Enhancing Healthcare Services through Effective APIs

Mathangi VijayShankar

Senad Ameti

Medibank Vision

Medibank vision is to create the best health and wellbeing for Australia

Connecting the enterprise to grow products that customers love through scalable & robust technology solutions

Create a culture of excellence, transparency and collaboration

API products are built with the highest standards of quality, security and compliance.

Delivering exceptional developer experience and continuously improving our platforms

Why are APIs crucial to this vision?

Enhanced Interoperability using standard models

Improved Patient Care and Engagement

Digital Channels streamlining Administration

Fostering Innovation, Collaboration in Healthcare ecosystem

Regulatory and Compliance Requirements

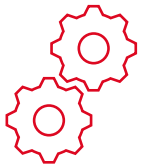
Adopting API product Mindset

- An outside-in, customer-centric approach to designing, developing, and managing API products.
- Understanding the needs of the API Consumers & building API products that are designed to meet those needs.

EASE OF CONSUMPTION

CUSTOMER ENGAGEMENT & HAPPINESS

LONG-TERM VALUE @SCALE



API MANAGEMENT & LIFECYCLE

Medibank Integration Landscape

AWS Lambda

Legacy Systems

- Outdated Technologies
- Lack of Vendor Support
- Struggles to meet current business needs
- Security constraints
- Lack of Test Coverage
- Lack of Automation
- Unable to scale easily

Impact of using Legacy Systems

- Lack of flexibility
- Complicated Data Exchange patterns
- Delayed timelines
- Increased costs

What is Devops Modernisation in API space?

Why do we need DevOps with API Management?

- Increased Productivity
- Greater Visibility
- Organizational Alignment
- Quick Feedback cycle
- Speed, Security, Scalability

API Lifecycle Management

- Essential to API Governance & Strategy
- Supports API Product delivery
- Steps to design, develop, test and consumer APIs.

