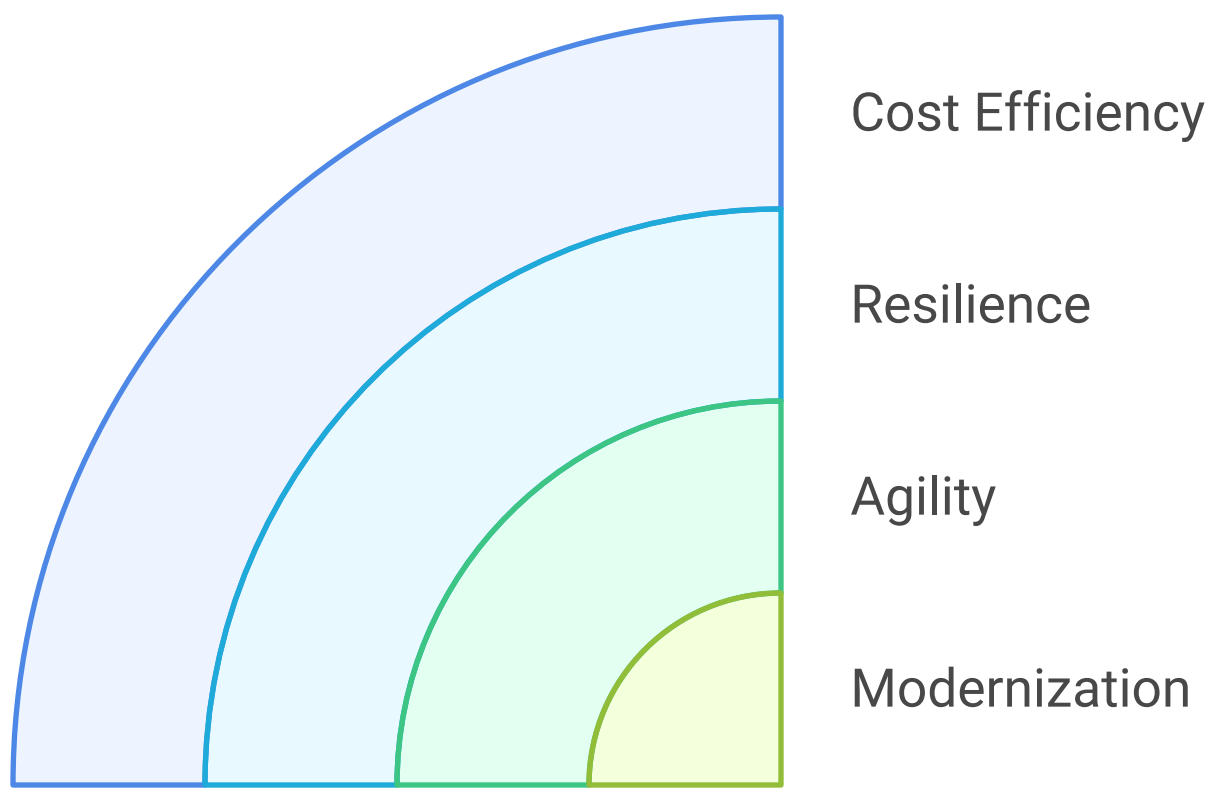# ⌗ Mainframe Application Modernization

Unlocking Agility, Resilience, and Cost Efficiency
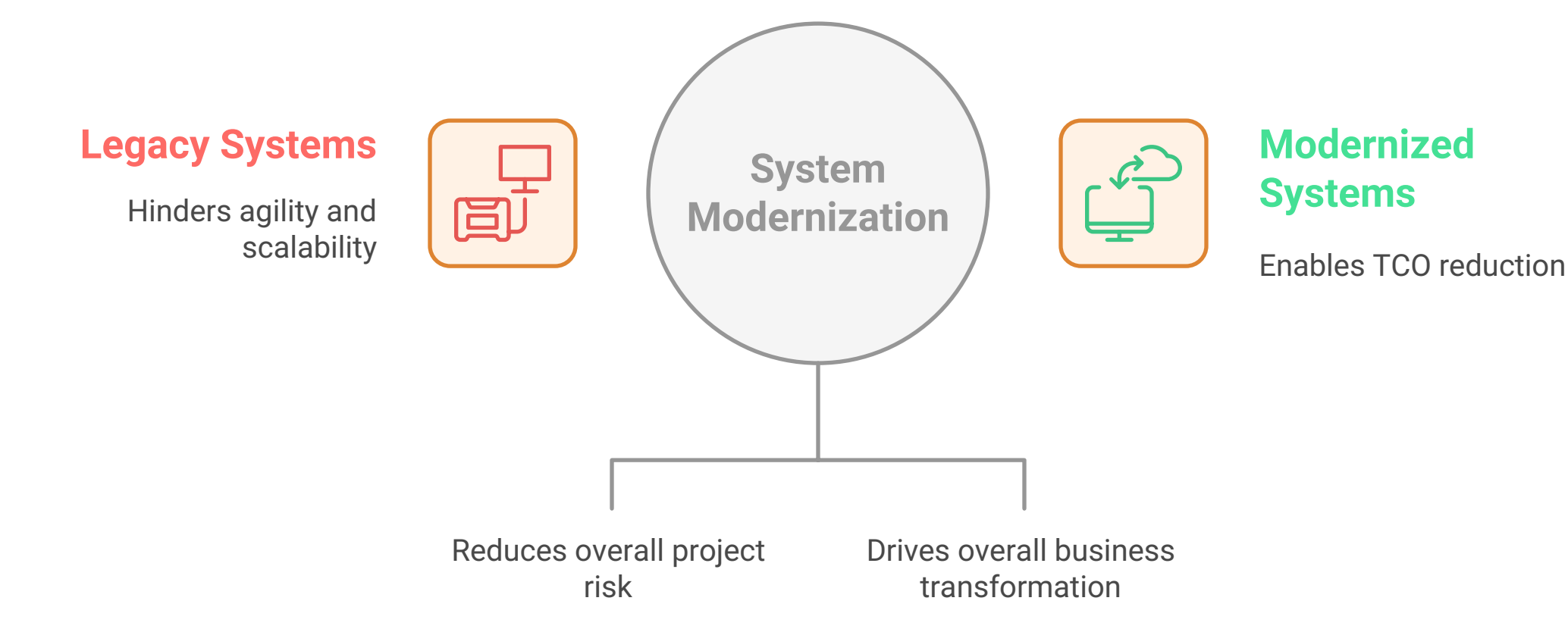
Mainframe Modernization Strategy

