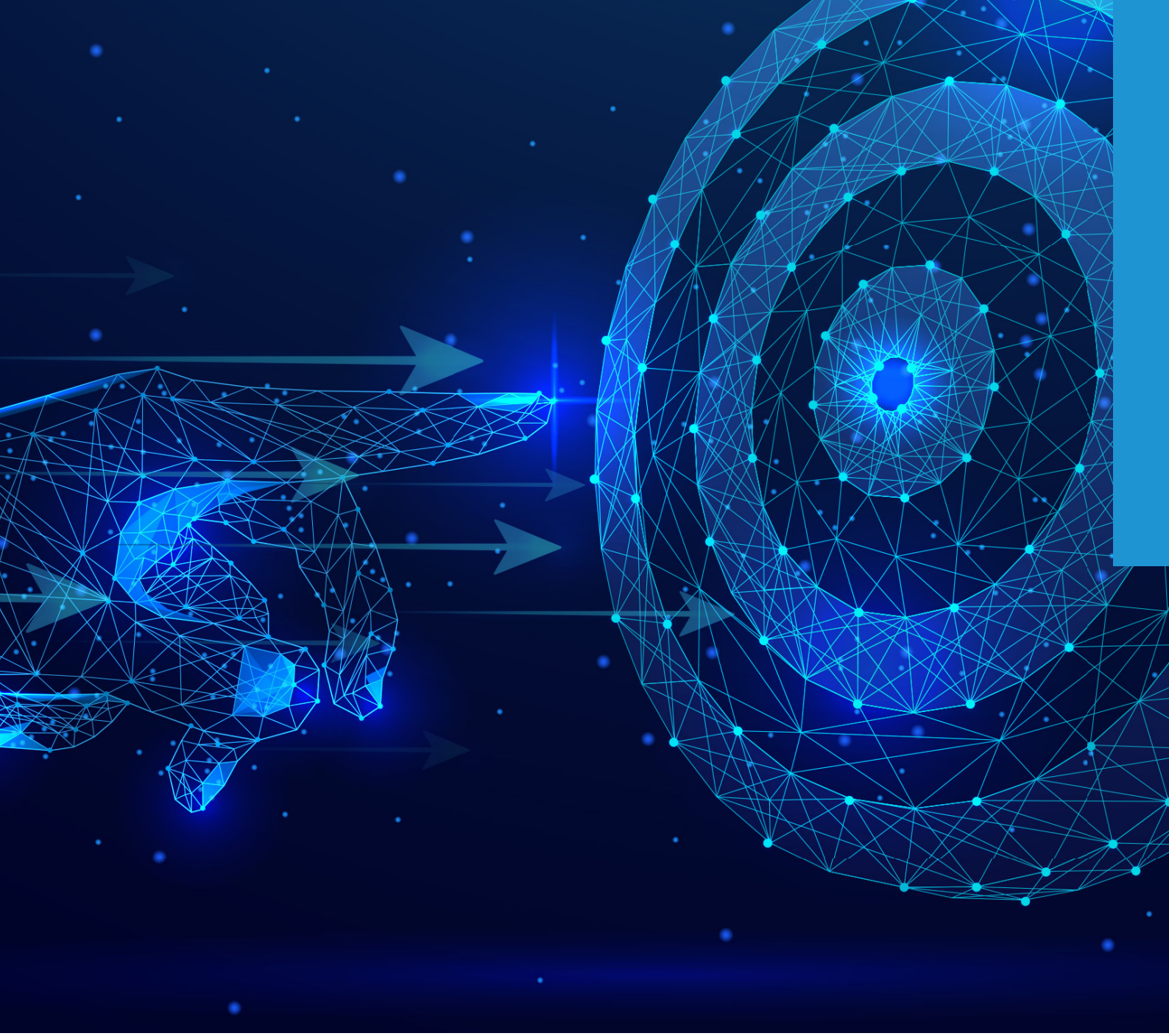


Target Cloud Operating Model

Olly Shaw et al - BETA Nov 2023

EQUAL EXPERTS WHITE PAPER

EQUAL EXPERTS

Contents

Introduction

Equal Experts have been part of cloud migrations at a number of large organisations. As such we have uncovered, learned and consolidated expertise in setting up, innovating and maintaining software services in the cloud.

Our clients wish to establish different ways of working and technology capabilities for emerging digital services, and heritage (previously on-premises estate) due to cost efficiencies arising from data centre migrations.

This report consists of contextualised learnings from our ongoing engagements and synthesised knowledge from a breadth of our other engagement contexts. This includes a British government department with 80 teams and 900 microservices, and a Middle Eastern financial services company with 7 teams and 40 microservices.

Aims

This report gives an overview of what good looks like for a cloud operating model, when a large organisation needs to simultaneously innovate in cloud-native digital services and enhance a heritage estate of critical heritage services.

We know from past experience there's no one way of working, one technology stack, or one blueprint for all problems. We outline the squads, tools, capabilities and technologies that are needed to build up digital capabilities and increase business agility.

Some caveats are necessary. This report deliberately does not contain:

- Advice on portfolio management
- Opinions on the exact composition of families, domains, and squads
- Advice on Data platforms / SaaS platforms
- In-depth detail on technology choices

Executive summary

Very broadly technology estates exist to:

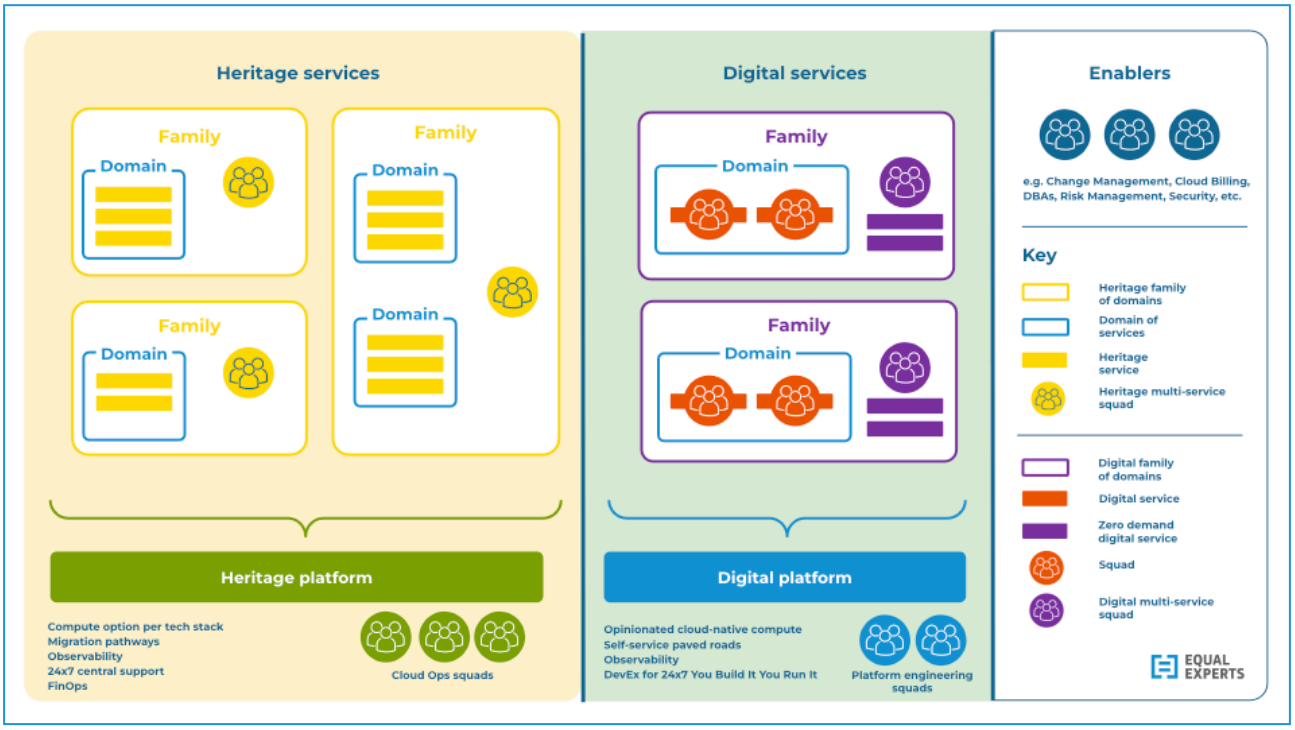
- Create new revenue streams and reduce operational costs; by innovating in cloud-native digital services.
- Protect existing revenue streams and reduce operational costs; by enhancing a heritage estate of critical heritage services.

We recommend organisations establish a cloud operating model by dividing application technology capabilities into **heritage services** and **digital services**. Heritage services have a high change cost and low demand, digital services have a low change cost and variable demand. This will enable appropriate levels of investment in ways of working, training, and technology according to feature demand and availability requirements. This should be complemented by appropriate domain organisation, service categorisation, and ways of working.

We advise that digital services be built and run as cloud-native workloads, by empowered engineering squads. These squads should be organised into domains and product families under the guidance of dedicated product managers. Organising squads by product family will allow a more optimal architecture to emerge. To run cloud-native workloads, we advise building a **Digital Platform**, of paved roads and opinionated patterns.

We recommend heritage services - COBOL monoliths, Java microservices, anything once on-premises - are maintained by multi-service squads due to low demand and cost efficiencies. Heritage services should also be organised by domain and product family. To run heritage workloads we suggest building a **Heritage Platform**, with migration pathways, a limited set of compute options and consistent baseline operability.

Which squad type is most appropriate for which service type can be guided by the squad selection matrix. We also offer guidance on the essential principles and practices for a digital operating model.



