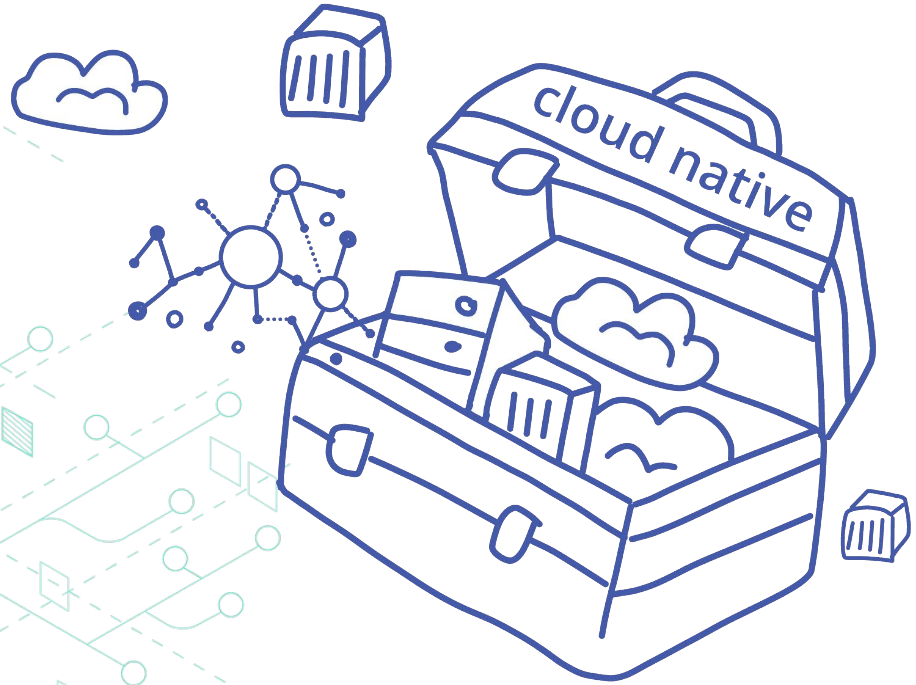
Cloud Native Transformation Patterns

A Method for Successful Cloud Migration

Jamie Dobson
@jamiedobson

Container Solutions

info@container-solutions.com
container-solutions.com



Cloud Native Patterns

A Method for Successful Cloud Migration

Jamie Dobson
@jamiedobson

Container Solutions

info@container-solutions.com
container-solutions.com

Container Solutions
Cloud Native Transformation Cards
MICROSERVICES ARCHITECTURE
To reduce the costs of coordination among teams delivering large monolithic applications, build the software as a suite of modular services that are built, deployed, and operated independently.

WealthGrid and Classic Mistakes

A Pattern Language for Cloud Native

Designing the Transformation

“All great literature is one of two stories; a man goes on a journey or a stranger comes to town.”

Leo Tolstoy

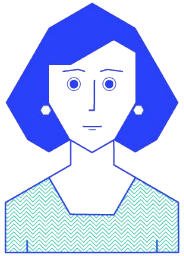
Meet

WEALTHGRID

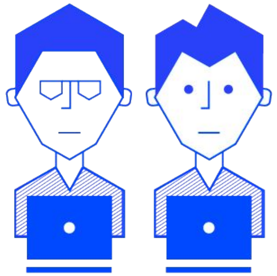
A successful, mid-size financial company

Meet the People

CEO

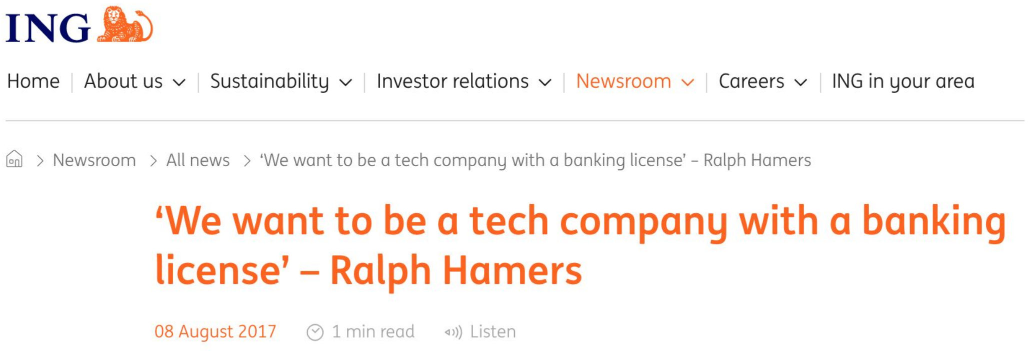


Jenny
a Technical Manager



Engineers

The Stranger is Coming…

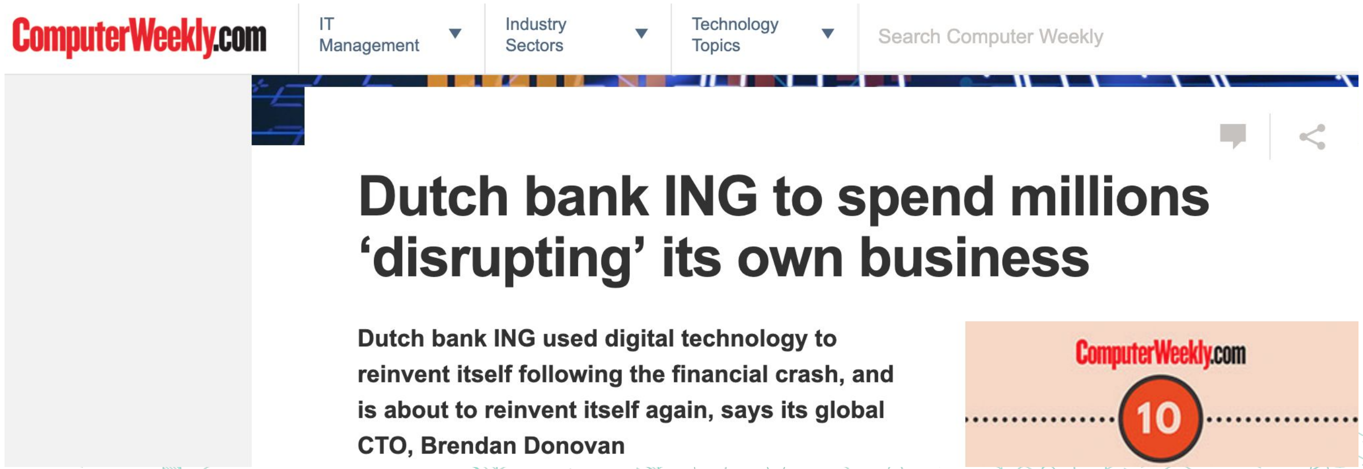


Home | About us | Sustainability | Investor relations | Newsroom | Careers | ING in your area

Newsroom > All news > 'We want to be a tech company with a banking license' – Ralph Hamers

'We want to be a tech company with a banking license' – Ralph Hamers

08 August 2017 1 min read Listen

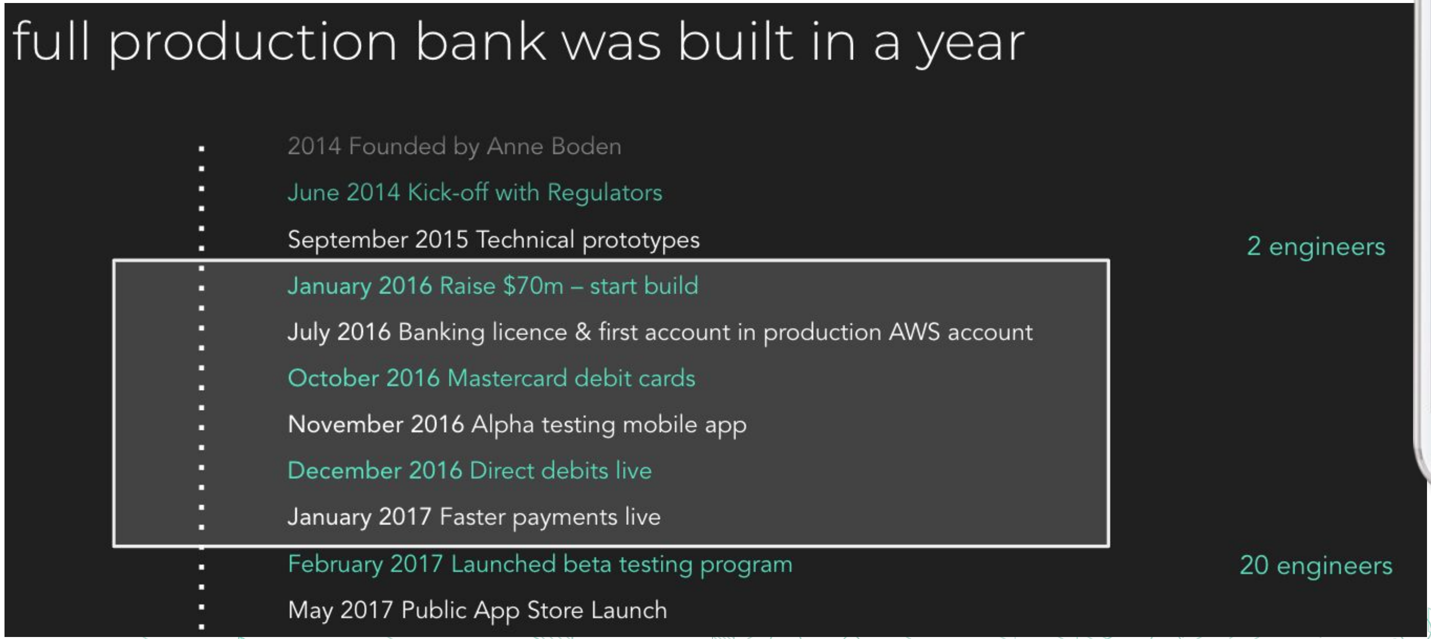


IT Management | Industry Sectors | Technology Topics | Search Computer Weekly

Dutch bank ING to spend millions 'disrupting' its own business

Dutch bank ING used digital technology to reinvent itself following the financial crash, and is about to reinvent itself again, says its global CTO, Brendan Donovan