Cost Efficiency

Resilience

Agility

Modernization

Executive Summary

- Legacy systems hinder agility and scalability.
- Modernization enables >30% TCO reduction.
- Phased approach reduces risk.
- Cloud-native systems drive transformation.
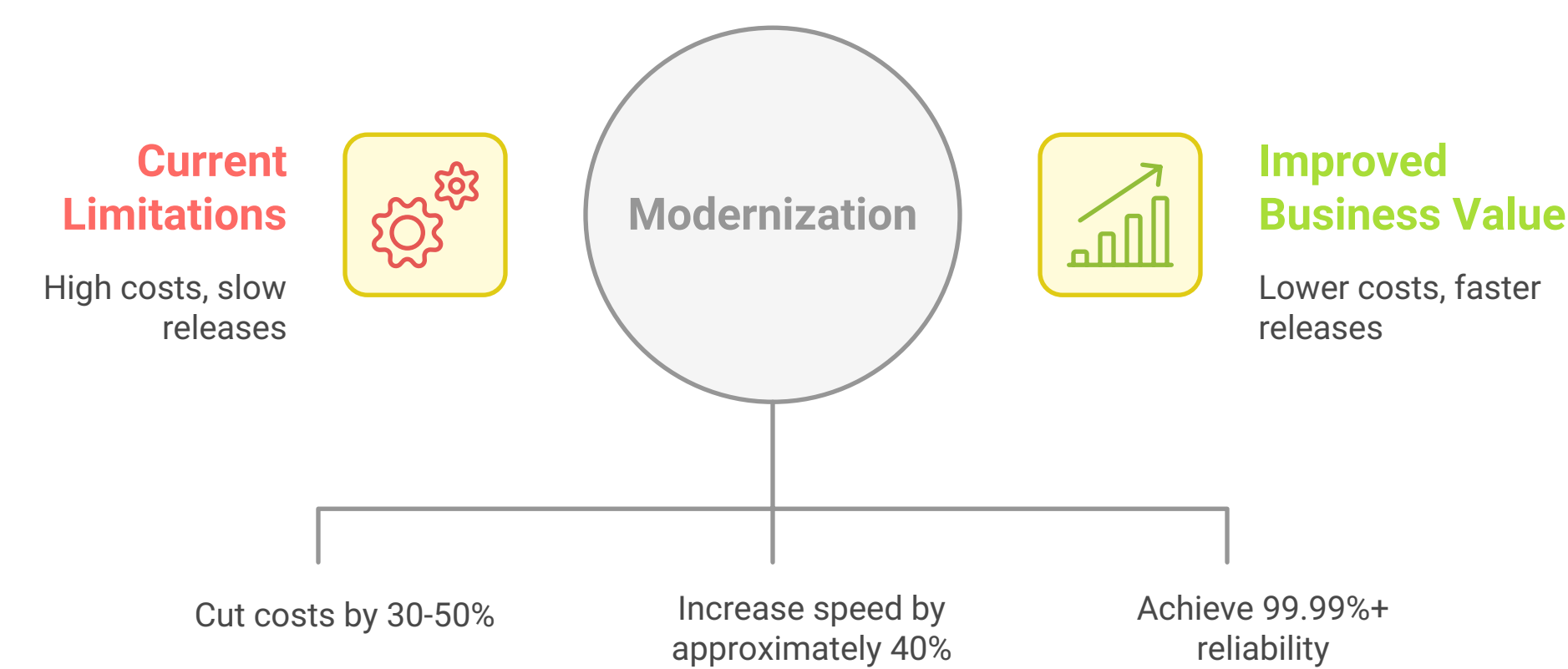
Modernization enables TCO reduction

Legacy Systems

Hinders agility and scalability

System Modernization

Modernized Systems

Enables TCO reduction

Reduces overall project risk

Drives overall business transformation

Why Modernize Now?

