Enabling delivery at scale at Australia’s fastest growing digital bank with an API driven Platform

Daniel McKeown
Area Lead, Cloud and Orchestration Platforms
ANZ Bank Australia

01 The Platform journey

02 The X Framework - Our Internal Developer Platform

03 Embracing GitOps for KRM

04 Progress and plans

05 The help I’m looking for..

06 Questions

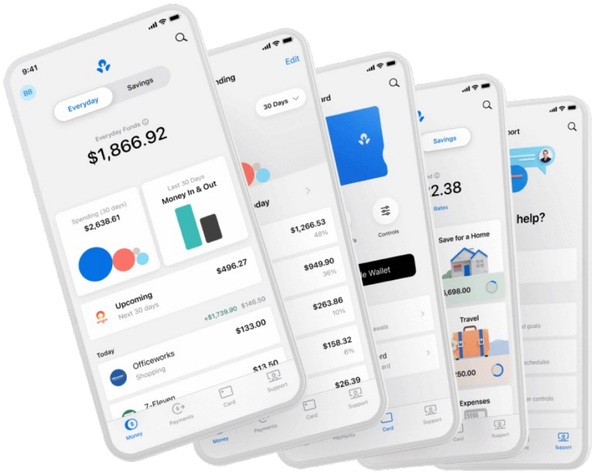
The Platform journey

A cloud-native digital bank

Launched in March 2022, ANZ Plus has already shipped its transact and save product and delivered a refinance home loan for pilot testing. ANZ Plus is Australia's fastest growing new digital banking service

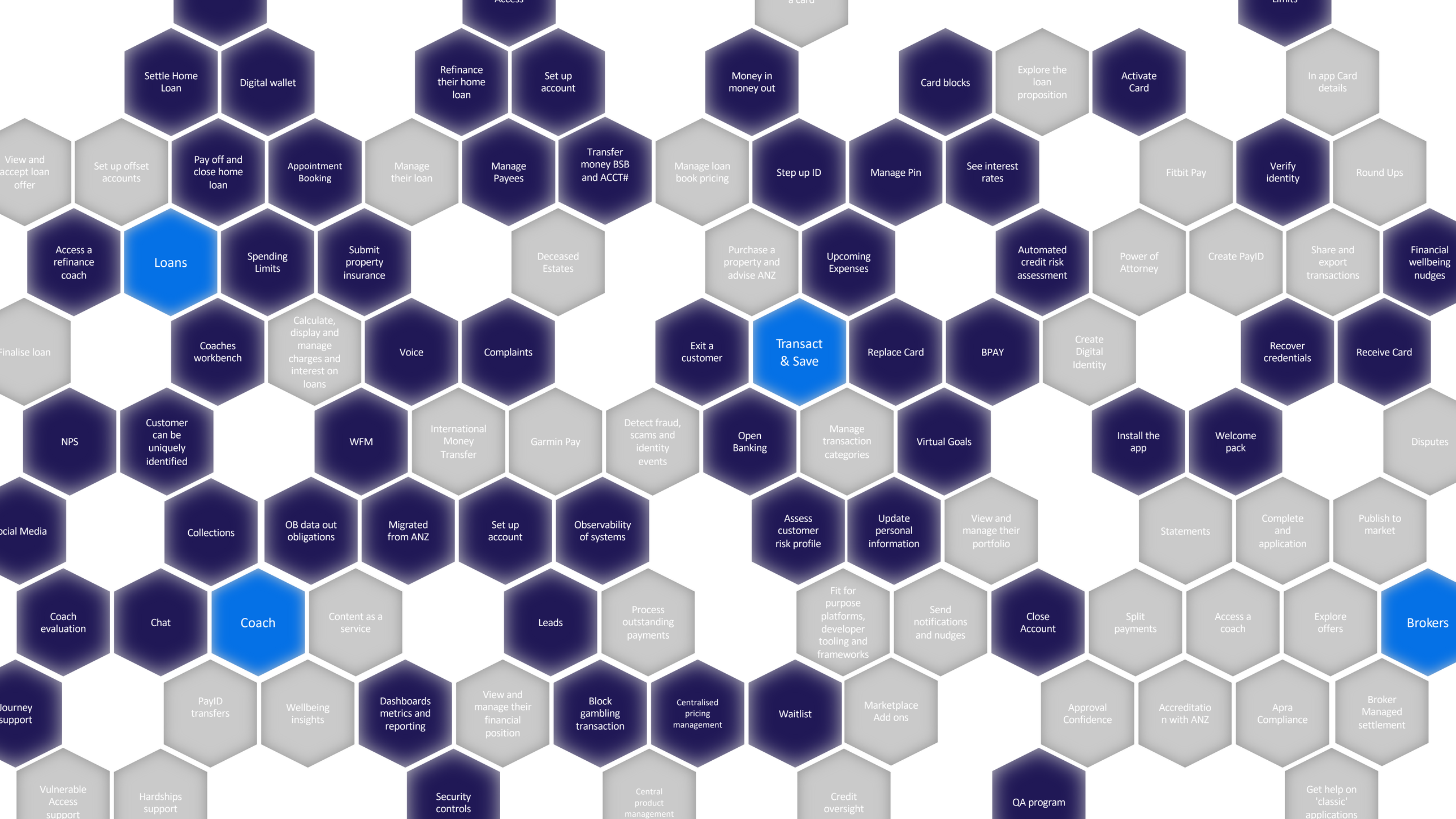
We're working on an ambitious roadmap of deliverables to uplift both user experience and to help our customers to improve their financial wellbeing