STARLING BANK



full production bank was built in a year

2014 Founded by Anne Boden

June 2014 Kick-off with Regulators

September 2015 Technical prototypes — 2 engineers

January 2016 Raise $70m – start build

July 2016 Banking licence & first account in production AWS account

October 2016 Mastercard debit cards

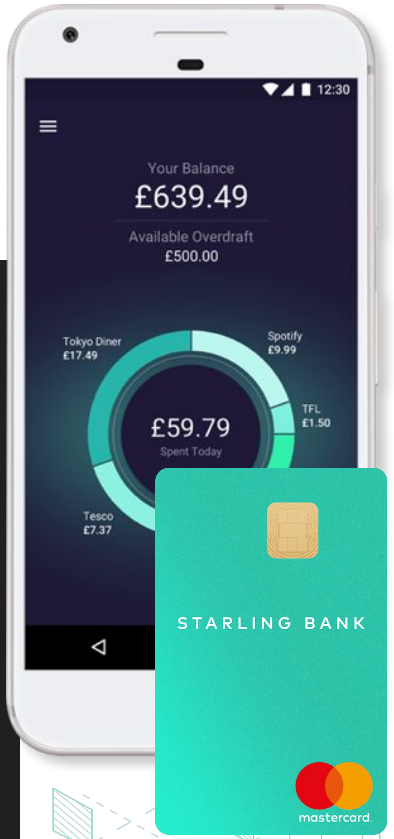
November 2016 Alpha testing mobile app

December 2016 Direct debits live

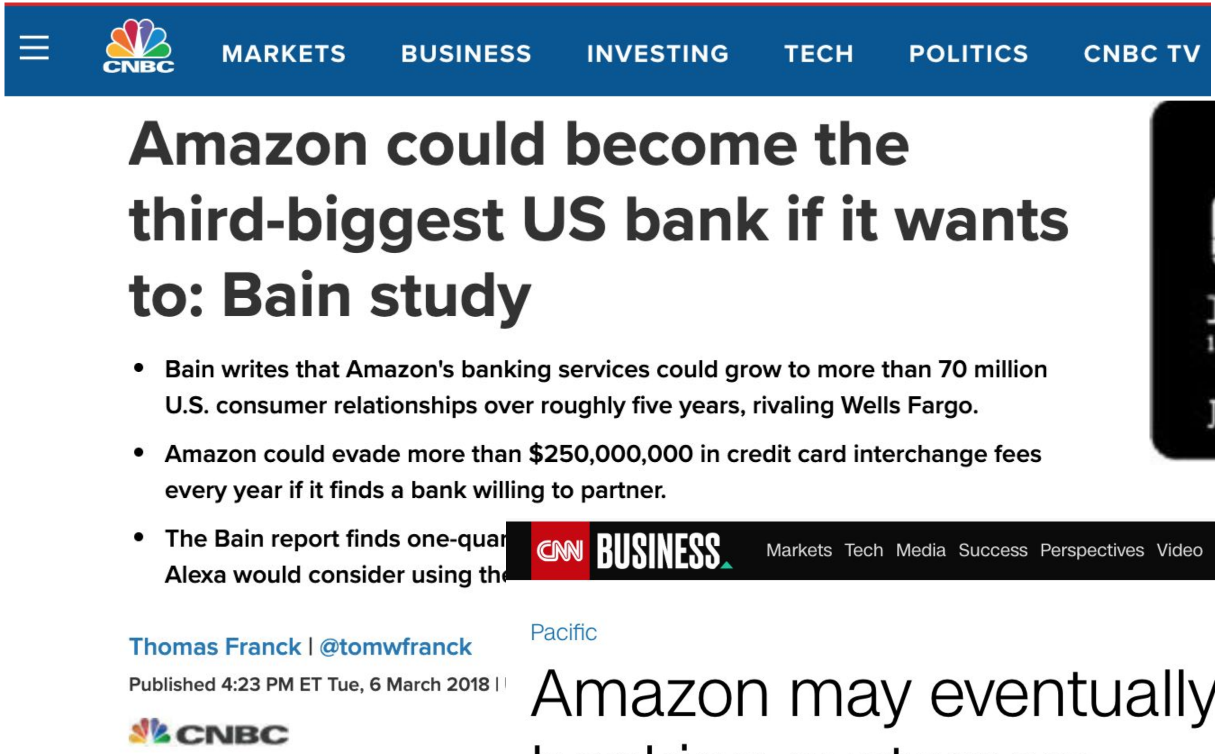
January 2017 Faster payments live

February 2017 Launched beta testing program — 20 engineers

May 2017 Public App Store Launch



Greg Hawkins, Starling Bank

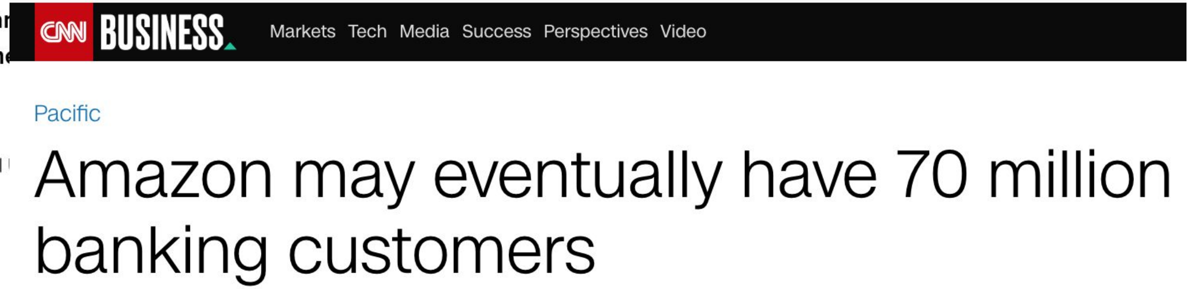


Amazon could become the third-biggest US bank if it wants to: Bain study

- Bain writes that Amazon's banking services could grow to more than 70 million U.S. consumer relationships over roughly five years, rivaling Wells Fargo.
- Amazon could evade more than $250,000,000 in credit card interchange fees every year if it finds a bank willing to partner.
- The Bain report finds one-quar
  Alexa would consider using the

Thomas Franck | @tomwfranck

Published 4:23 PM ET Tue, 6 March 2018 |

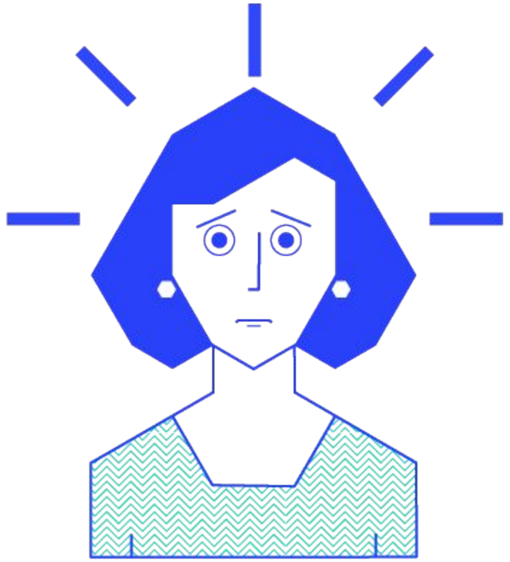


Pacific

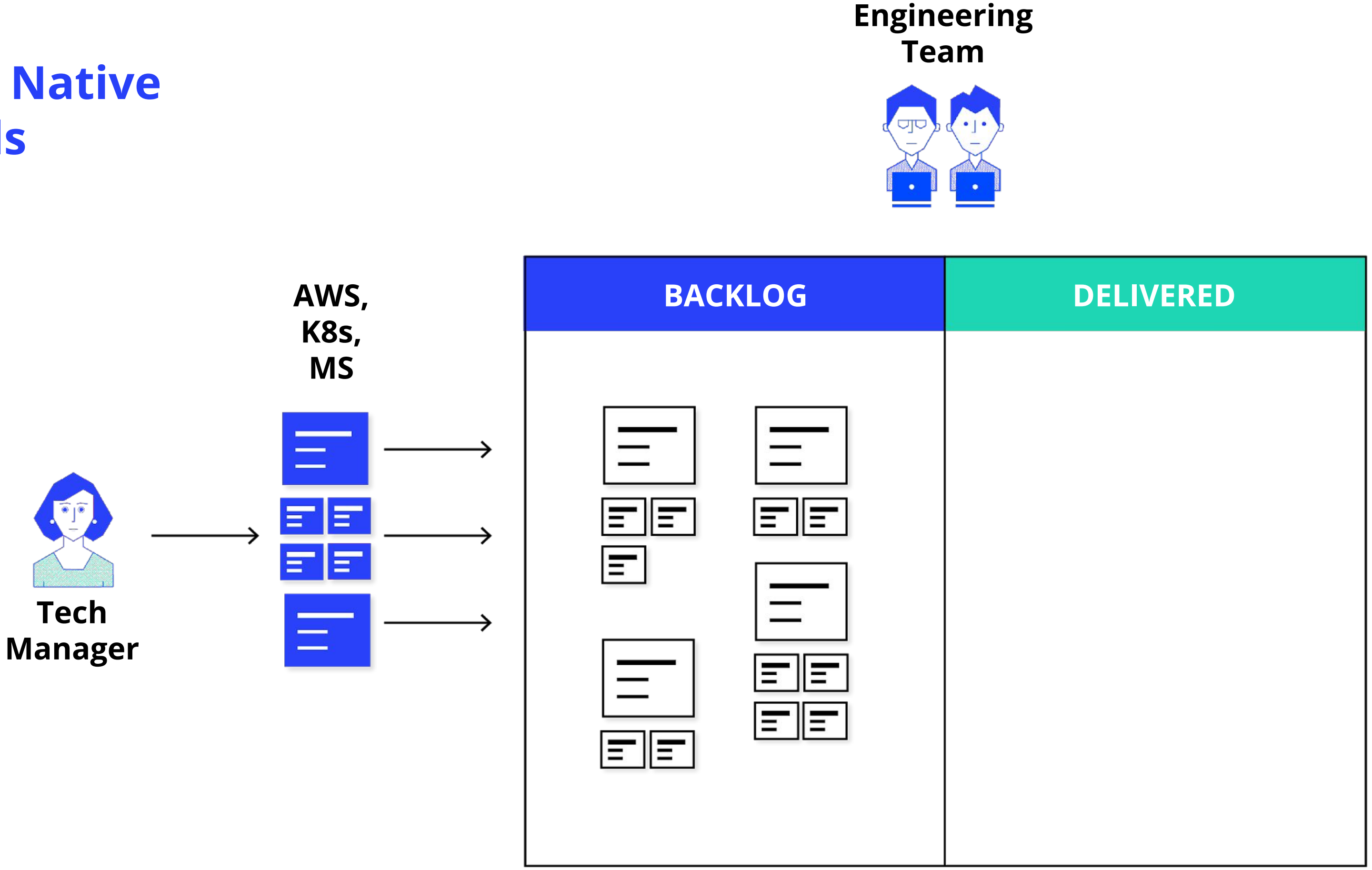
Amazon may eventually have 70 million banking customers

We Must DO Something!

Jenny’s wakeup call

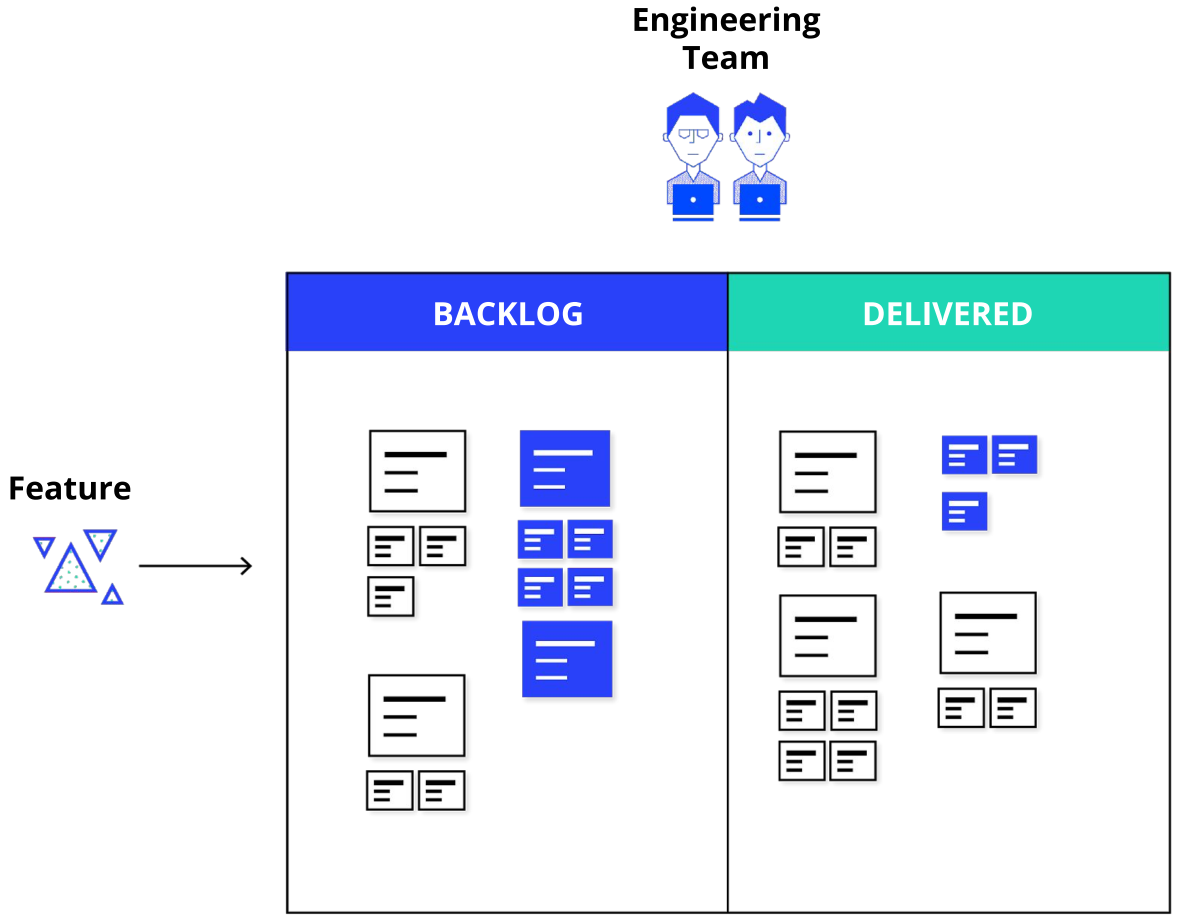


Use Cloud Native Tools



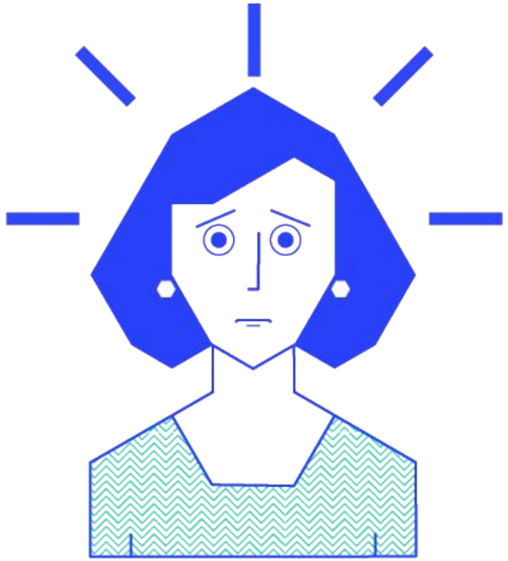
6-12 month later...

Only old stuff + a bit of CN have been delivered

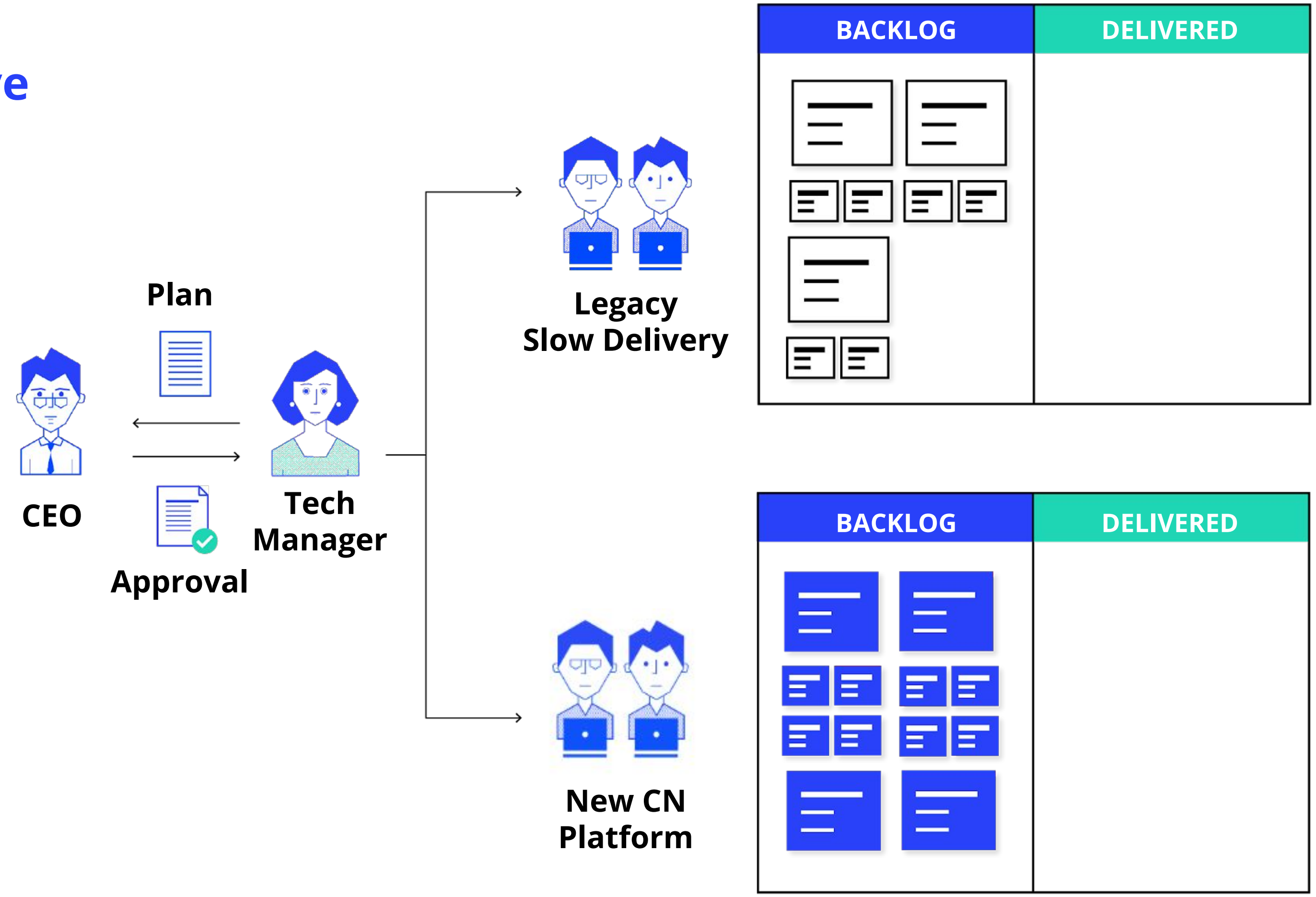


We Must DO Something ELSE!