Drivers:

- Cost, talent scarcity, compliance, agility
- Value at Stake:
- TCO cut by 30–50%
- Time-to-market boost ~40%
- 99.99%+ reliability

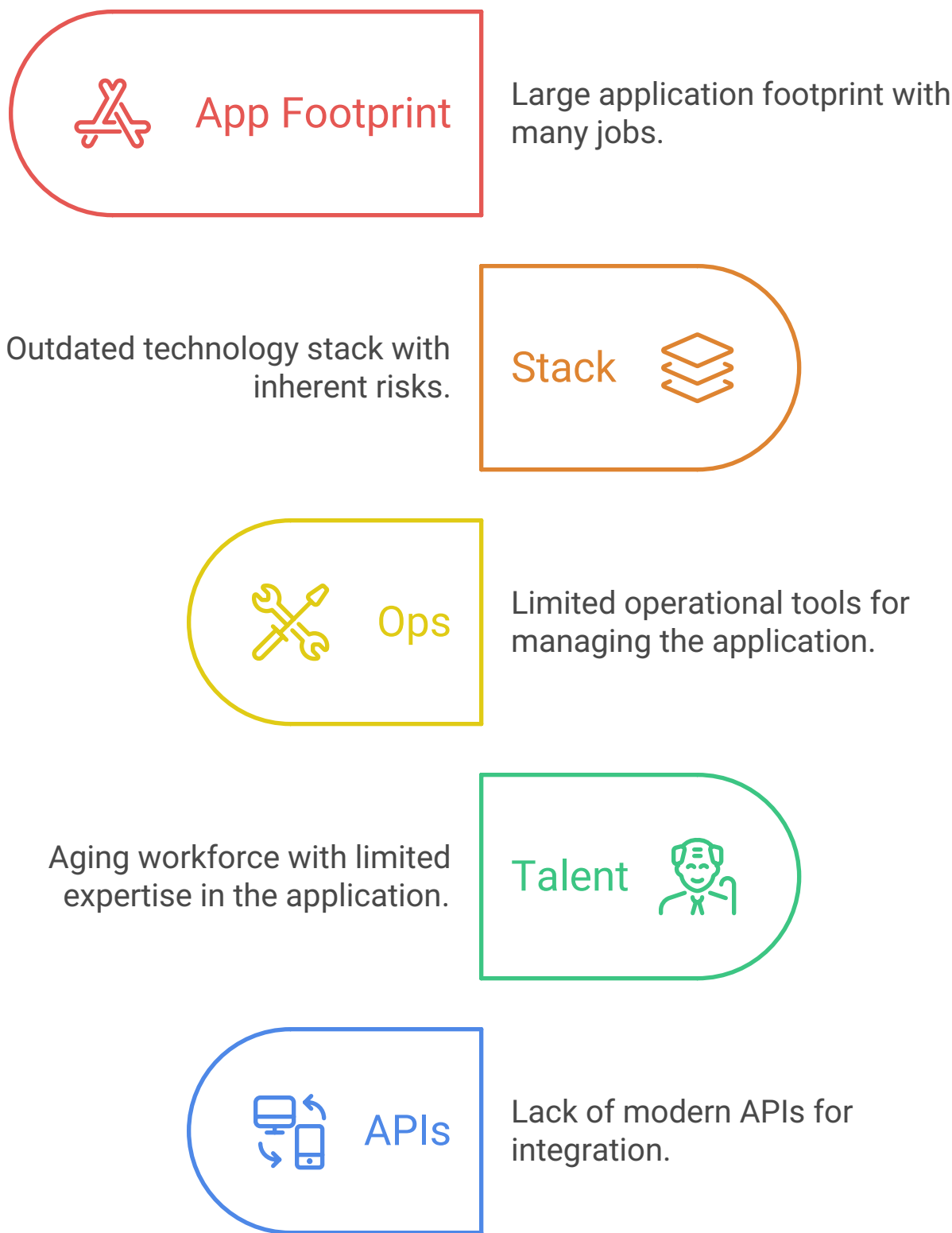# Modernization for Business Value

Current Limitations

High costs, slow releases

Modernization

Improved Business Value

Lower costs, faster releases

Cut costs by 30-50%

Increase speed by approximately 40%

Achieve 99.99%+ reliability

Current State Assessment

• App Footprint: ~1.2M LOC, 450+ jobs (High Risk)
• Stack: COBOL, JCL, VSAM (High Risk)
• Ops: Limited tools (Med Risk)
• Talent: Aging (High Risk)
• APIs: Lacking (High Risk)

Application Risks

App Footprint — Large application footprint with many jobs.

Stack — Outdated technology stack with inherent risks.

Ops — Limited operational tools for managing the application.