Operating model overview - heritage services, digital services, and enablers

Operating model

Services

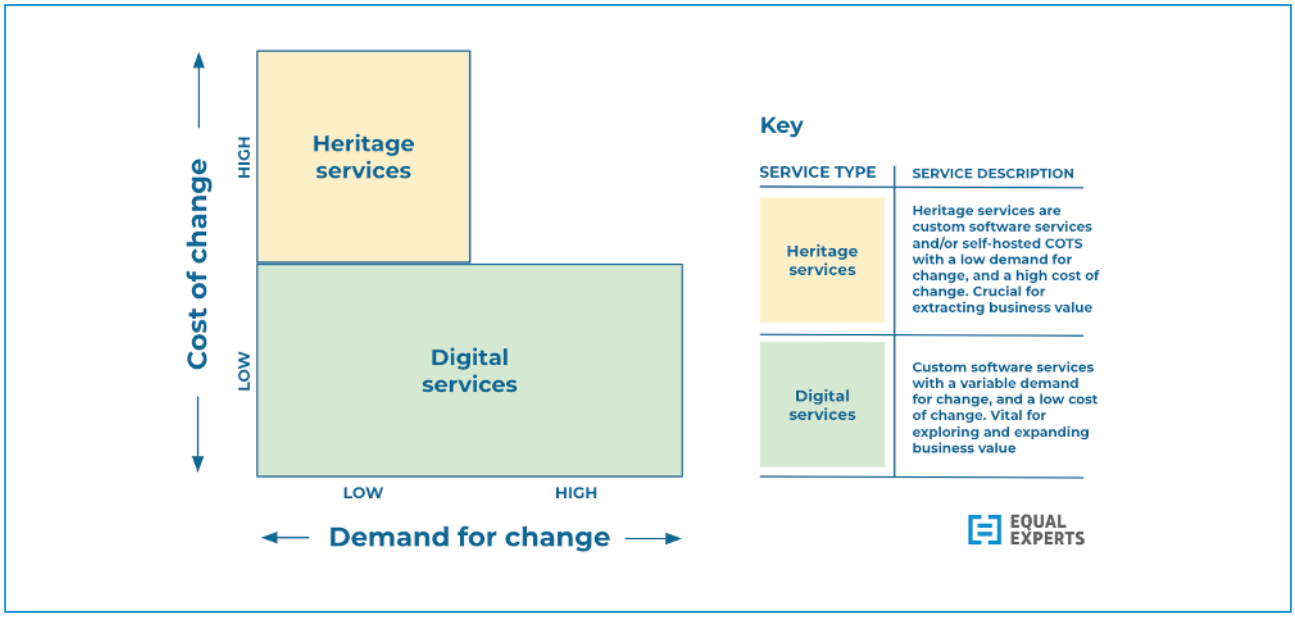
Software services are varied. They differ in technology, squad ownership, criticality, age, feature demand and many other dimensions. They may be customer facing products, or internal supporting functionality to other products and services.

> *Encourage product and technology leaders to collaborate on a clear definition of desired business outcomes and digital products, and then establish product families and product domains. This will create contextual alignment for squads, and help them to make better decisions*

A cloud operating model implements organisational structures that allow for rapid iteration and improvement in digital products. This is best accomplished by segregating technology capabilities according to demand for business change, and cost of business change.

Name	Description
Demand for business change	User expectations for new product features in a software service
Cost of business change	Transaction cost of implementing an idea and providing it to users in a software service

This produces technology capabilities consisting of **heritage services** and **digital services**. It enables different levels of investment in ways of working, employee training, technology stack etc. according to business needs. It acts as a driver of future business agility.



Services selector - heritage services and digital services

Heritage services are custom software services and/or self-hosted COTS with a low demand for change and a high cost of change. They lack dedicated funding and are likely to have been in operation for years. Their reliability is crucial to the ongoing extraction of business value.

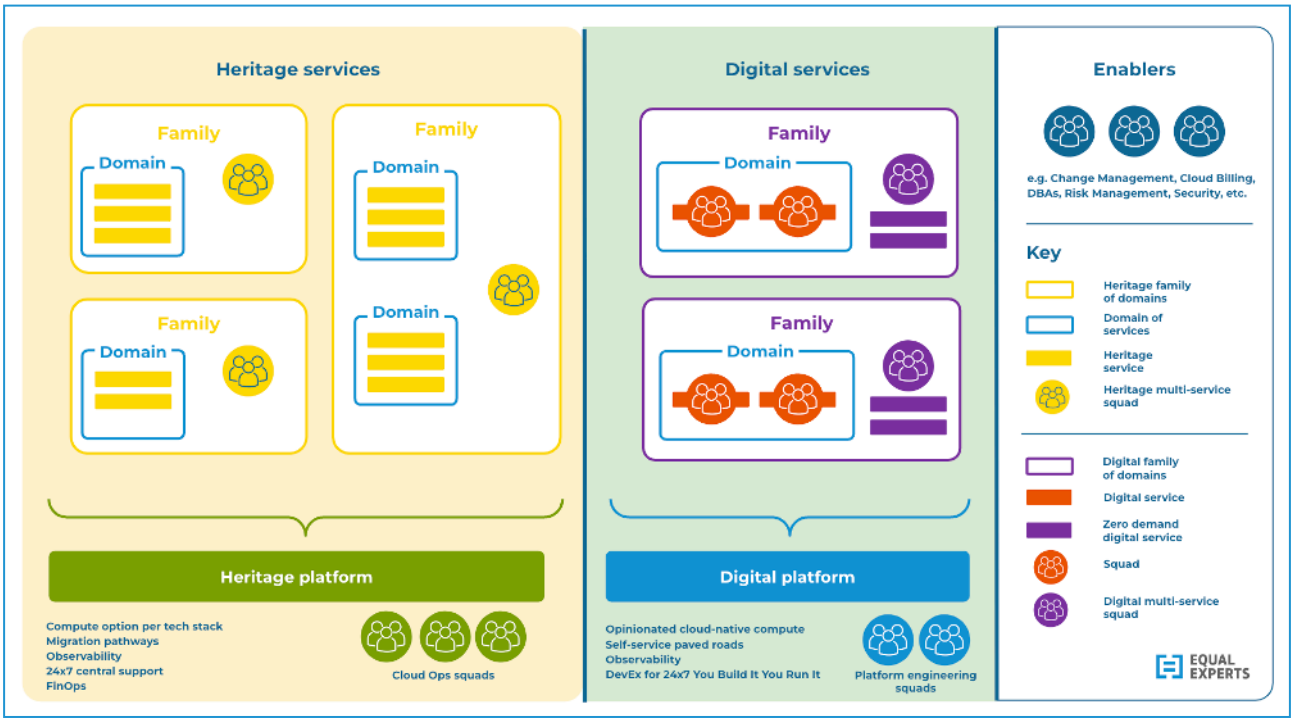
Digital services are custom software services with a variable demand for change, and a low cost of change. They have dedicated funding, until demand tails off to zero. Their malleability and reliability are vital for ongoing exploration and expansion of business value.

Socialise this heuristic for heritage vs. digital services:

"If it's ever been on-premises then it's a heritage service, if it's cloud native then it's a digital service"

Run a workshop to plot future investment in heritage services.

Below is a high-level operating model diagram, which describes how digital services and heritage services can be organised and co-exist as different technology capabilities. The same enabling teams are available to all squads, such as change management and security.



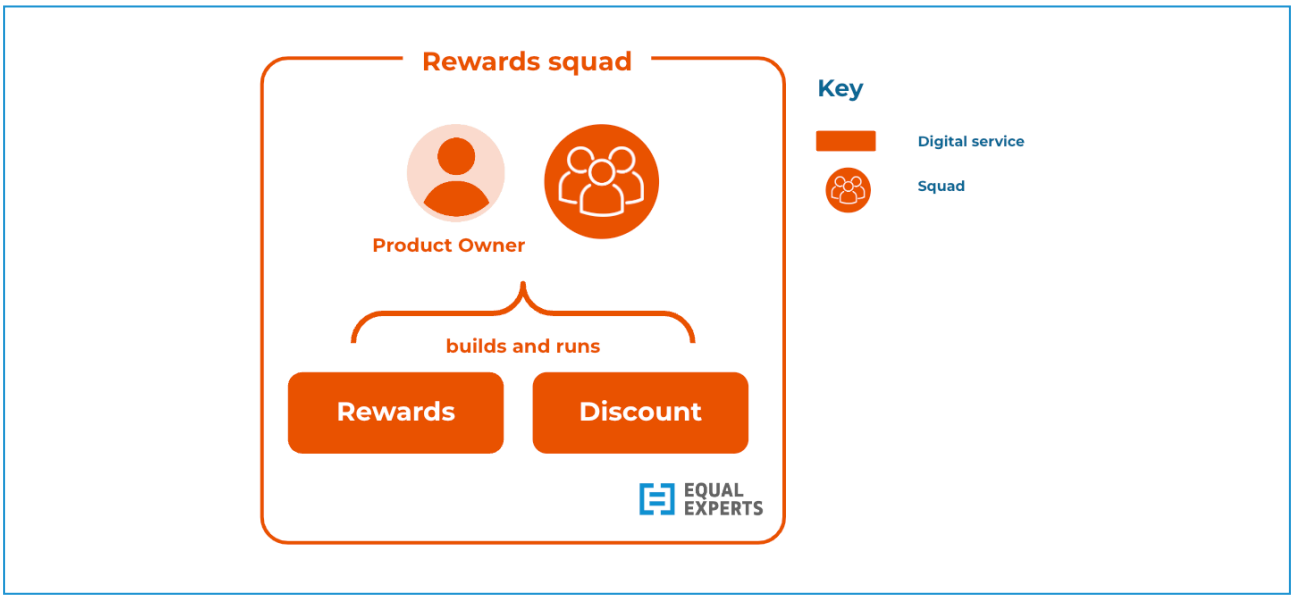
Operating model overview - heritage services, digital services, and enablers

Assign a technical architect and technical delivery manager to oversee the delivery of digital services across all product families. Ensure they are responsible for eliminating/softening dependencies and encouraging collaboration across squads.

Gradually build up to one architect and one delivery manager per product family

Squads

A squad is a cross-functional team of individuals with different skill sets. Squads flex and change to have the skills, knowledge, and experience they need for the service(s) they own.



An example digital service squad

Squad selection

The type of squad that is required to build, maintain, and run software services is a key decision in any cloud operating model. Maintenance is an important consideration alongside build and run, because all live software services will require indefinite BAU maintenance work (e.g. library upgrades, vulnerability fixes) and occasional product feature updates.

We recommend per-service squad selection is calculated from feature demand and service availability requirements.