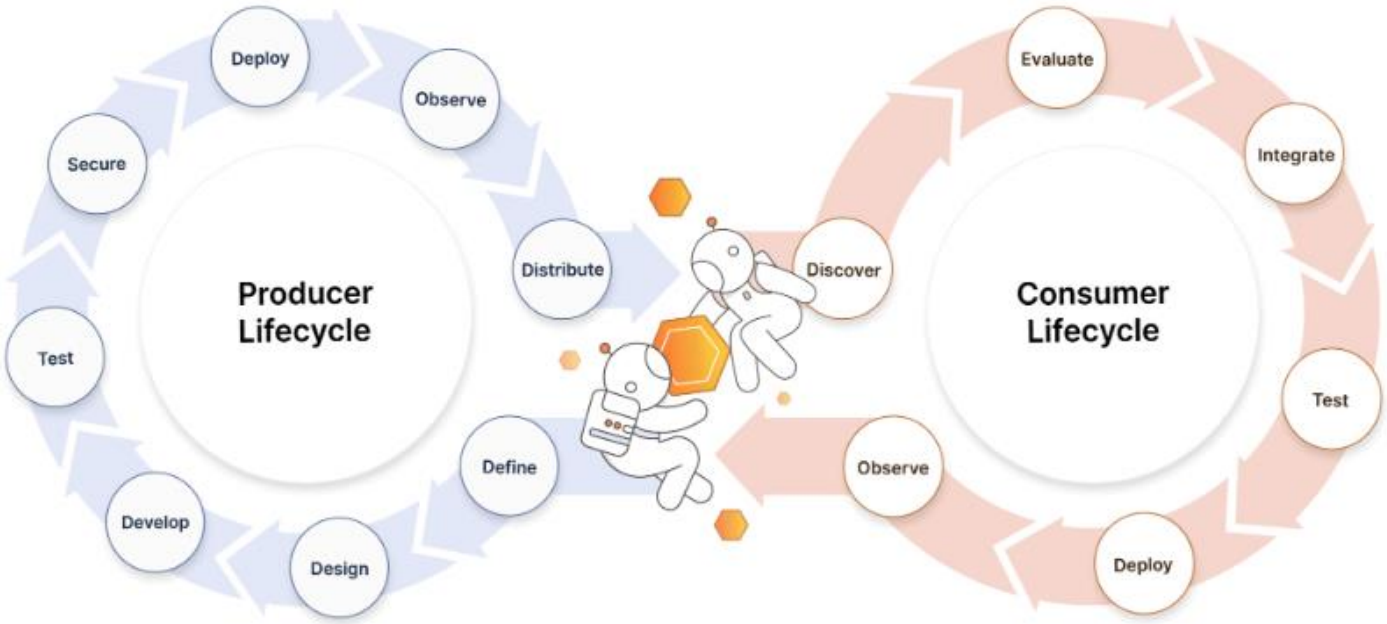


Figure 1: API Lifecycle management

Design And Dev Process

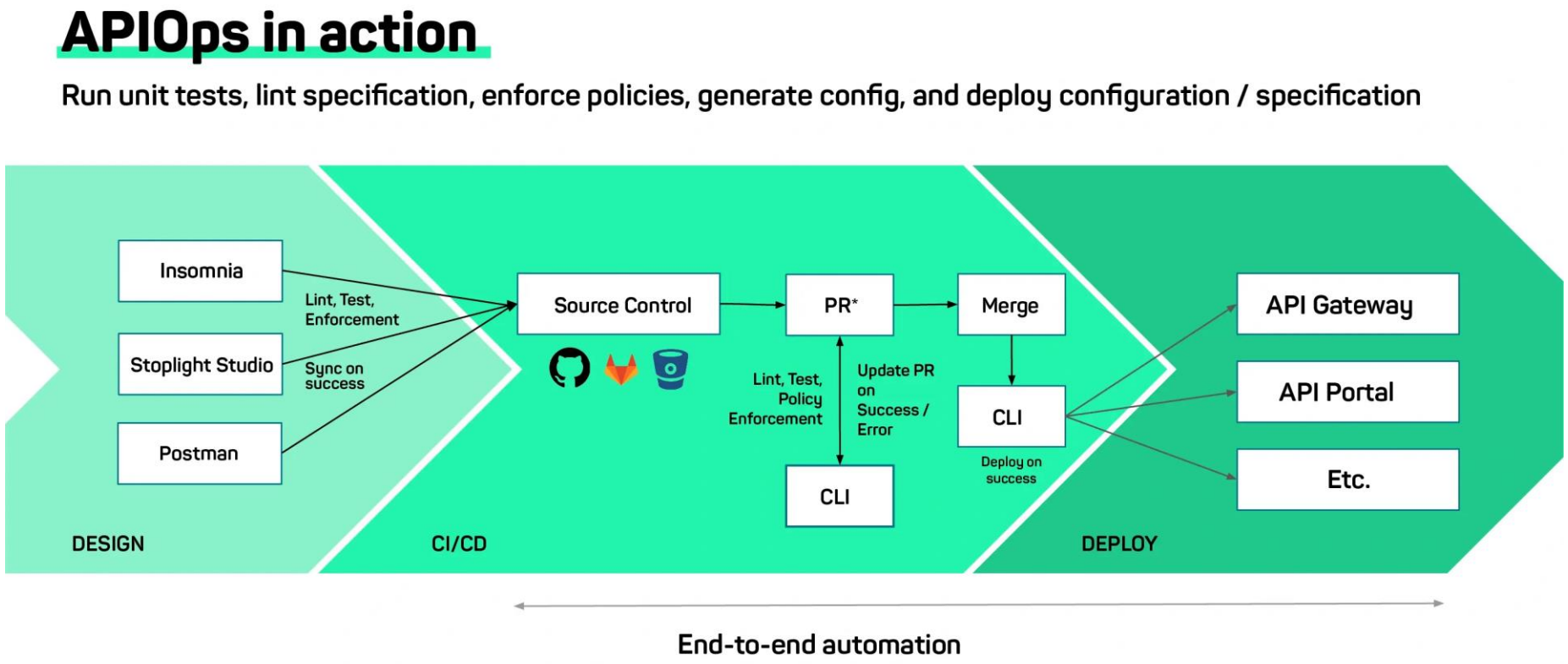
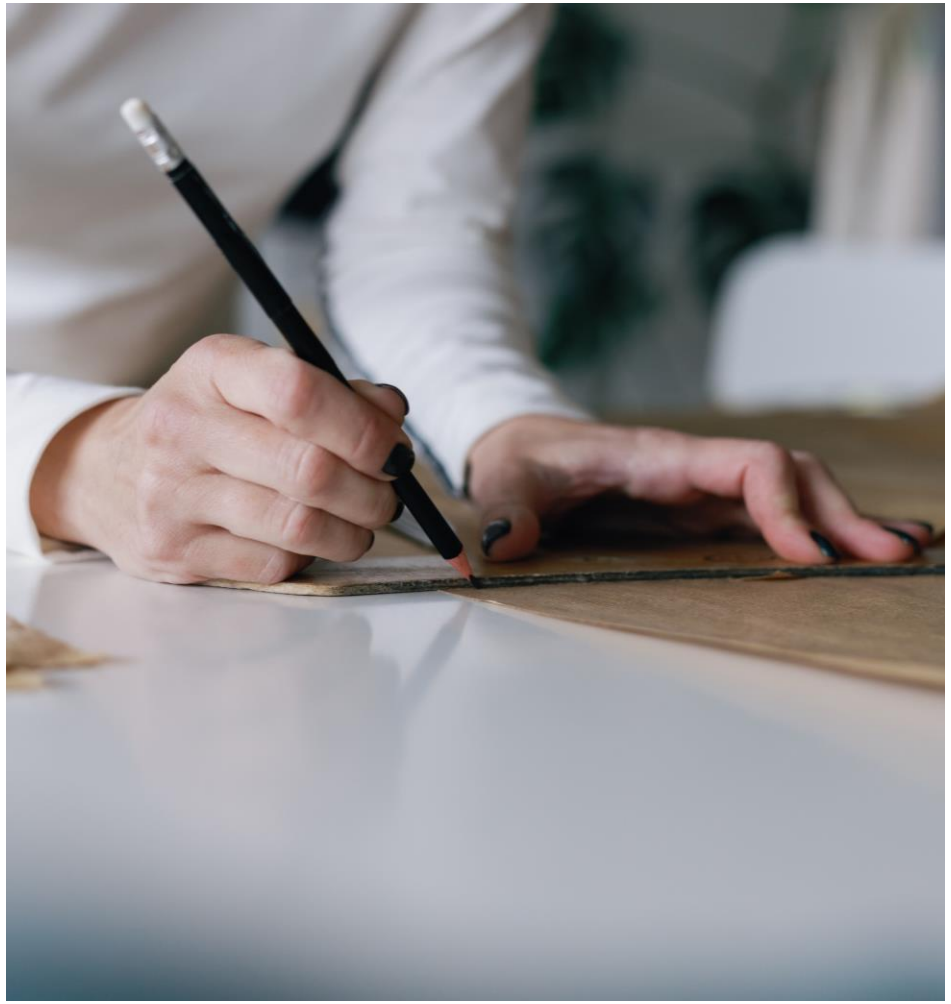


Figure 1: ApiOps in Action from codecentric.de

Define & Design

- Follow API Design First approach
- Business & stakeholder Involvement
- Prototyping and mocking for faster feedback
- Early Design enables parallel development



Development

Use modern technologies and tooling

- Use of Spring Boot & AWS as the backbone of the Integration platform
- Custom Spring Boot starter
- API Spring Boot template
- Java with gradle (PMD, checkstyle, spotsbugs)
- Use of python and AWS Lambdas
- SonarQube
- Secure Coding practices