Talent — Aging workforce with limited expertise in the application.
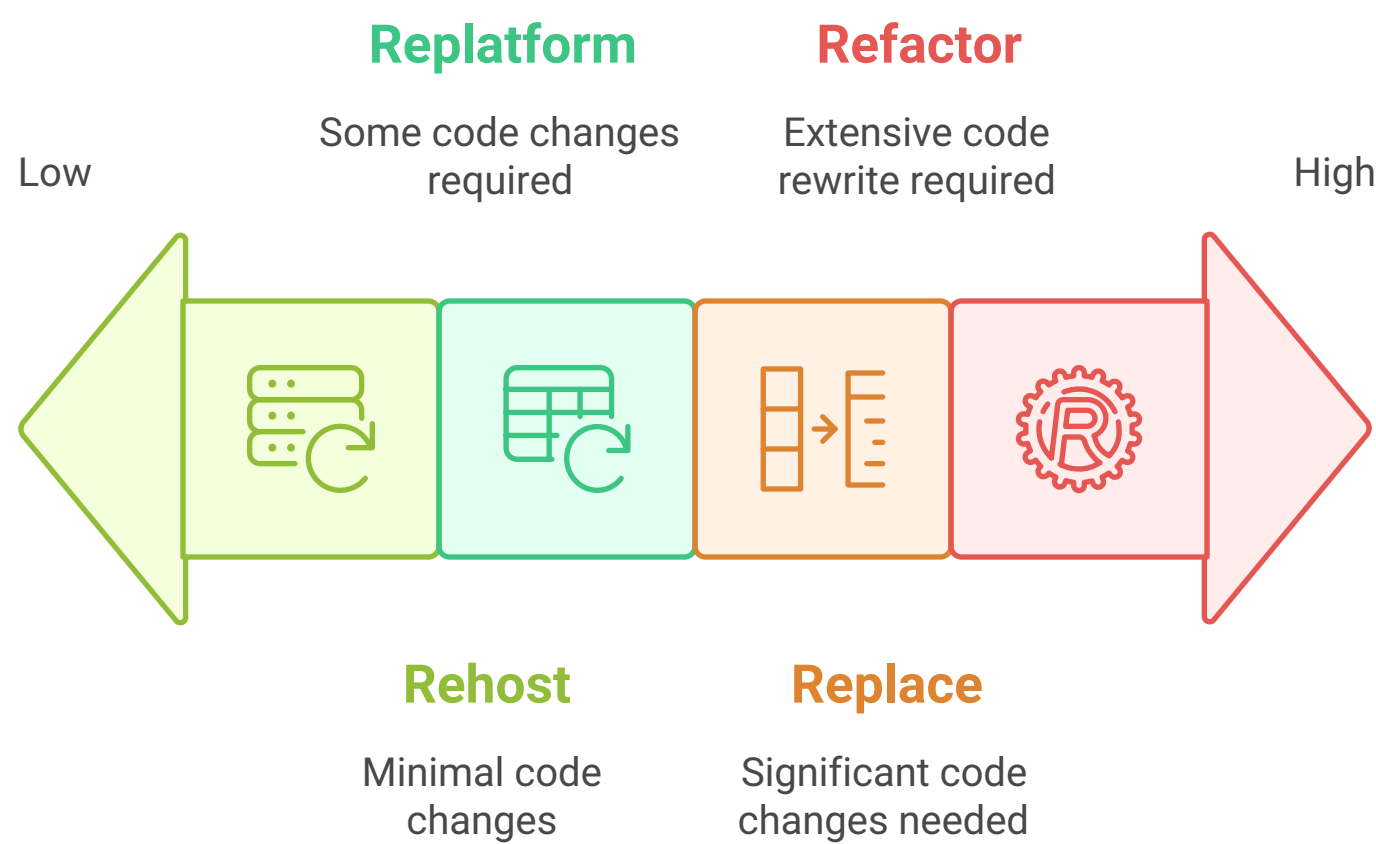
APIs — Lack of modern APIs for integration.

Modernization Options – Framework

Options:
• Rehost (low risk)
• Replatform (balanced)
• Refactor (high risk)
• Replace (medium risk)

Recommended: Hybrid approach

Modernization options ranked by risk level.

Replatform

Some code changes required

Refactor

Extensive code rewrite required

Low

High

Rehost

Minimal code changes

Replace

Significant code changes needed

Target State Vision (Architecture)

- Kubernetes, APIs (REST, GraphQL)
- Event-driven (Kafka, Pub/Sub)
- CI/CD DevSecOps pipelines
- Observability: Prometheus, Grafana, ELK

Architecture components

Kubernetes and APIs

Focuses on container orchestration and API technologies like REST and GraphQL.

Utilizes Kafka and Pub/Sub for asynchronous communication between services.

Event-Driven Architecture

CI/CD Pipelines

Implements CI/CD DevSecOps pipelines for automated software delivery.

Employs Prometheus, Grafana, and ELK for monitoring and logging.