Jenny’s second wakeup call

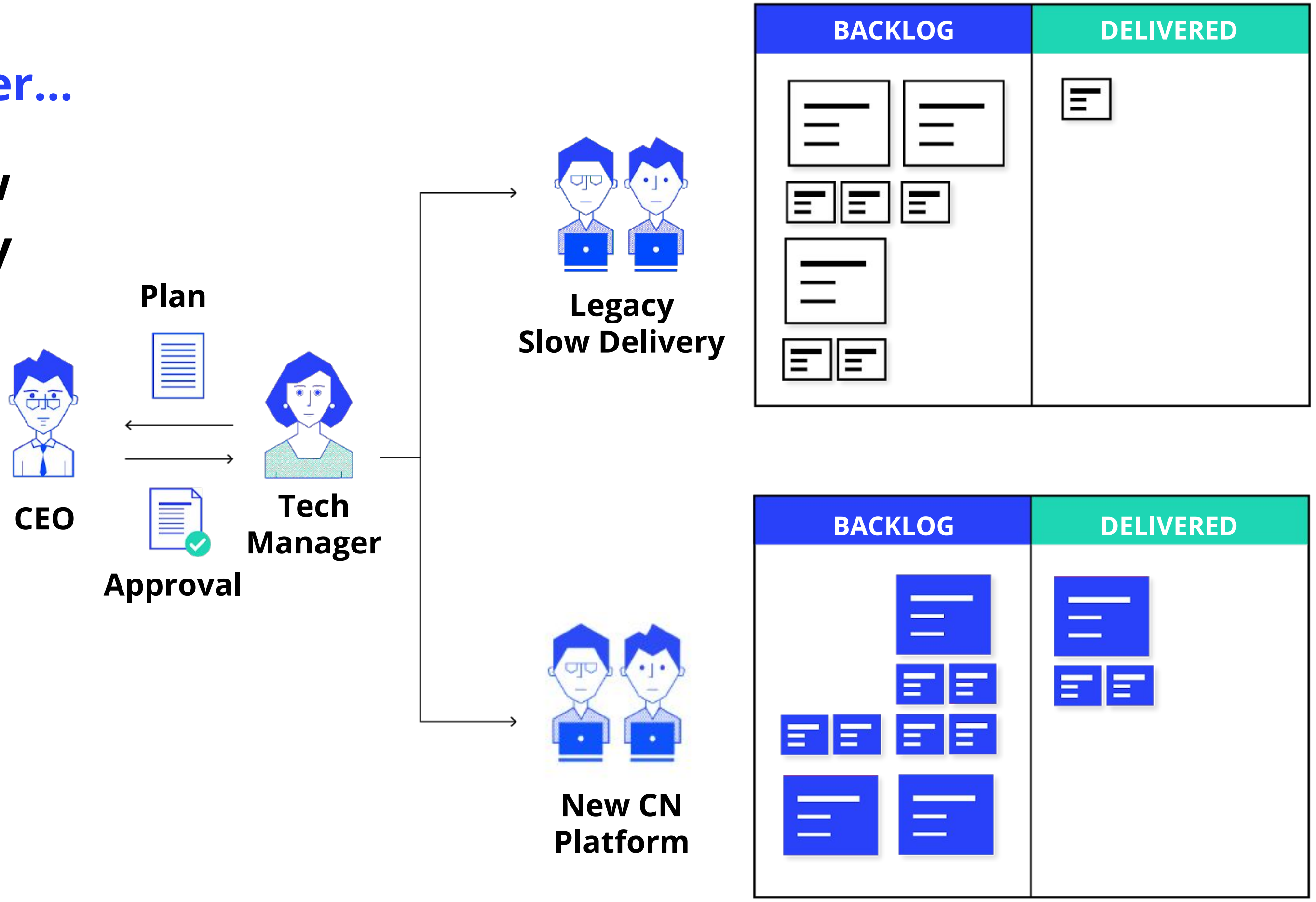


Cloud Native Rewrite



6-12 month later...

Almost no new features + only 30% on CN have been delivered



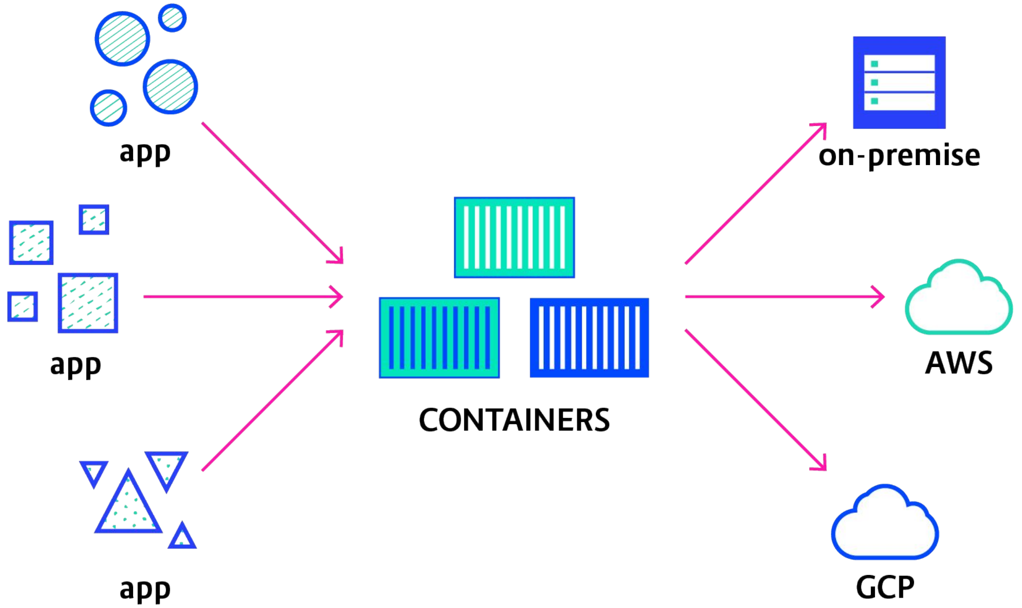
Why is it so difficult?

They’ve Never Done Cloud Native Before

Stage	NO PROCESS	WATERFALL	AGILE	CLOUD NATIVE	NEXT
ARCHITECTURE	Emerging from trial and error	Tightly coupled monolith	Client server	Microservices	Functions
PROVISIONING	Manual	Scripted	Config. management (Puppet/Chef/Ansible)	Orchestration (Kubernetes)	Serverless
INFRASTRUCTURE	Single server	Multiple servers	VMs (pets)	Containers/ hybrid cloud (cattle)	Edge computing

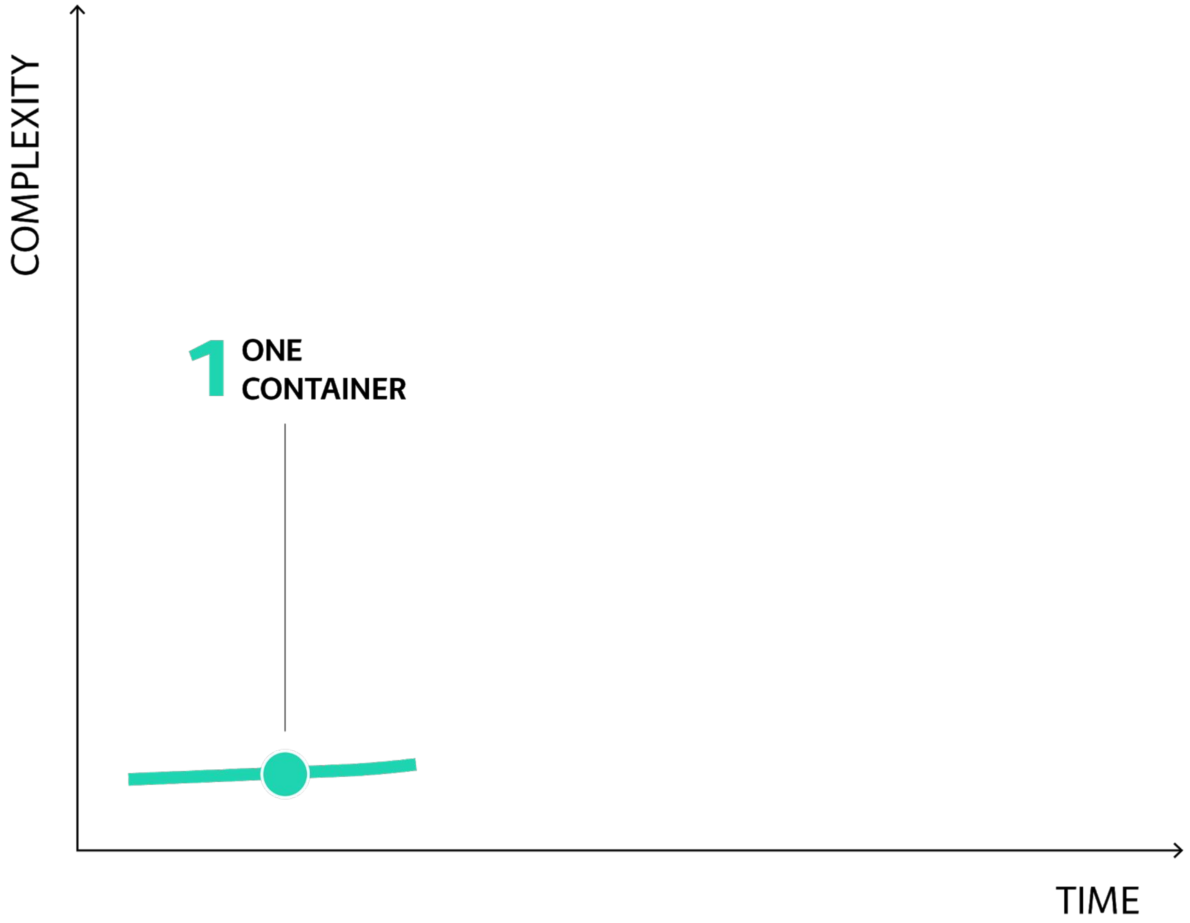
Cloud Native

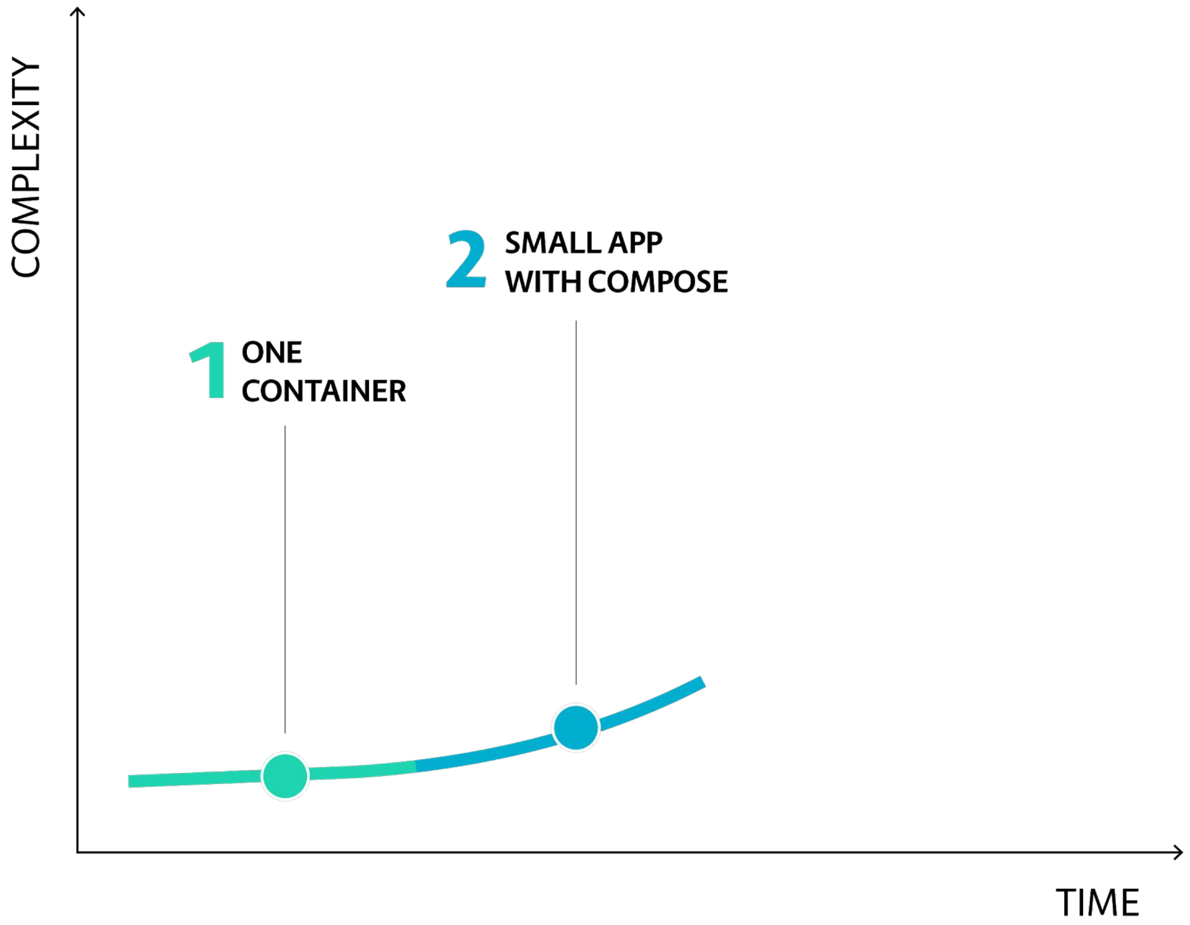
Public Cloud,
Microservices,
Containers (Docker),
Dynamic Scheduling
(Kubernetes),
etc.