To support this complex multi-year program, we require a reliable, secure and efficient API driven Platform, to deliver at pace, in a highly regulated environment



Customers	Deposits
250k	$6B
March '23	March '23

Why is ANZ Plus investing in an API driven Platform?


Settle Home Loan
Digital wallet
Refinance their home loan
Set up account
Money in money out
Card blocks
Explore the loan proposition
Activate Card
In app Card details
View and accept loan offer
Set up offset accounts
Pay off and close home loan
Appointment Booking
Manage their loan
Manage Payees
Transfer money BSB and ACCT#
Manage loan book pricing
Step up ID
Manage Pin
See interest rates
Fitbit Pay
Verify identity
Round Ups
Access a refinance coach
Loans
Spending Limits
Submit property insurance
Deceased Estates
Purchase a property and advise ANZ
Upcoming Expenses
Automated credit risk assessment
Power of Attorney
Create PayID
Share and export transactions
Financial wellbeing nudges
Finalise loan
Coaches workbench
Calculate, display and manage charges and interest on loans
Voice
Complaints
Exit a customer
Transact & Save
Replace Card
BPAY
Create Digital Identity
Recover credentials
Receive Card
NPS
Customer can be uniquely identified
WFM
International Money Transfer
Garmin Pay
Detect fraud, scams and identity events
Open Banking
Manage transaction categories
Virtual Goals
Install the app
Welcome pack
Disputes
ocial Media
Collections
OB data out obligations
Migrated from ANZ
Set up account
Observability of systems
Assess customer risk profile
Update personal information
View and manage their portfolio
Statements
Complete and application
Publish to market
Coach evaluation
Chat
Coach
Content as a service
Leads
Process outstanding payments
Fit for purpose platforms, developer tooling and frameworks
Send notifications and nudges
Close Account
Split payments
Access a coach
Explore offers
Brokers
Journey support
PayID transfers
Wellbeing insights
Dashboards metrics and reporting
View and manage their financial position
Block gambling transaction
Centralised pricing management
Waitlist
Marketplace Add ons
Approval Confidence
Accreditation with ANZ
Apra Compliance
Broker Managed settlement
Vulnerable Access support
Hardships support
Security controls
Central product management
Credit oversight
QA program
Get help on 'classic' applications

and we are not stopping anytime soon

Platform journey

01 Origins

Our vision was to abstract away the complexities of the cloud, but…we first had to certify, secure, provision, and get regulatory approval for the underlying tools and infrastructure required for launch ANZ Plus

