications.
5. Establish monitoring and optimization processes:
   - Use Amazon CloudWatch to monitor data transfer volumes and processing efficiency.
   - Establish regular reviews of decentralized architecture performance and optimization opportunities.

Resources

Related best practices:

- LSSUS03-BP02 Process data closer to source
- LSSUS02-BP01 Implement sustainability proxy metrics pipeline for research workloads
- LSSUS03-BP01 Optimize Data Management for Sustainability in Life Sciences

Related documents:

- Advance environmental sustainability in clinical trials using AWS
- Successful Decentralized Clinical Trials: A True Possibility with AWS in the Post-Pandemic Era

Conclusion

The Life Sciences Lens provides architectural best practices across the six pillars of the AWS Well-Architected Framework for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. The Framework provides a set of questions that allows you to review an existing or proposed architecture. It also provides a set of AWS best practices for each pillar. Using the Framework in your life sciences workloads assists you in producing stable and efficient systems, which allows you to focus on your functional requirements.

Contributors

The following individuals and organizations contributed to this document:

- Denny Daugherty, Principal Technical Account Manager, Amazon Web Services
- Eva Donaldson, Sr. Technical Account Manager, Amazon Web Services
- Ian Sutcliffe, Principal HCLS Tech Strategist, Amazon Web Services
- Jyothi Jayaraman, Sr. Partner Solutions Architect, Amazon Web Services
- Ran Zhang, Enterprise Support Lead, Amazon Web Services
- Avinash Gogineni, Enterprise Support Lead, Amazon Web Services
- Suresh Sankar, Enterprise Support Lead, Amazon Web Services
- Koushik Das, Enterprise Support Lead, Amazon Web Services
- Krishna Limbu, Technical Account Manager, Amazon Web Services
- Venu Rapolu, Technical Account Manager, Amazon Web Services
- Raghu Syama, Enterprise Support Lead, Amazon Web Services
- Harshika Thusu, Technical Account Manager, Amazon Web Services
- Barbara Bogdanescu, Sr. Technical Account Manager, Amazon Web Services
- Nadeem Bulsara, Principal Solutions Architect, Amazon Web Services
- Cameron Smith, Sr. Solutions Architect, Amazon Web Services
- Michael Smalley, Enterprise Support Manager, Amazon Web Services
- Preetkumar Shah, Technical Account Manager, Amazon Web Services
- Subha Kalia, Sr. Technical Account Manager, Amazon Web Services
- Dakshaja Vaidya, Technical Account Manager, Amazon Web Services
- Stewart Matzek, Sr. Technical Writer, Amazon Web Services
- Madhuri Srinivasan, Sr. Technical Writer, Amazon Web Services
- Matthew Wygant, Sr. TPM Guidance, Amazon Web Services

Document revisions

The following table describes the documentation releases for the Life Sciences Lens.

Change	Description	Date
Initial release	Initial lens release of the Life Sciences Lens.	December 30, 2025

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2025 Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.