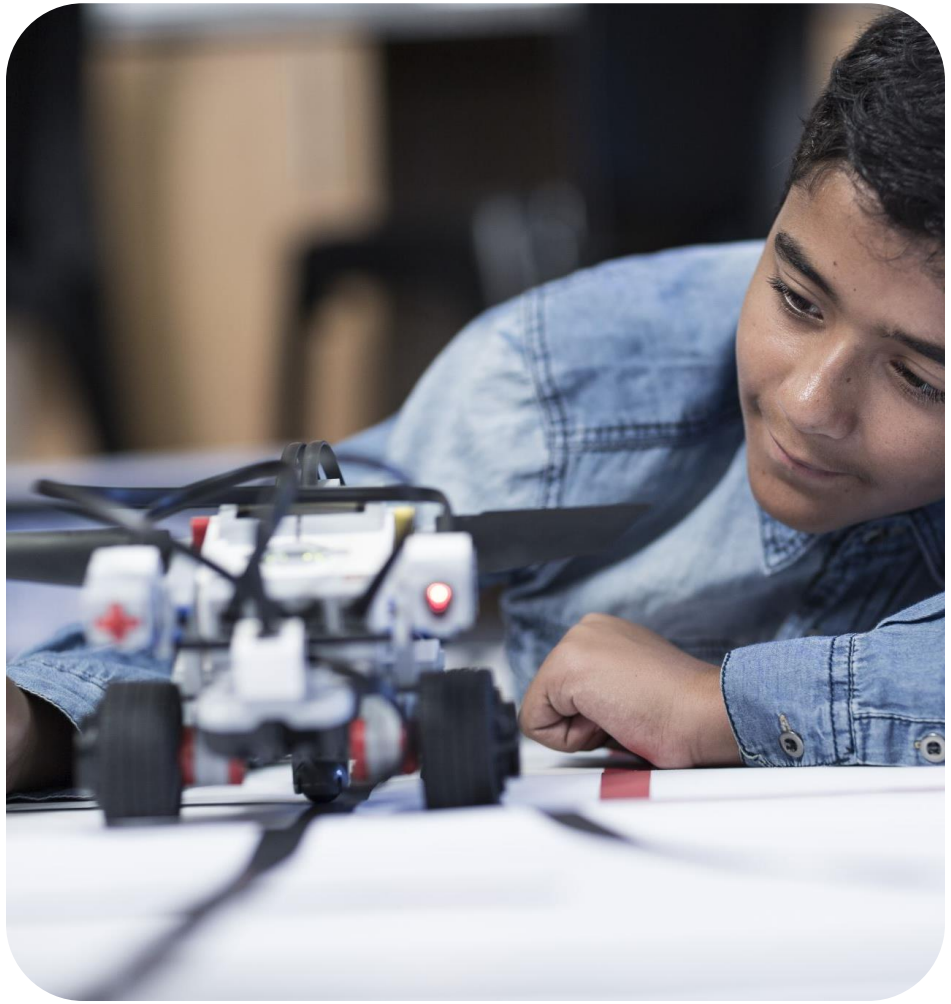
Testing

We do not treat Testing as simply a phase in the API Management Lifecycle.