02 Launch

We went live 18 months ago, and the Platform has been rock-solid. However, Inconsistencies and repo sprawl meant that the developer experience we offered wasn’t great, and the Platform was hard to run

03 Partner

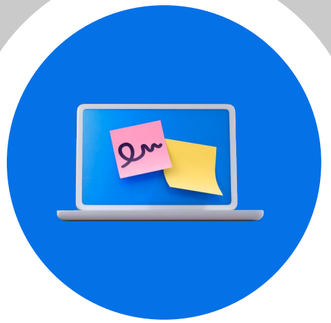
ANZx partnered with Google to help design our next iteration

04 Build

We’re now well progressed with the build, with many capabilities already in production.

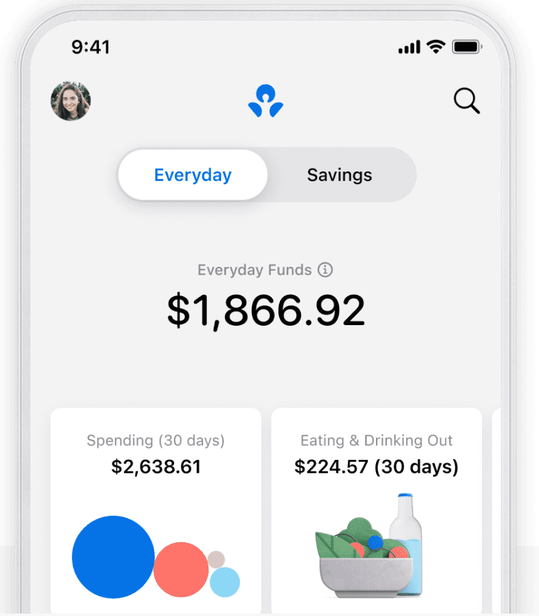
05 Adopt & Extend

Most workloads have now been onboarded onto the new Platform,.and we are now extending it as we learn



The X Framework - Our Internal Developer Platform

We’ve built an API driven Platform known internally as the X Framework



An API driven Platform

Key Platform user journeys have been automated using CRD interfaces and a GO templating codebase that executes in both pipelines and operators. Fleet wide changes can now be made from a single location in code.

All-in on GitOps for KRM

All KRM, including Kubernetes Config Connector (KCC) for GCP resources, is deployed using GitOps and OCI which improves security, DR posture, and is a big enabler for fleet, automated cluster provisioning and flexible cluster-based tenancy models.

Self-service tenancies

Our Workspace construct allow teams to manage their projects, namespaces, environments and IAM via CRDs. Application repos are created under Workspaces from templates with pipelines and other DevOps tooling configured out of the box.

Infra is now managed alongside applications

With everything in the one place, application engineers no longer need to interact with separate terraform repositories, resulting in better PRs, atomic changes, release management and increased autonomy.

Benefits of The X Framework

Asset transparency

Domain modelling our cloud and DevOps ecosystem not only allows us to automatically provision key user journeys, but we also get the benefits of an CMDB by publishing our CRDs to a queryable datastore

Reduced developer cognitive load

Provide golden paths based on interfaces of high-level intent, with a strong focus on user experience and documentation