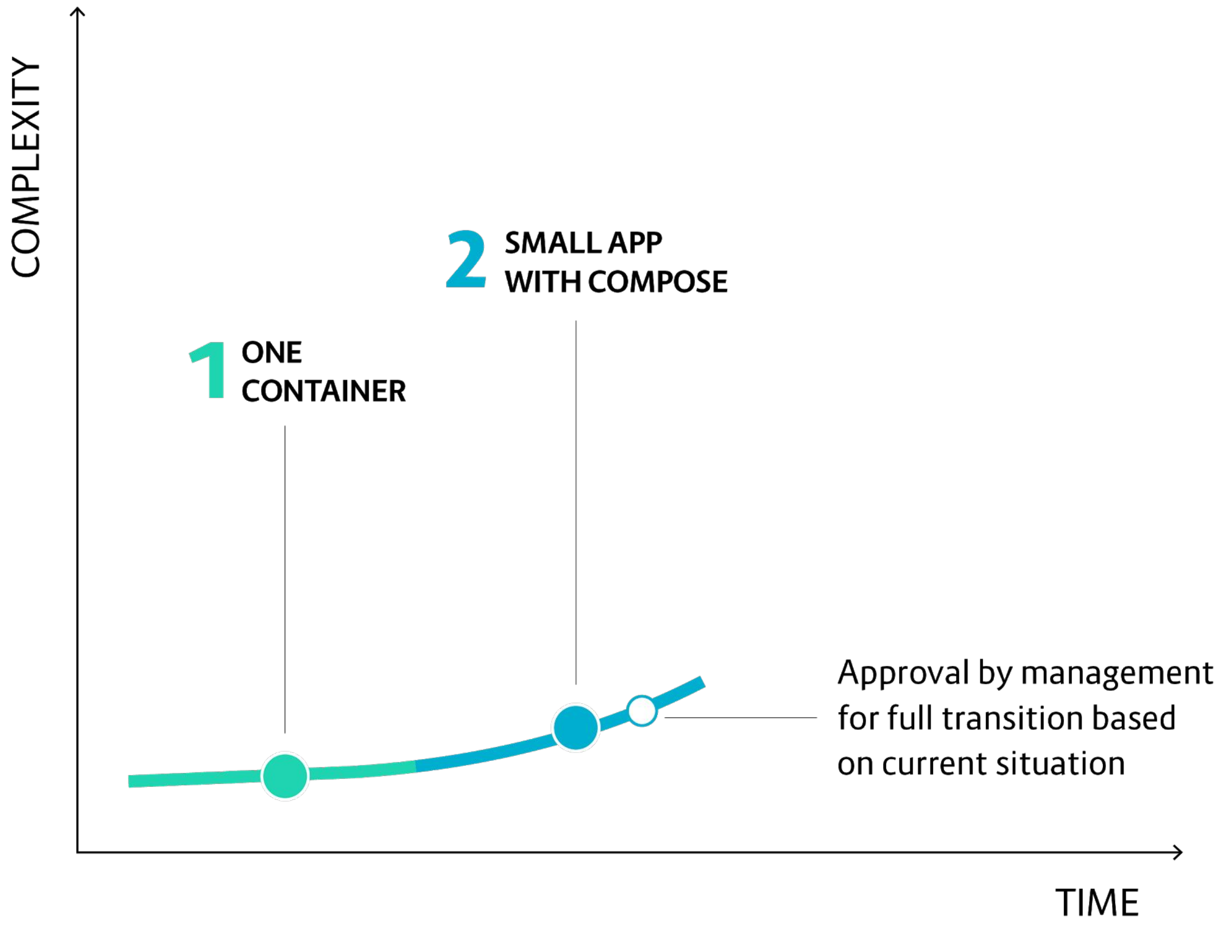


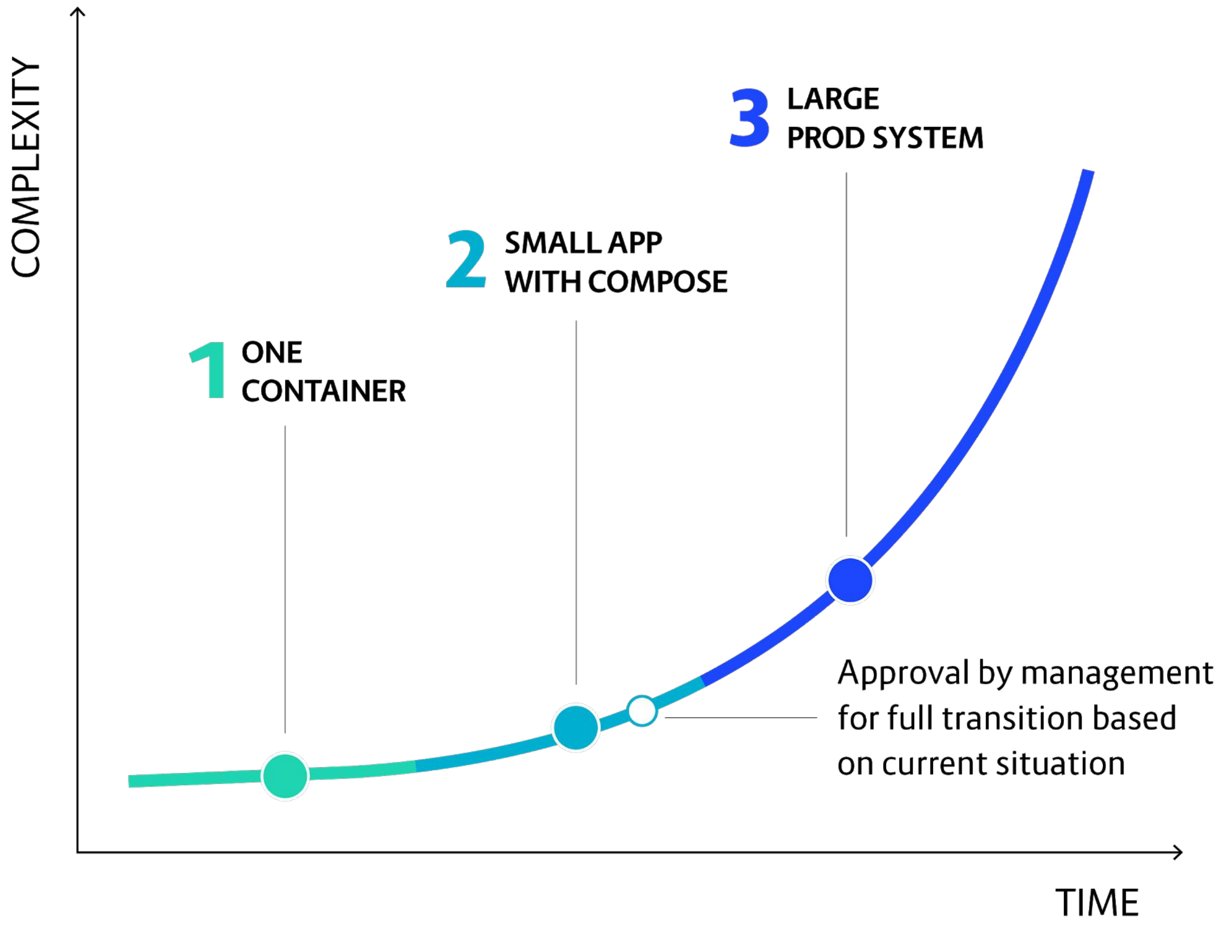
Maturity Matrix

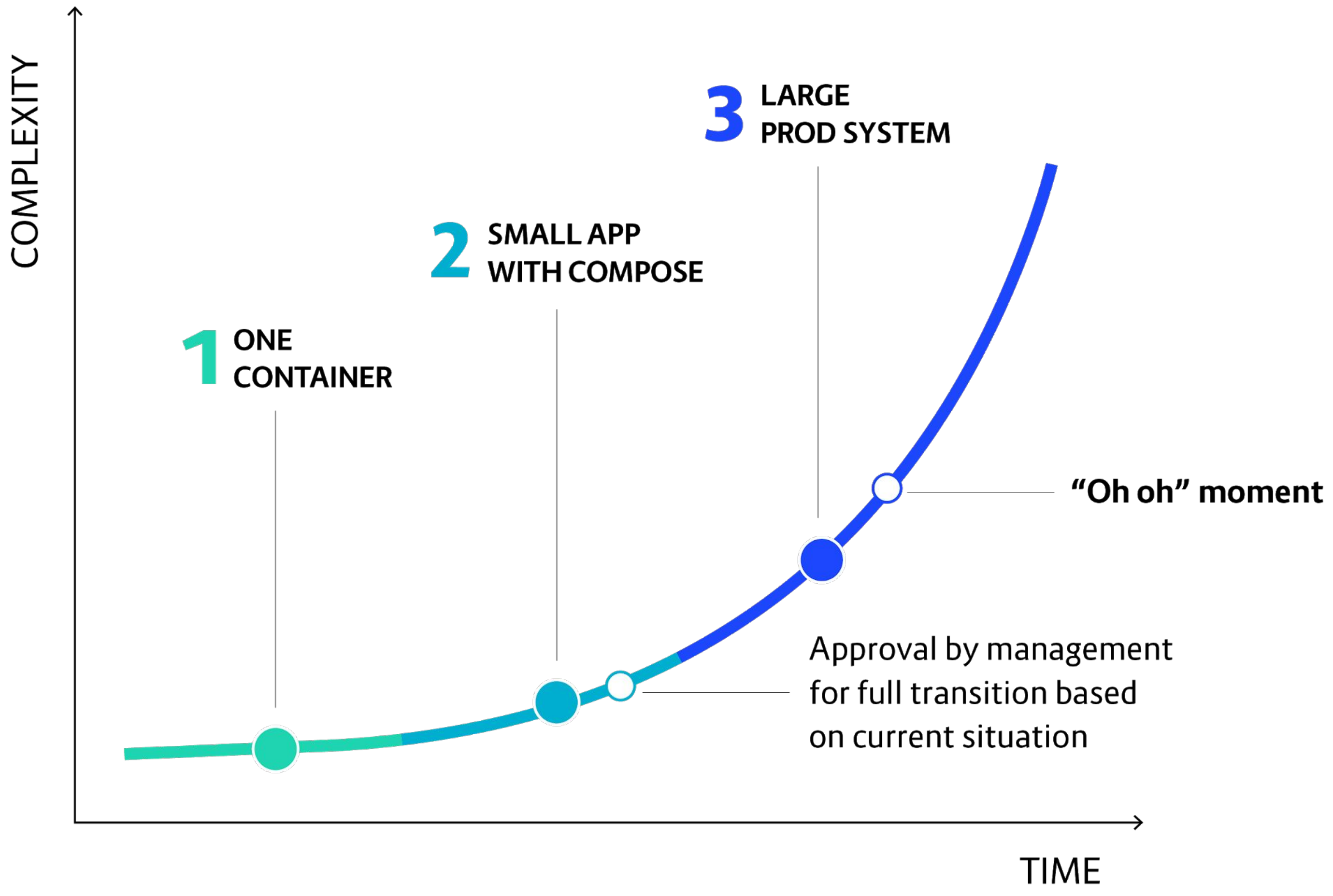
Stage	NO PROCESS	WATERFALL	AGILE	CLOUD NATIVE	NEXT
CULTURE	Individualist	Predictive	Iterative	Collaborative	Experimental
PROD/SERVICE DESIGN	Arbitrary	Long-term plan	Feature driven	Data driven	All driven
TEAM	No organization, single contributor	Hierarchy	Cross-functional teams	DevOps / SRE	Internal supply chains
PROCESS	Random	Waterfall	Agile (Scrum/Kanban)	Design Thinking + Agile + Lean	Distributed, self-organized
ARCHITECTURE	Emerging from trial and error	Tightly coupled monolith	Client server	Microservices	Functions
MAINTENANCE	Respond to users complaints	Ad-hoc monitoring	Alerting	Full observability & self-healing	Preventive ML, AI
DELIVERY	Irregular releases	Periodic releases	Continuous Integration	Continuous Delivery	Continuous Deployment
PROVISIONING	Manual	Scripted	Config. management (Puppet/Chef/Ansible)	Orchestration (Kubernetes)	Serverless
INFRASTRUCTURE	Single server	Multiple servers	VMs (pets)	Containers/ hybrid cloud (cattle)	Edge computing