Name	Description	Target measure	Digital services impact	Heritage services impact
Service feature demand	User expectations for new product features. Does not follow a linear progression, as some business events may be seasonal e.g. end of tax year or one-off e.g. regulatory changes	Deployment frequency - ranging from daily deploys down to BAU only deploys	High before and during launch. Continues to be high for differentiators. Slowly tails off to zero for commodities	Permanently low to zero
Service availability	Exposure to failure. Falls into a per-organisation range based on number of nines. Requires substantial investment in engineering effort to increase as is levels	Availability target - ranging from 99.0% (max 7 hours loss per month) up to 99.9% (max 43 mins loss per month)	Updated by financial exposure learnings, and actual losses incurred, during chaos testing and live incidents	Updated by financial exposure learnings, and actual losses incurred, during chaos testing and live incidents

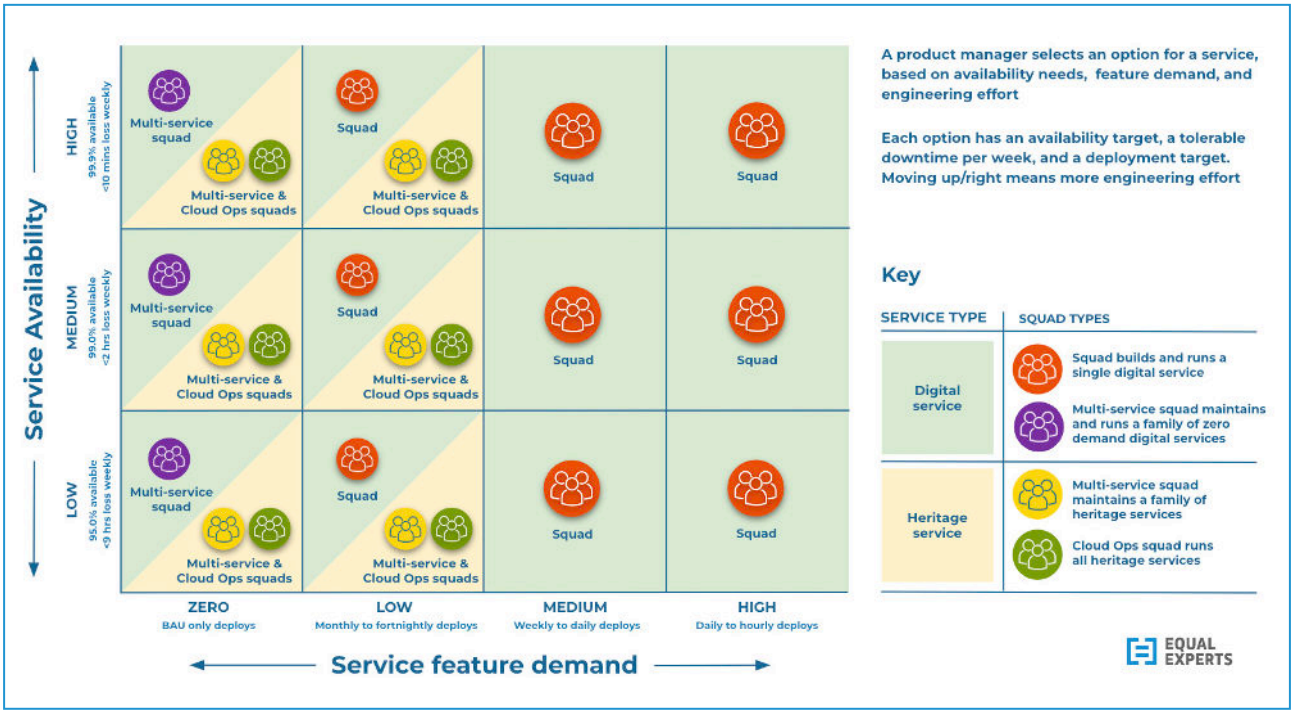
There are many dimensions of interest for a software service and its owning squad, e.g. live traffic volume, known security vulnerabilities, technology stack. It's important that squads are empowered to make their own decisions, and are supplied with the necessary contextual alignment to match their autonomy.

Capture service availability levels as achieved, and make the data available to all squads. Don't pursue theoretical ideals such as 99.99% availability, unless genuinely required for regulatory reasons. 99.99% is costly to achieve, and usually unnecessary

Create a range of availability levels from 99.0% to 99.9%, map targets onto all heritage and digital services, and ensure graceful degradation is possible when critical services are unavailable

A squad has a single owner - the accountable budget holder. For a digital service, that is the product owner who owns the dedicated funding for ongoing delivery. For a foundational service, that is the product lead who would own any funding for future feature development. The squad owner needs to be in charge of prioritisation decisions on product features, reliability features, and ongoing BAU maintenance work. This ensures people have the correct incentives to balance user needs, speed of delivery, and technical quality.

A heritage service with an reliable, fast and frequent release pipeline may be able to support a dedicated service squad to build it. A heritage service is unlikely to achieve daily builds. Investment would be better spent on migrations.

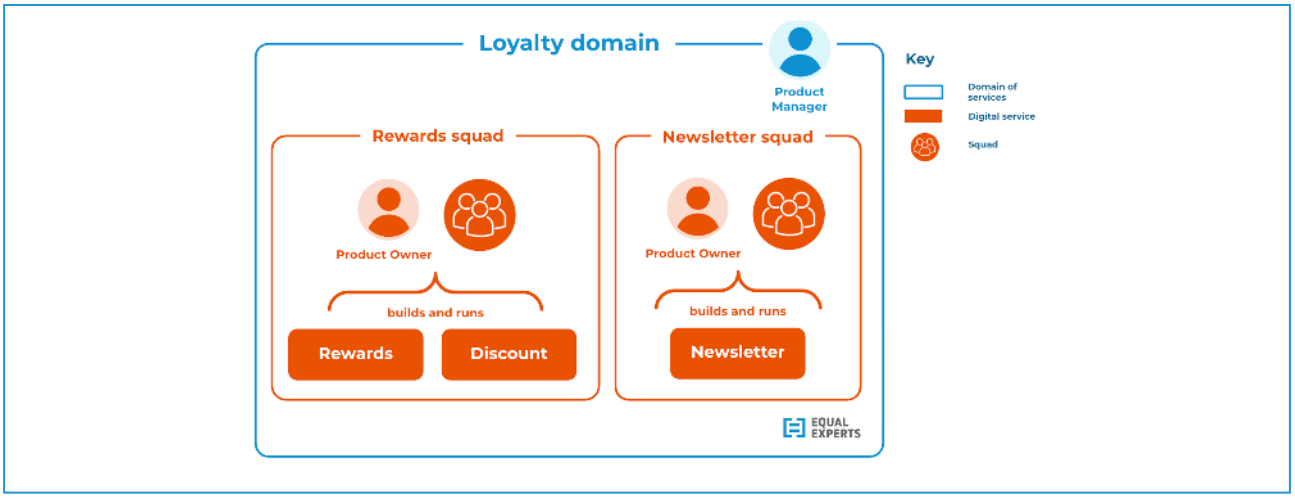


Squads selector - digital services and heritage services

Domains

A domain is a logical grouping of software services with an established product affinity. Both the software services and squads within a domain are considered to be siblings. A domain has a product manager, to ensure a single business owner can make prioritisation decisions across squads and their own product owners.

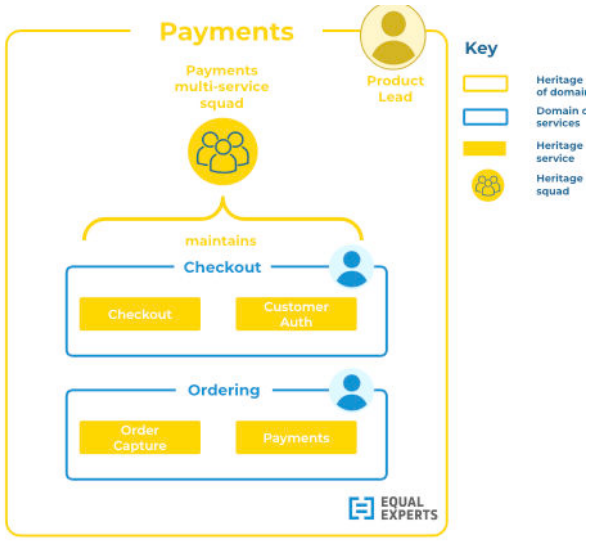
Implementing domains by product affinity minimises cognitive load, simplifies knowledge sharing between squads, encourages squads to focus on user outcomes, and ensures that rapid prioritisation decisions can be made when necessary. Other implementation routes e.g. technology affinity result in matrix management, and unavoidably slow decision-making.



An example product domain

Families

A family is a logical grouping of domains with an established product affinity. Both the software services and squads within a family are considered to be relatives. A family has a product lead, to ensure a single business owner can make prioritisation decisions when necessary across all squads and services.



An example product family

Find the product families and product domains that product managers already recognise within your organisation, and use those constructs to forge consensus on how to deliver heritage and foundational services in the future

Heritage services

Heritage services are custom software services and/or self-hosted COTS with a low demand for change and a high cost of change. They are often crucial services that are at the core of many organisations.

Heritage services need to be maintained and run to a high standard. The consistently low demand for business change means all heritage services can be considered zero-demand services. There will be an occasional need for new product features, perhaps due to regulatory changes or a burst of market demand.

A heritage service will comprise one or more previous-generation, cloud-last workloads. The cost of change is significant, even for BAU maintenance work, and any on-premises workloads are exposed to high running costs, security vulnerabilities, and other operational risks.

A heritage platform is the appropriate home for heritage services. A heritage platform is a cloud-based modern software platform, formed of managed services from the cloud provider. It provides a dedicated migration pathway for each type of heritage workload, hosted in on-premises data centres or raw cloud infrastructure. It's owned by a platform product manager, who is empowered to balance ongoing investment, maintenance and support needs of the most critical heritage services.