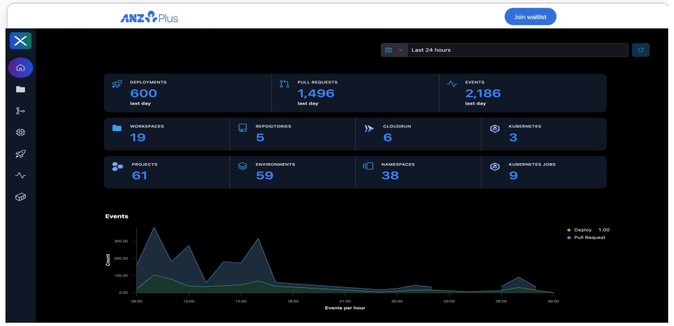
Reduced operator toil

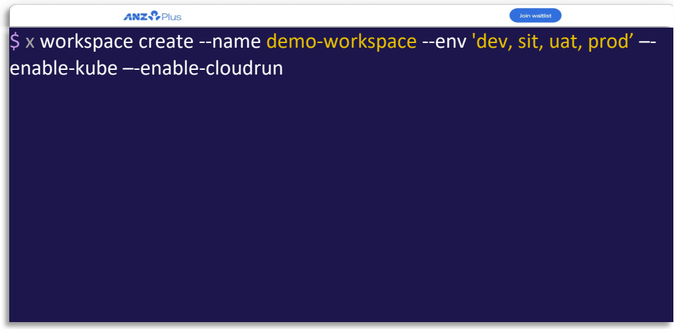
Platform config is deployed centrally employing a plugin model. Fleet wide changes across independent tenant repositories can be made from a single location in code.

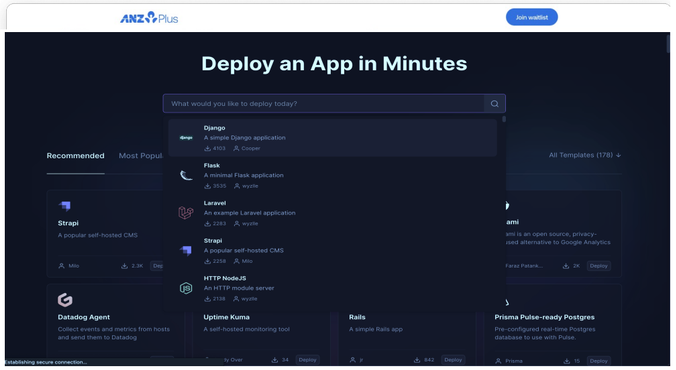
Community Governance

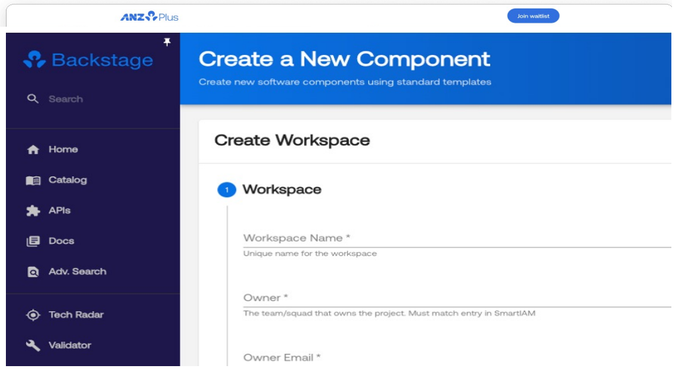
An open-sourced inspired governance model, with strong representation from application teams, provides consistency as well as program-wide buy in