COMPLEXITY
1 ONE CONTAINER
TIME

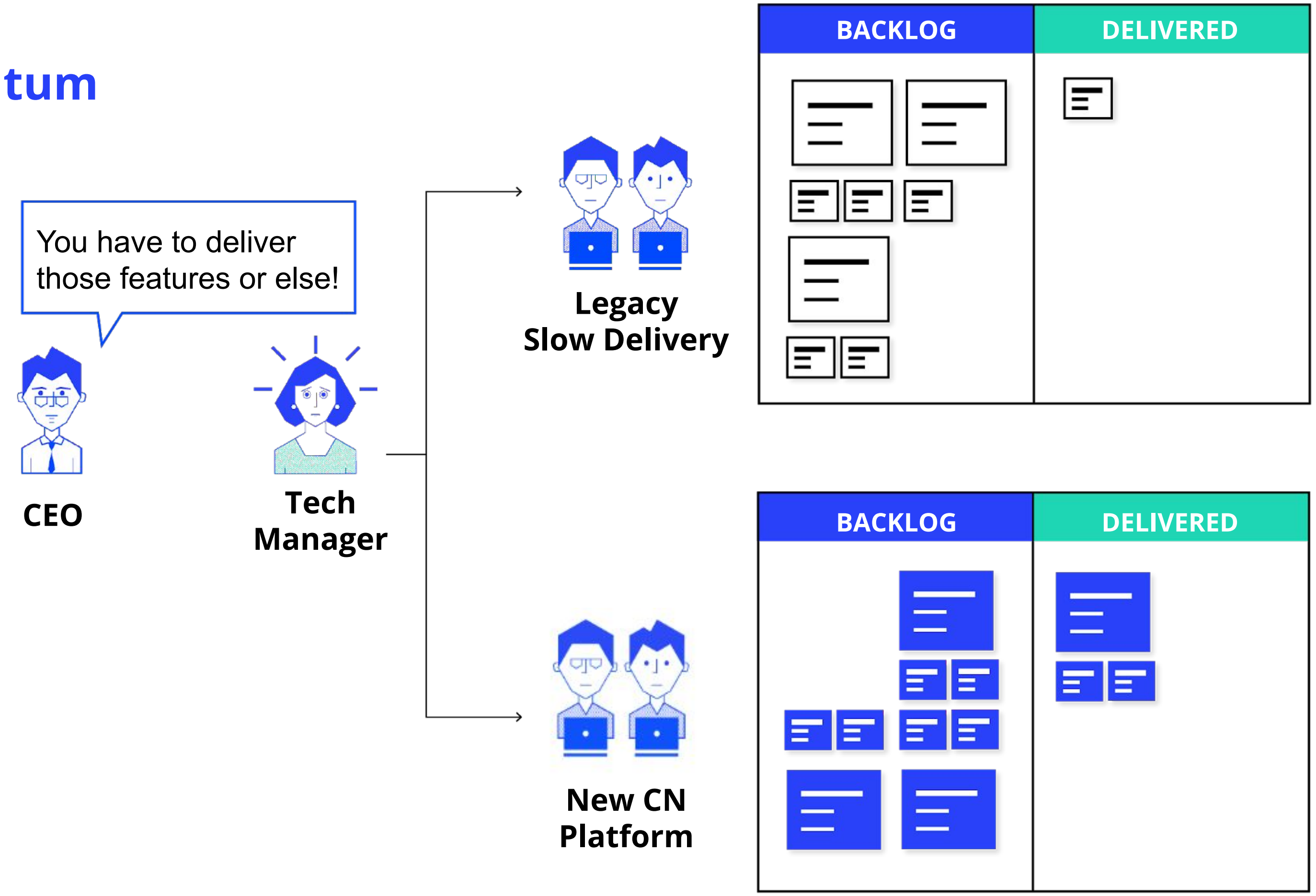

COMPLEXITY
1 ONE CONTAINER
2 SMALL APP WITH COMPOSE
TIME


COMPLEXITY
1 ONE CONTAINER
2 SMALL APP WITH COMPOSE
Approval by management for full transition based on current situation
TIME


COMPLEXITY
1 ONE CONTAINER
2 SMALL APP WITH COMPOSE
3 LARGE PROD SYSTEM
Approval by management for full transition based on current situation
TIME

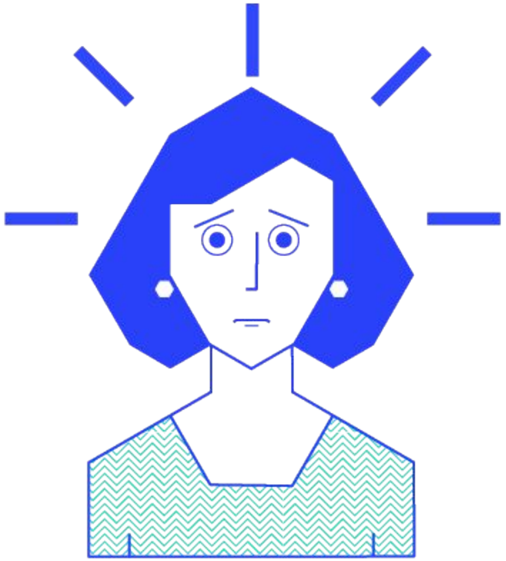

COMPLEXITY
TIME
1 ONE CONTAINER
2 SMALL APP WITH COMPOSE
3 LARGE PROD SYSTEM
"Oh oh" moment
Approval by management for full transition based on current situation

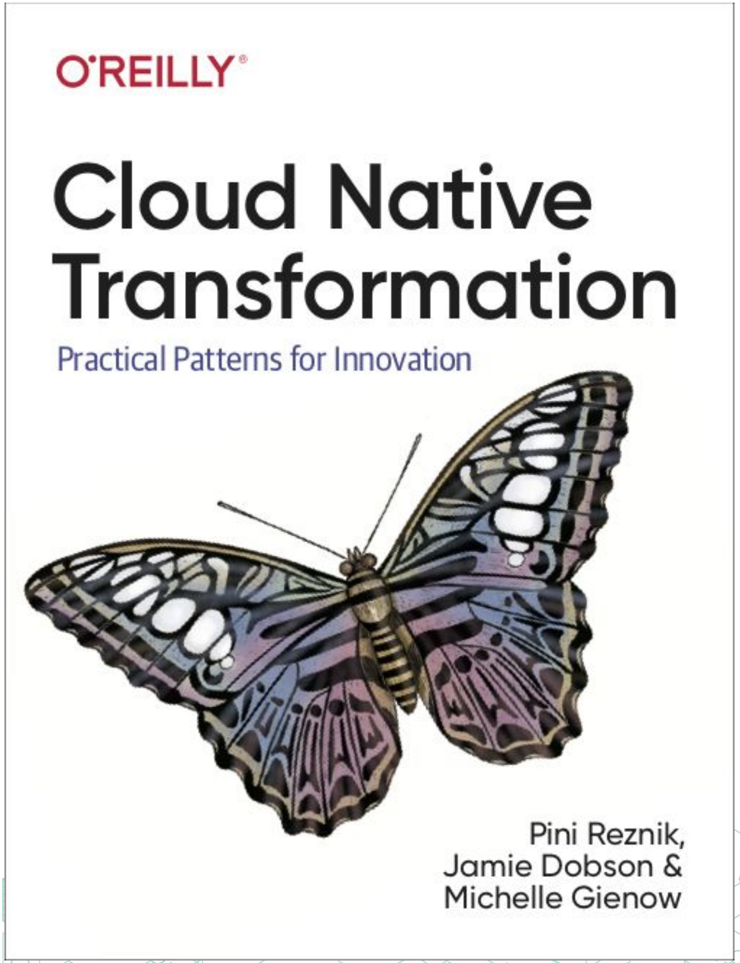
The Ultimatum



We Must DO Something Else AGAIN!

Jenny’s third wakeup call





O'REILLY

Cloud Native Transformation

Practical Patterns for Innovation

Pini Reznik, Jamie Dobson & Michelle Gienow

What is a Pattern Language?

A Collection of Design Decisions

Patterns, Languages and Designs

Pattern is a Word:

Table
Chair
Sofa
…

Patterns, Languages and Designs

Pattern is a Word:

Table
Chair
Sofa
…

Patterns, Languages and Designs

Pattern is a Word:

Table
Chair
Sofa
...

Languages consist of Words:

Furniture language

Patterns, Languages and Designs

Pattern is a Word:

Table
Chair
Sofa
...

Languages consist of Words:

Furniture language

Designs are Stories:

There is a square table with 4 chairs and a sofa in a room.

What is a Cloud Native Transformation Pattern Language?

Our **cloud native pattern language** is a collection of **design decisions** about cloud native practices and technologies and the **context** in which they work.

Example Patterns

The Business Case and Microservices

Structure

Definition
In This Context:
Therefore:
Consequently:
Related Patterns:

Definition - Business Case

When an organisation’s leadership does not fully comprehend the advantages that result from a Cloud Native migration, providing a strong Business Case will allow them to understand and support the project without hesitation.

A company is experiencing pressure from external advisors or internal tech teams to move to Cloud Native. The executive team is contemplating making the move to CN, but this is the first such transformation the company has undertaken and there is only a partial understanding of the complexity of a CN migration and the benefits that will come from it.

In This Context:

The benefits of the transformation are not clear to the executive team, so they may not support the initiative or even give it serious consideration.

- The traditional model is for organisations to be massively risk averse, to minimise uncertainty at all costs.
- Change-averse culture avoids new technologies or experimental approaches.
- Cloud Native architectures are conceptually different from traditional approaches, merging careful up-front planning with flexible and mutable,experimentation-based implementation.
- Tech teams are eager to get started with the transformation, even before business case is established

Therefore:

Create a formal business case to help educate the organisation's executive team, taking into account the benefits to be gained from Cloud Native. The business case needs to include key CN advantages, including acceleration of business velocity, scalability, potential cost savings, and enhanced recruitment and retention of tech staff.

Consequently:

The business case for a CN transformation is clear and the company's decision makers have a clear understanding of the advantages CN confers and are ready to move forward. They are prepared to allocate the necessary budget and resources that such a large project will require.

Definition - Microservices Architecture

To reduce the costs of coordination between teams delivering large monolithic applications, build the software as a series of microservices that are built, deployed and operated independently.

A company has decided to move to Cloud Native and is looking at the ways to increase the velocity of feature development and to optimise their utilization of cloud resources. The size of the development/engineering staff can range from a few tens, for a small to medium business, up to a few thousand for a large enterprise.

In This Context:

Delivery of large monolithic applications developed by large teams require long and complex coordination and extensive testing, leading to longer TTM (Time to Market). Hardware utilisation by such applications is inefficient, which leads to waste of resources.

- People tend to delay painful moments; since integration and delivery are typically painful, their frequency tends to decrease as system longevity increases.
- Larger monolithic systems are increasingly more difficult to understand as they grow in size and complexity
- Monoliths are easier to work with than modular applications so long as they are small enough to be understood by each developer.
- Conway's law: architecture tends to resemble the organisational structure.

Therefore:

Split applications into smaller microservices that can be built, tested, deployed and run independently from other components.

- Independent components allow different teams to make progress at their own pace faster-moving teams are not held back by slower ones and to use the most appropriate tools for each situation.
- Independence and freedom of choice are achieved in a tradeoff with reduced standardisation and certain types of reusability.

Consequently:

New systems are created from a large number of small components with a complex web of connections.

Small and independent teams work on separate modules and deliver them with only limited coordination across the teams.

Related Patterns:

Cross-functional teams, CI, CD, Common Services, Libraries & Tools, Communication Through API, Dynamic Scheduling, Monitoring.

Some of the Patterns

Strategy

- Business Case
- Vision First
- Transformation Strategy
- Hierarchy of Values
- Executive Commitment
- Learning Loop
- Reflective Breaks
- Designated Strategist
- Strategic Formulation

Risks

- No regret moves
- Option and hedges
- Big bets
- Reduce cost of experimentation
- Reduce cost of refactoring
- Data driven decision making
- De-risking tech projects
- Central security policies
- Exit strategy vs. vendor locking;
- Open Source
- Highly secure systems

Growth and Safety

- Psychological Safety
- Honest Feedback
- Internal Evangelism
- Team Building for Level Two relationships
- Blameless Inquiry
- Value Proficient and Creative Teams Equally
- Mentoring
- Coaching
- Empathic Listening
- Whiteboards everywhere
- Ongoing Education
- Learning Organisation
- Knowledge Sharing

Teams

- Transformation Champion
- Cross-Functional Teams
- SRE
- Strangle monolithic Organisation
- Remote teams
- DevOps teams
- Design organisation for failure
- Joint Responsibilities
- Manage for proficiency
- Manage for Creativity
- Co-Located team
- Involve the business
- Platform team
- Core team
- Gradual onboarding
- Communicate through tribes
- Delegate power

- Development Reference architecture
- Reproducible development environments
- Starter pack
- Self service
- Common services, libs and tools
- Demo applications;SUMMARY
- Reducing dependencies;SUMMARY
- Release strategies (canary, blue/green, etc.)