- Test and verify API contracts

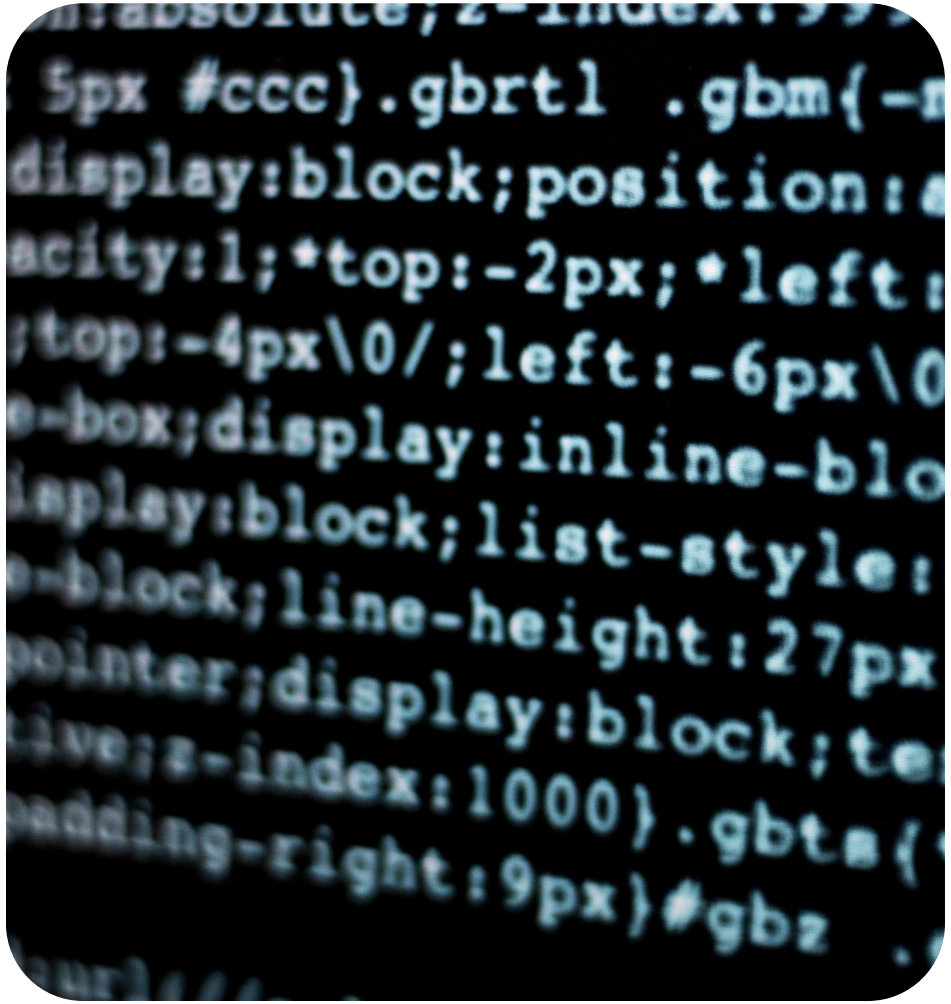
- Unit Testing (minimum 80% coverage)
- Integration Testing (with Wiremock)
- Automated Testing via Postman
- Exploratory Testing

- Performance & PEN testing etc



CI/CD

- Pre-PR checks
- Vulnerability scanning
- Code Quality checks
- Image Creation & Deployment to Dev
- Automated deployment to all environments
- Automated Production releases inclusive of Change management Automation.