Observability Tools

Modernization Options – Framework

Options:

- Rehost (low risk)
- Replatform (balanced)
- Refactor (high risk)
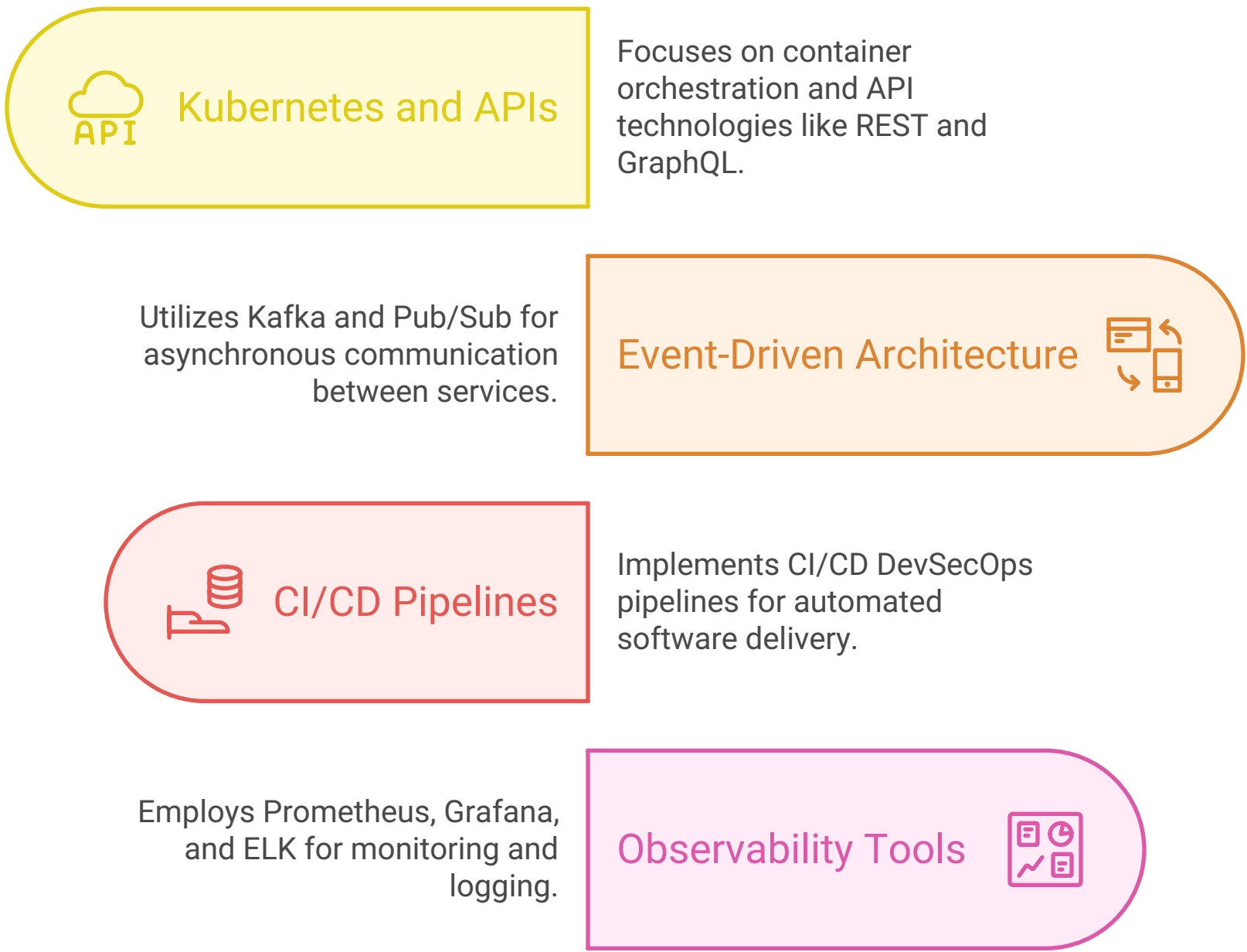- Replace (medium risk)
- Recommended: Hybrid approach

Modernization options ranked by risk level.

Low Risk

Replatform

Some code changes, moderate effort

Refactor

Extensive code rewrite, long-term investment

High Risk

Rehost

Minimal changes, quick deployment

Replace

Significant changes, new system implementation

Target State Vision (Architecture)

