Cloud Native Transformation

Practical Patterns for Innovation

Pini Reznik
@pini42

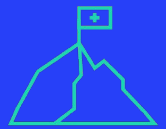
Container Solutions

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

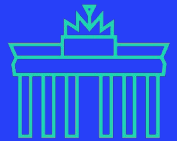
Container Solutions

Amsterdam

London

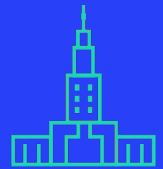


Zurich



Berlin

Montreal



Warsaw

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

Leo Tolstoy

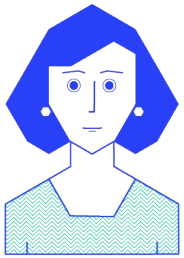
Meet

WEALTHGRID

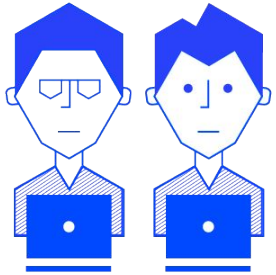
A successful, mid-size financial company

Meet the People

CEO



Jenny
a Technical Manager



Engineers

WEALTHGRID

The Stranger is Coming...

ING

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

ComputerWeekly.com | 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

ComputerWeekly.com 10

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

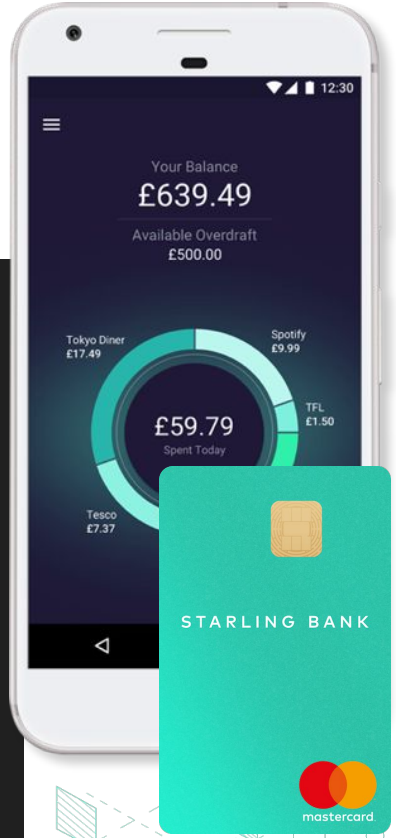
November 2016 Alpha testing mobile app

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

CNBC MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV

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 th

CNN BUSINESS Markets Tech Media Success Perspectives Video

Thomas Franck | @tomwfranck
Published 4:23 PM ET Tue, 6 March 2018 |
CNBC

Pacific

Amazon may eventually have 70 million banking customers

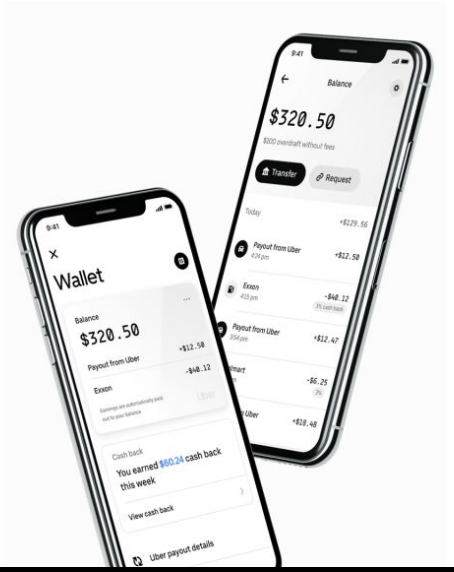
TECH | TRANSPORTATION | UBER

Uber Money is the company's latest attempt to expand into financial services

Debit cards for drivers, credit cards for riders

By Andrew J. Hawkins | @andyjayhawk | Oct 28, 2019, 3:31pm EDT

SHARE

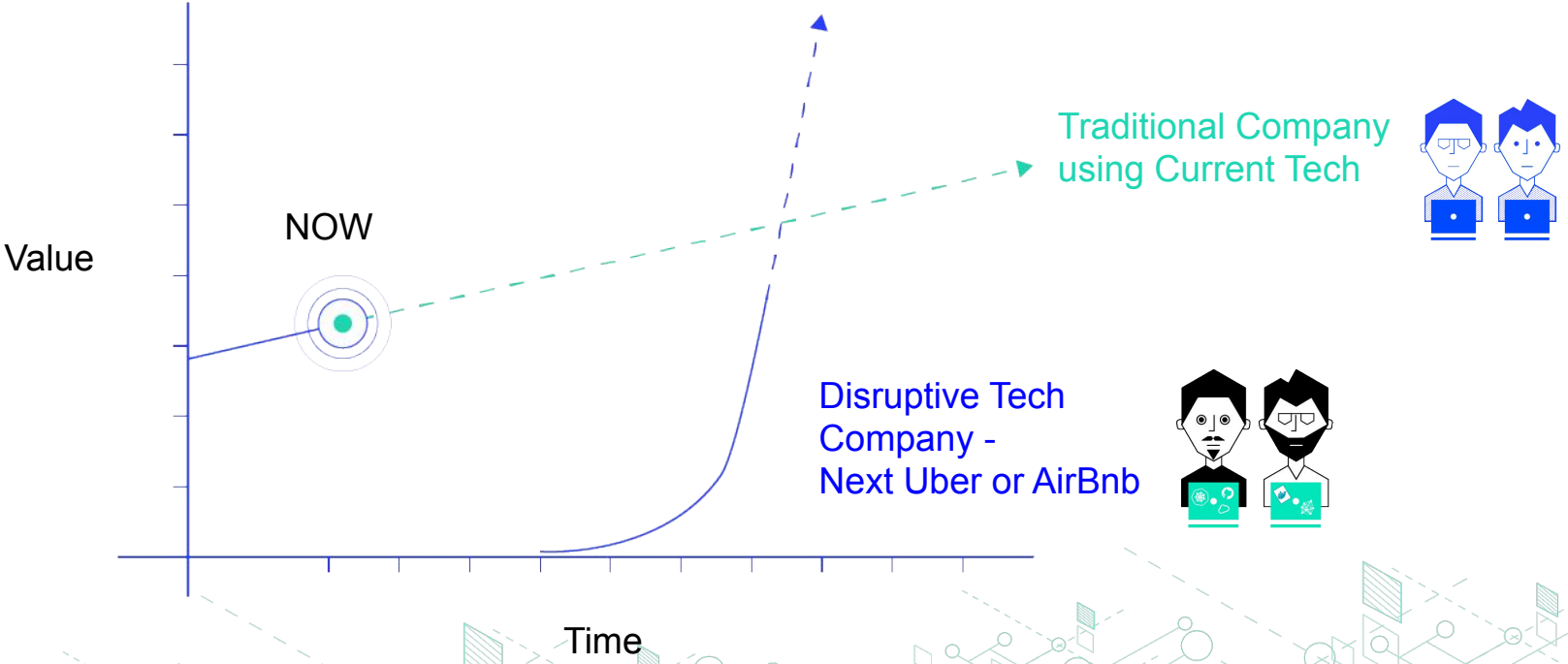


US | Oct 28, 2019

Introducing Uber Money

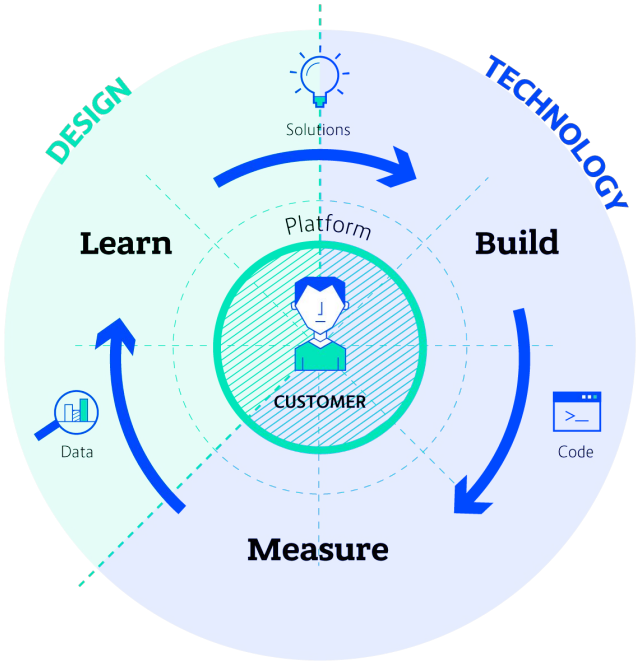
— Written by **Peter Hazlehurst, Head of Uber Money**

They are coming fast!



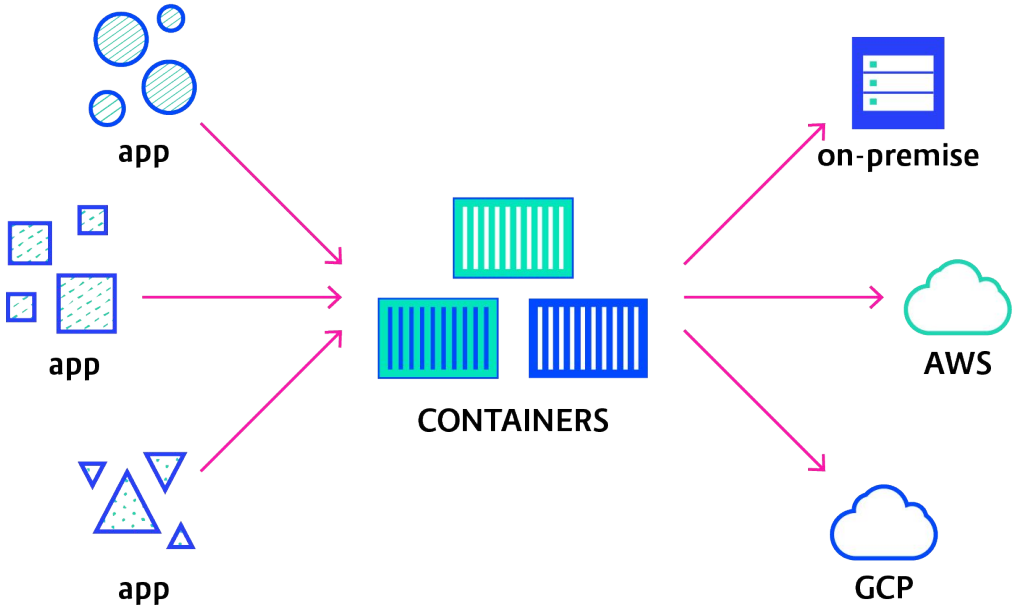
Why?

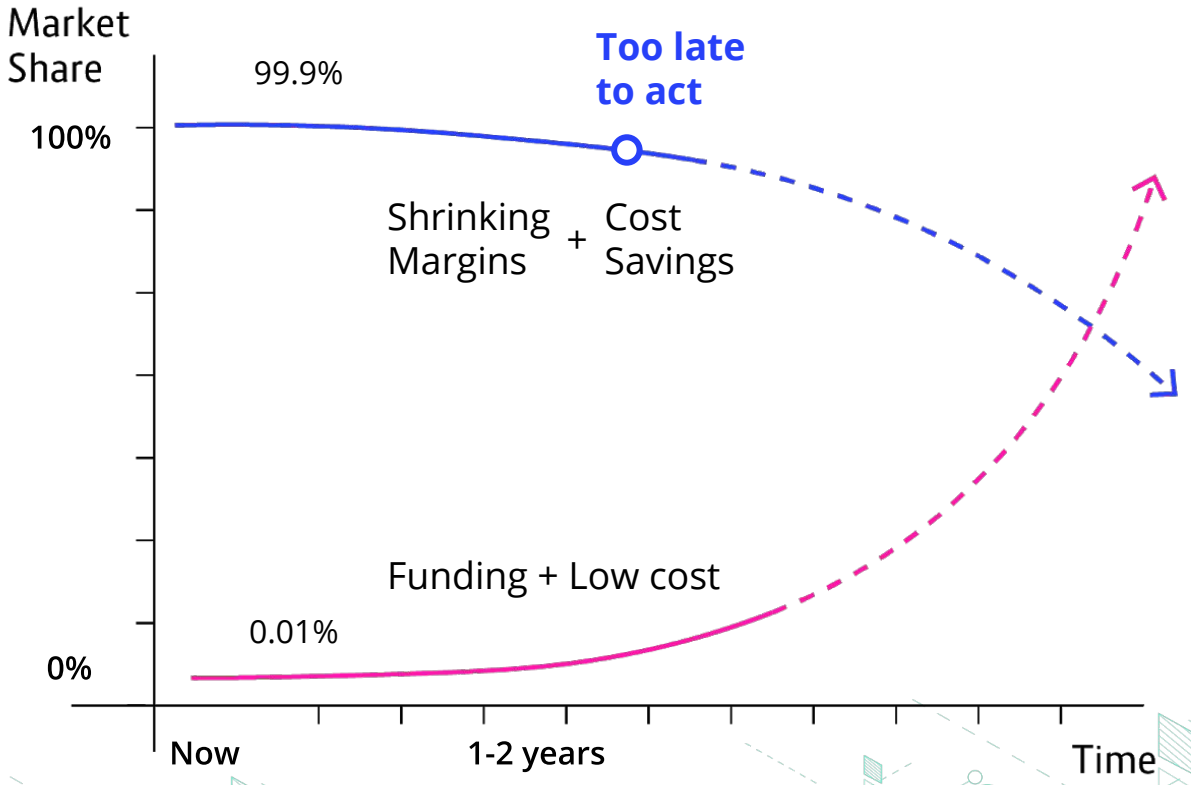
Because they deliver faster and more frequently by using modern technologies



Cloud Native

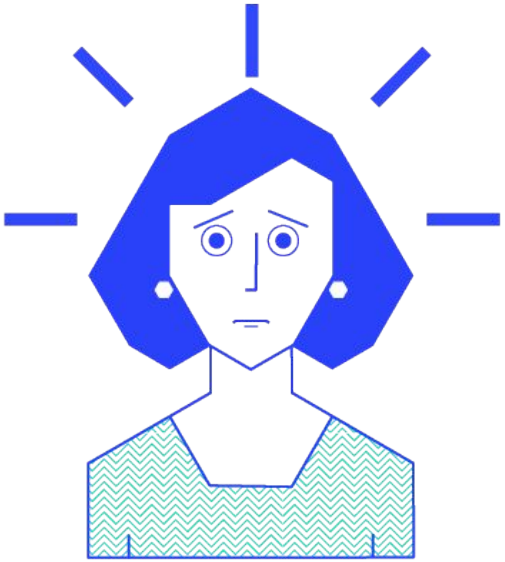
Public Cloud,
Microservices,
Containers (Docker),
Dynamic Scheduling
(Kubernetes),
etc.