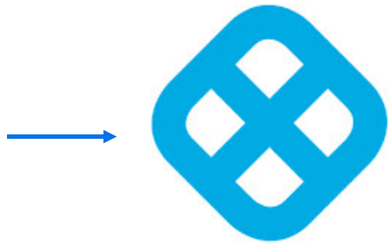
The X Framework Interfaces











Google Cloud

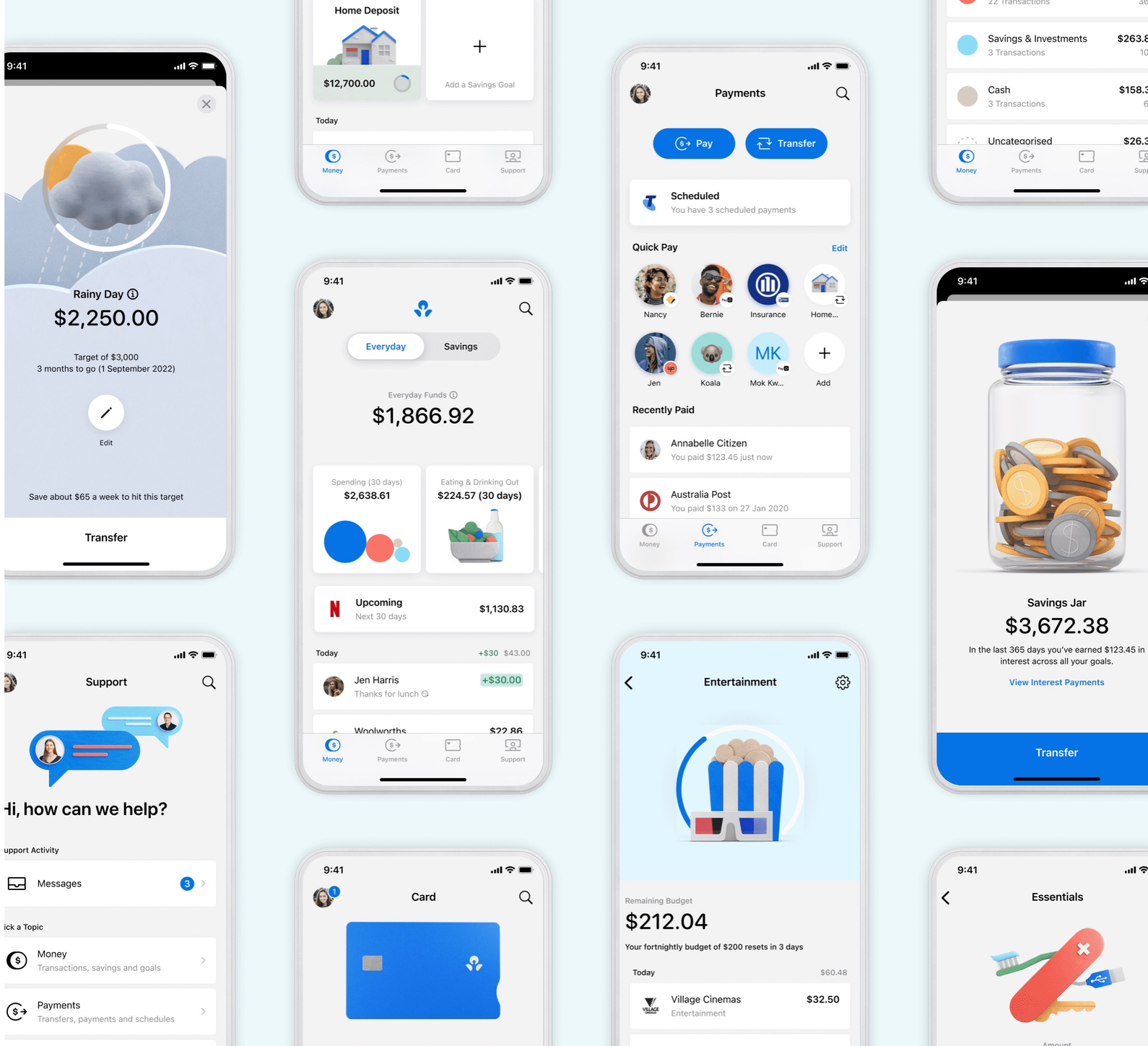
GitHub Actions

Embracing GitOps for KRM

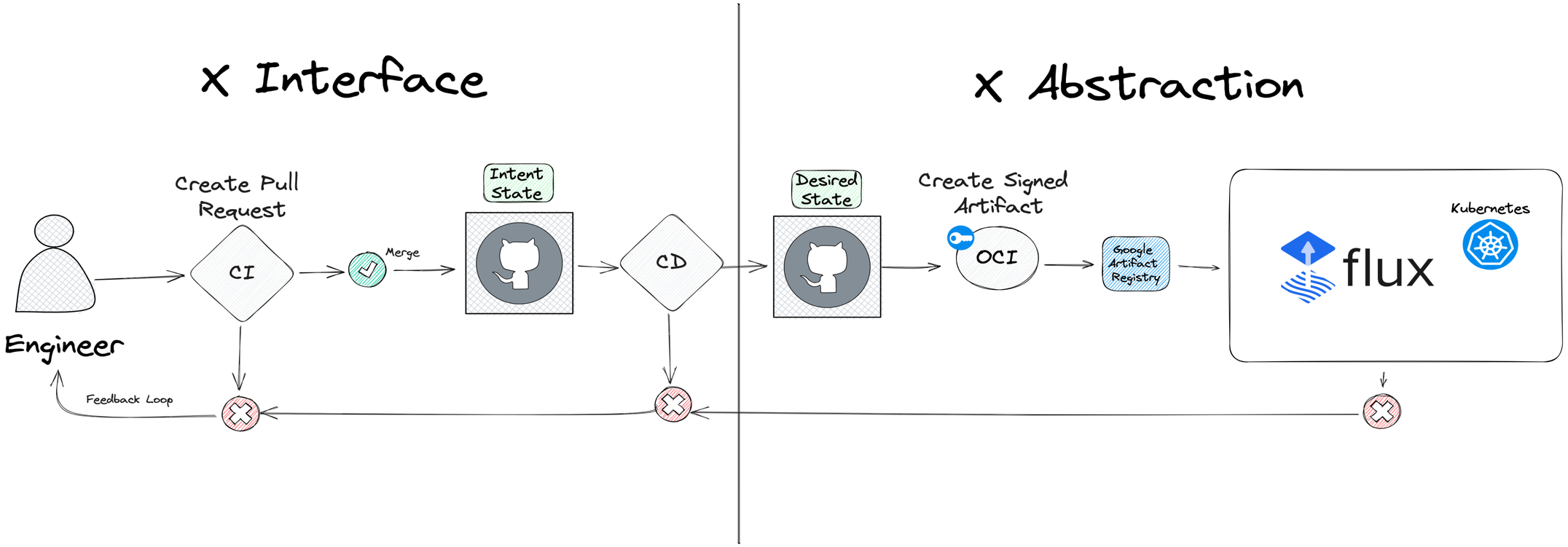
Challenges with independent push based deployments for KRM

Prior to going all in on GitOps, we faced multiple challenges caused by app teams using kubectl apply to deploy resources directly from independent repositories

- Large, static, long-lived clusters
- Reliant on velero backups for disaster recovery
- Configuration drift
- Unmanaged dangling resources
- Inconsistent environments
- Different CD pipelines and processes for PROD and NOTPROD



Benefits of GitOps for KRM



Disaster recovery

GitOps simplifies disaster recovery through version controlled declarative infrastructure, enabling easy rollbacks to known working configs and provides a clear, auditable history for rapid recovery

Fleet wide changes

Utilizing our CRD abstractions and templating, in conjunction with GitOps, we can seamlessly implement fleet-wide changes with ease and efficiency

Cluster provisioning

Leveraging OCI and keeping our configuration DRY we can effortlessly create and provision clusters on demand whilst deploying applications

Progress and plans

Service creation

- Project creation
- Kubernetes Config Connector
- GitHub repository template
- Namespace creation
- Service Archetypes

Cloud features

- Google Secret Manager
- K8's ingress & egress
- Load Balancers and Subnets
- Cloud Run
- DataFlow
- Cloud Functions
- Cloud DNS
- Pub/Sub
- BigQuery
- Cloud KMS
- Spanner

Governance

- Azure AD Groups
- Service accounts
- Google Cloud IAM
- servicenow Application Portfolio Management
- Xplore Risk & Controls

Dev tooling

- Github Actions
- Backstage Component & Techdocs
- Google Artifact Registry
- sonarqube Code Quality
- BLACKDUCK Software Composition Analysis
- CHECKMARX Static Application Security Testing