- Kubernetes, APIs (REST, GraphQL)
- Event-driven (Kafka, Pub/Sub)
- CI/CD DevSecOps pipelines
- Observability: Prometheus, Grafana, ELK
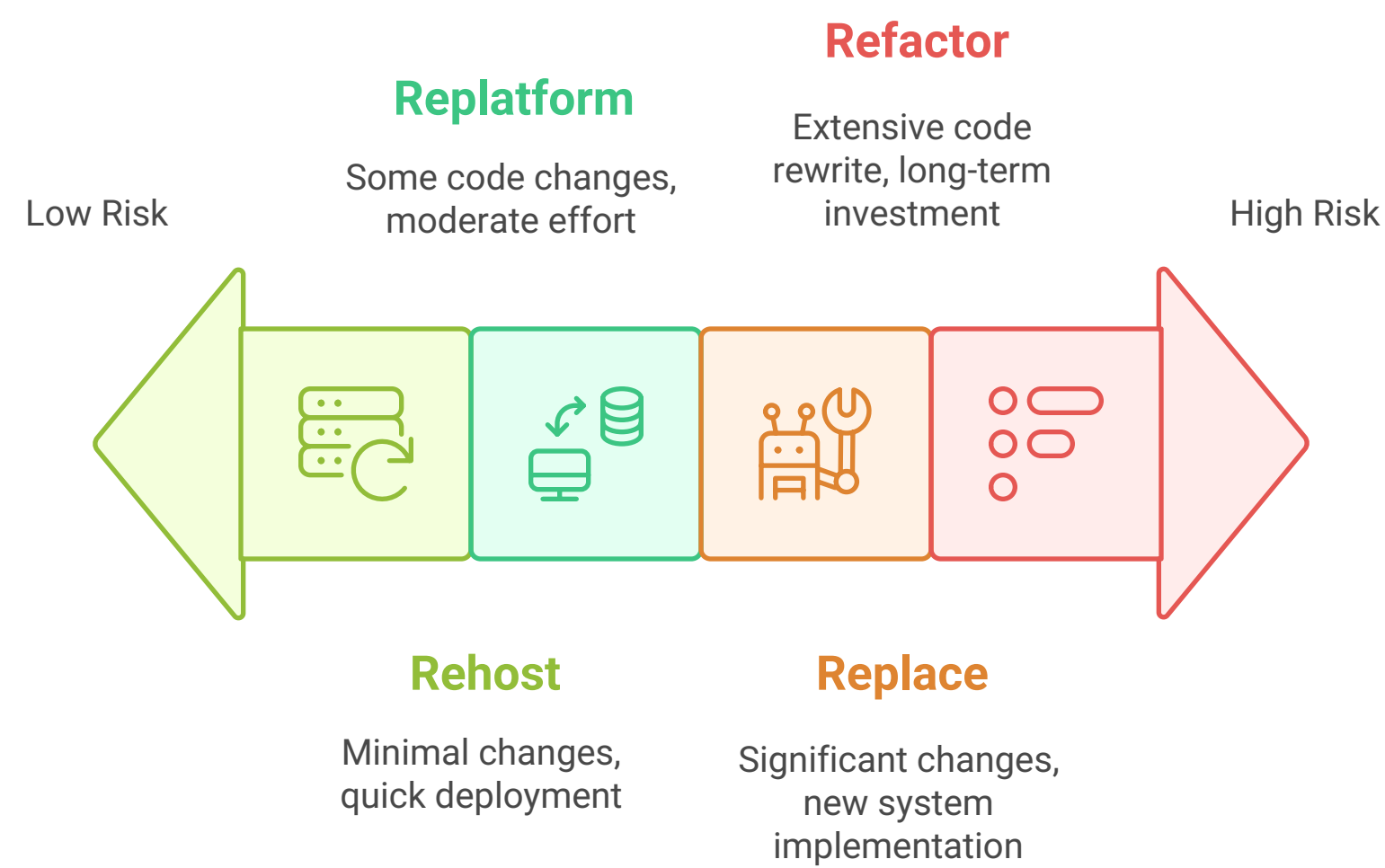
Architecture components

Kubernetes and APIs
Focuses on container orchestration and API technologies like REST and GraphQL.

Event-Driven Architecture
Utilizes Kafka and Pub/Sub for asynchronous communication between services.

CI/CD Pipelines
Implements CI/CD DevSecOps pipelines for automated software delivery.

Observability Tools
Employs Prometheus, Grafana, and ELK for monitoring and logging.