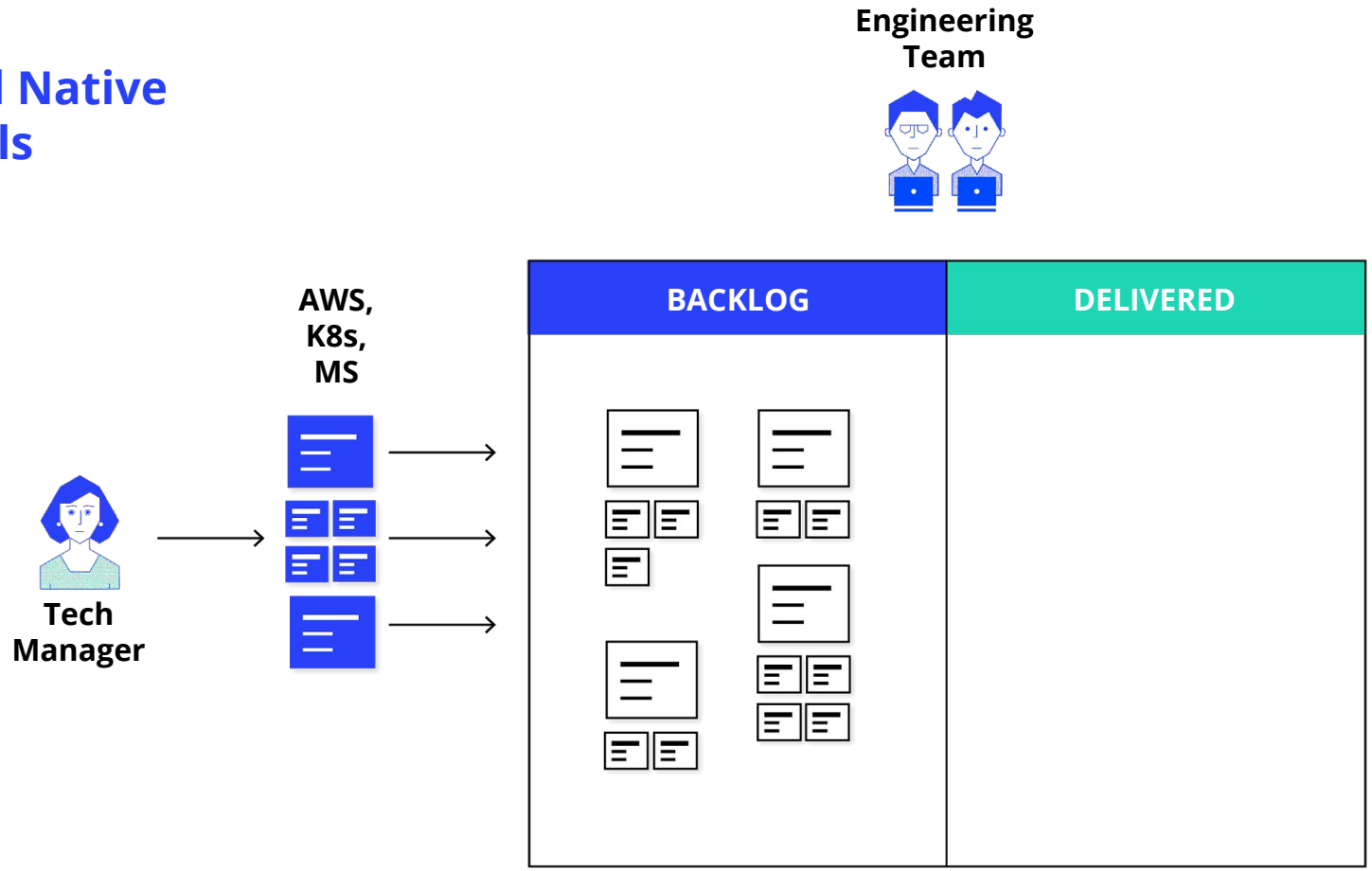

Market Share
99.9%
Too late to act
100%
Shrinking Margins
+
Cost Savings
Funding + Low cost
0.01%
0%
Now
1-2 years
Time

We Must DO Something!

Jenny's wakeup call

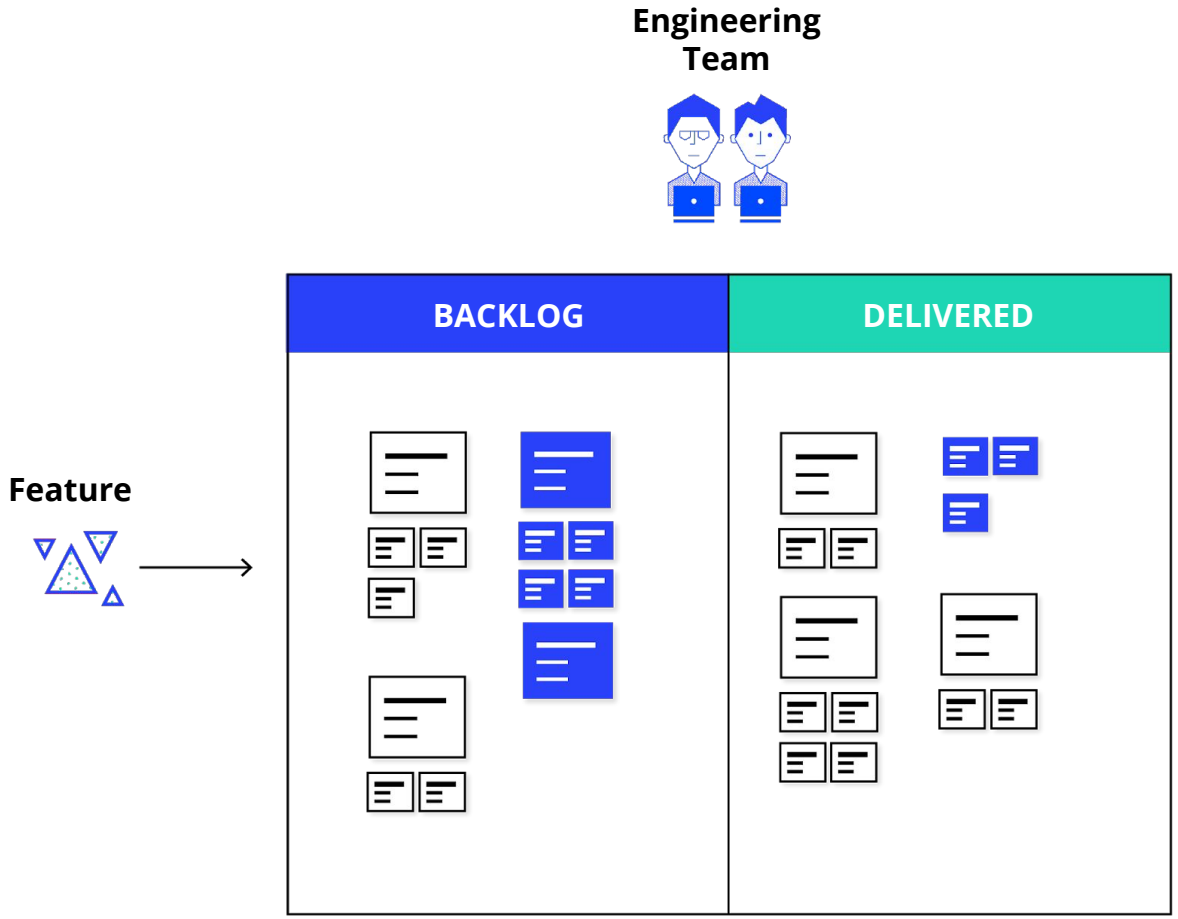


Use Cloud Native Tools



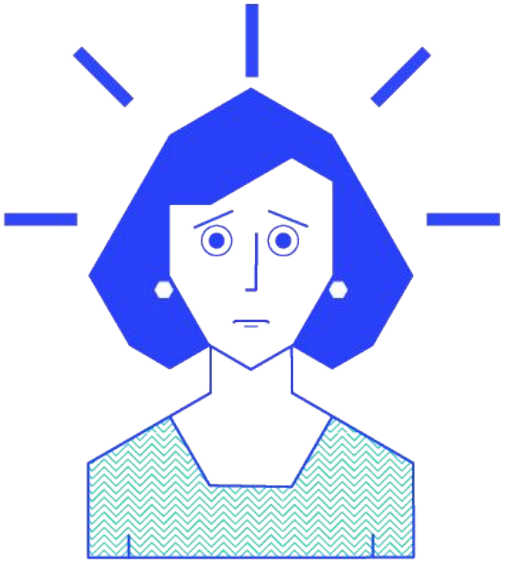
6-12 month later...

Only old stuff +
a bit of CN have
been delivered

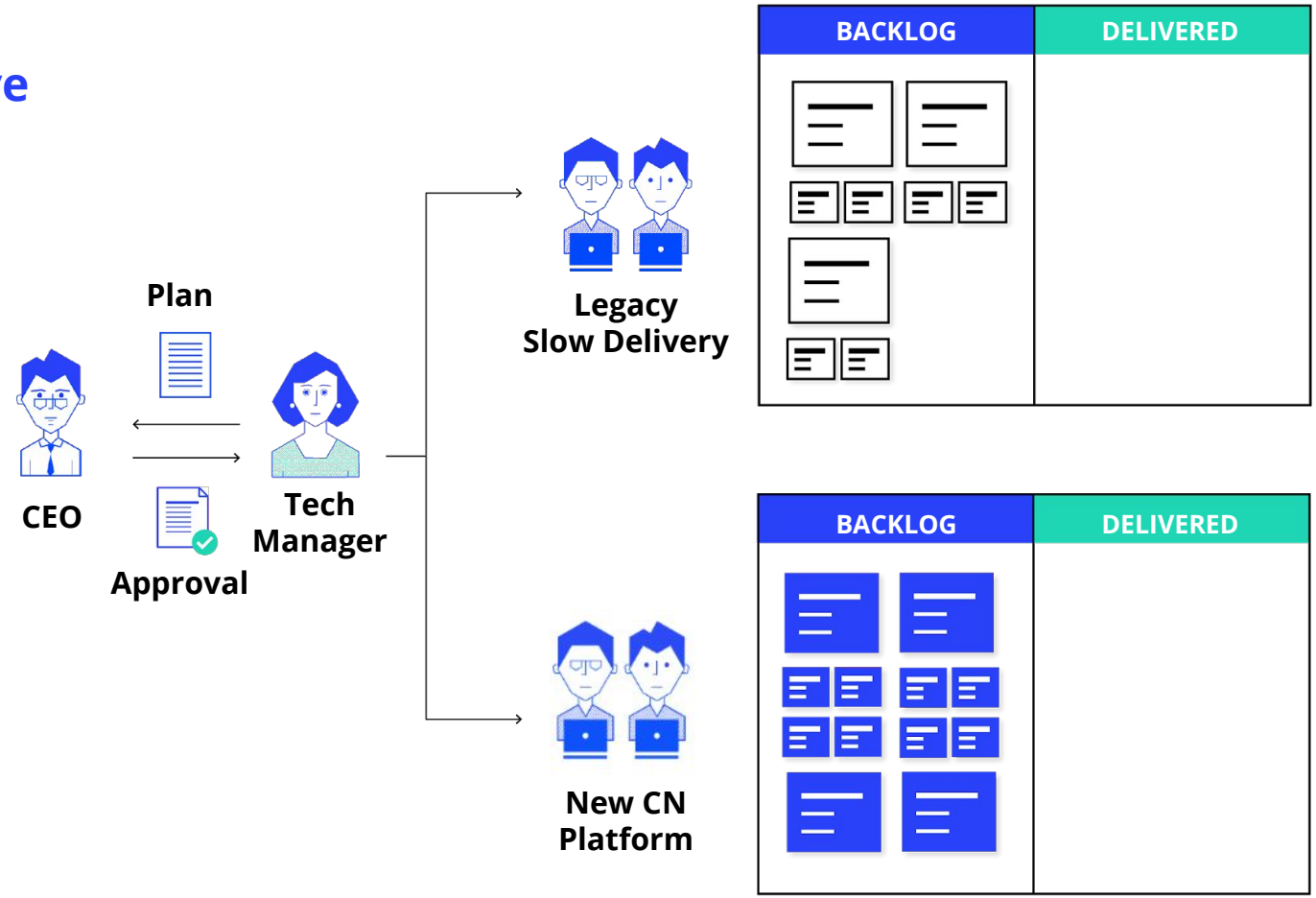


We Must DO Something ELSE!

Jenny's second wakeup call

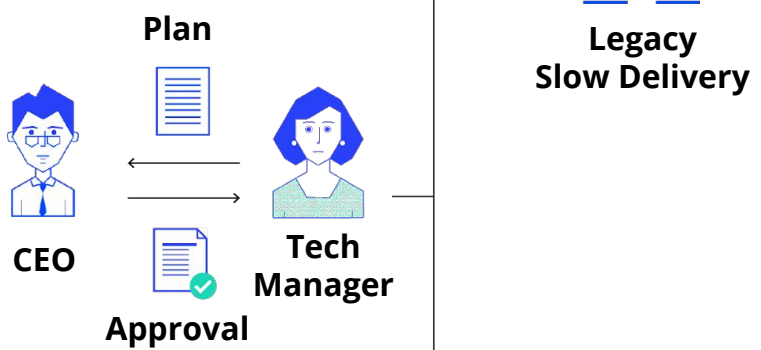


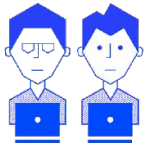
Cloud Native Rewrite



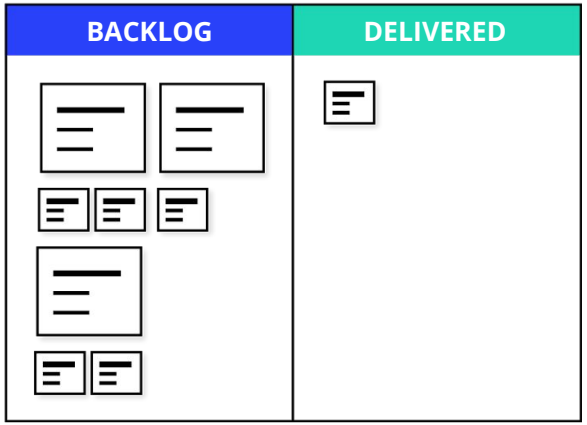
6-12 month later...

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

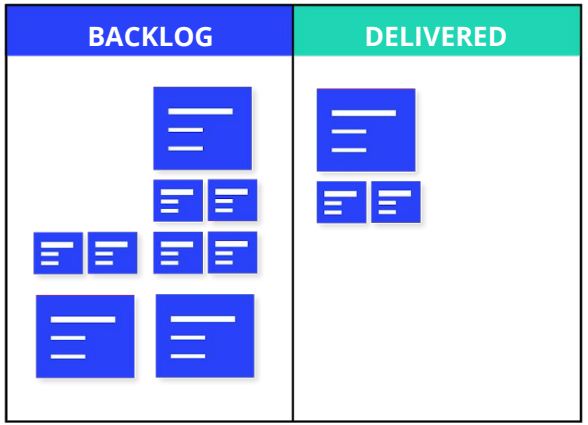




Legacy
Slow Delivery



New CN
Platform



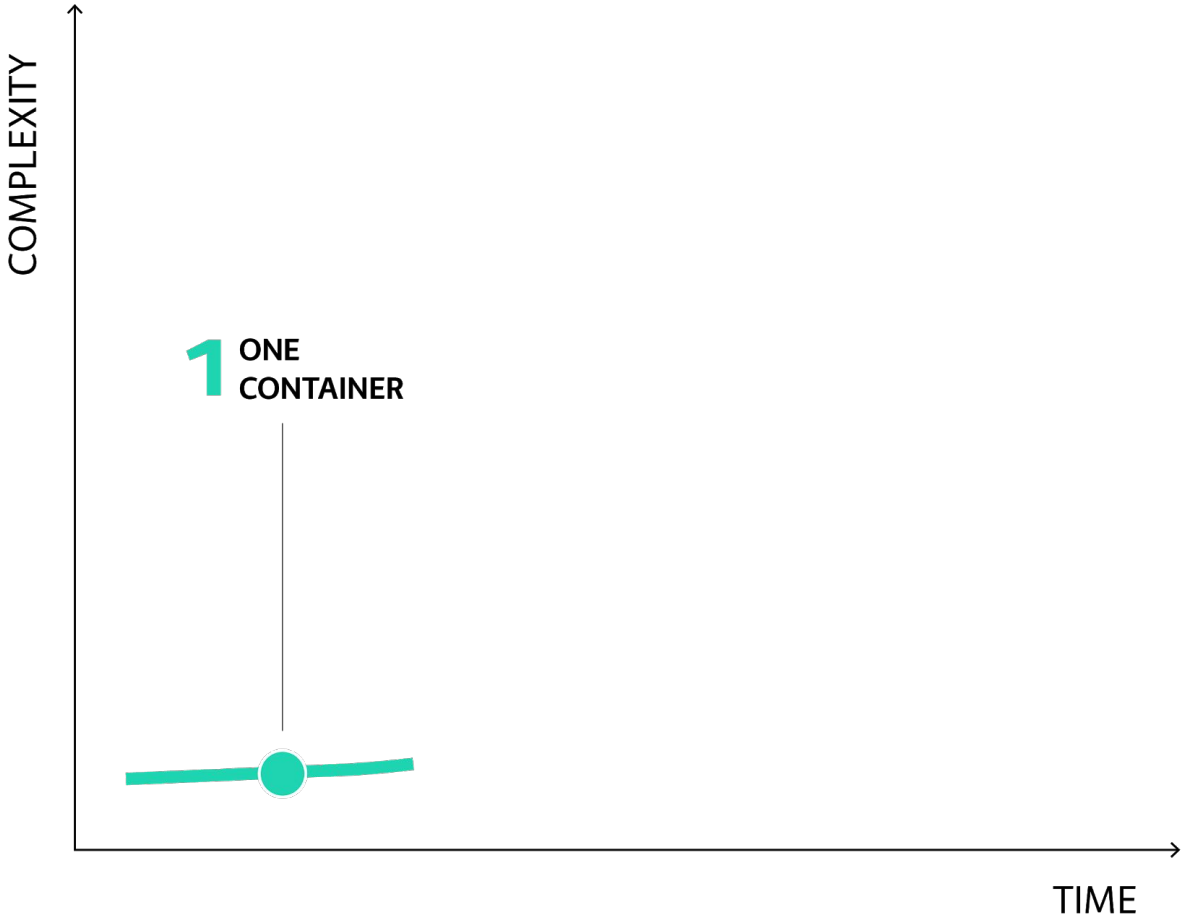
Why is it so difficult?