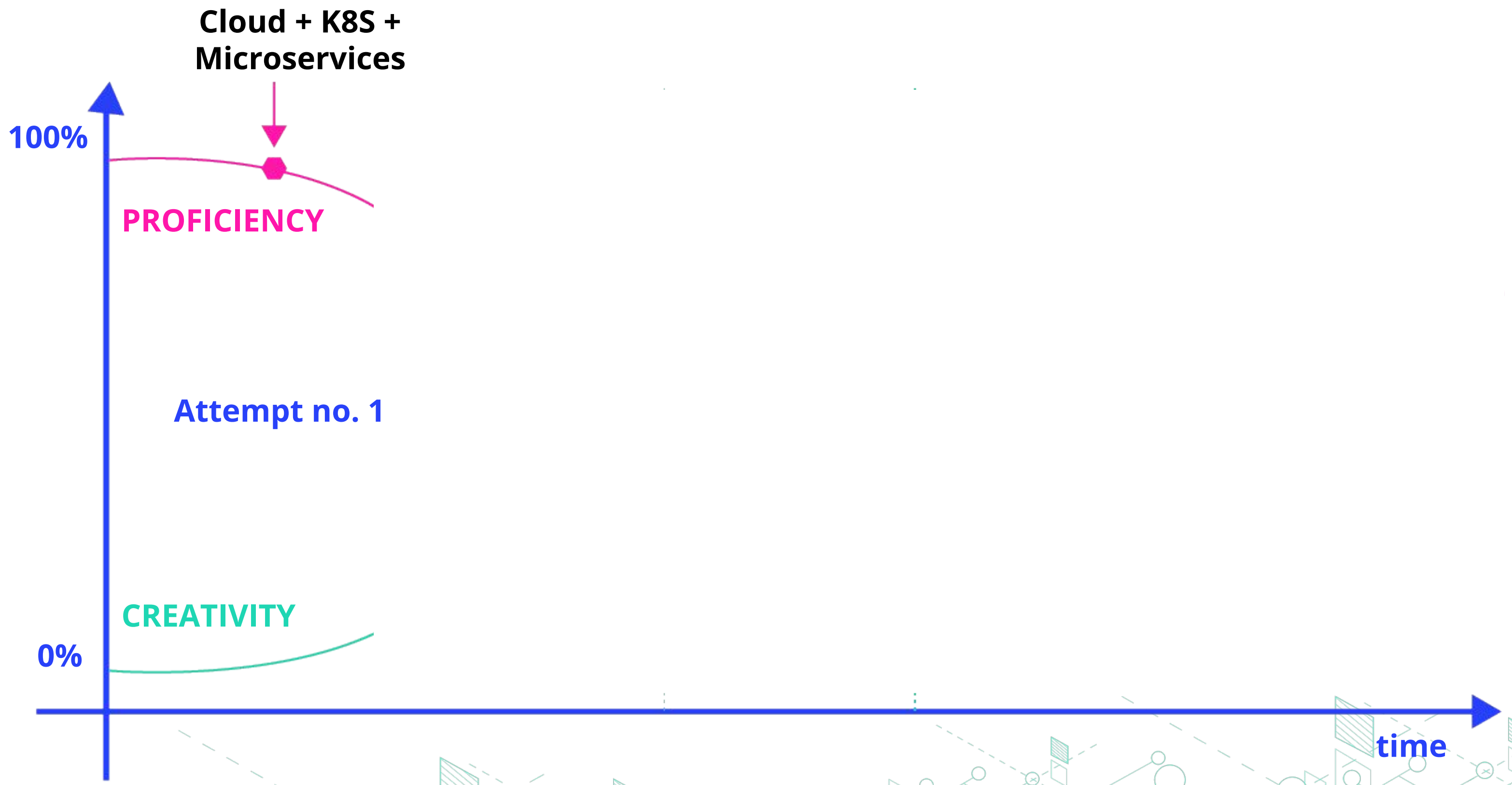
Architecture

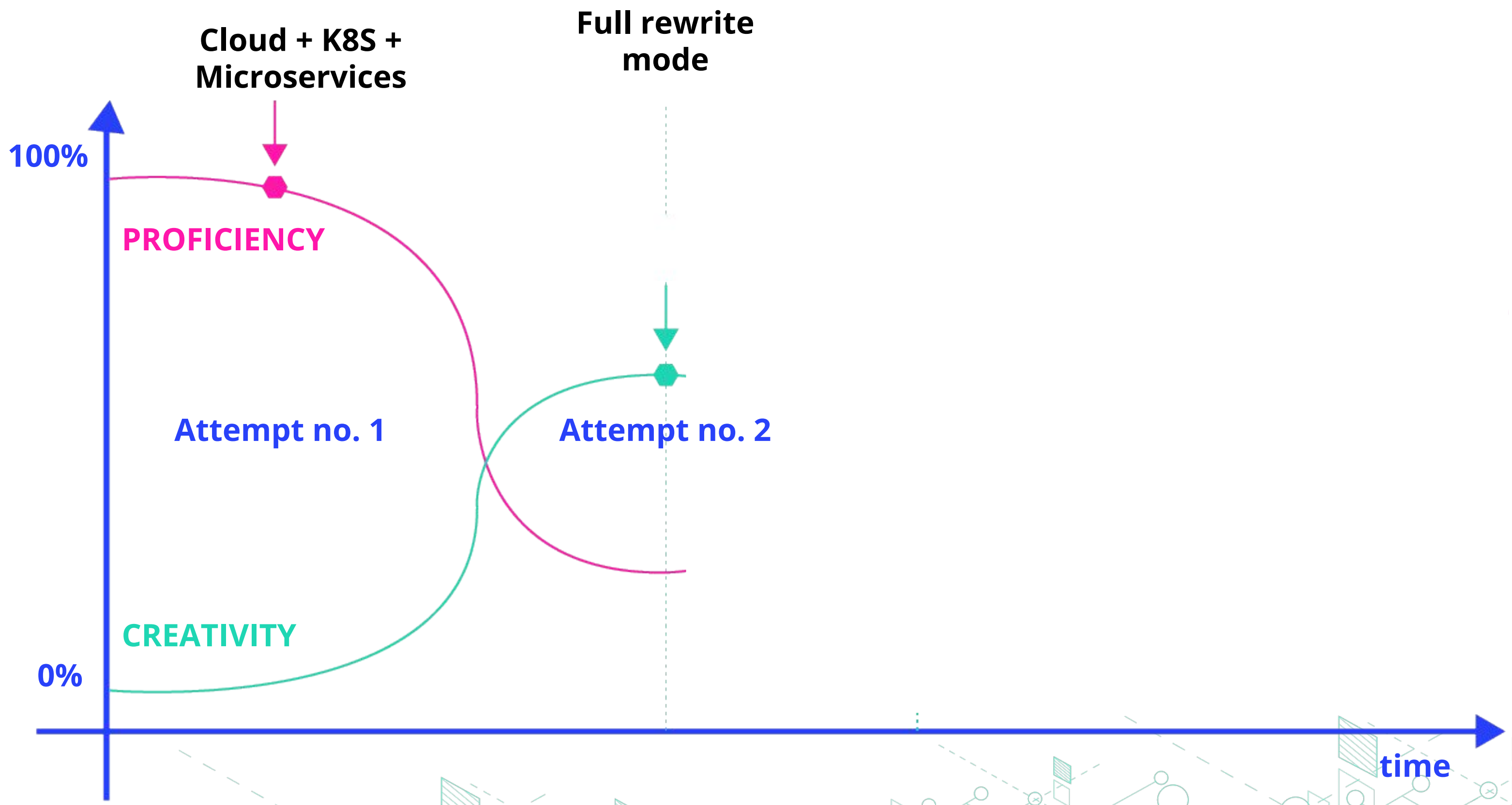
- Distributed System
- Architecture visualisation
- Strangle monoliths
- Avoid reinventing the wheel
- Microservices architecture
- Communication through APIs
- Containerised services
- CI
- Automated testing
- Non blocking long running tests
- CD
- Observability
- Automated infrastructure
- Serverless
- Service mesh
- Dynamic scheduling
- Public Cloud
- Private Cloud

Process

- Room for ongoing improvements
- 3 horizons
- Periodic check-ups
- Focus on bottlenecks
- Design thinking for radical innovation
- Agile for new development
- Lean for optimisation
- Bias for action
- Delayed Automation
- MVP
- Shut down old systems
- Proof of Concept (PoC)
- Exploratory Experiments
- Measure what matters
- Production Readiness
- Lift & Shift At End
- A/B testing
- Value stream mapping
- Shareholders mapping
- Gap analysis;

What Happened So Far?


Cloud + K8S +
Microservices
100%
PROFICIENCY
Attempt no. 1
CREATIVITY
0%
time


Cloud + K8S + Microservices
Full rewrite mode
100%
PROFICIENCY
Attempt no. 1
Attempt no. 2
CREATIVITY
0%
time