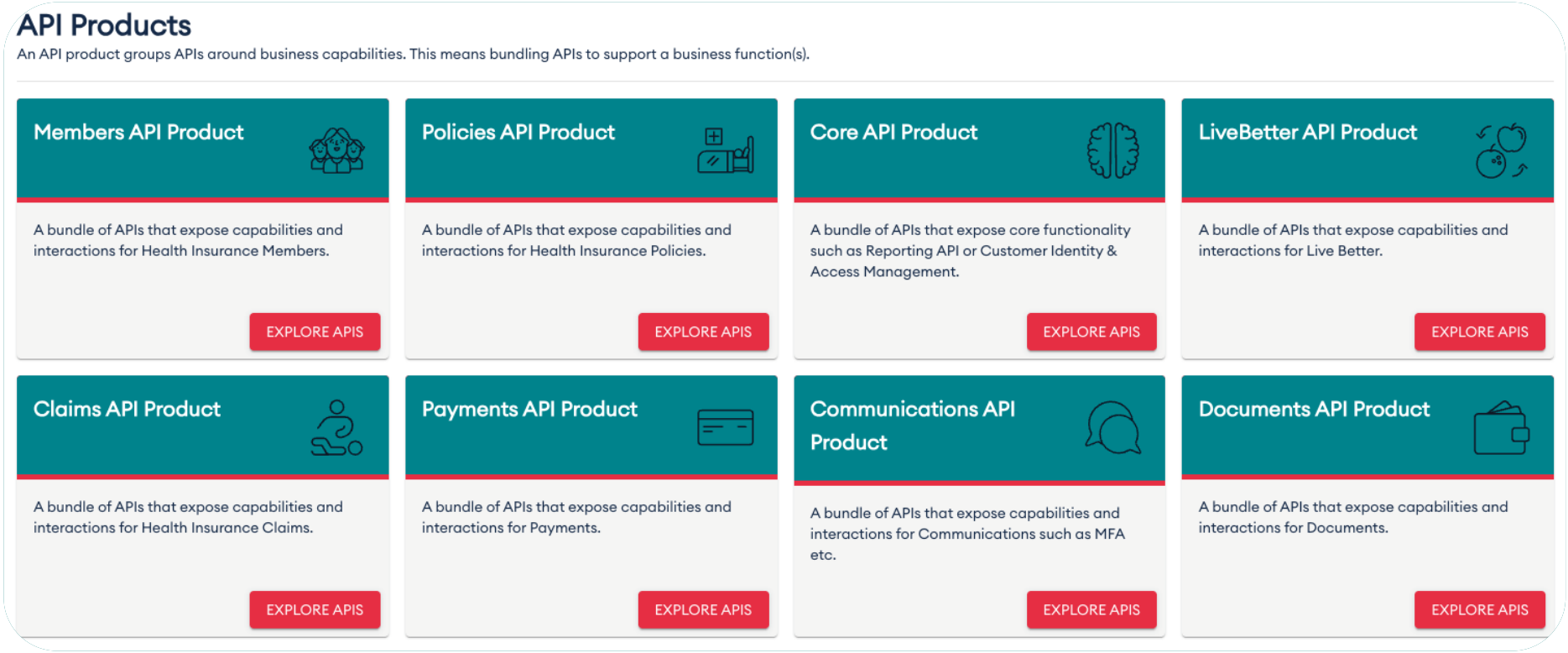
Monitoring & Observability

- Collecting, visualizing and alerting on API telemetry in production.
  - Events
  - Traces
  - Logs
  - Metrics
- Security Alerting (DDOS, Rate limiting etc)
- Performance post deployment
- Real-time analytics
- Dashboards

- Observe and decommission APIs based on their usage patterns.

Distribute and showcase

- API discoverability via Portal or Catalog
- Ease of collaboration for teams
- Surface important information about APIs



Cultural mindset shift

- Security-first
- Collaboration
- Ownership
- Responsibility
- Feedback loops

API Team Principles

Tech Principles

- Automation (at every level possible)
- Curiosity and Innovation
- Consistency
- API Visibility & Discoverability
- Observability
- Quality
- Reusability
- Reliability
- Future Tech Selection

Team Behavior Principles

- Accountability and Responsibility (end to end)
- Foster Curiosity and Innovation
- Active collaboration
- Fail Fast & Learn
- Share knowledge
- Feel free to speak up
- Celebrate Successes and Failures
- Raise areas for improvement

Ways of working

Build Fast. Learn Fast. Repeat.

Challenges in Modernizing Legacy Systems

Challenges Faced

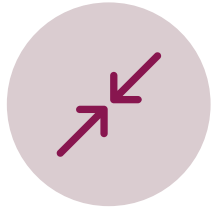
Budget Constraints

External Partner Integration and Support

Consumer Onboarding

Lack of consumable documentation leading to reverse engineering

Lack of knowledge



Resistance to change

Strategies of dealing with Legacy Services

Rewrite of Services

Decommissioning Strategies

Shift Left improvements

Wrapper Techniques

Containment strategies

Modernizing services

Identifying High value Services

Aligning services with Business Programs rather than running technology upliftment programs

Improvement of Security Footprint

Educating consumer and provider systems to adopt modern practices

Thank You.

Questions ?