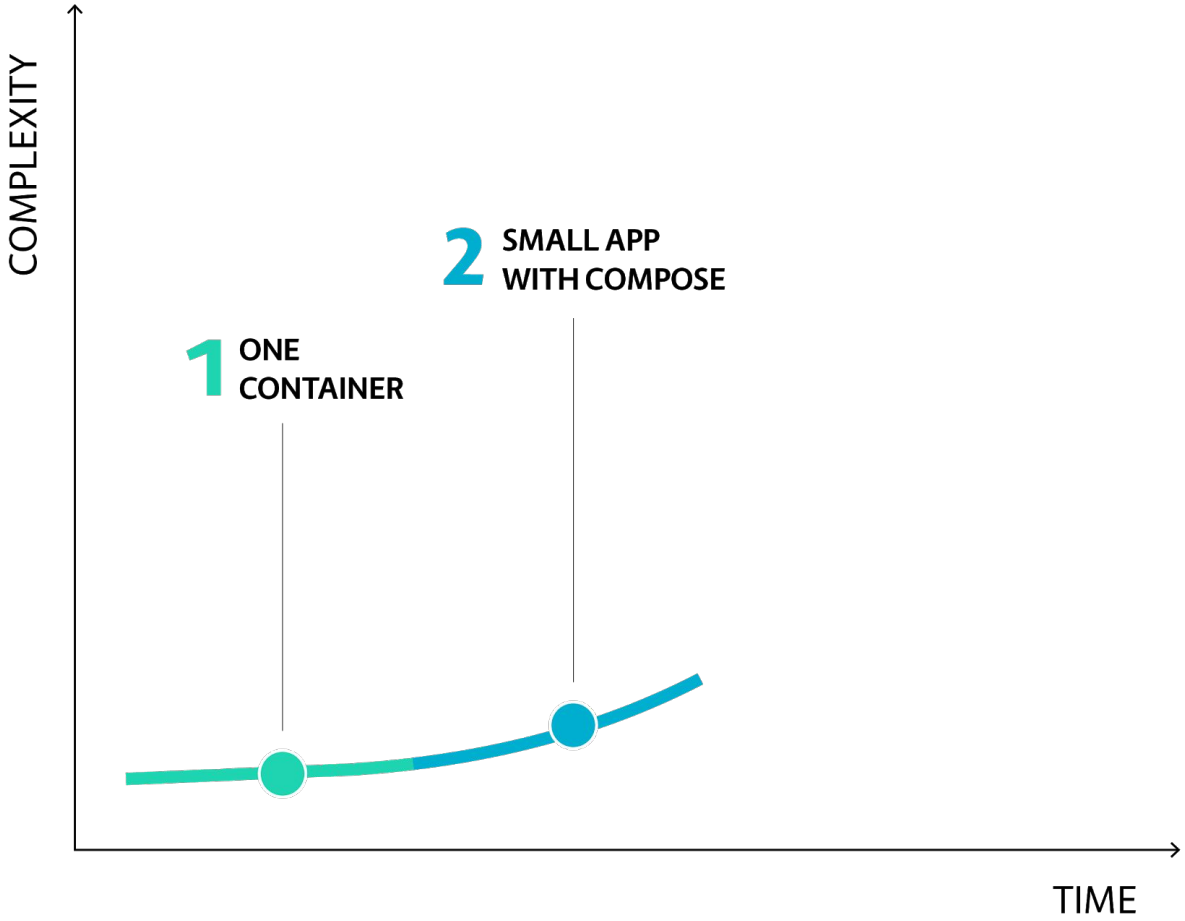
Because - Cloud Native is New, Complex and requires new ways of thinking!

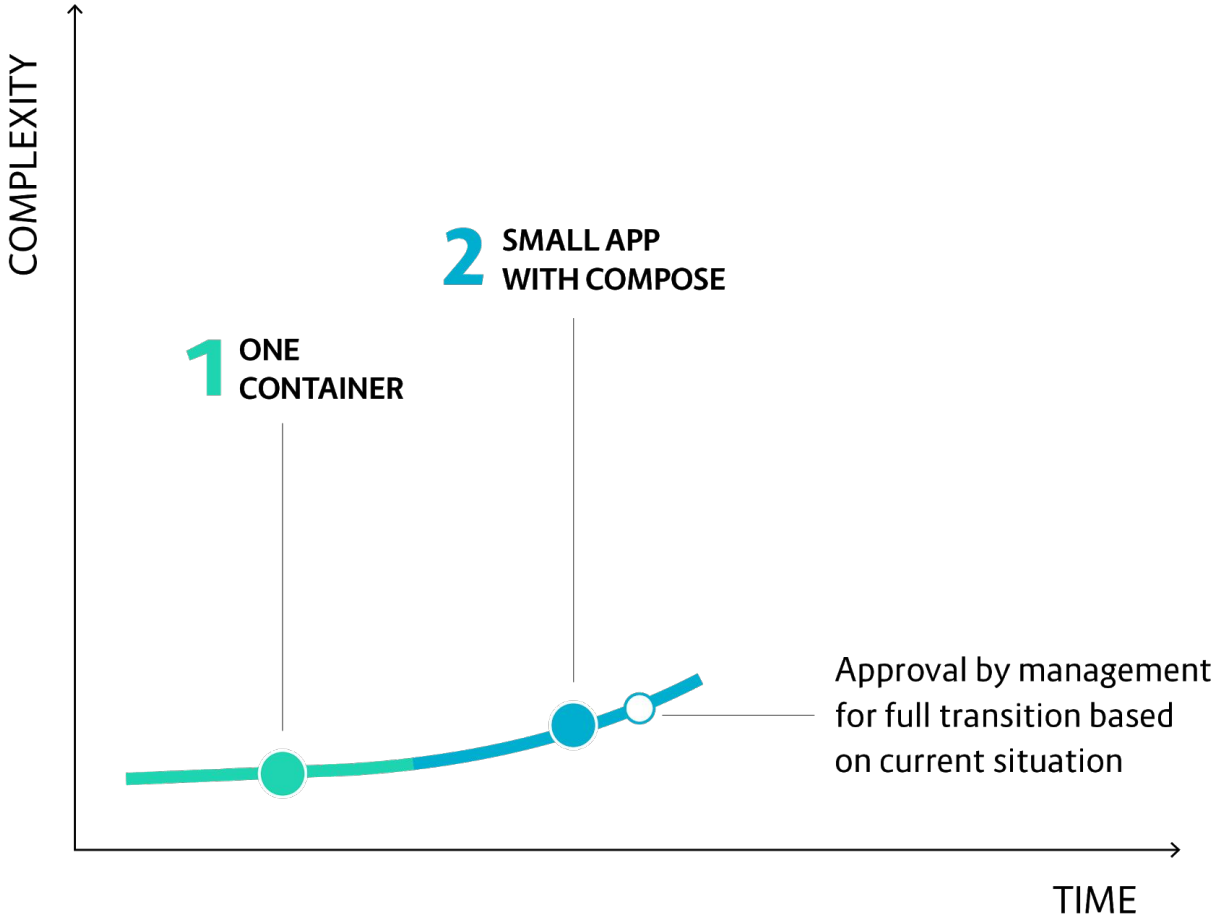
What is CNCF?

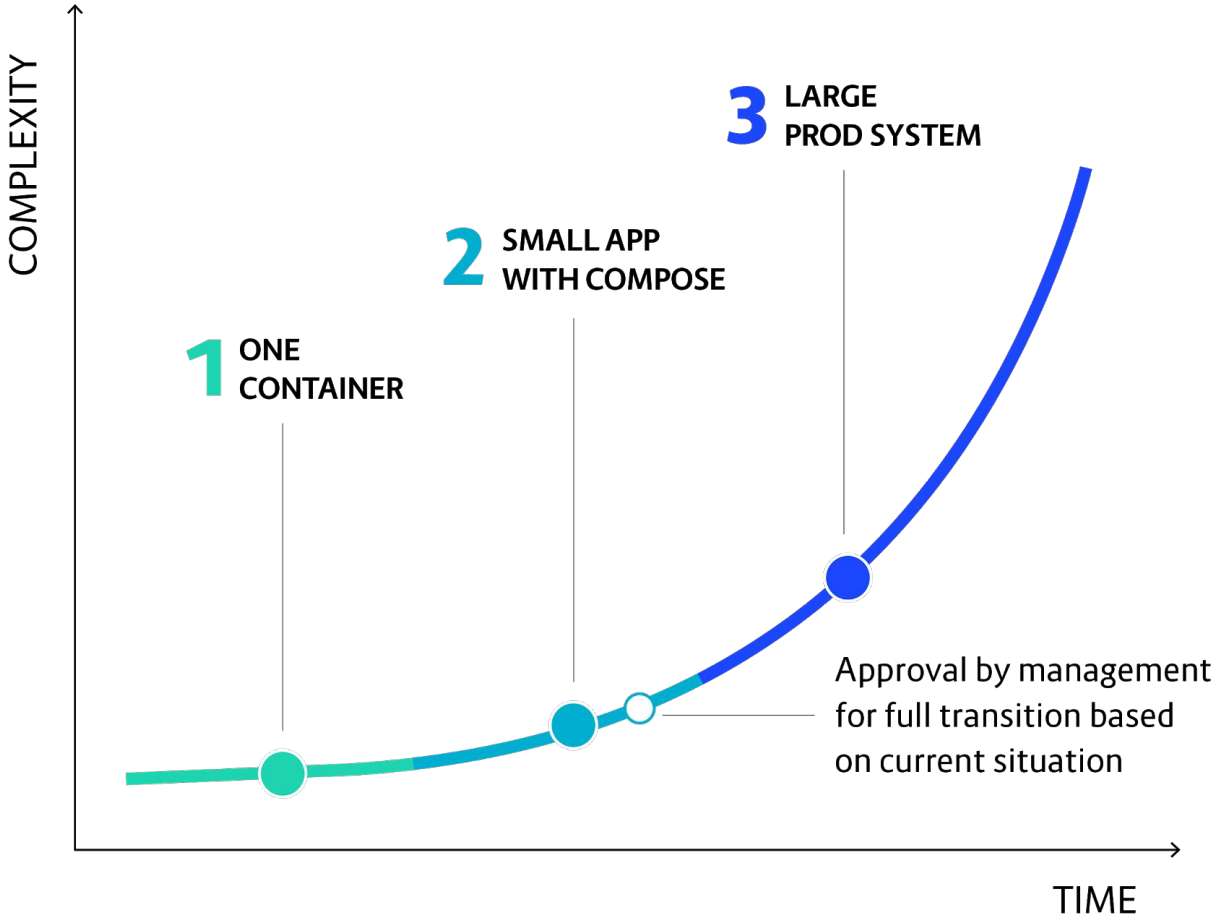
CNCF is an open source software foundation dedicated to making cloud-native computing universal and sustainable. Cloud-native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud-native technologies enable software developers to build great products faster.