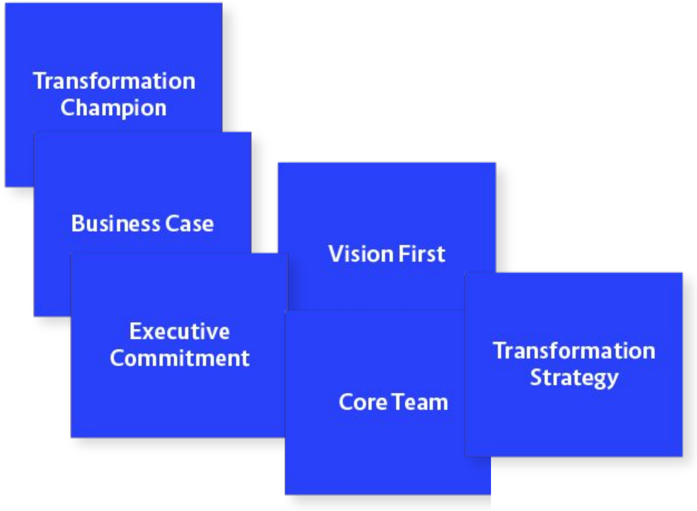
Design the Transformation

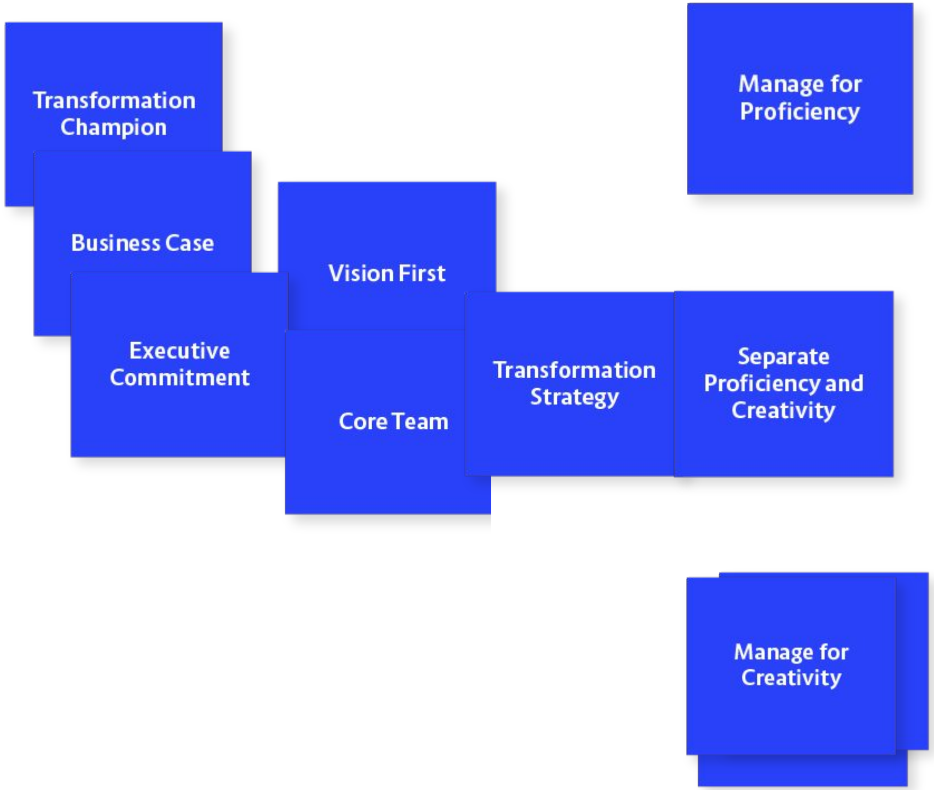
By using Cloud Native Patterns Language

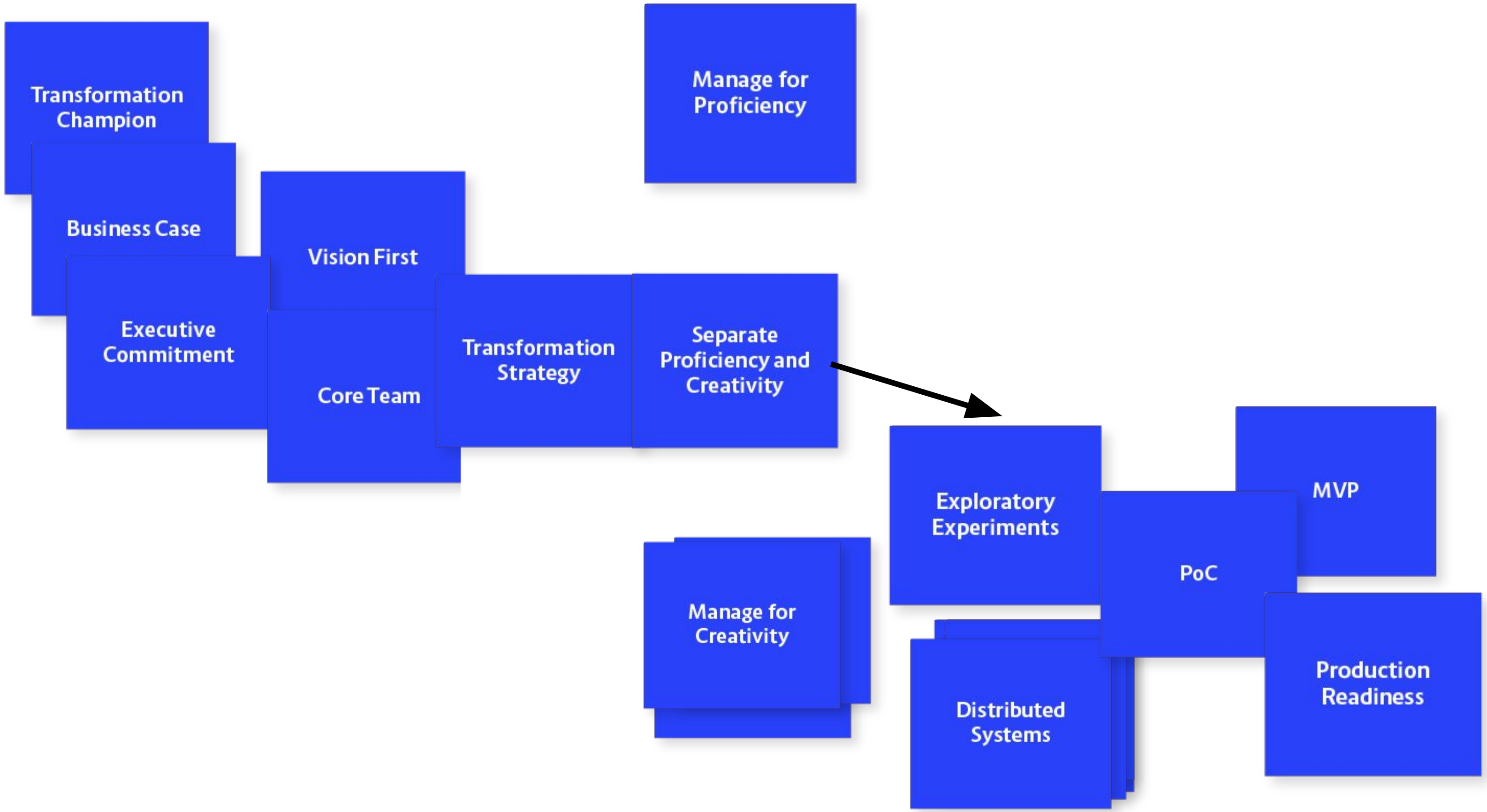
Transformation Champion

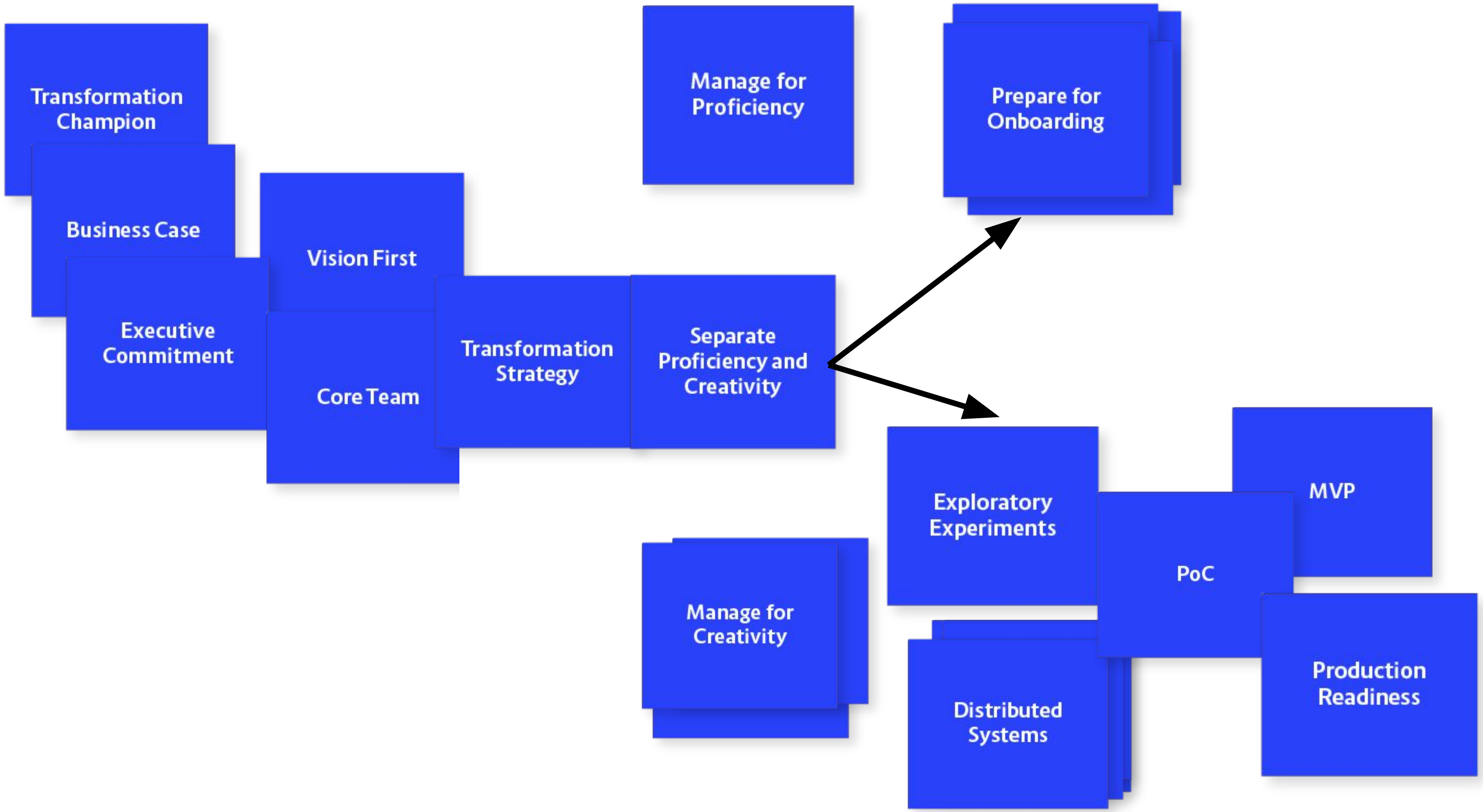
Business Case

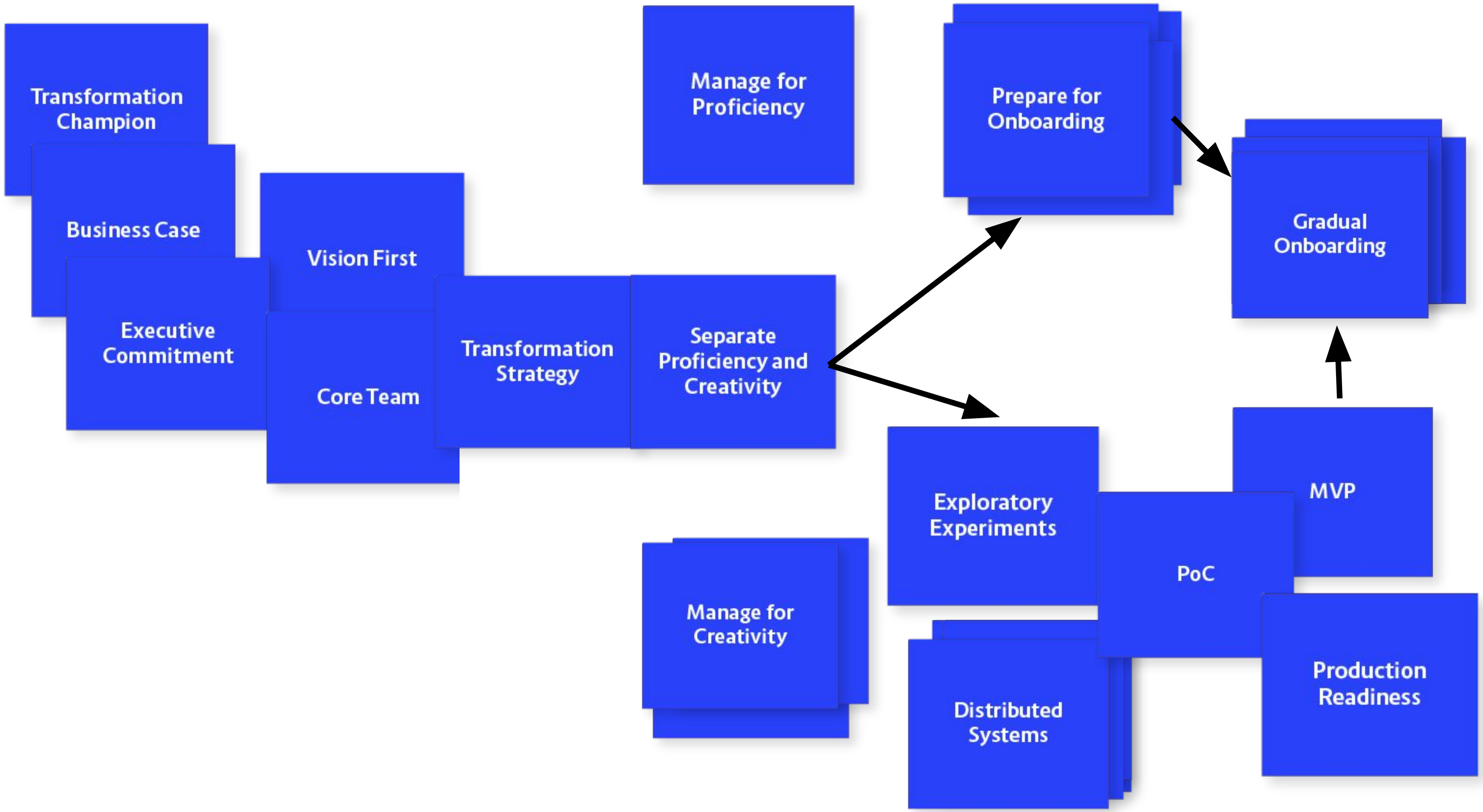
Executive Commitment

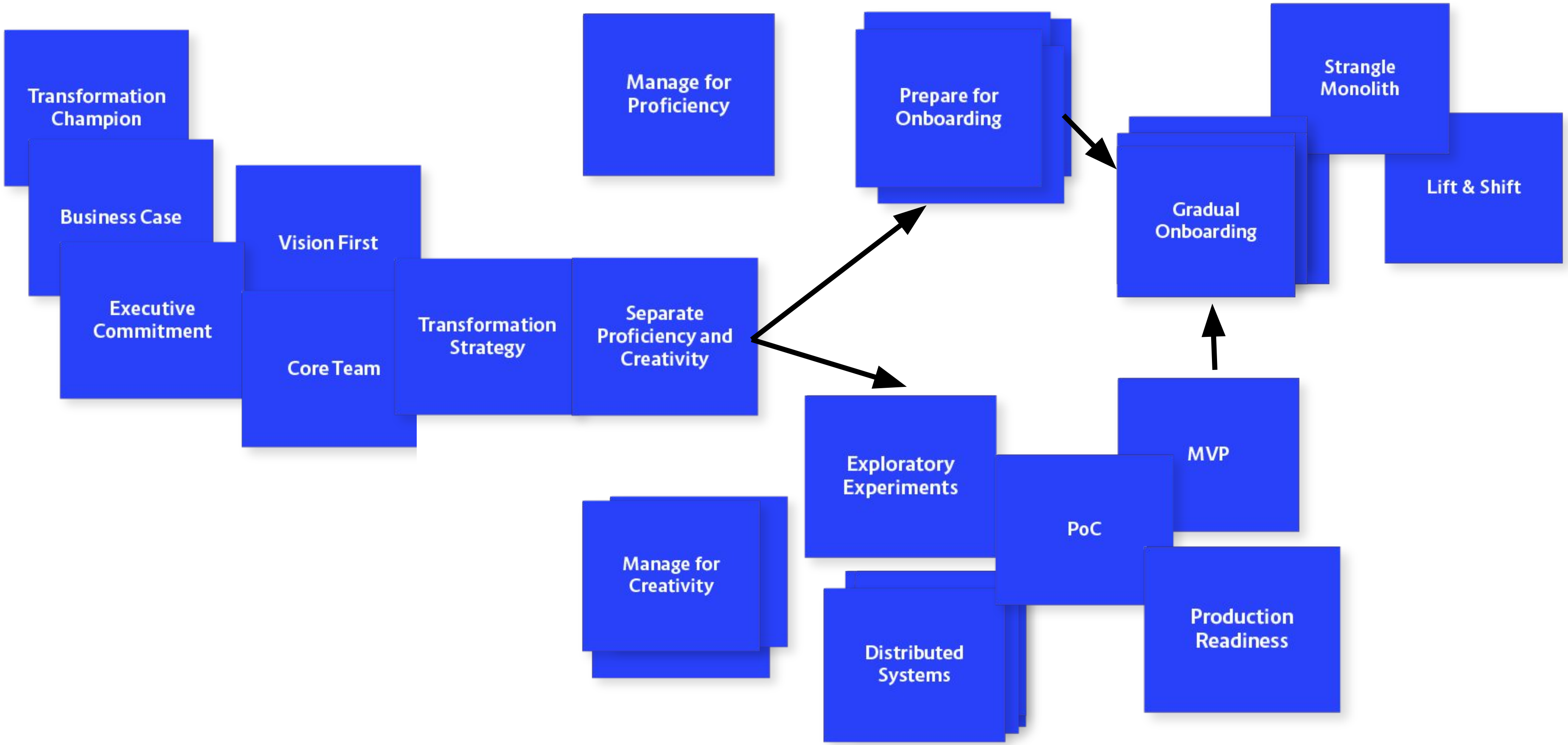

Transformation Champion
Business Case
Vision First
Executive Commitment
Core Team
Transformation Strategy


Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Manage for Proficiency
Separate Proficiency and Creativity
Manage for Creativity