A heritage platform is operated by two Cloud Ops squads:

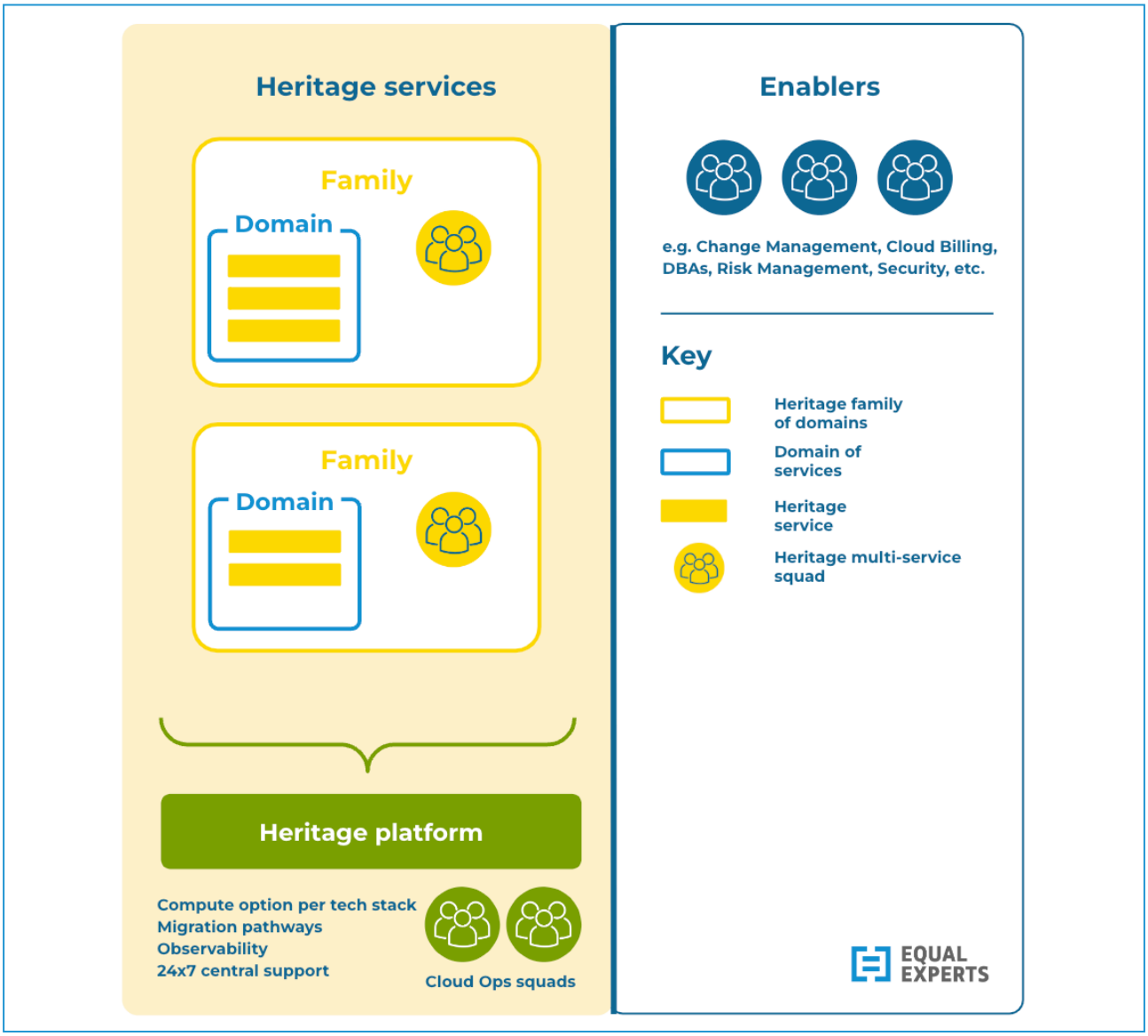
- Enhancements. Responsible for capabilities and migration pathways
- Support. Responsible for 24x7 on-call support for all heritage services

Invest in a Heritage Platform to run heritage services. Lift and shift COBOL, PHP, and Java workloads onto it, to minimise migration costs and deliver ahead of any cloud migration deadline.

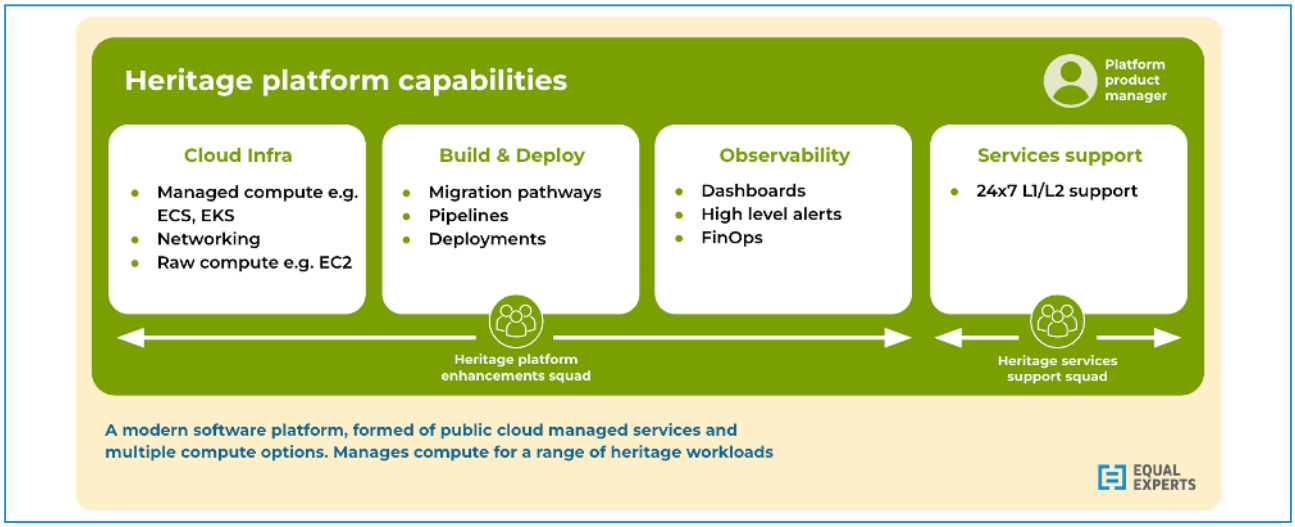
Balance engineering effort with future run costs and operational risks, by building the migration pathways for common stacks in your organisation.

For example:
COBOL monoliths to EC2
Java microservices from KOPS to EKS
Any other containerised workloads to ECS

A heritage service runs indefinitely until its eventual replacement by SaaS or new digital capabilities. In a minority of situations, a heritage service might see a higher level of feature demand, and its product owner has to decide whether to immediately fund a rebuild. A heritage service cannot be migrated to a digital platform, as it signifies a very different level of investment. This is discussed more in Migrate Goldilocks heritage services first.



Heritage services operating model - families, domains, services, and heritage platform



Heritage platform capabilities and squads

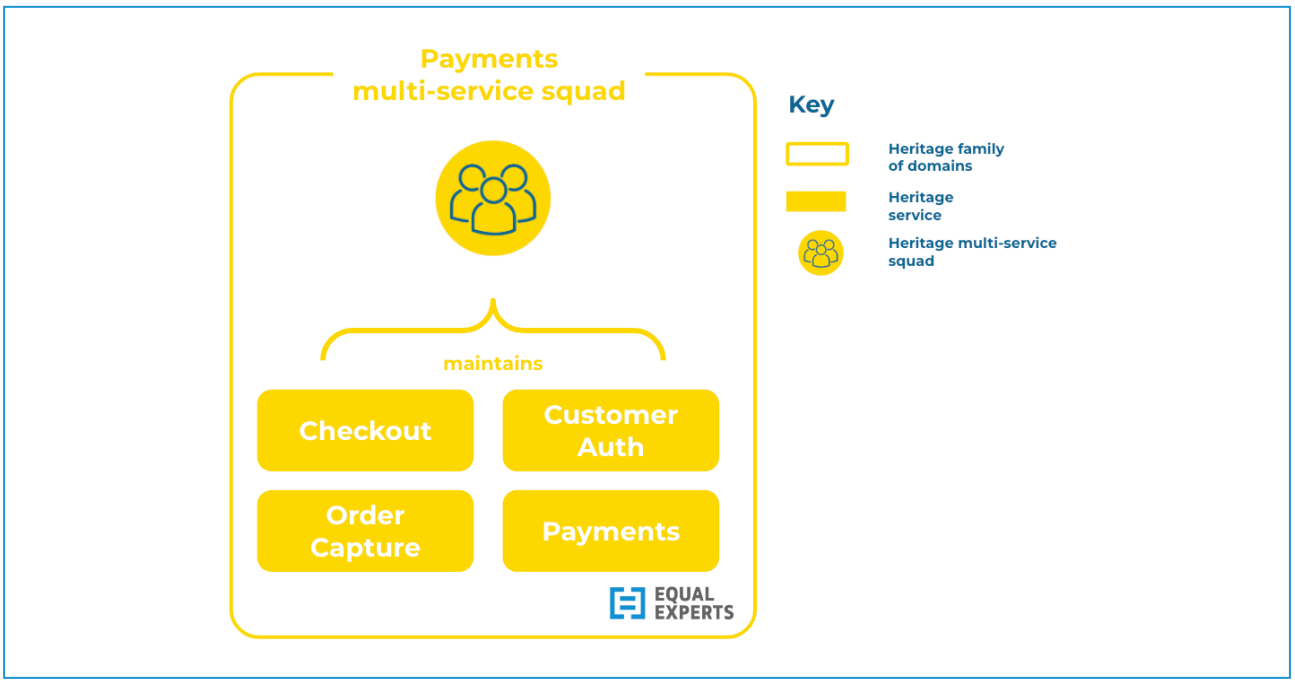
Squads

There's insufficient demand for business change in individual heritage services to justify dedicated squads. Nor is there enough demand to merit practices such as You Build It You Run It. Multi-service squads are the cost-effective choice for heritage services.

Multi-service squads

A multi-service squad is a cross-functional team of individuals that is accountable for the maintenance of all heritage services within a product family. By definition, workloads have low or zero demand for new product features. Ongoing maintenance work is necessary to preserve technical quality.

A multi-service squad has a product lead, who can prioritise urgent maintenance work across heritage workloads when necessary.