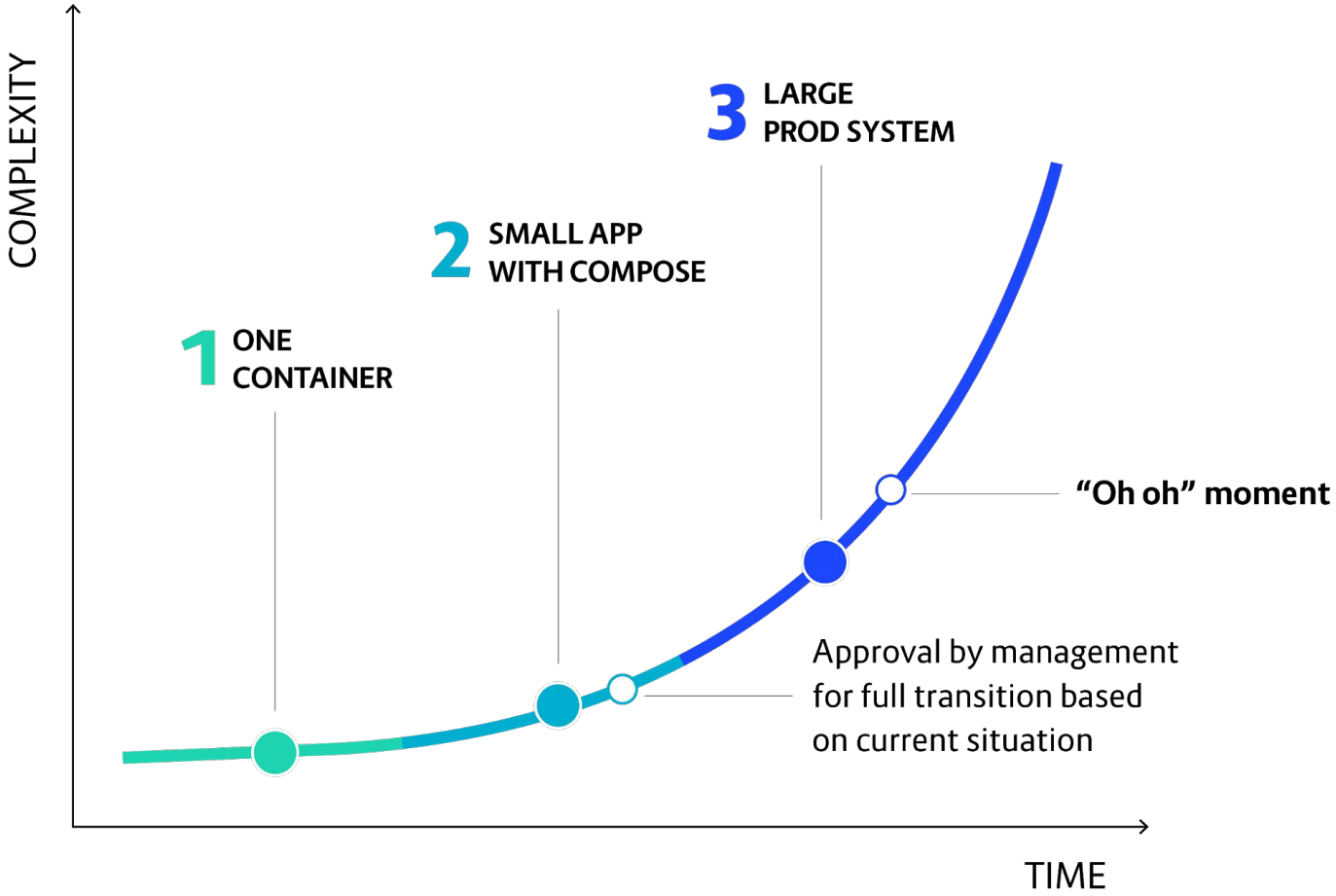
JOIN

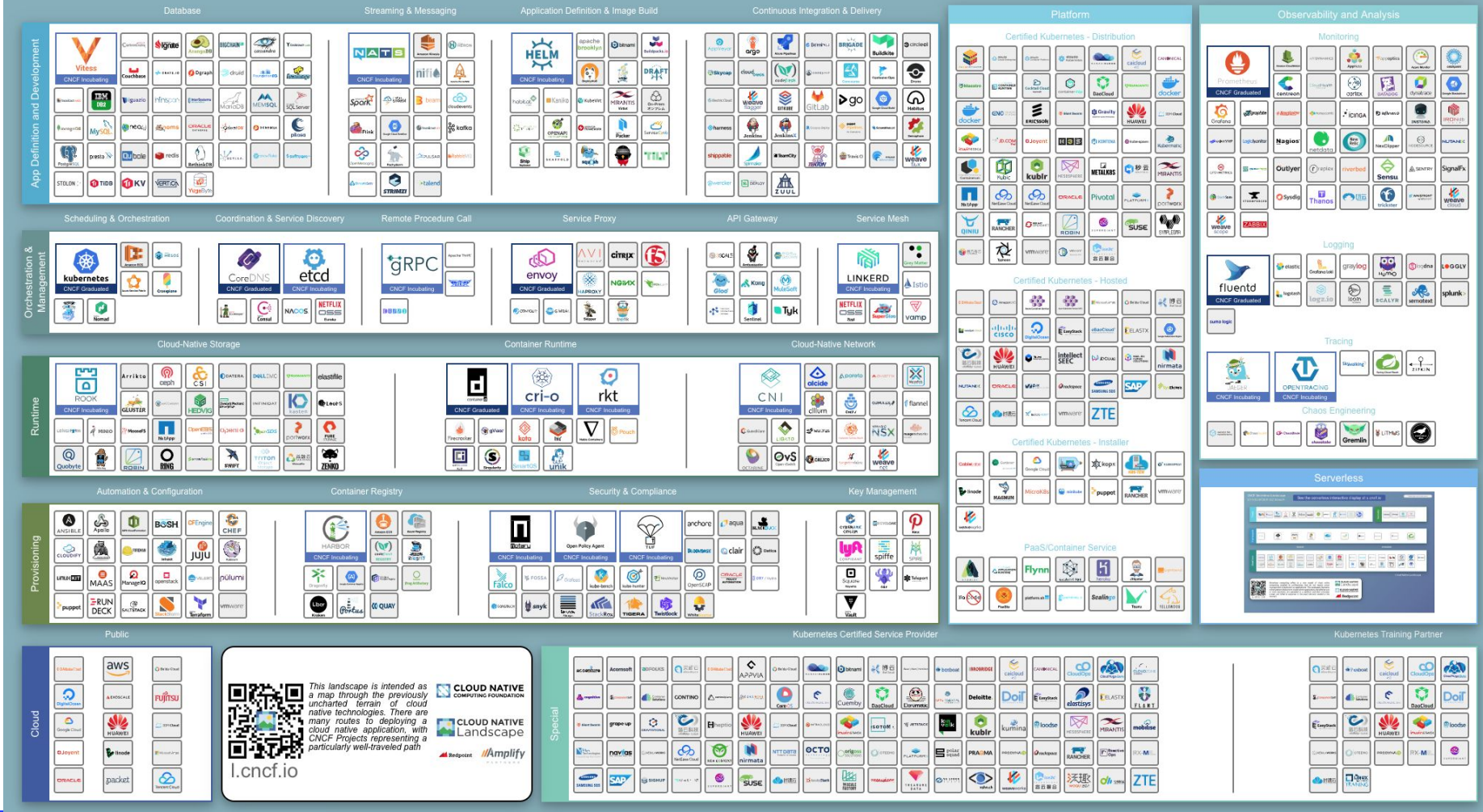

COMPLEXITY
1
ONE
CONTAINER
TIME


COMPLEXITY
2 SMALL APP WITH COMPOSE
1 ONE CONTAINER
TIME


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


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


COMPLEXITY
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
TIME

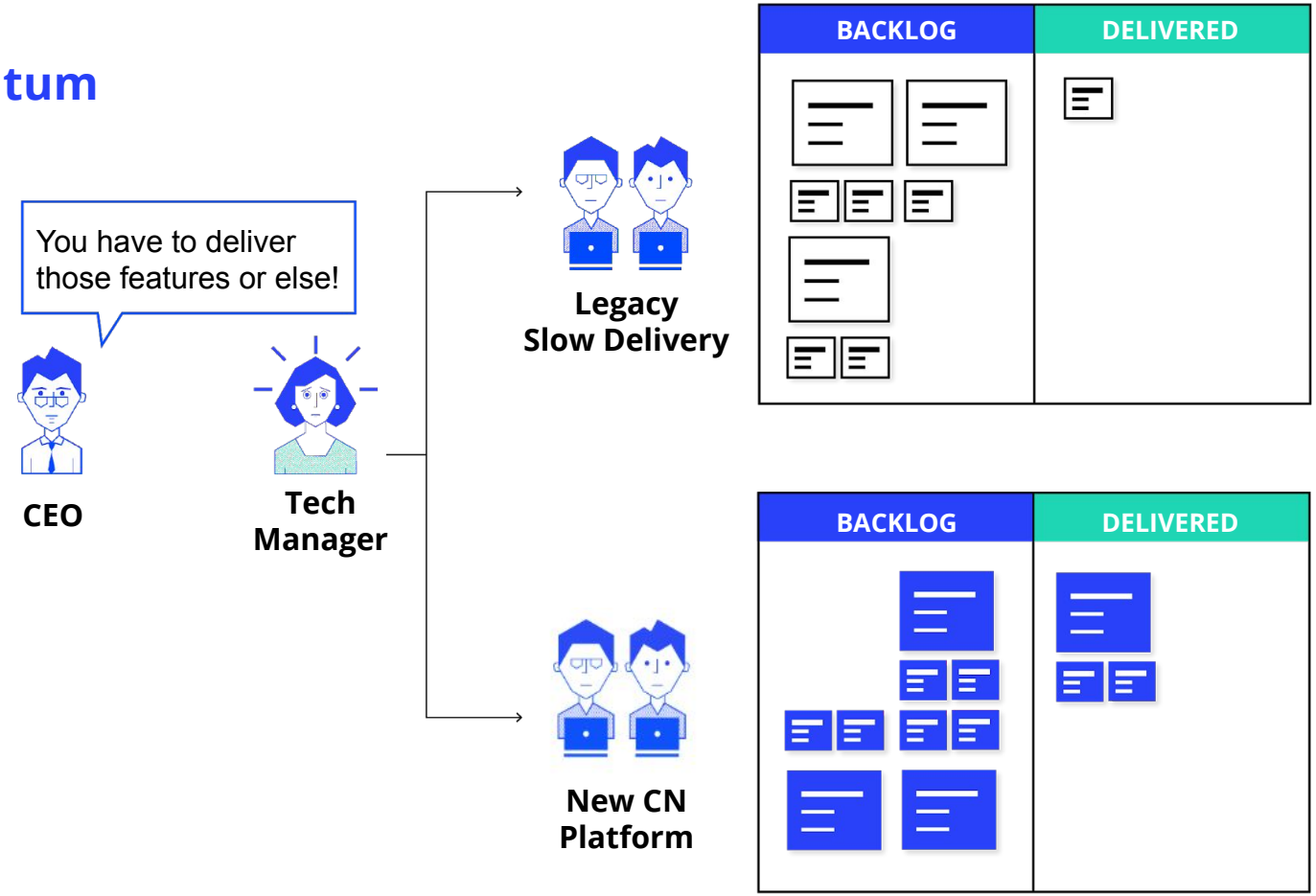

Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Vitess
CNCF Incubating
NATS
CNCF Incubating
HELM
CNCF Incubating
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Orchestration & Management
kubernetes
CNCF Graduated
CoreDNS
CNCF Incubating
etcd
CNCF Incubating
gRPC
CNCF Incubating
envoy
CNCF Graduated
LINKERD
CNCF Incubating
Cloud-Native Storage
Container Runtime
Cloud-Native Network
Runtime
ROOK
CNCF Incubating
cri-o
CNCF Incubating
rkt
CNCF Incubating
CNI
CNCF Incubating
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Provisioning
HARBOR
CNCF Incubating
Notary
CNCF Incubating
Open Policy Agent
CNCF Incubating
TUF
CNCF Incubating
Public
Kubernetes Certified Service Provider
Cloud
Special
aws
packet
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path
l.cncf.io
CLOUD NATIVE COMPUTING FOUNDATION
CLOUD NATIVE Landscape
Redpoint
Amplify
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Prometheus
CNCF Graduated
Logging
fluentd
CNCF Graduated
Tracing
OPENTRACING
CNCF Incubating
Chaos Engineering
Serverless
Kubernetes Training Partner

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

Maturity Matrix

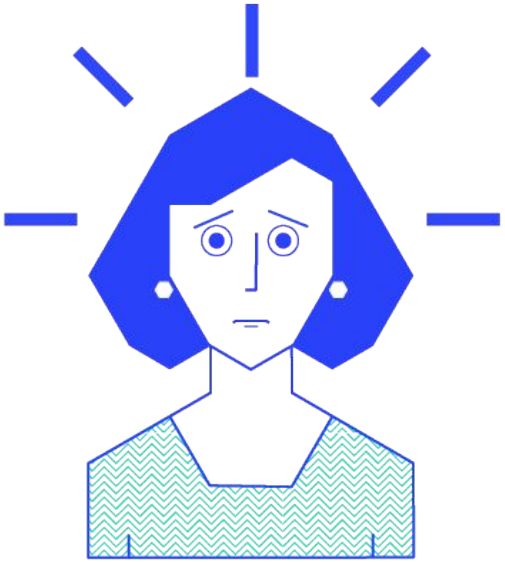
Stage	NO PROCESS	WATERFALL	AGILE	CLOUD NATIVE	NEXT
CULTURE	Individualist	Predictive	Iterative	Collaborative	Experimental
PROD/SERVICE DESIGN	Arbitrary	Long-term plan	Feature driven	Data driven	All driven
TEAM	No organization, single contributor	Hierarchy	Cross-functional teams	DevOps / SRE	Internal supply 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

The Ultimatum



We Must DO Something Else AGAIN!