Transformation Roadmap

- Phase 1 (0–6 mo): Inventory, DevOps setup
- Phase 2 (6–18 mo): Replatforming, API, CI/CD
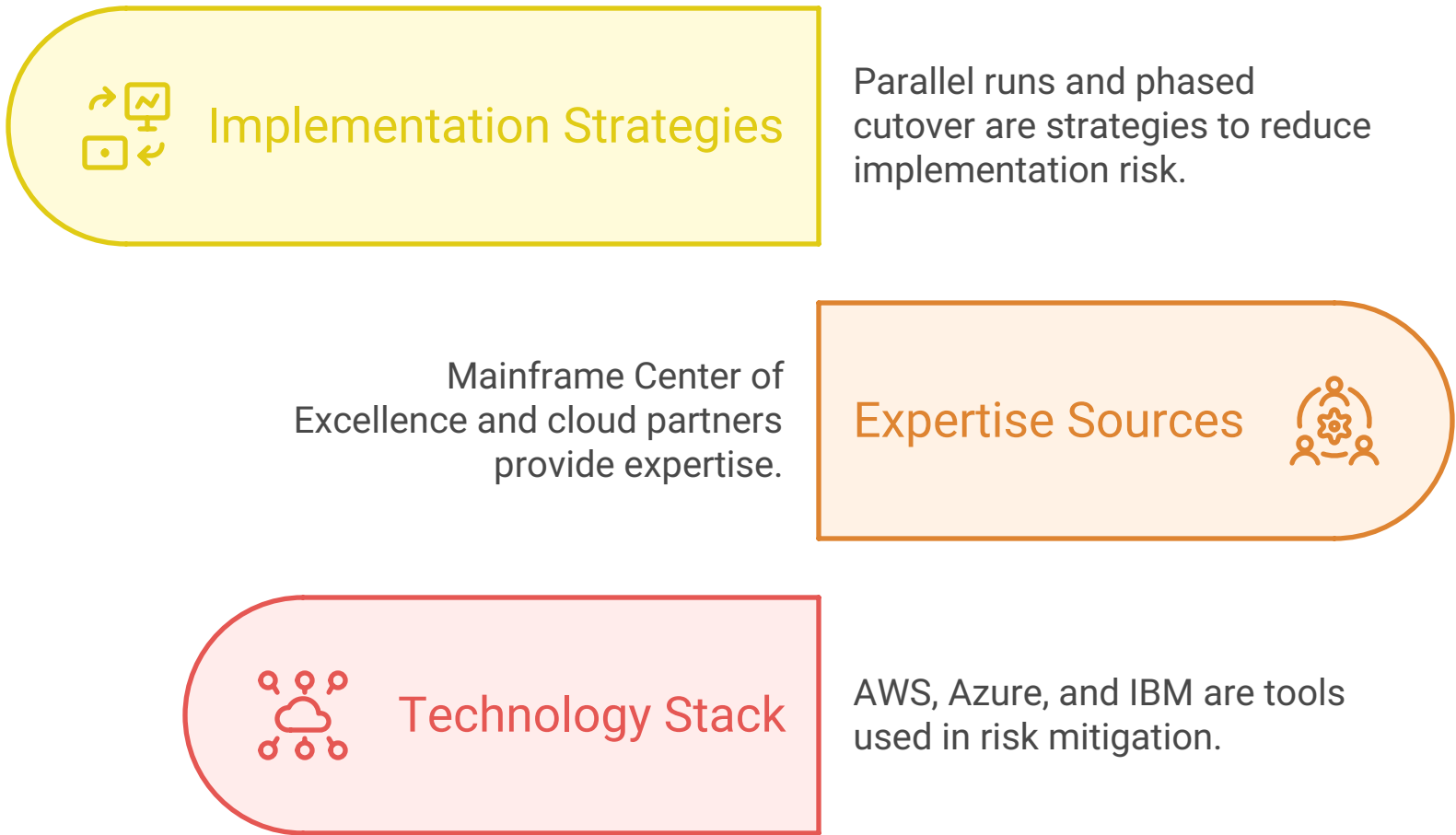- Phase 3 (18–24+ mo): Refactor & legacy exit

Strategic Transformation Roadmap

0–6 mo
Inventory and DevOps setup

18–24+ mo
Refactor and legacy exit

6–18 mo
Replatforming, API, CI/CD
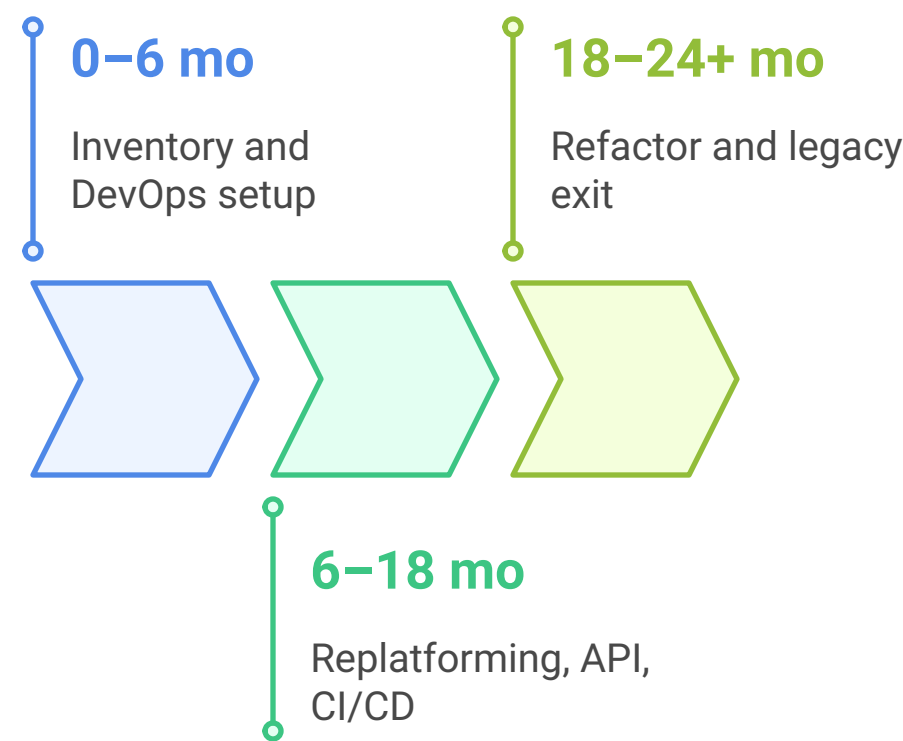
Risk Mitigation & Governance

- Parallel runs, phased cutover
- Mainframe CoE, cloud partners
- Tools: AWS, Azure, IBM

Risk Mitigation and Governance Elements

Implementation Strategies

Parallel runs and phased cutover are strategies to reduce implementation risk.