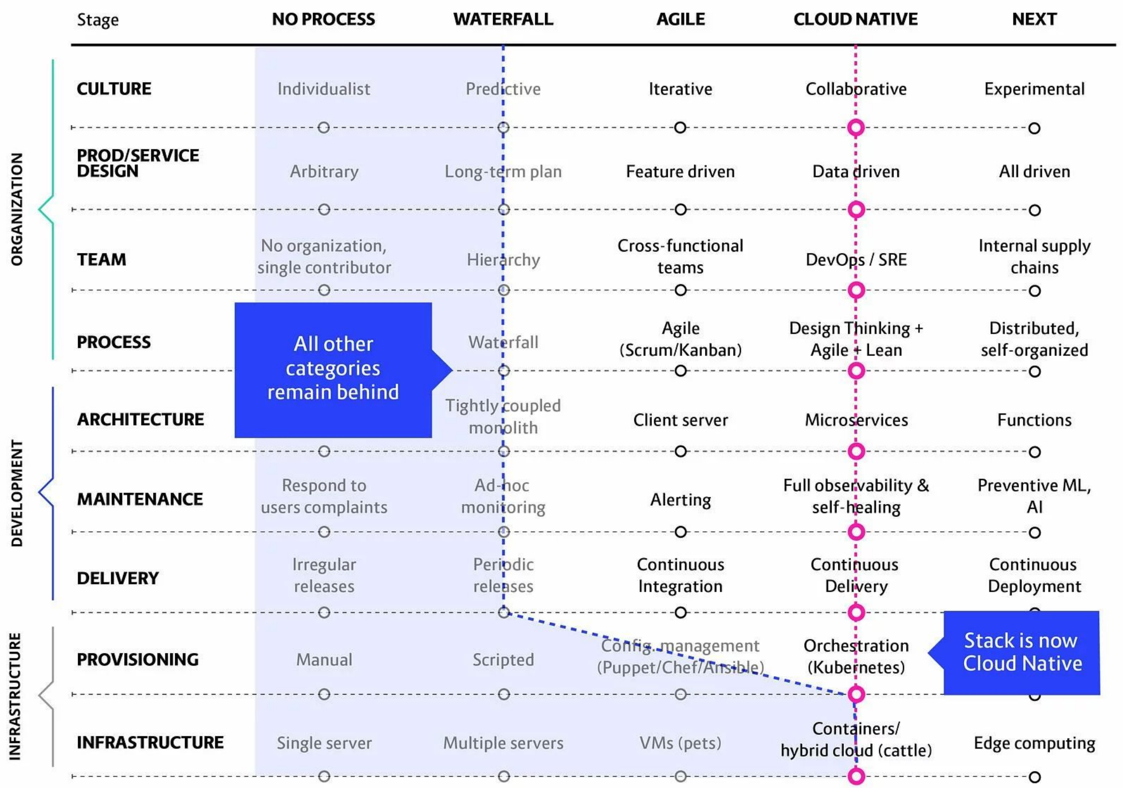

Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Manage for Proficiency
Separate Proficiency and Creativity
Manage for Creativity
Exploratory Experiments
Distributed Systems
PoC
MVP
Production Readiness


Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Separate Proficiency and Creativity
Manage for Proficiency
Prepare for Onboarding
Exploratory Experiments
PoC
MVP
Production Readiness
Manage for Creativity
Distributed Systems

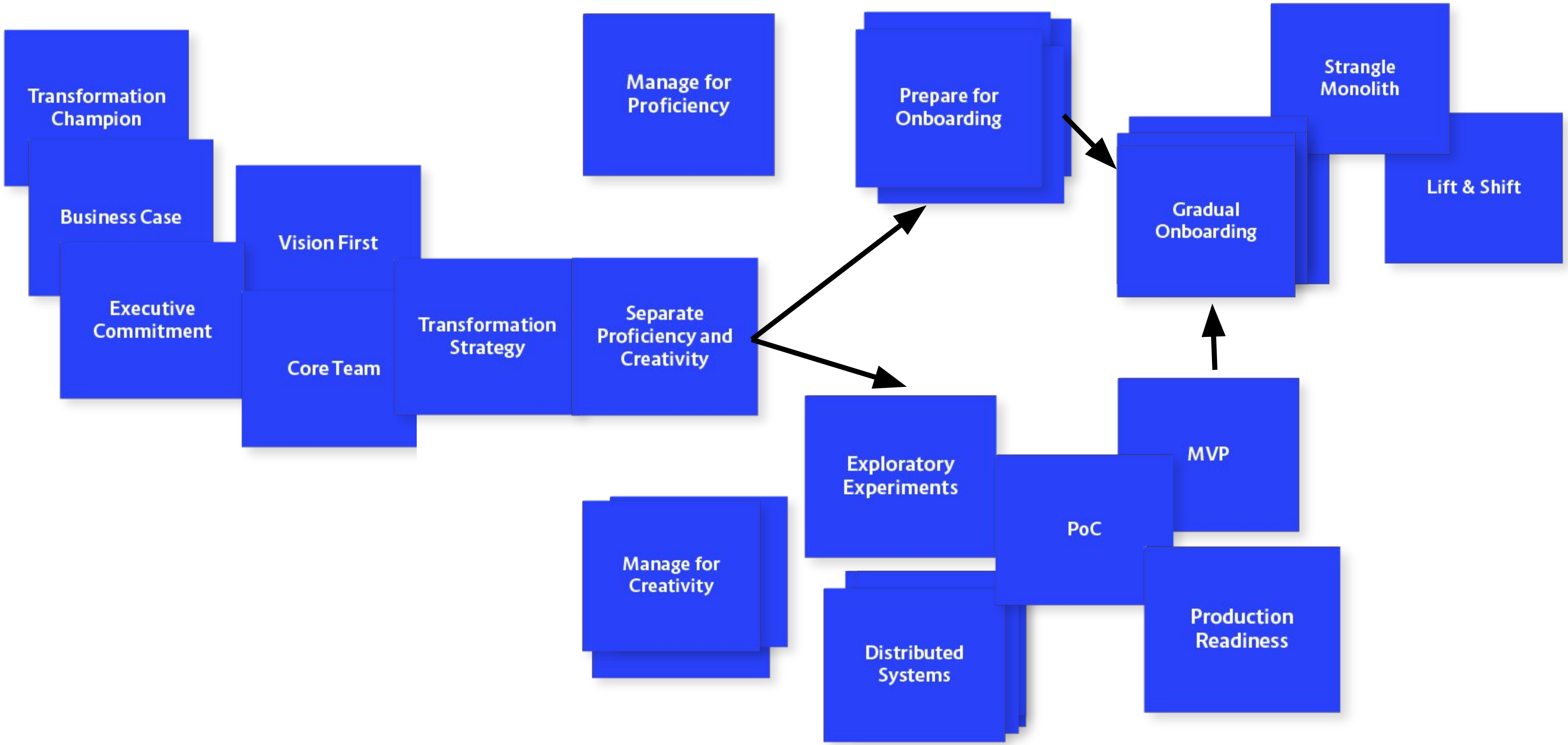

Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Separate Proficiency and Creativity
Manage for Proficiency
Prepare for Onboarding
Gradual Onboarding
Exploratory Experiments
PoC
MVP
Production Readiness
Manage for Creativity
Distributed Systems


Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Separate Proficiency and Creativity
Manage for Proficiency
Manage for Creativity
Prepare for Onboarding
Gradual Onboarding
Strangle Monolith
Lift & Shift
Exploratory Experiments
Distributed Systems
PoC
MVP
Production Readiness

SCENARIO 1: **MOVING TO CLOUD INFRASTRUCTURE - "LIFT & SHIFT"**



	Stage	NO PROCESS	WATERFALL	AGILE	CLOUD NATIVE	NEXT
ORGANIZATION	**CULTURE**	Individualist	Predictive	Iterative	Collaborative	Experimental
	PROD/SERVICE DESIGN	Arbitrary	Long-term plan	Feature driven	Data driven	All driven
	TEAM	No organization, single contributor	Hierarchy	Cross-functional teams	DevOps / SRE	Internal supply chains
	PROCESS		Waterfall	Agile (Scrum/Kanban)	Design Thinking + Agile + Lean	Distributed, self-organized
DEVELOPMENT	**ARCHITECTURE**		Tightly coupled monolith	Client server	Microservices	Functions
	MAINTENANCE	Respond to users complaints	Ad-hoc monitoring	Alerting	Full observability & self-healing	Preventive ML, AI
	DELIVERY	Irregular releases	Periodic releases	Continuous Integration	Continuous Delivery	Continuous Deployment
INFRASTRUCTURE	**PROVISIONING**	Manual	Scripted	Config management (Puppet/Chef/Ansible)	Orchestration (Kubernetes)	
	INFRASTRUCTURE	Single server	Multiple servers	VMs (pets)	Containers/ hybrid cloud (cattle)	Edge computing


Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Separate Proficiency and Creativity
Manage for Proficiency
Manage for Creativity
Prepare for Onboarding
Gradual Onboarding
Strangle Monolith
Lift & Shift
Exploratory Experiments
Distributed Systems
PoC
MVP
Production Readiness

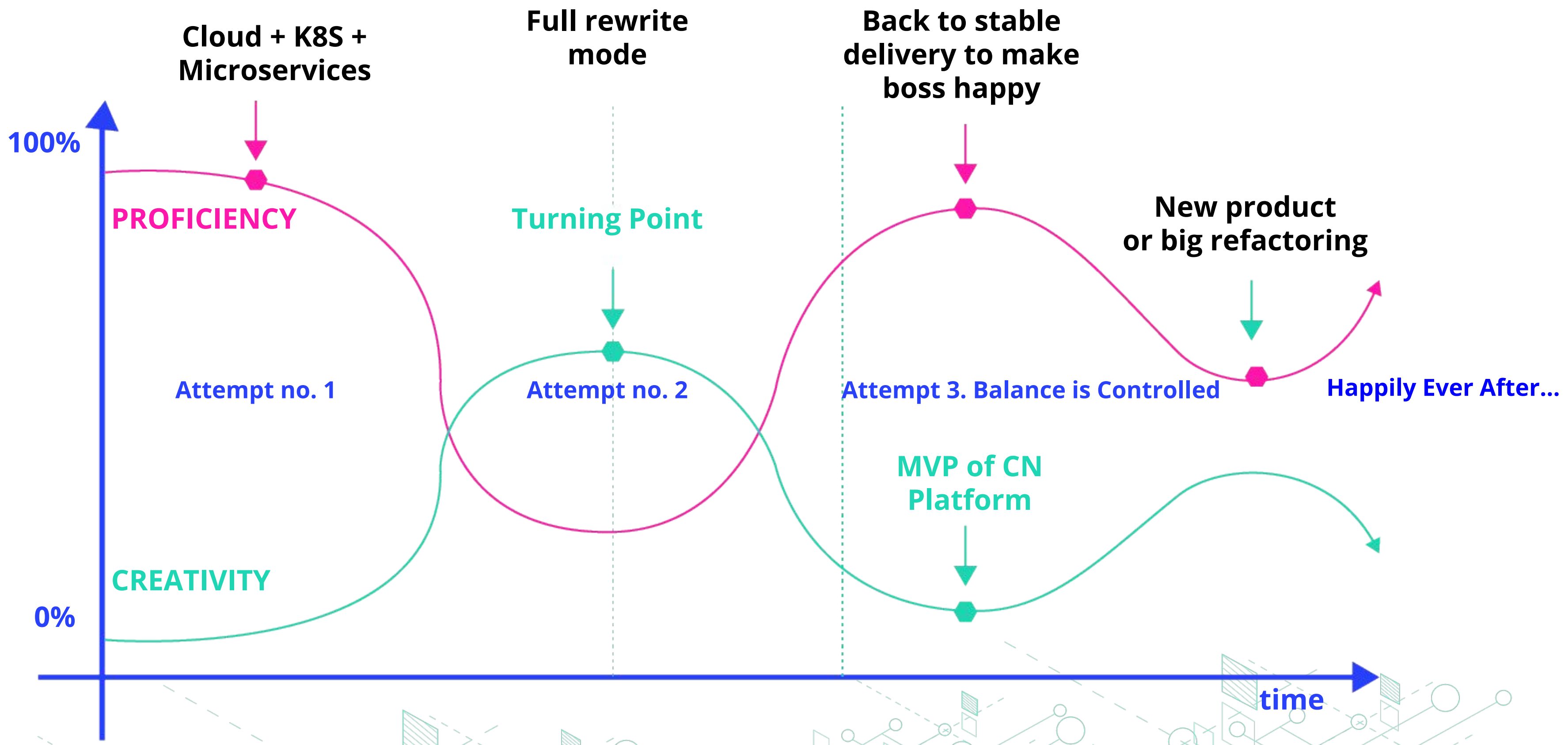
Quick, schedule a meeting – your Business Cards are here!
MOO
NO REGRET MOVES
BLAMELESS INQUIRY
CORE TEAM
SERVERLESS
AUTOMATED INFRASTRUCTURE
ARCHITECTURE DRAWING
Container Solutions
Cloud Native Transformation Patterns
Get your free deck here:
Order your Cloud Native Transformation book here:

Culture Patterns

"Culture is a set of living relationships working toward a shared goal. It's not something you are. It's something you do."

The Culture Code
Daniel Coyle

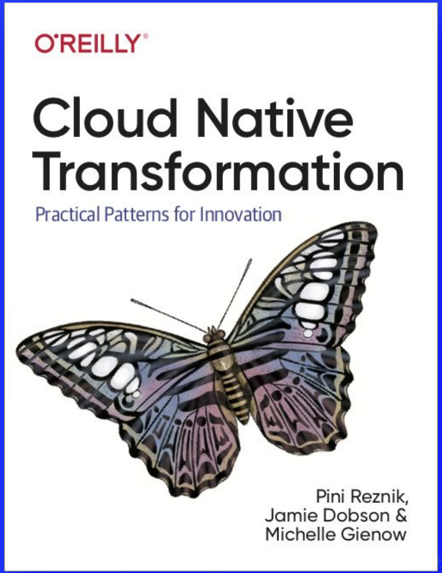
- Strategic Formulation
- Focus on Bottle Necks
- Learning Loop
- Joint Responsibilities
- Avoid Reinventing the Wheel
- Reflective Breaks
- Creative Capabilities
- Psychological Safety
- Value Proficient & Creative Teams Equally
- Measure What Matters
- Honest Feedback
- Blameless Inquiry


Cloud + K8S + Microservices
Full rewrite mode
Back to stable delivery to make boss happy
100%
PROFICIENCY
Turning Point
New product or big refactoring
Attempt no. 1
Attempt no. 2
Attempt 3. Balance is Controlled
Happily Ever After...
MVP of CN Platform
CREATIVITY
0%
time

The Stranger is Coming…

… you are ready now!

A Common Cloud Native Transformation Scenario to Avoid: **'Lift and Shift'.** Read the blog:


O'REILLY
Cloud Native Transformation
Practical Patterns for Innovation
Pini Reznik, Jamie Dobson & Michelle Gienow

Get your **free** pack of pattern cards to map your journey

Container Solutions
Cloud Native Transformation Cards
MICROSERVICES ARCHITECTURE
To reduce the costs of coordination among teams delivering large monolithic applications, build the software as a suite of modular services that are built, deployed, and operated independently.

PLAN YOUR CLOUD NATIVE TRANSFORMATION

Get your **free** pack of pattern cards to map your journey