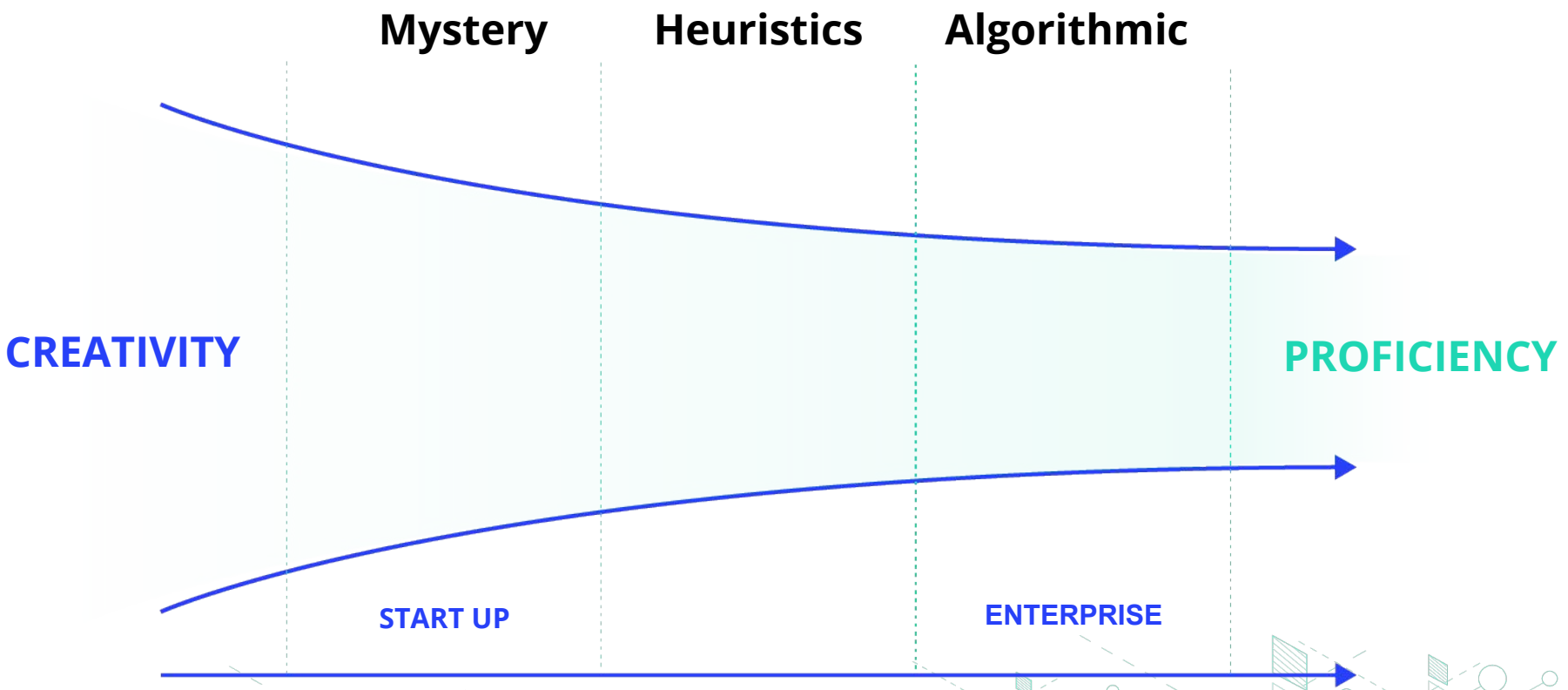
Jenny's third wakeup call

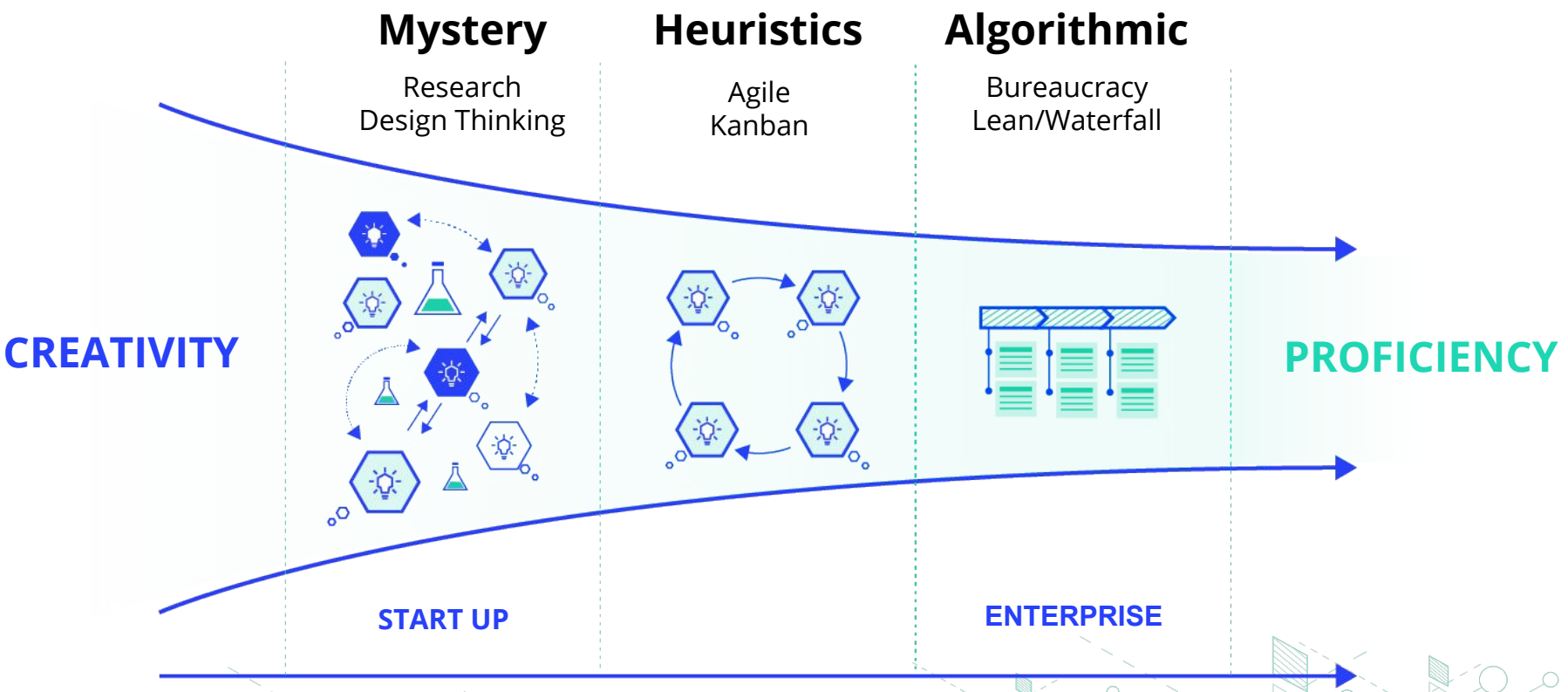


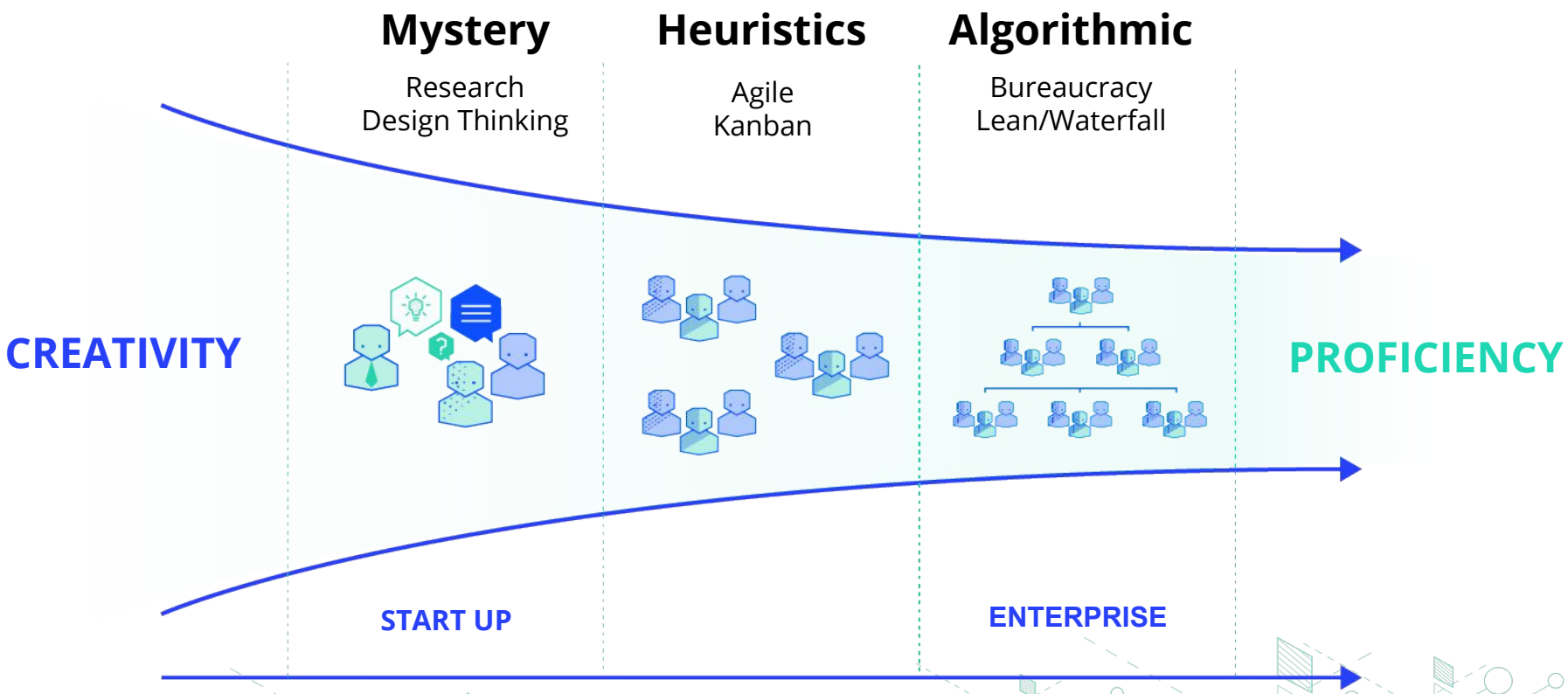
Let’s introduce some tools …

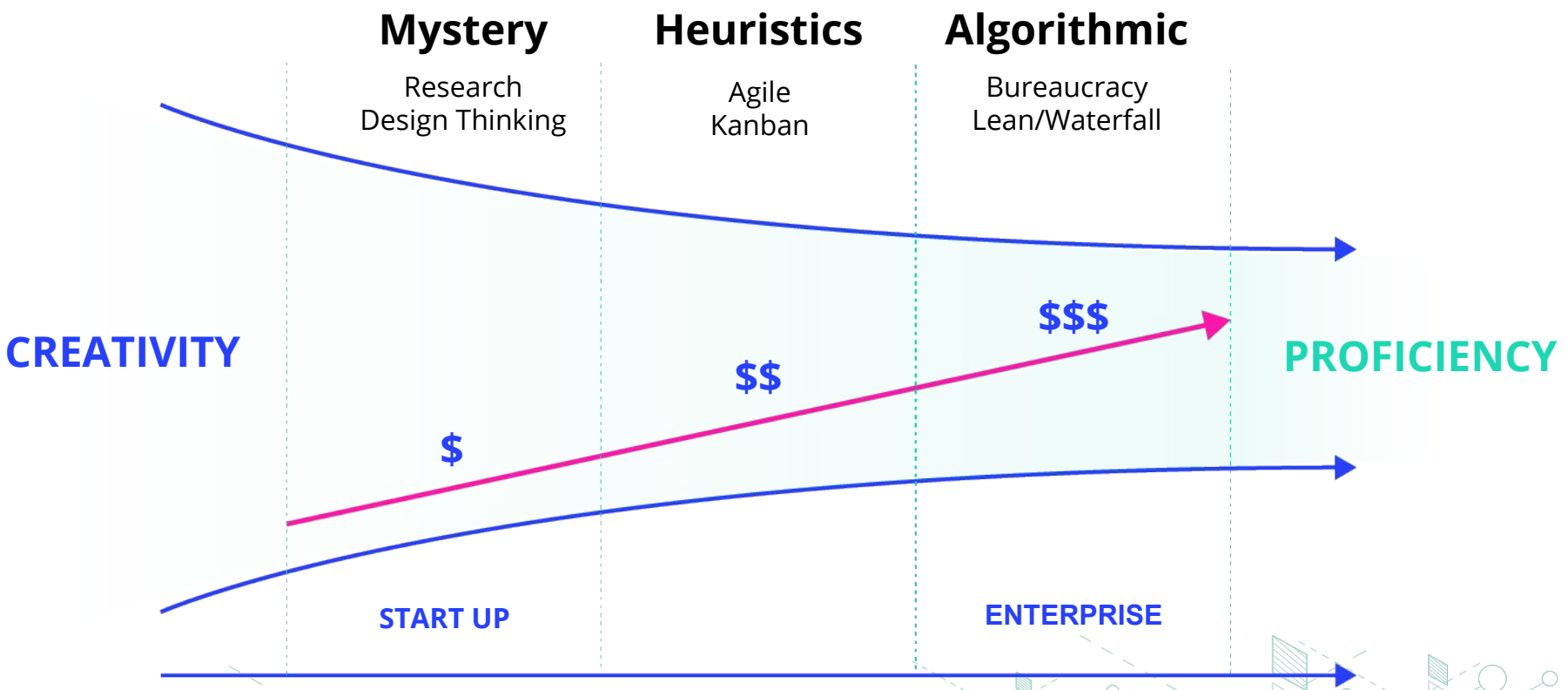
Tool no. 1

Creativity vs. Proficiency

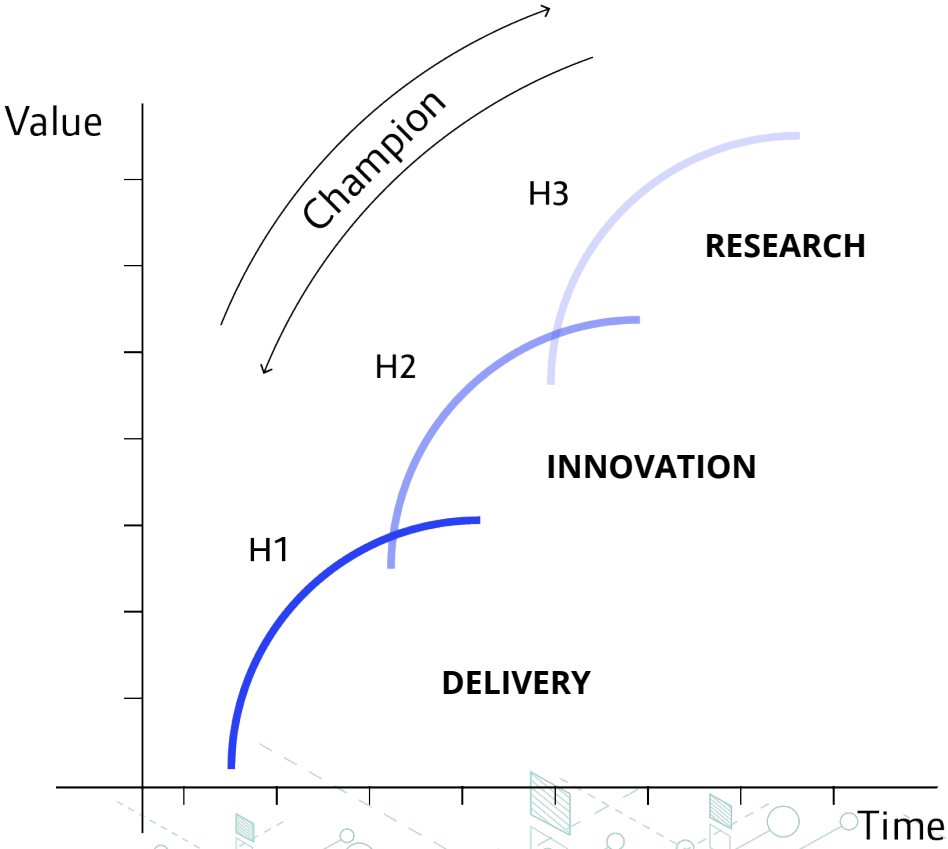

Mystery
Heuristics
Algorithmic
CREATIVITY
PROFICIENCY
START UP
ENTERPRISE

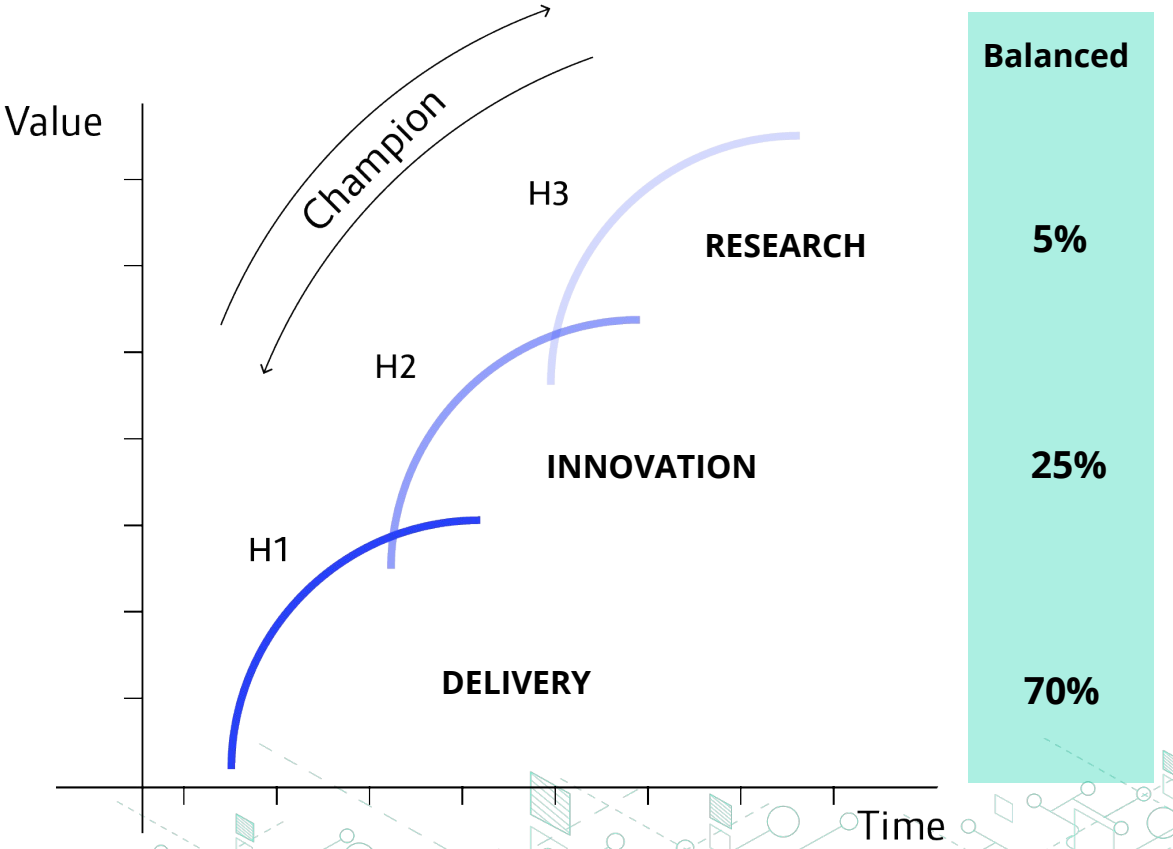

Mystery
Research
Design Thinking
Heuristics
Agile
Kanban
Algorithmic
Bureaucracy
Lean/Waterfall
CREATIVITY
PROFICIENCY
START UP
ENTERPRISE


Mystery
Research
Design Thinking
Heuristics
Agile
Kanban
Algorithmic
Bureaucracy
Lean/Waterfall
CREATIVITY
PROFICIENCY
START UP
ENTERPRISE

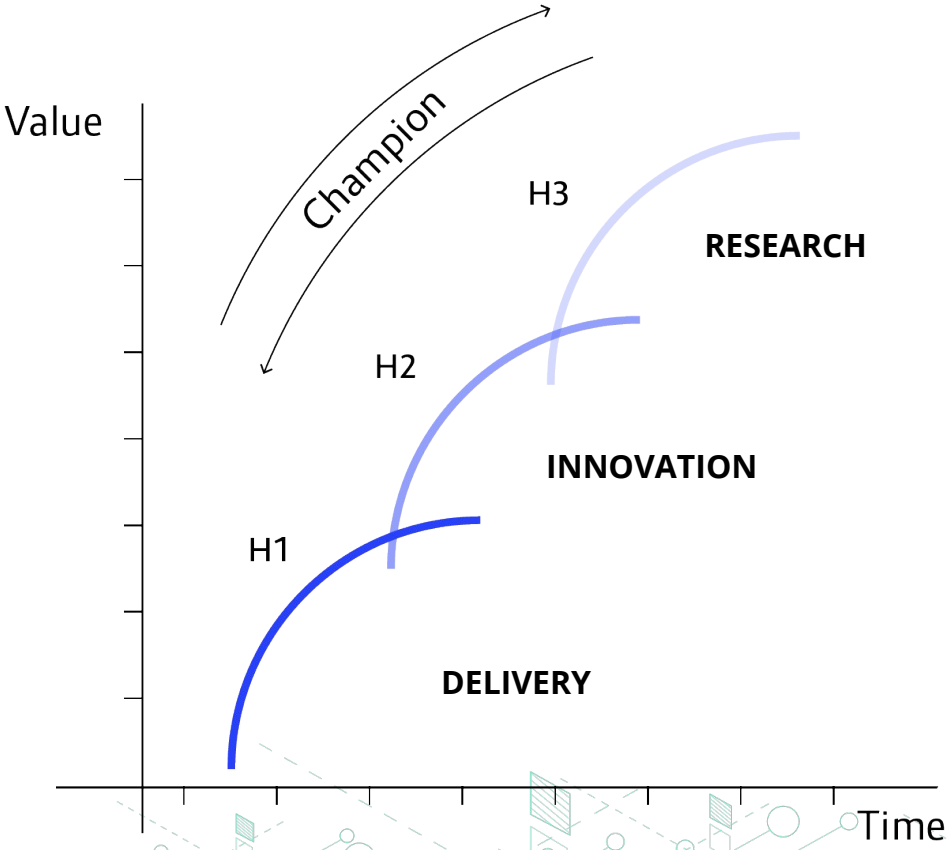

Mystery
Heuristics
Algorithmic
Research
Design Thinking
Agile
Kanban
Bureaucracy
Lean/Waterfall
CREATIVITY
PROFICIENCY
$
$$
$$$
START UP
ENTERPRISE


ice
water
phase transition
this is where you find success
STRATEGY
enterprise
PROFICIENCY
ALL BLACKS
NAVY SEALS
VISION
Creativity
startup
PIXAR
IDEO
HIGH REPETITION
HIGH FEEDBACK
SMALL SET OF RULES
EMPHASISE SUCCESFUL SKILS
PERSONAL SAFETY STANDARDS
exchange
INNOVATION
and
PROFICIENCY
PURPOSE
SUPPORT
SAFETY
AUTONOMY
TEAM DYNAMICS