An example multi-service squad for heritage services

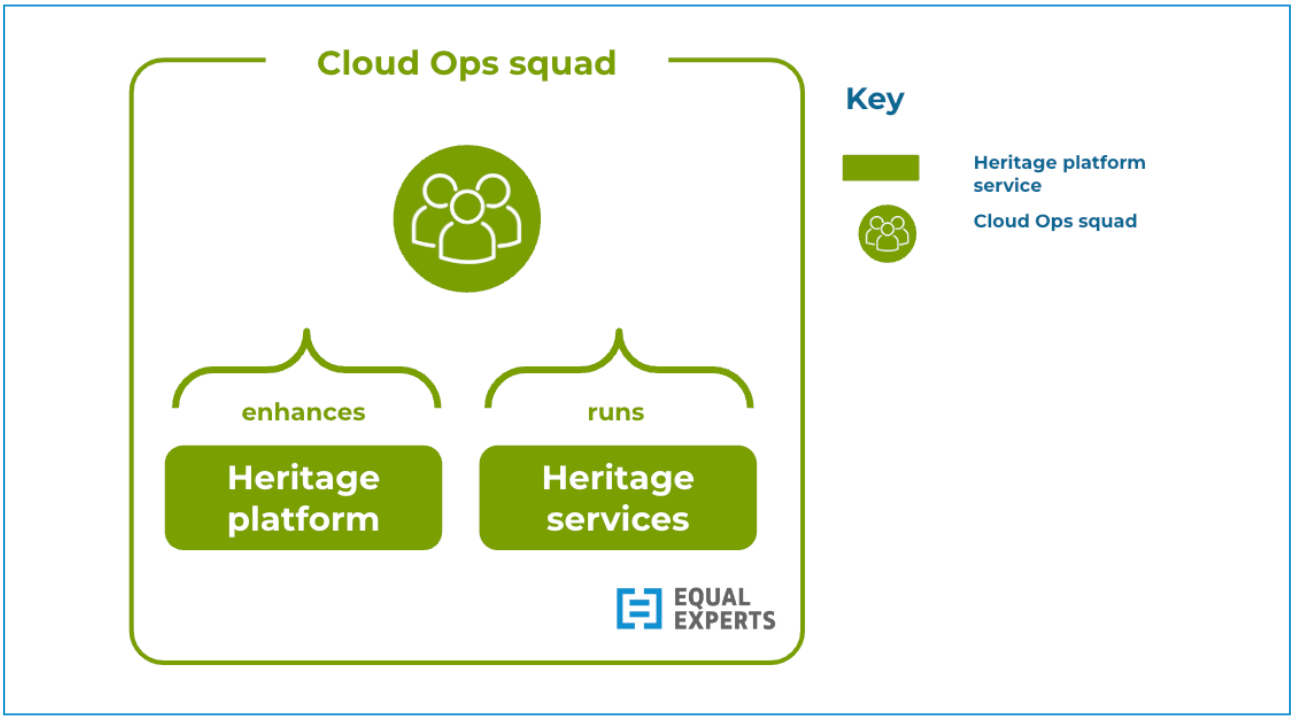
Cloud Ops squads

A Cloud Ops squad is a team of application support engineers, well versed in cloud infrastructure and operations. They are accountable for improvements to the heritage platform, or the ongoing 24x7 support of heritage services running on the heritage platform. If a Cloud Ops squad becomes too large due to the number of specialist engineers required per compute option, it can be split into multiple squads by compute option.

A Cloud Ops team can be staffed by application support engineers who have previously maintained and supported heritage services. Their business knowledge and operational experiences are likely to be invaluable.

Offer new career pathways and training opportunities to application support employees, as heritage services are lifted and shifted from on-premises to a heritage platform:

- *Join the Enhancements squad, and develop cloud infrastructure*
- *Join the Support squad and support heritage services as cloud workloads*
- *Join a multi-service squad and maintain multiple heritage services*

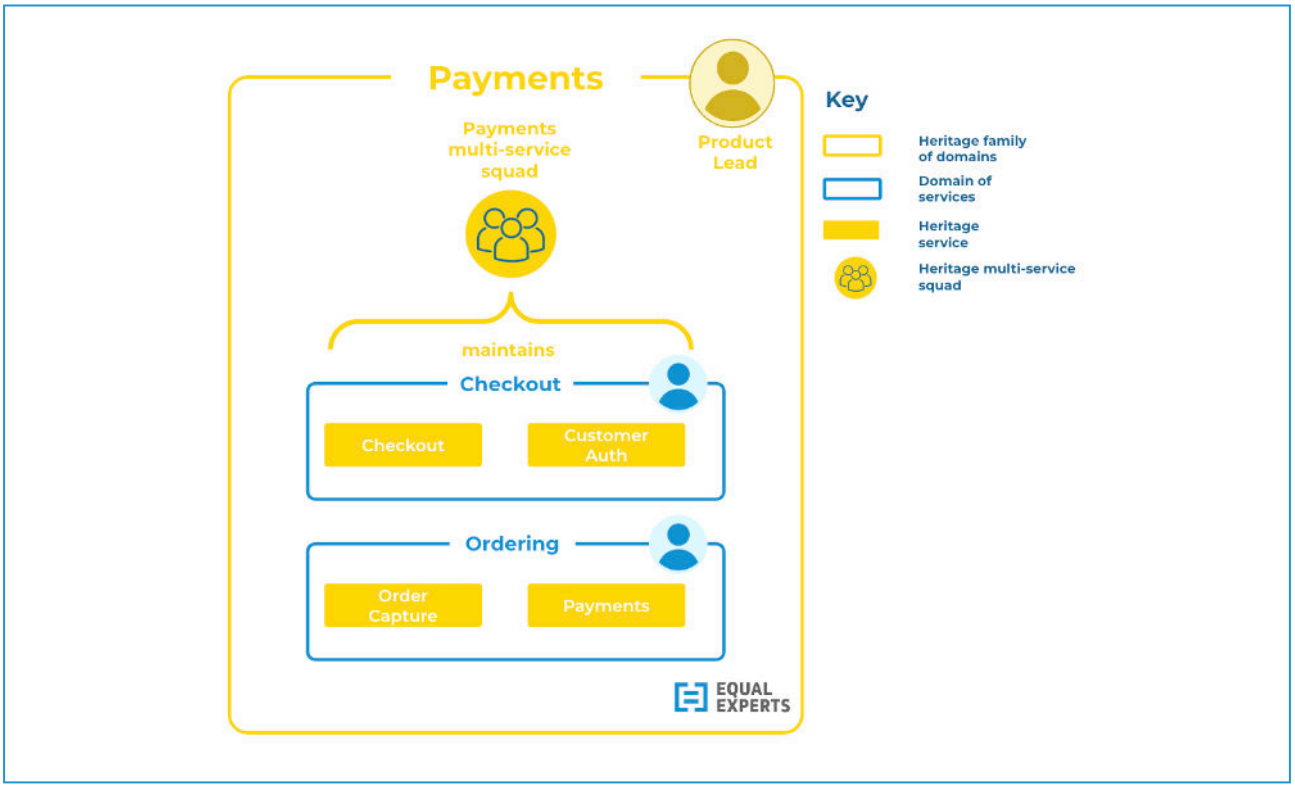


An example Cloud Ops squad

Domains and families

A family of heritage services is a logical grouping with an established product affinity. This means a single business owner able to make swift prioritisation decisions across many live workloads with distinct maintenance and operational needs.

If it's not possible to group heritage services by product family, the fallback is to organise by compute options on the heritage platform. A focus on technology over business function introduces complex prioritisation problems and sub-optimal architectural choices, due to Conway's Law.



An example product family of domains and heritage services

Digital services

Digital services need to be designed for iteration and experimentation to satisfy user needs. Demand for change varies through the lifecycle of a digital service. There is high demand as features are continually added and made available to a growing population of users, or trialled by a small population in preparation for a wider rollout. If a digital service is a differentiator, that high demand could persist for the long term. If it is a commodity, demand will gradually slow down until the product owner can declare zero demand.

A fast, frequent flow in digital service delivery is essential. Releases can be de-risked and accelerated by decoupling the technical action of deploy from the business action, with feature toggles, phased rollouts, branch by abstraction, etc. A new digital service can be introduced into production gradually, until 100% live traffic and switching off any predecessor heritage workloads is a complete non-event.

A digital platform allows digital service squads to rapidly innovate, and run their own live services to an exceptional standard. A digital platform is a cloud-based custom platform, built on top of managed services from the cloud provider. It's owned by a platform product manager, who allocates investment based on the emerging needs of digital service squads, and drives the vision of enabling squads to constantly deliver innovation to users.

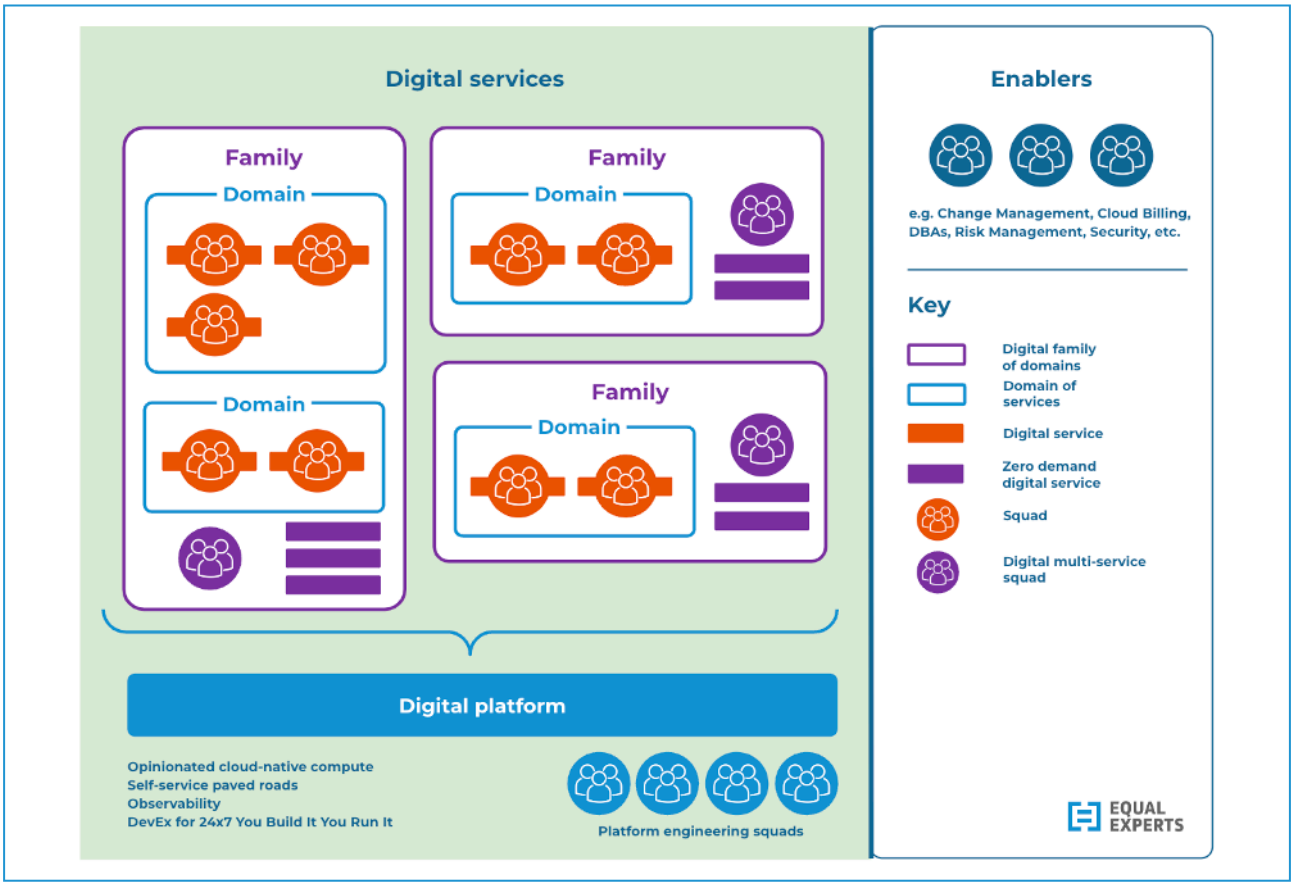
A digital platform offers self-service paved roads, and capabilities built around a standard compute option that creates a force multiplier effect. Lightweight, opinionated compute is vital to remove friction and accelerate digital services delivery. See Do more with less by picking compute that reduces maintenance and How digital platforms fail by Adam Hansrod.