Mainframe Center of Excellence and cloud partners provide expertise.

Expertise Sources
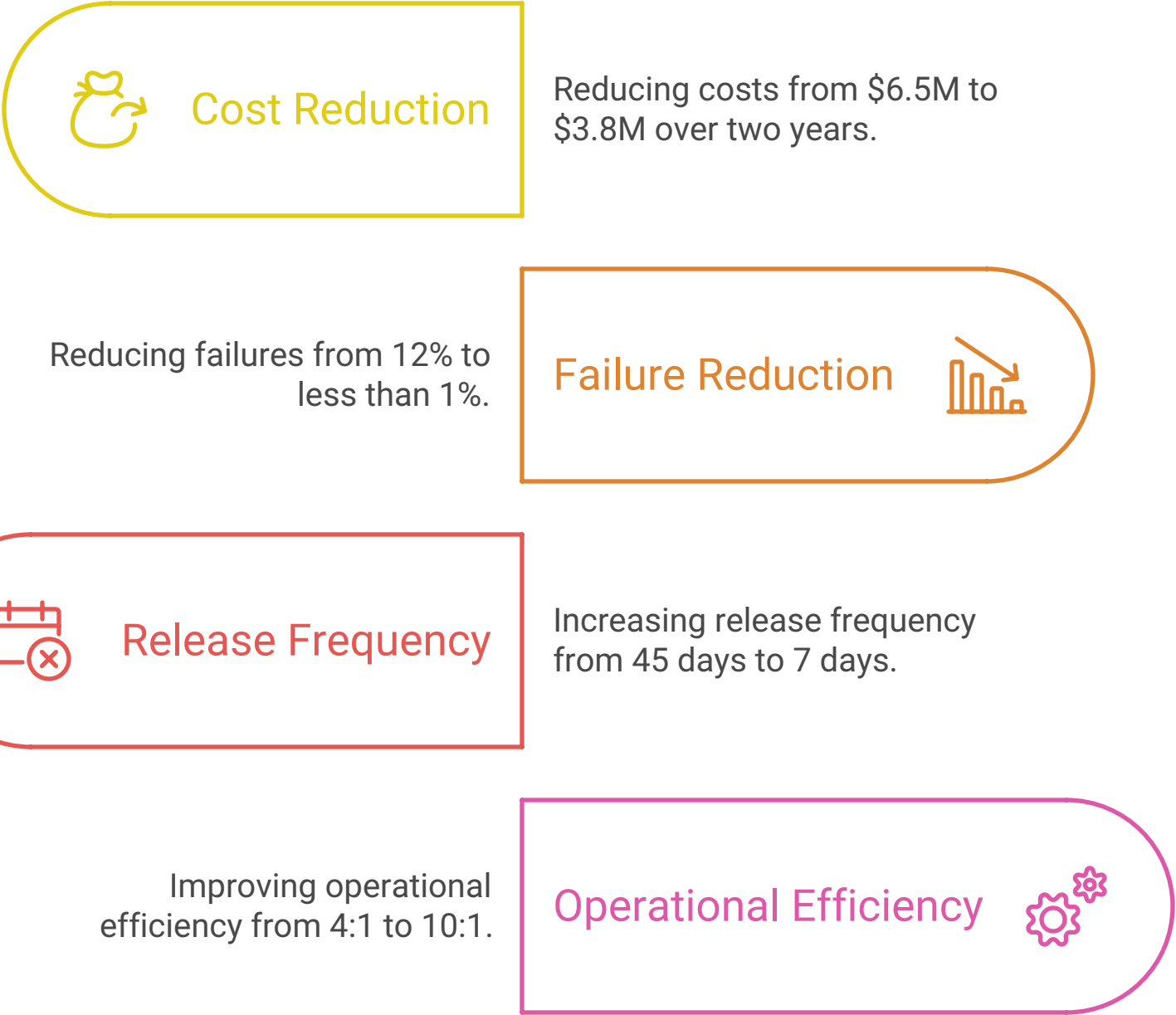
Technology Stack

AWS, Azure, and IBM are tools used in risk mitigation.

Success Metrics & KPIs

- Cost: $6.5M -> $3.8M (2Y)
- Failures: 12% -> <1%
- Releases: 45d -> 7d
- Ops: 4:1 -> 10:1

Performance improvements

Cost Reduction
Reducing costs from $6.5M to $3.8M over two years.

Failure Reduction
Reducing failures from 12% to less than 1%.

Release Frequency
Increasing release frequency from 45 days to 7 days.

Operational Efficiency
Improving operational efficiency from 4:1 to 10:1.

Next Steps & Ask

• Fund discovery phase
• Form squad
• Confirm partner
- Launch PoC

Project Development Sequence

Fund Discovery Phase
Secure initial funding for the project

Form Squad
Assemble a team to work on the project

Confirm Partner
Finalize partnerships for collaboration

Launch PoC
Initiate the proof of concept phase