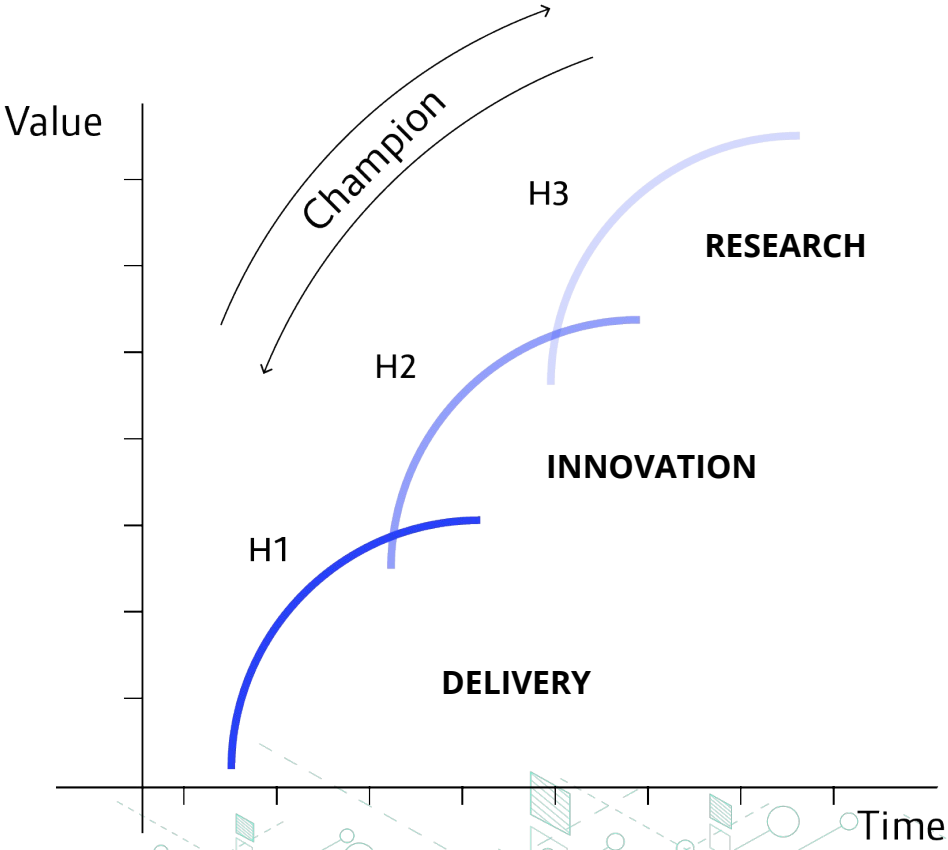

Value
Champion
H3
RESEARCH
H2
INNOVATION
H1
DELIVERY
Time



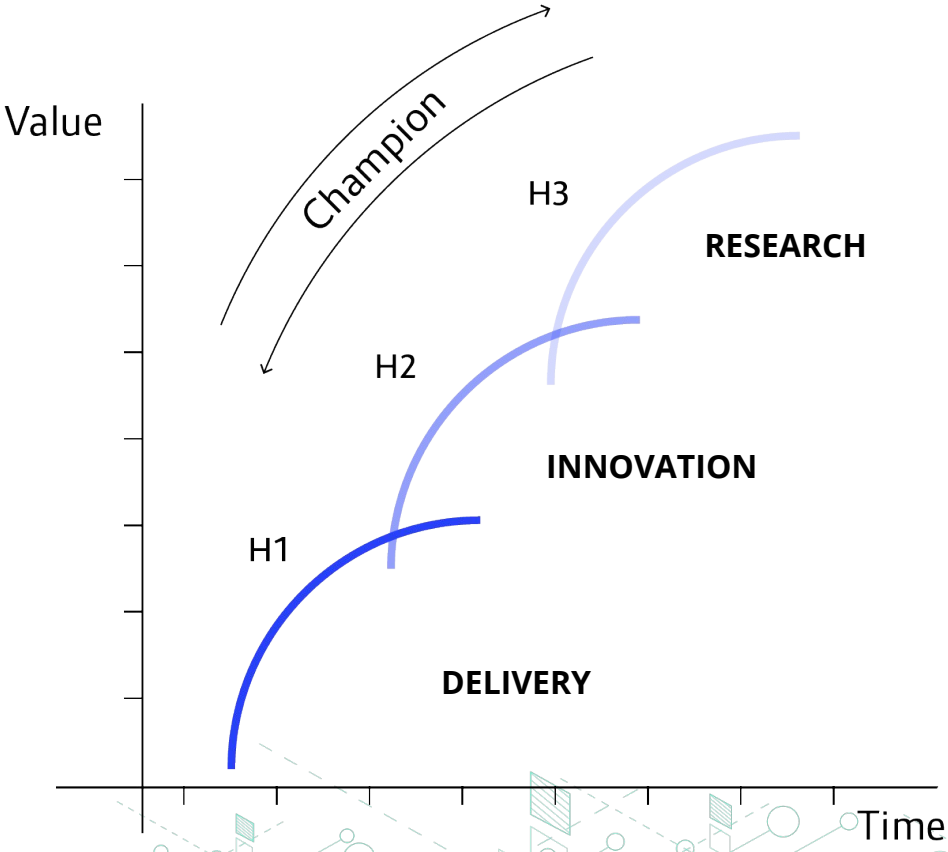
Balanced
5%
25%
70%



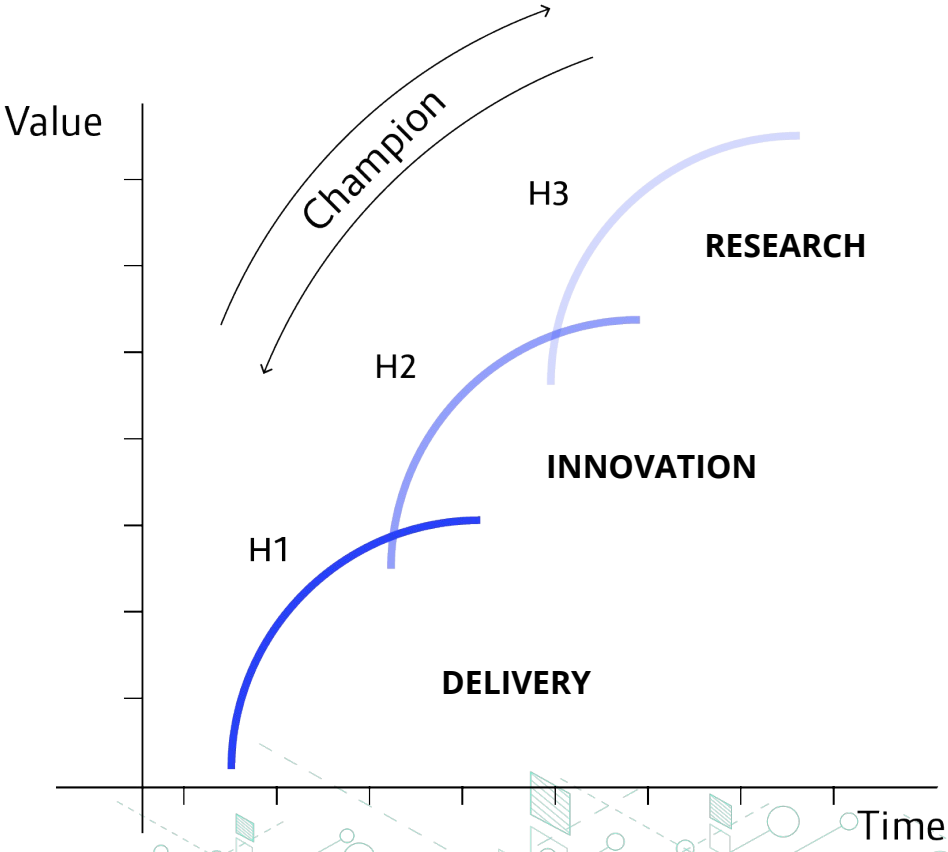
	Balanced	
RESEARCH	5%	1%
INNOVATION	25%	4%
DELIVERY	70%	95%



Balanced		
5%	1%	10%
25%	4%	80%
70%	95%	10%



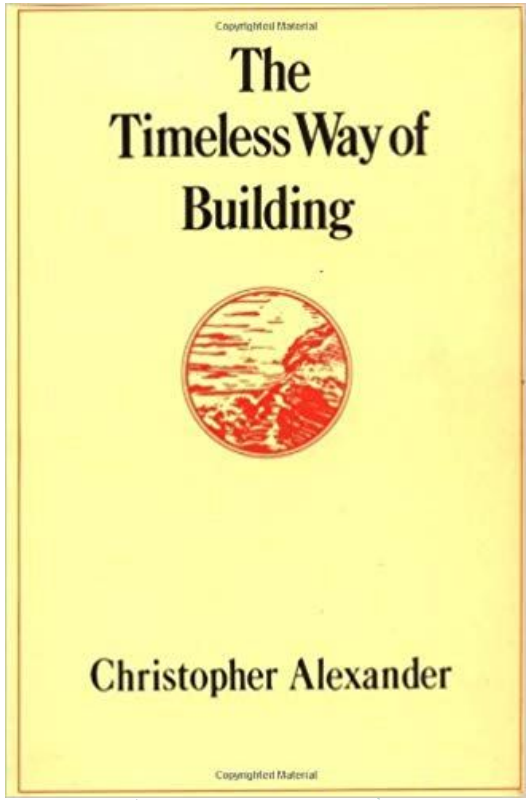
Balanced			
5%	1%	10%	100%
25%	4%	80%	0%
70%	95%	10%	0%

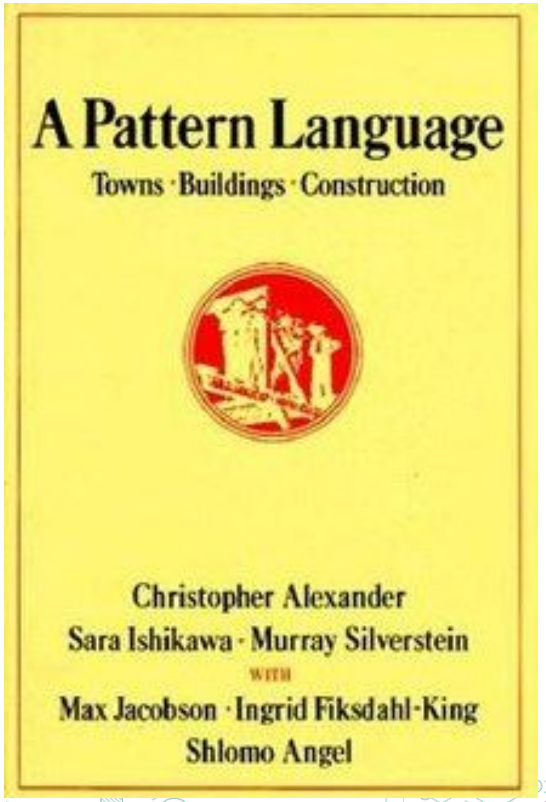


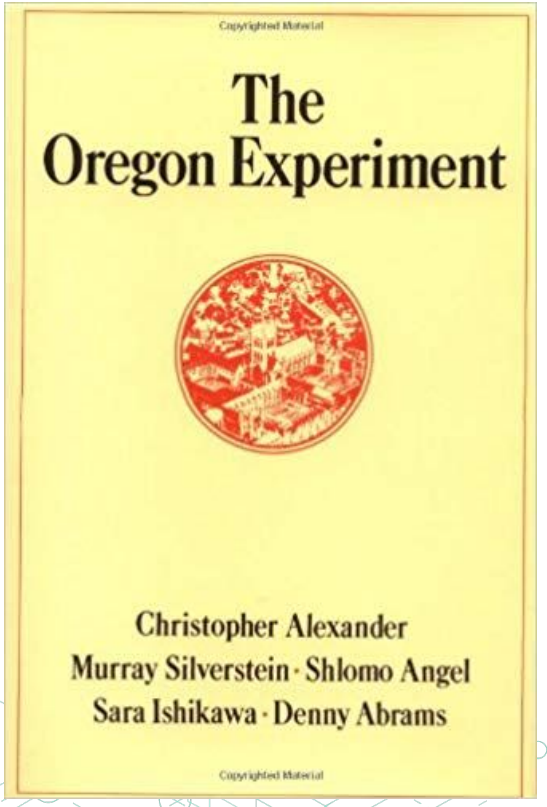
Balanced	Enterprise	Startup	University
5%	1%	10%	100%
25%	4%	80%	0%
70%	95%	10%	0%

Tool no. 2

Patterns


Copyrighted Material
The Timeless Way of Building
Christopher Alexander
Copyrighted Material


A Pattern Language
Towns · Buildings · Construction
Christopher Alexander
Sara Ishikawa · Murray Silverstein
WITH
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel


Copyrighted Material
The Oregon Experiment
Christopher Alexander
Murray Silverstein · Shlomo Angel
Sara Ishikawa · Denny Abrams
Copyrighted Material

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.

Vision First

The company needs to define a clear and achievable vision, which will later be translated into specific executable steps.

Vision First

The company needs to define a clear and achievable vision, which will later be translated into specific executable steps.

In this context:

The combination of limited experience and lack of extra time and flexibility for research leads to pursuing CN implementation using “well known ways”.

- Without an overall consistent vision, different teams will make independent and, frequently, conflicting architectural decisions
- In many companies, Enterprise Architects are responsible for creating a detailed architecture. Many Enterprise Architects lack sufficient theoretical or practical experience in the Cloud Native approach.
- Agile methodologies, widely adopted in the contemporary business world, create pressure to produce results early and onboard teams to new systems very quickly.

Vision First

The company needs to define a clear and achievable vision, which will later be translated into specific executable steps.

In this context:

The combination of limited experience and lack of extra time and flexibility for research leads to pursuing CN implementation using "well known ways".

- Without an overall consistent vision, different teams will make independent and, frequently, conflicting architectural decisions
- In many companies, Enterprise Architects are responsible for creating a detailed architecture. Many Enterprise Architects lack sufficient theoretical or practical experience in the Cloud Native approach.
- Agile methodologies, widely adopted in the contemporary business world, create pressure to produce results early and onboard teams to new systems very quickly.

Therefore:

You should define and visualize the architecture of the whole system upfront.

It can either be requested from external sources or uncovered by series of small research and prototyping projects ranging from few hours to few days each.

It's important to keep the vision high level to allow freedom of choice during implementation, yet also detailed enough to provide clear guidance (which will help avoid common pitfalls).

Vision First

The company needs to define a clear and achievable vision, which will later be translated into specific executable steps.

In this context:

The combination of limited experience and lack of extra time and flexibility for research leads to pursuing CN implementation using “well known ways”.

- Without an overall consistent vision, different teams will make independent and, frequently, conflicting architectural decisions
- In many companies, Enterprise Architects are responsible for creating a detailed architecture. Many Enterprise Architects lack sufficient theoretical or practical experience in the Cloud Native approach.
- Agile methodologies, widely adopted in the contemporary business world, create pressure to produce results early and onboard teams to new systems very quickly.

Therefore:

You should define and visualize the architecture of the whole system upfront.

It can either be requested from external sources or uncovered by series of small research and prototyping projects ranging from few hours to few days each.

It’s important to keep the vision high level to allow freedom of choice during implementation, yet also detailed enough to provide clear guidance (which will help avoid common pitfalls).

Consequently:

All teams have a clear guiding principle for the implementation phase.

The teams can start producing the lower level architecture, and translate it to the backlog of tasks. Therefore, Executive Commitment paired with leadership by the Transformation Champion are essential to have in place for successful vision creation.

Vision First

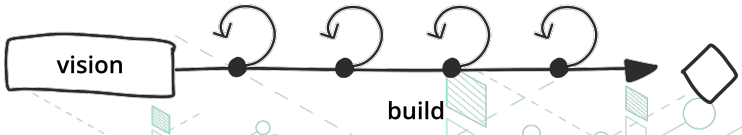
Defining a guiding vision as very first step helps set the right path through an uncertain environment.

The company needs to define a clear and achievable vision, which will later be translated into specific executable steps.

In this context:

The combination of limited experience and lack of extra time and flexibility for research leads to pursuing CN implementation using “well known ways”.

- Without an overall consistent vision, different teams will make independent and, frequently, conflicting architectural decisions
- In many companies, Enterprise Architects are responsible for creating a detailed architecture. Many Enterprise Architects lack sufficient theoretical or practical experience in the Cloud Native approach.
- Agile methodologies, widely adopted in the contemporary business world, create pressure to produce results early and onboard teams to new systems very quickly.



Therefore:

You should define and visualize the architecture of the whole system upfront.