To maximise economies of scale, a digital platform needs multiple platform engineering squads:

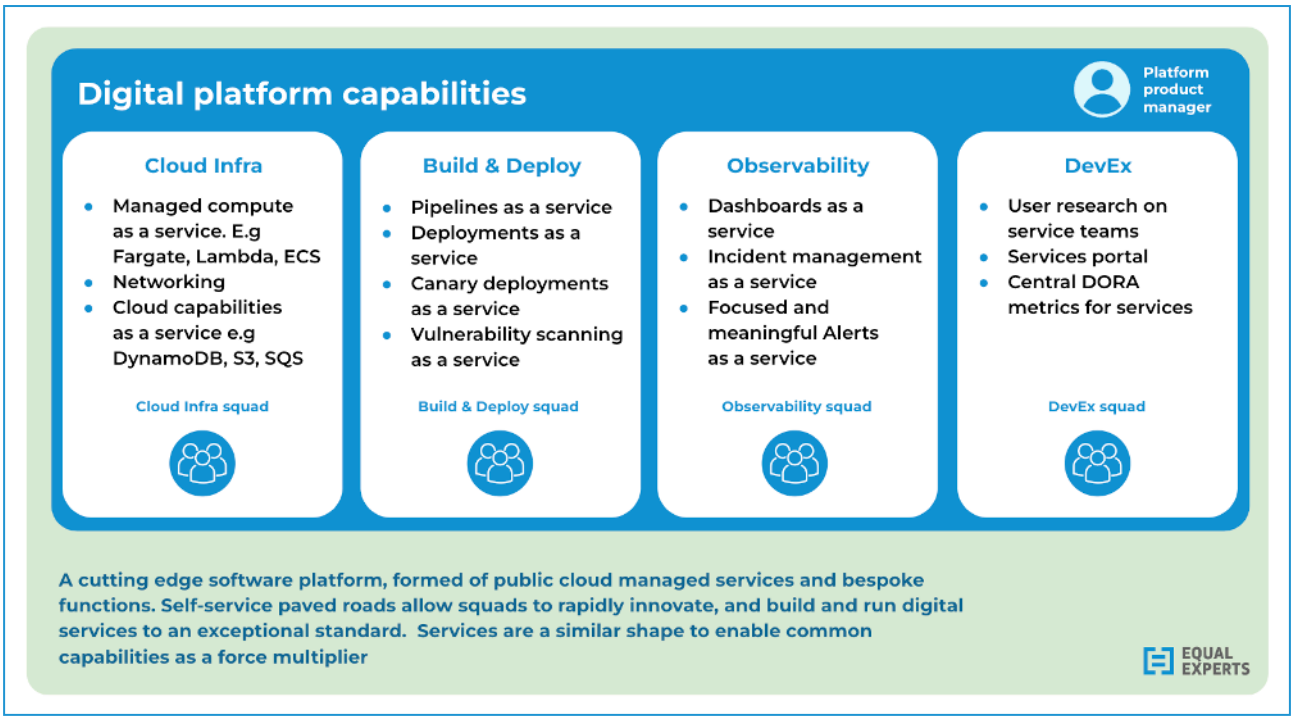
- Cloud Infra. Builds core, self-service platform capabilities
- Build & Deploy. Creates pipelines as a service, canary deploys, etc.
- Observability. Offers monitoring, alerting, incident routing
- DevEx. Works with digital service squads to tighten feedback loops for Continuous Delivery and You Build It You Run It

It's important to remember that the platform itself will require the same level of observability, build and deployment patterns etc. The platform squads will work together to ensure engineering excellence for the platform itself. Some of the tools for the platform's customers will be directly usable, some will be bespoke.

Build a **Digital Platform** *to run digital services. Provide opinionated platform capabilities based around a single compute option, so squads can quickly create cloud-native workloads and collect feedback from users*



Digital services operating model - families, domains, services, and platform



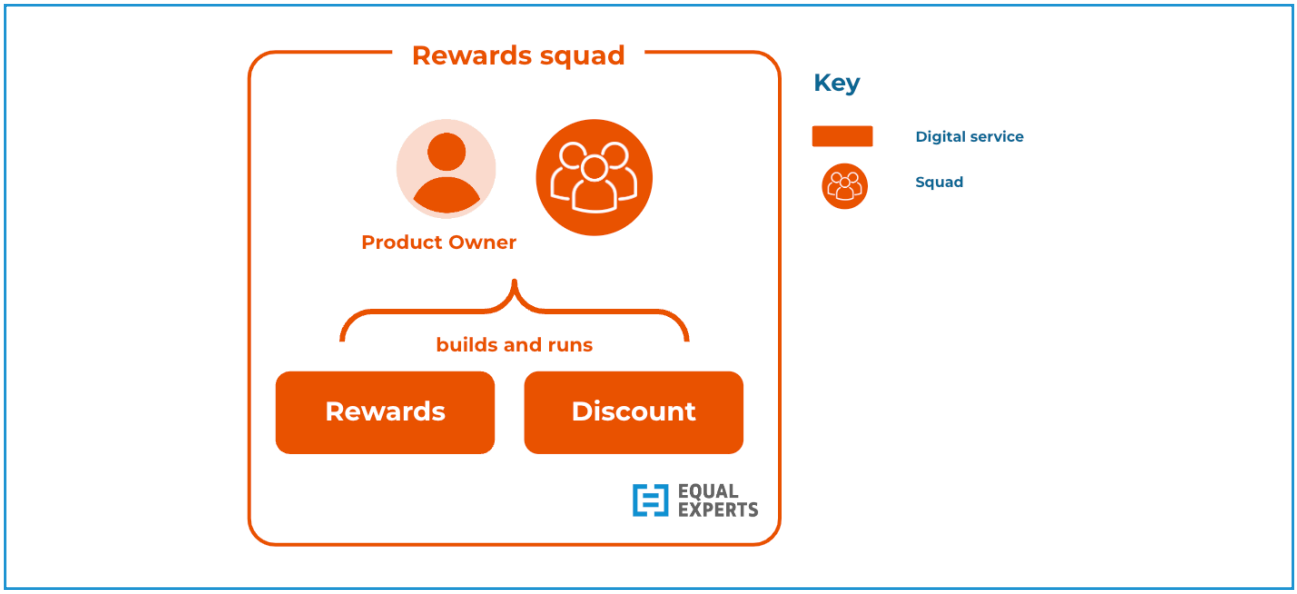
Digital platform - capabilities and squads

Squads

Digital service squads

A digital service squad is a cross-functional team of complementary individuals with different skill sets, who come together to research, design, build, and run a live digital service. Required skill sets depend on the digital service. For example, a squad owning a user-facing portal needs frontend engineers, designers, and user researchers.

A digital service squad has a product owner who balances priorities between new feature work, defect fixes, and regular maintenance work. When necessary, a squad collaborates with other squads in the same product domain, or squads across the product family, in order to deliver a shared user outcome.



An example digital service squad

> *Ensure squads for digital services have a mix of technical, delivery, design and other roles. User researchers and user experience analysts need to be in squads and contribute to delivery outcomes. They might be shared between teams for expediency, but do not put them into a functionally siloed team of their own*