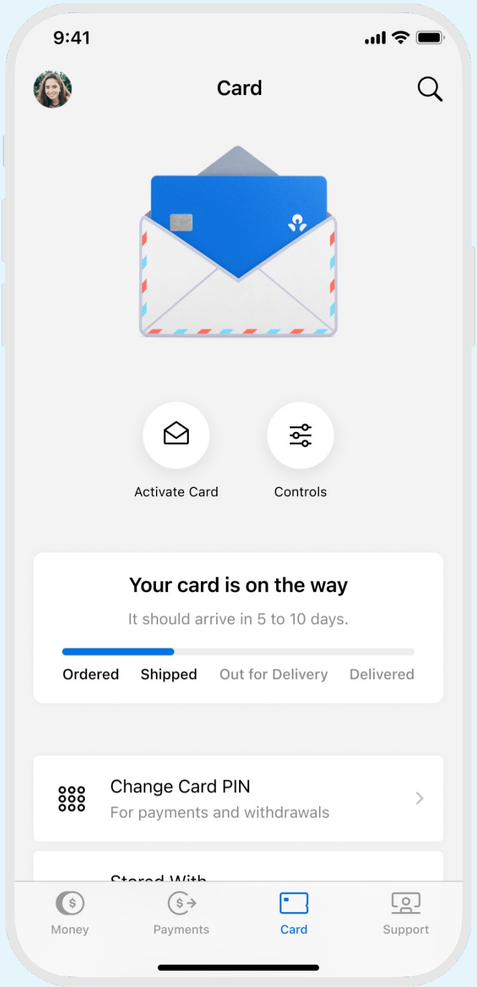
Integration & Ops

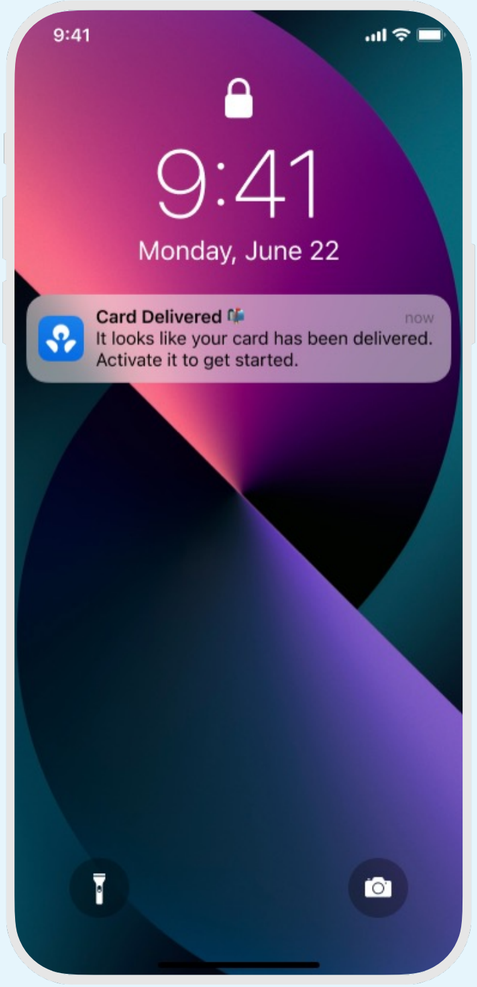
- Temporal
- LaunchDarkly Feature Flagging
- FORGEROCK Identity integration
- NOBL9 SLO Management
- Cloud Monitoring

The help I’m looking for..

Is to hear from the community about your experience..

- Have you gone through a similar journey progressing from simply provisioning cloud infrastructure and DevOps tools, to finally building a Platform?
- When designing your Platform, did you bake off Crossplane and KCC against building your own custom orchestration framework? What option did you go with and why?
- What is your view on rendering intent to primitives server-side vs in pipelines? Do all primitives need to be stored in git, or is intent enough?
- What is your entry criteria for building an operator? Are they worthwhile for simple use cases?
- How much of your cloud provider and DevOps tooling estate have you been able to configure through Platform APIs?
- What approach does your team use to automatically provision clusters with KRM config originating from multiple tenant repositories?





Questions

Thank you