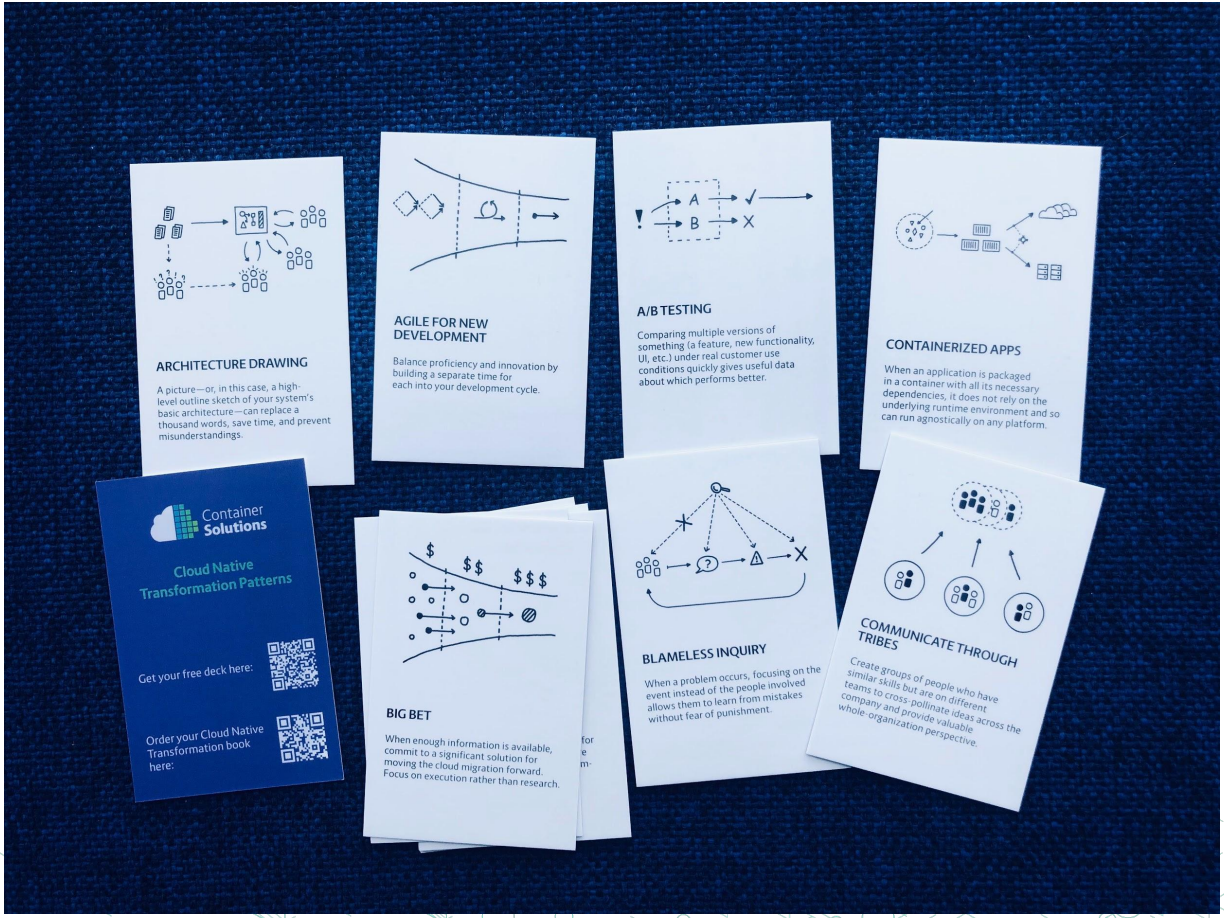
It can either be requested from external sources or uncovered by series of small research and prototyping projects ranging from few hours to few days each.

It’s important to keep the vision high level to allow freedom of choice during implementation, yet also detailed enough to provide clear guidance (which will help avoid common pitfalls).

Consequently:

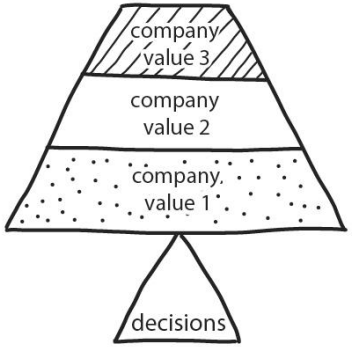
All teams have a clear guiding principle for the implementation phase.

The teams can start producing the lower level architecture, and translate it to the backlog of tasks. Therefore, Executive Commitment paired with leadership by the Transformation Champion are essential to have in place for successful vision creation.


ARCHITECTURE DRAWING
A picture—or, in this case, a high-level outline sketch of your system's basic architecture—can replace a thousand words, save time, and prevent misunderstandings.
AGILE FOR NEW DEVELOPMENT
Balance proficiency and innovation by building a separate time for each into your development cycle.
A/B TESTING
Comparing multiple versions of something (a feature, new functionality, UI, etc.) under real customer use conditions quickly gives useful data about which performs better.
CONTAINERIZED APPS
When an application is packaged in a container with all its necessary dependencies, it does not rely on the underlying runtime environment and so can run agnostically on any platform.
Container Solutions
Cloud Native Transformation Patterns
Get your free deck here:
Order your Cloud Native Transformation book here:
BIG BET
When enough information is available, commit to a significant solution for moving the cloud migration forward. Focus on execution rather than research.
BLAMELESS INQUIRY
When a problem occurs, focusing on the event instead of the people involved allows them to learn from mistakes without fear of punishment.
COMMUNICATE THROUGH TRIBES
Create groups of people who have similar skills but are on different teams to cross-pollinate ideas across the company and provide valuable whole-organization perspective.

Patterns for Strategy & Risk Reduction

- Dynamic Strategy
- Value Hierarchy
- Business Case
- Executive Commitment
- Transformation Champion
- Vision First
- Objective Setting
- Periodic Checkups
- Data-Driven Decision Making
- Involve the business
- Three horizons
- Learning Loop
- Learning Organization
- Measure What Matters
- Research Through Action
- Gradually Raising the Stakes
- No Regret Moves
- Options and Hedges
- Big Bet
- Reduce Cost of Experimentation
- Exit Strategy Over Vendor Lock-in
- Reflective breaks



VALUE HIERARCHY

When an organization's values are clearly stated and prioritized, day-to-day decisions can be made without seeking consent or permission/approval.

BUSINESS CASE

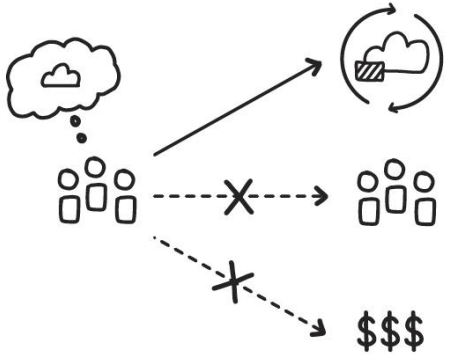
Before launching a Cloud Native transformation, an enterprise's leadership must make sure the initiative is needed and that the benefits will justify the investment.

EXECUTIVE COMMITMENT

To ensure allocation of sufficient resources and reasonable delivery time frames, large scale projects such as Cloud Native transformation require strong commitment from the top executive team.

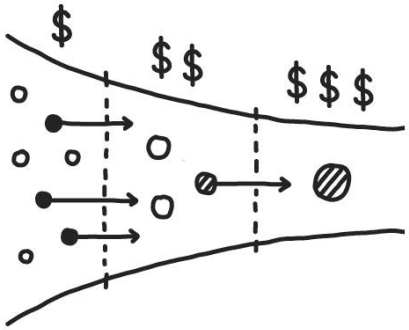
DYNAMIC STRATEGY

Today's tech-driven marketplace is a constantly shifting environment, no matter what business you are in—so your game plan needs to shift right along with it.



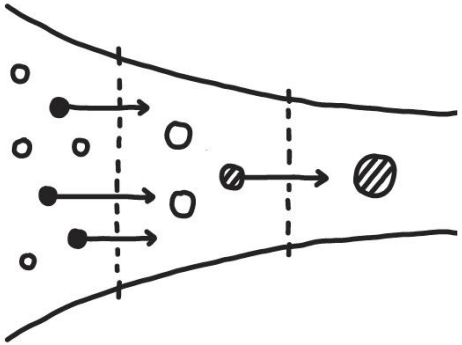
MEASURE WHAT MATTERS

People optimize their actions based on how their work is measured. Assessing the wrong things leads people to optimize for the
wrong goals.



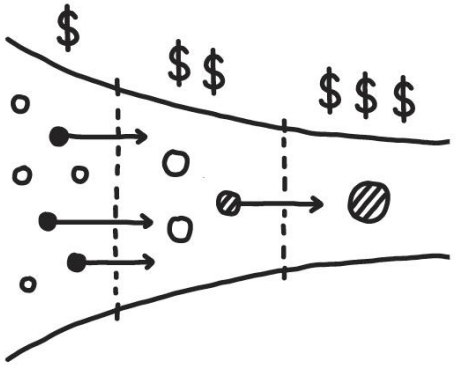
BIG BET

When enough information is available, commit to a significant solution for moving the cloud migration forward. Focus on execution rather than research.



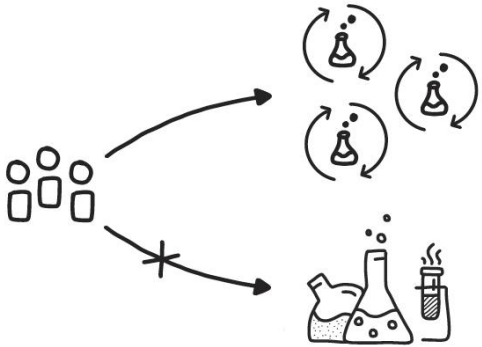
OPTIONS & HEDGES

Research has created deeper understanding and a few potentially promising transformation paths have begun to emerge. Continue reducing the risk by focusing on the most promising options and developing them further.



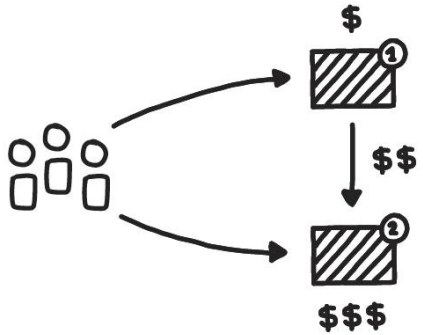
NO REGRET MOVES

Small, quick actions that require little investment of time and money but increase knowledge, reduce risk, and benefit the entire organization—inside or outside of a transformation scenario.



REDUCE COST OF EXPERIMENTATION

When someone has an idea that requires validation, the costs of doing experiments around it needs to be as low as possible.

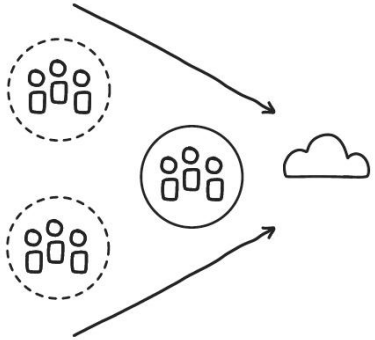


EXIT STRATEGY OVER VENDOR LOCK-IN

Public cloud vendors can handle all aspects of building and operating a CN platform, and their tools are often excellent—but when committing to a vendor/technology/platform it's important to identify an alternate solution and any costs associated with switching over.

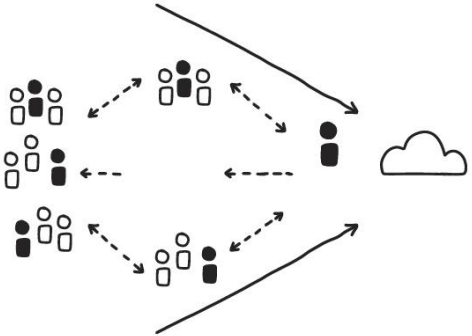
Patterns for Organization & Culture

- Core Team
- Build-Run Teams (“CN DevOps”)
- Platform Team
- SRE Team
- Remote Teams
- Co-Located Teams
- Communicate Through Tribes
- Manage for Creativity
- Manage for Proficiency
- Strangle Monolithic Organization
- Gradual Onboarding
- Design Thinking for Radical Innovation
- Designated strategist
- MVP platform
- Agile for New Development
- Lean for Optimization
- Internal Evangelism
- Ongoing Education
- Exploratory Experiments
- Proof of Concent (POC)
- Decide Closest to the Action
- Productive Feedback
- Psychological Safety
- Personalized Relationships for Co-Creation
- Blameless Inquiry
- Productive feedback



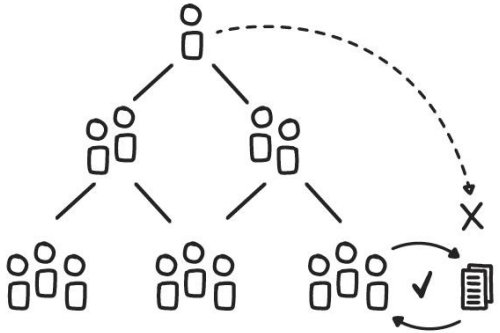
CORE TEAM

Dedicate a team of engineers and architects to the task of uncovering the best transformation path and implementing it along the way
This reduces risk embedded in the transformation while the team gains experience helpful for onboarding the remaining teams later.



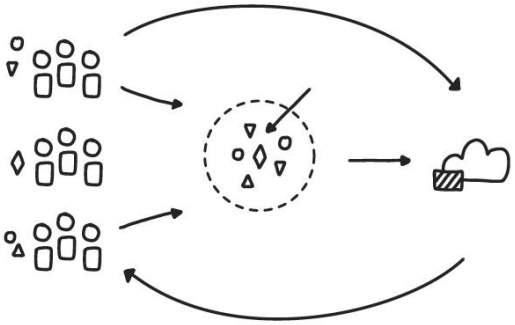
INTERNAL EVANGELISM

Provide plenty of information about the transformation across the entire company right from the start to create understanding, acceptance of, and support for the initiative.



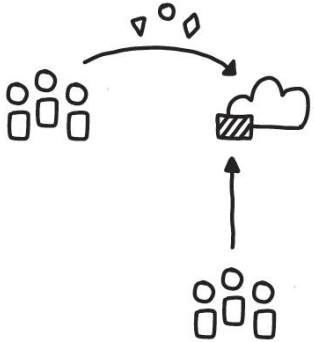
DECIDE CLOSEST TO THE ACTION

Those nearest to a change action get the first chance to make any decisions related to it.



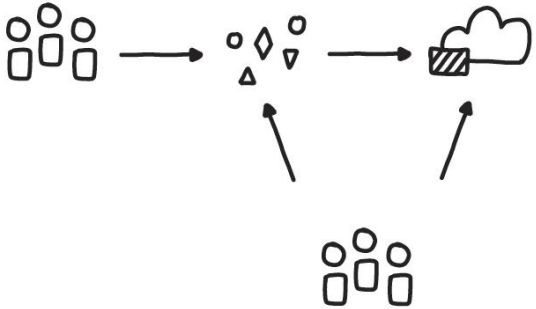
BUILD-RUN TEAMS

Dev teams have full authority over the services they build, not only creating but also deploying and supporting them.



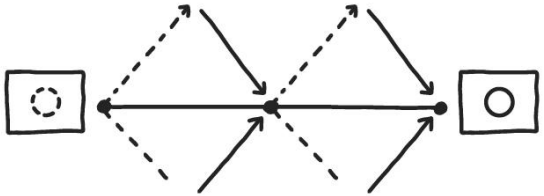
PLATFORM TEAM

Create a team to be in charge of architecting, building, and running a single, consistent, and stable CN platform for use by the entire organization so that developers can focus on building applications instead of configuring infrastructure.



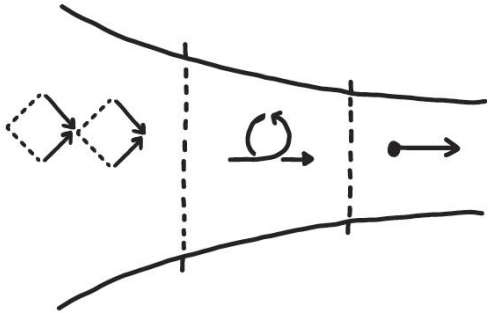
SRE TEAM

The SRE (Site Reliability Engineering) team helps the development teams to maintain and improve the application (not the platform or infrastructure).



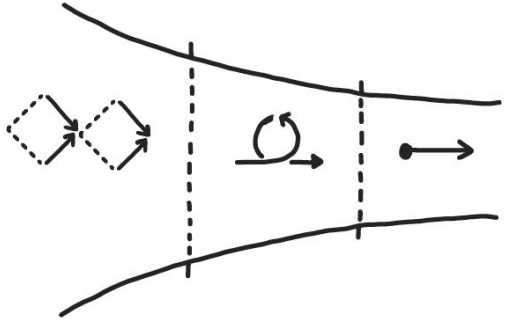
DESIGN THINKING FOR RADICAL INNOVATION

Whether faced with a radical new idea or a big problem, Design Thinking can be used as a process for first brainstorming a robust list of solutions and then narrowing it down to the best possibilities for actual exploration.



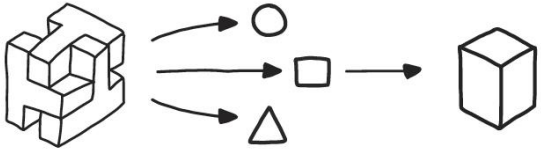
AGILE FOR NEW DEVELOPMENT

Balance proficiency and innovation by building a separate time for each into your development cycle.