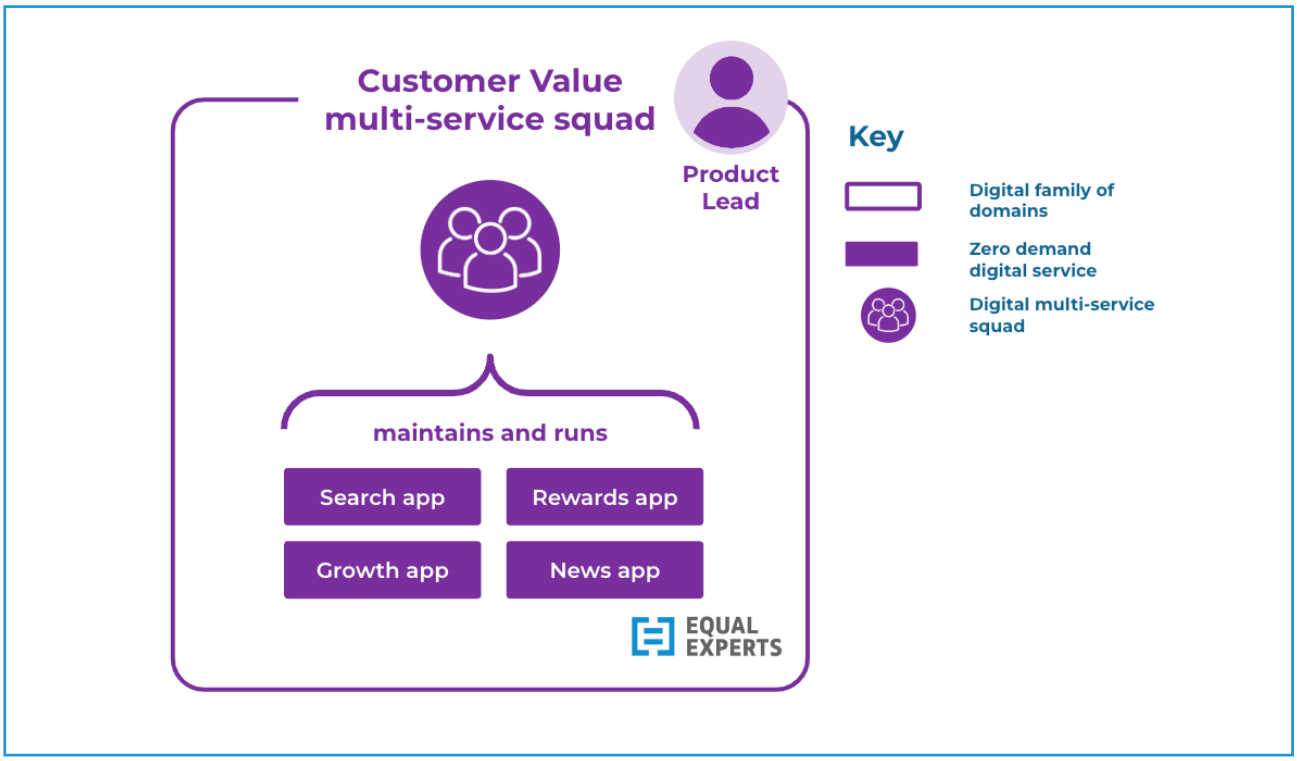
Multi-service squads

It isn't cost-effective for every digital service to be built and run by its own squad forever. When demand for change slows down, and a digital service has been live for at least 3 months with its own squad, its product owner can declare zero demand. At that point, the squad selector is revisited and the digital service is transferred into the multi-service squad for that same product family.

A multi-service squad is a cross-functional team of individuals who are accountable for the maintenance of all zero demand digital services within a product family. BAU maintenance work, monitoring, live deployments, and incident response are all performed. See why developers doing maintenance mode is best for your business by Steve Smith.

A multi-service squad has a product lead for the product family, who:

- decides when a live digital service has reached zero demand, and can be transferred from its squad into the multi-service squad, for maintenance mode
- prioritises urgent maintenance work across digital workloads in that family
- decides when a digital service has sufficient new demand to merit its transfer back from the multi-service squad into a digital service squad, for further development

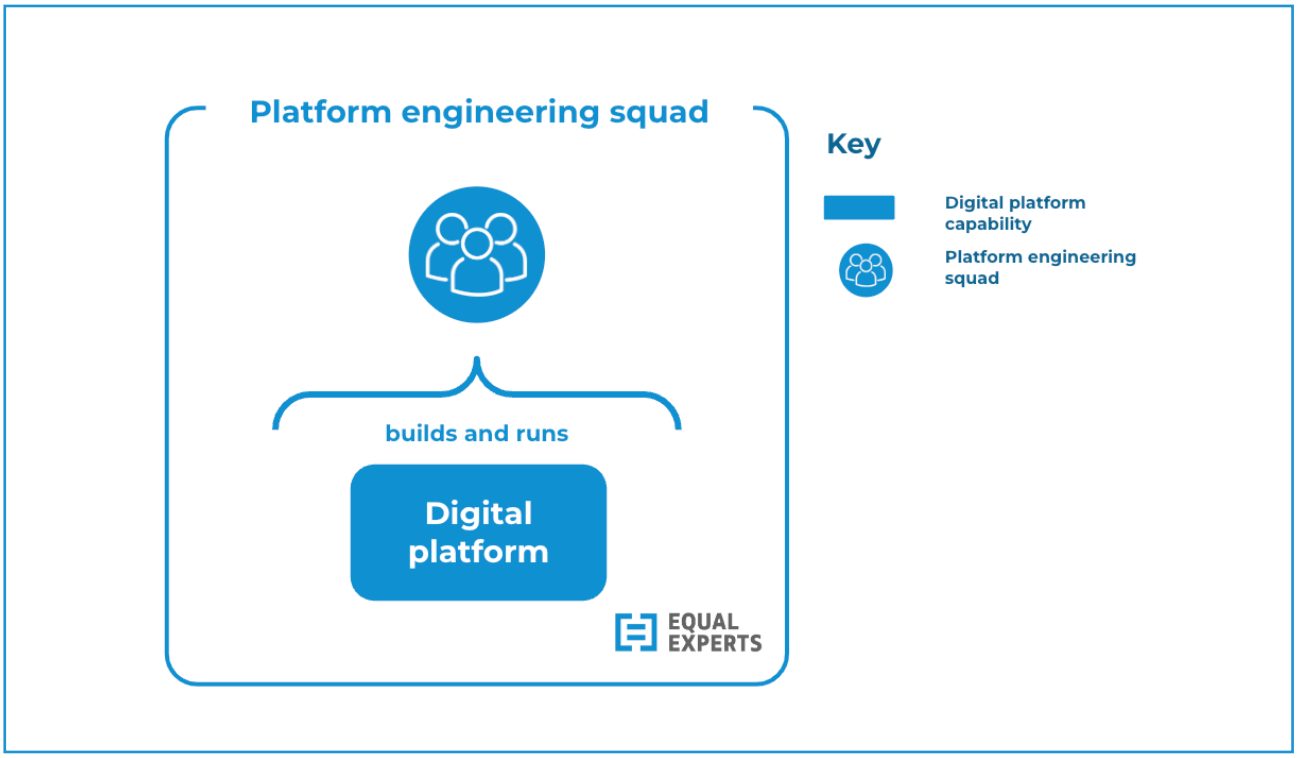


An example multi-service squad

Platform engineering squads

A platform engineering squad is a team of platform engineers, well versed in the use of cloud infrastructure and operations as accelerators for digital service squads. They are accountable for building and running all digital platform capabilities. They do not support any live digital services on the digital platform, as that is covered by digital service and multi-service squads.

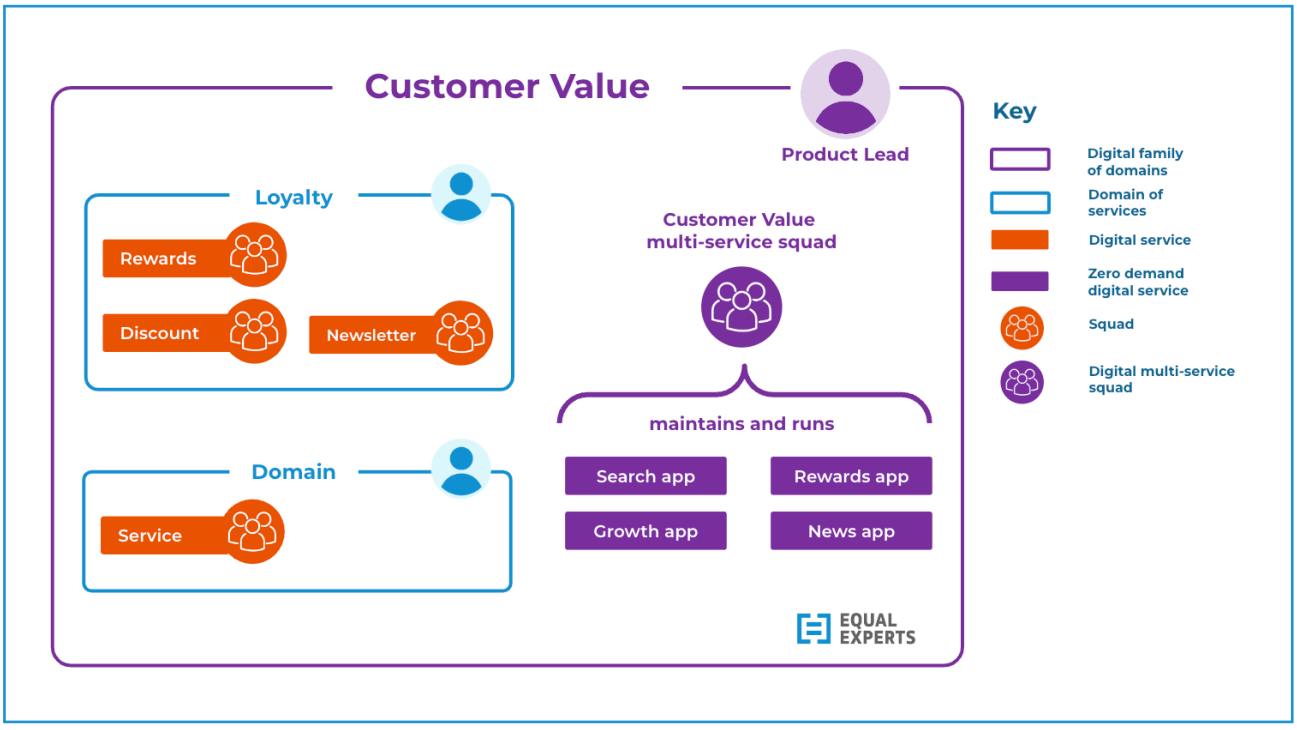
A platform engineering squad reports into the platform product manager, who oversees the vision and capabilities of the digital platform, across all platform engineering squads.



An example platform engineering squad

Domains and families

A family of digital services is a logical grouping with an established affinity around one or more digital product capabilities. There's a single business owner, who is able to prioritise work across in-development, live, and zero demand digital services when necessary.



An example product family of product domains, digital services, and a multi-service squad with zero demand services

Principles

Well reasoned principles enable planning for the future, and an effective response to changing situations. As the digital operating model evolves over time, these principles can be used to assess if proposed changes are appropriate.

Focus investment where it matters

Investment needs to be concentrated where it can have the most positive business impact. Digital services and heritage services must be treated differently because the demand for business change is so much greater than the latter. Invest in heritage services until their product leads consider the cost of change to have become tolerable. Invest in digital services until their product leads believe the demand for change has been satisfied. The latter will possess far more business opportunities, and take longer to achieve.

Don't overthink the migration of heritage services into a heritage platform. COBOL and PH monoliths running on EC2 is fine. Java microservices running on EKS is fine.

Don't overinvest when there is demand for business change is so much higher elsewhere. Lift and shift is perfectly acceptable

Prioritise fast feedback

Accelerating feedback loops is essential. It's important to learn quickly, in all aspects of the software lifecycle. Swift feedback allows engineers to identify and rectify issues early in the development process, leading to fewer defects and greater quality. It allows squads to gather real world feedback and insights, enabling them to make informed decisions about future development directions. It ensures the desires of a product owner are aligned closely with user expectations, leading to lower opportunity costs and higher revenues.

Maximise lightweight cloud services

Use lightweight managed services from the cloud provider wherever possible. Build differentiators and rent commodities. Favour the lower cognitive load, fewer security vulnerabilities, and faster development time of managed cloud services over fears of vendor lock-in. Avoid not invented here syndrome, and free up squads to focus on business problems, not esoteric technology roadblocks.

Don't commit to any technology choices that require daily babysitting. Kafka, Kubernetes, and Istio are classic examples - even when managed by the cloud provider, they need daily attention and massively increase unplanned work for squads. Use tools like SQS, AppRunner, and AppMesh instead

Optimise for resilience, not robustness

Everything fails all the time, and most failures are caused by surprise circumstances, not deployments. Being able to rapidly respond to failure is more cost effective than having failures less often, and a faster MTTR is more important than a lower MTBF. Create a production environment in which all services can gracefully adapt to unpredictable conditions. Replace expensive, slow, and unrealistic test environments that aren't live with cheaper, faster, and realistic observability tools that are live. See Why stopping deployments doesn't stop incidents by Caitlin Smith et al.

Insist upon engineering excellence

Foster a culture of quality engineering practices. Ensure that technical leadership keep standards high, senior engineers have the time to mentor more junior engineers, and junior engineers are confident enough to take risks and learn from mistakes. Teach engineers at all levels the importance of continuous delivery practices such as test-driven development, feature toggles, and trunk-based development. Encourage squads to hold themselves to account with a technical team charter they draw up themselves.

Share service management responsibilities

Everyone in a squad or an enabling role has a responsibility to ensure that regulatory checks and business value delivery are continuously happening, side by side. This is applicable to all software services. People need to work together to create streamlined, efficient service management capabilities that satisfy internal compliance checks and any external regulations e.g. who has permissions to change which software service, when did the last deployment happen, where are data backups physically hosted.

Create automated integrations between GitLab CI, PagerDuty, and ServiceNow for all heritage and digital services. This removes friction and toil from day to day engineering tasks, and it also creates a reliable, fully automated audit trail for any future compliance efforts

Create employee learning pathways

Sign up to organisation-wide cloud training, encourage cross-squad rotations, and fund leadership courses to boost skills liquidity across squads. Give senior engineers the chance to lead new digital service squads. Place junior engineers in multi-service squads, to expose them to live traffic and added responsibilities. Allow individuals to find new opportunities within the organisation, if and when the digital operating model moves away from their current skill set.

Practices

These are some digital operating model practices. Other practices will emerge as the principles are applied to different scenarios.

Governance

Make a budget holder wholly accountable

Empower the budget holder for a heritage service or digital service to be accountable for both feature demand and availability.A single decision maker should prioritise product features alongside operational features, to strike a balance between future functionality and current reliability.

Prevent any scenario in which someone prioritises functionality and someone else prioritises reliability. Incentives are optimised for a squad when the same person is accountable for building and running a live service.

Implement contextual change management

Put the appropriate safeguards in place for all services, which are aligned with ITIL standards and based upon cost of change and demand for change:

- For heritage services, use Normal change requests and a CAB for live deployments due to change complexity and partially automated tasks
- For digital services, use Standard change requests and self approvals for live deployments due to opportunity costs and fully automated tasks

Ensure emergency changes are not treated as a special route. Implement an emergency patch deployment as a regular deployment solely with different versioning, and implement rollbacks as regular deployments with known good versions. Eliminate the need for manual sign-offs and empower squads to make their own decisions, wherever possible.

Recognise other platform needs

There will be many exceptions to the heritage services and digital services within a digital operating model. Examples include a data mesh running data services, or a SaaS platform running internal IT capabilities. Acknowledge these workloads are not well suited to a heritage platform for low demand, high change mature workloads, nor a digital platform for high demand, cloud-native workloads. Don't bend a platform to fit a workload, and don't bend a workload to fit a platform.

Bootstrap

Migrate `Goldilocks` heritage services first

When heritage services need to be migrated from on-premises or raw cloud infrastructure to the heritage platform, don't migrate the least risky workload first, or the most valuable. Balance value, cost, and risk by prioritising one or a few Goldilocks services - those that are just valuable enough, just costly enough, just risky enough to warrant an early migration.

Any migration is non-trivial, and requires planned, budgeted effort from engineers with specialised skill sets. Bake the lessons learned in those migrations from blockers, incidents, and retrospectives back into the dedicated migration pathways offered by the heritage platform, to smooth the journey for other heritage workloads in need of migration.

Precede a digital service with discovery

Start a new digital service with a product discovery. Research the user needs that are unmet, and how a new or existing service might satisfy those needs. The product manager for the domain can then decide if a new digital service is warranted, and balance the identified business need with other needs in the same domain. See Find out which type of discovery is right for you by Liz Leakey.

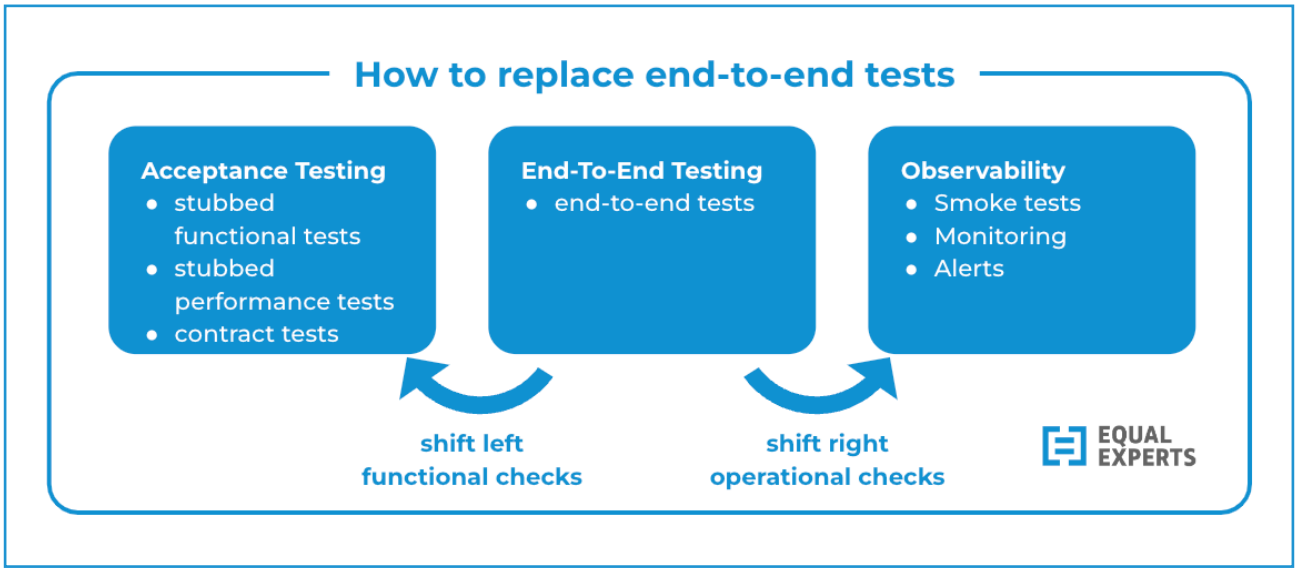
When the business need is next to be delivered, the product manager can estimate service availability and feature demand by examining the financial impact and user demand relative to pre-existing services. Then they can use the squads selector to confirm if a new digital service squad is required.

Testing

Incrementally replace end-to-end testing

Do not invest in functional end-to-end tests connected to unowned dependencies, and replace any such test suite as necessary. Recognise that end-to-end testing isn't cost effective, as no test environment can be genuinely live-like, and test execution time and non-determinism are directly proportional to test scope.

An end-to-end test can be replaced with a combination of other techniques. Shift functional checks left into unit tests of behaviour, stubbed acceptance tests of functionality, and contract tests of unowned dependencies. Shift operational checks into smoke tests, monitoring, and alerts in production. Start with only one test environment for digital services, restrict it to exploratory testing of unowned dependencies, and don't add other test environments unless production observability and fast rollback on a particular event are deemed impossible.



Delivery assurance

Measure the DORA metrics

Implement the DORA metrics by Dr. Nicole Forsgren et al, and visualise them in a services portal. The DORA metrics in the Accelerate book are

- Deployment frequency - how often production deployments happen
- Deployment lead time - time from mainline code commit to its deployment
- Change failure rate - how often live failures are triggered by deployments
- Time to restore service - how long it takes to resolve a failure
- Rework rate - how often squads have to complete non-value adding tasks

Many people misunderstand the DORA metrics as a target, and it's unhelpful. The question isn't if a heritage service has a lower deployment frequency than a digital service. The question is if a digital service has a lower deployment frequency than itself last month. That's a starting point for a conversation with a squad, and any delivery assurance actions can be shared with other squads for potential learnings. See How to win at delivery assurance with DORA metrics by Steve Smith.

Measure first deployment lead time

Use a specialised form of Deployment Lead Time to track the first live deployment of a digital service, before its DORA metrics become active. Build an automated measure of first deployment lead time for the digital platform, and incorporate it into the services portal. Implement the measure as the first code commit to the first production deployment.

> *Implement First Deployment Lead Time across all digital services as soon as possible, to visualise how much business value is currently unrealised in a queue for architecture panel dates, investment committee dates, etc. Assess the impact of current governance structures on deployment lead times*

Measure platform onboarding time

Use a specialised form of Deployment Lead Time to track platform onboarding speed. Build an automated measure of platform onboarding time for the heritage platform, and the digital platform, and incorporate it into the services portal. Implement the measure as platform signup to the first production deployment. Remove any obstacles to that first deployment, on the heritage platform or the digital platform.