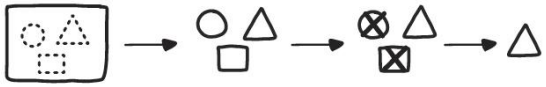
LEAN FOR OPTIMIZATION

When a stable system delivers the value that's intended and is not a target for technical innovation, focus on improving the system by continuously and incrementally improving delivery and maintenance processes with emphasis on repeatability.



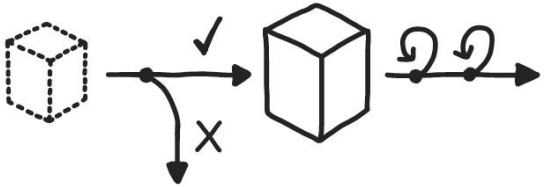
EXPLORATORY EXPERIMENTS

When dealing with a complex problem with no obvious available solution, run a series of small experiments to evaluate the possible alternatives and learn by doing.



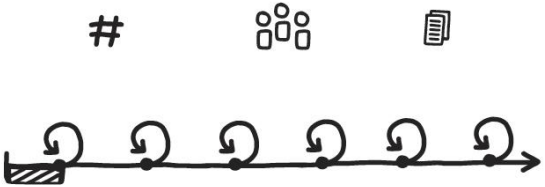
PROOF OF CONCEPT (POC)

Before fully committing to a solution that can significantly affect the future, build a small prototype to demonstrate viability and gain better understanding.



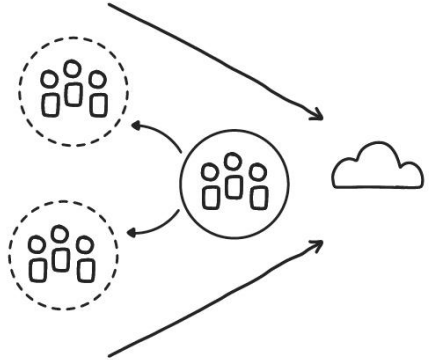
MVP PLATFORM

Once Exploratory Experiments and PoCs have uncovered a probable path to success, build a simple version of a basic but fully functional and production-ready platform with one to three small applications running on it in production.



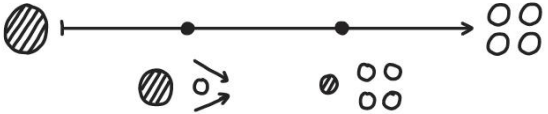
ONGOING EDUCATION

Continuously introduce new ways and improve existing ones to help teams continually develop their CN knowledge and skills.



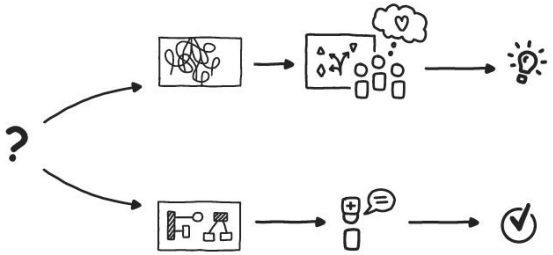
GRADUAL ONBOARDING

One to three months before the new platform goes live, begin training a couple of teams at a time with a pause between each cohort to incorporate feedback and improve the process/ materials.



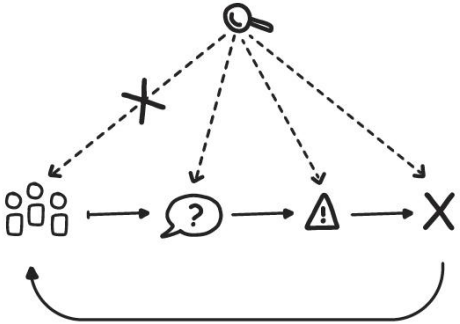
STRANGLE MONOLITHIC ORGANIZATION

Just as the new tools, technologies, and infrastructure gradually roll out over the course of a transformation initiative, the organization and its teams must also evolve to work with them properly.



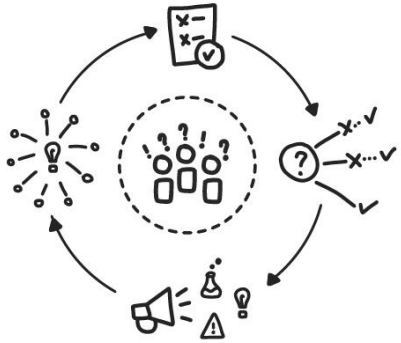
PERSONALIZED RELATIONSHIPS FOR CO-CREATION

Solutions to complex problems are best created collaboratively by teams with high levels of interpersonal connection.



BLAMELESS INQUIRY

When a problem occurs, focusing on the event instead of the people involved allows them to learn from mistakes without fear of punishment.

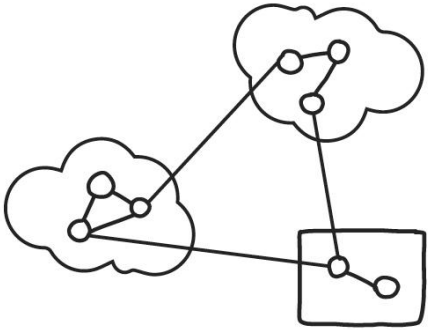


PSYCHOLOGICAL SAFETY

When team members feel they can speak up, express concern, and make mistakes without facing punishment or ridicule, they can think freely and creatively, and are open to taking risks.

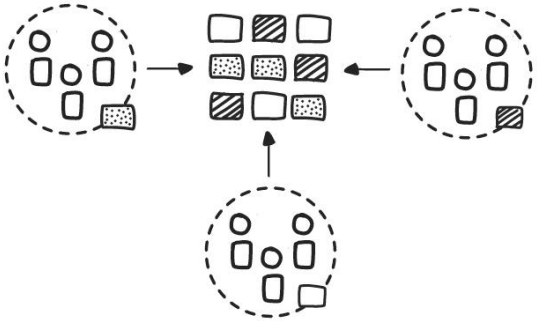
Patterns for Development & Process

- Open Source Internal Projects
- Distributed Systems
- Automated Testing
- Continuous Integration
- Reproducible Dev Environments
- No Long Tests in CI/CD
- Microservices Architecture
- Architecture drawing
- Secure system from the start
- Communicate Through APIs
- Reference Architecture
- Developer Starter Pack
- Demo Applications
- Strangle Monolithic Application
- Delayed Automation
- Avoid Reinventing the Wheel
- A/B Testing
- Serverless



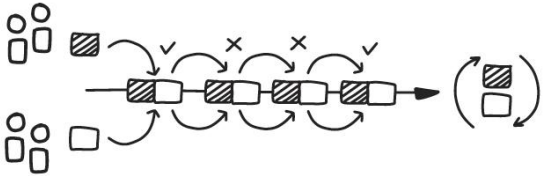
DISTRIBUTED SYSTEMS

When software is built as a series of fully independent, loosely coupled services,
the resulting system is, by design, fast, resilient, and highly scalable.



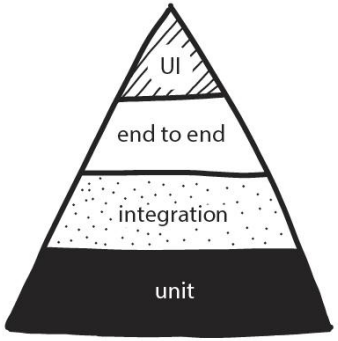
MICROSERVICES ARCHITECTURE

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.



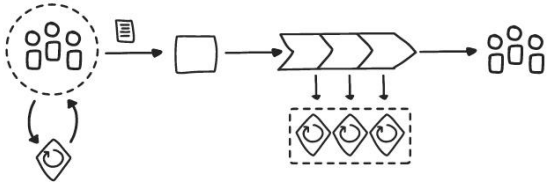
CONTINUOUS INTEGRATION

Frequent integration of small iterative changes speeds overall delivery and improves the quality of the code.



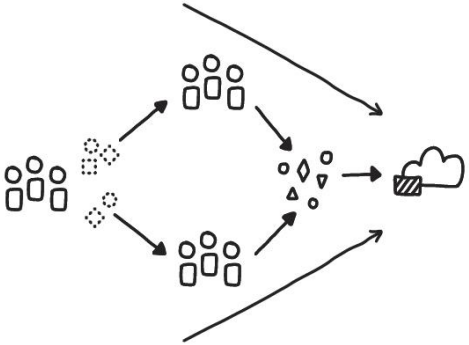
AUTOMATED TESTING

Shift responsibility for testing from humans (manual) to an automated testing framework so the quality of the released products is consistent and continuously improving, allowing developers to deliver faster while spending more of their time improving features to meet customer needs.



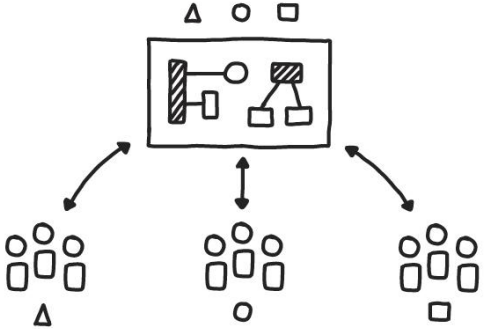
REPRODUCIBLE DEV ENVIRONMENTS

Developers need to test their daily work in an environment that is easy to spin up and that matches production tooling as closely as possible.



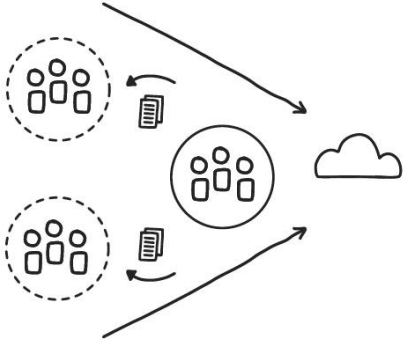
DEMO APPLICATIONS

Teams onboarded to the new CN system receive demo applications as an educational starting point for building their own CN applications.



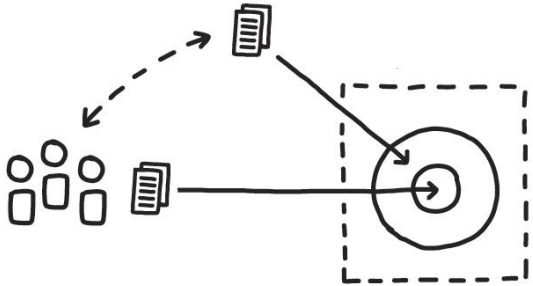
REFERENCE ARCHITECTURE

Provide an easily accessible document laying out a standardized system architecture for all teams to use for building their applications/components. This ensures higher architectural consistency and lowers development costs via better reusability.



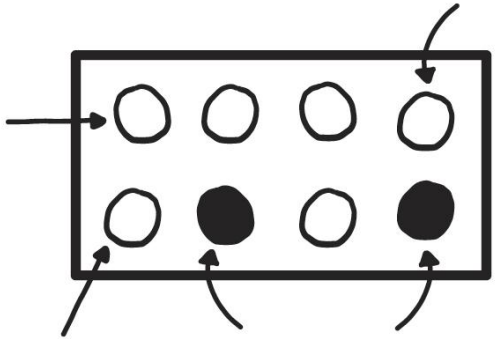
DEVELOPER STARTER PACK

Provide a “starter kit” of materials, guides, and other resources to help new teams onboard to the new cloud native system quickly and with confidence.



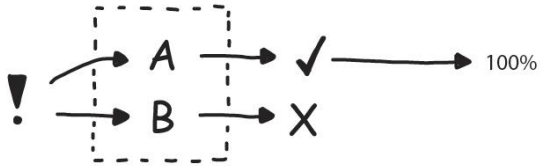
OPEN SOURCE INTERNAL PROJECTS

Use open source solutions for any software need that is not directly related to the company's core business value.



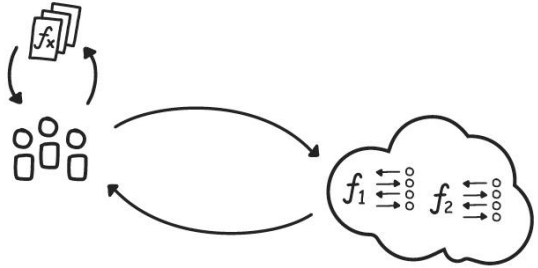
AVOID REINVENTING THE WHEEL

When possible, purchase solutions for any need that is not your actual core business instead of trying to custom-build perfect tools.



A/B TESTING

Comparing multiple versions of something (a feature, new functionality, UI, etc.) under real customer use conditions quickly gives useful data about which performs better.



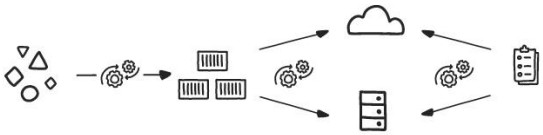
SERVERLESS

The soon-to-arrive future is event-driven, instantaneously scalable services (functions) on the cloud.

Patterns for Infrastructure & Cloud

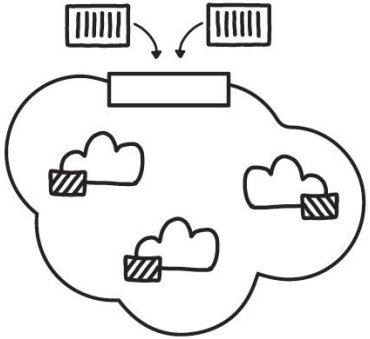
- Private Cloud
- Automated Infrastructure
- Self Service
- Dynamic Scheduling
- Containerized Apps
- Observability
- Public cloud

- Continuous Delivery
- Continuous Deployment
- Full Production Readiness
- Risk-Reducing Deployment Strategies
- Lift and Shift at the End



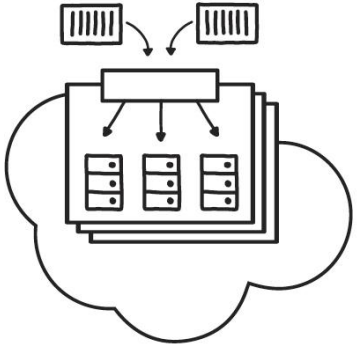
AUTOMATED INFRASTRUCTURE

The absolute majority of operational tasks need to be automated. Automation reduces interteam dependencies, which allows faster experimentation and leads in turn to faster development.



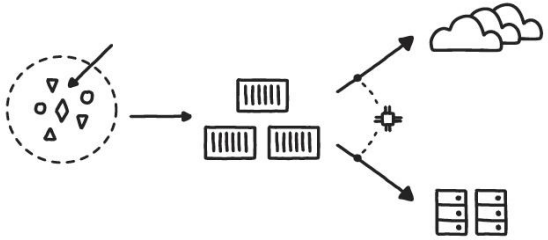
PUBLIC CLOUD

When someone has an idea that requires validation, the costs of doing experiments around it needs to be as low as possible.



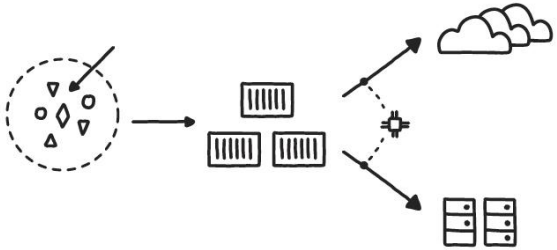
PRIVATE CLOUD

A private cloud approach, operated either over the internet or on company-owned on-premises infrastructure, can offer the benefits of cloud computing services like AWS while restricting access to only select users.



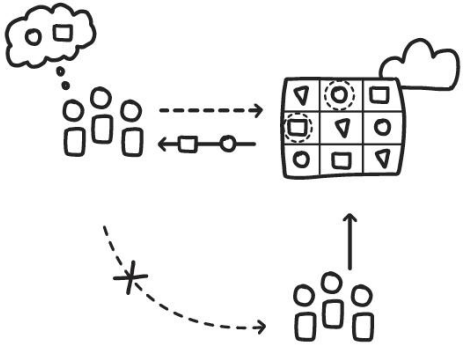
CONTAINERIZED APPS

When an application is packaged in a container with all its necessary dependencies, it does not rely on the underlying runtime environment and so can run agnostically on any platform.



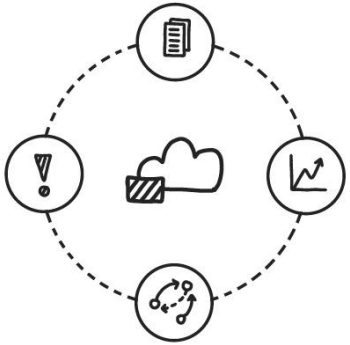
DYNAMIC SCHEDULING

An orchestrator (typically Kubernetes) is needed to organize the deployment and management of microservices in a distributed container-based application to assign them across random machines
at the instant of execution.



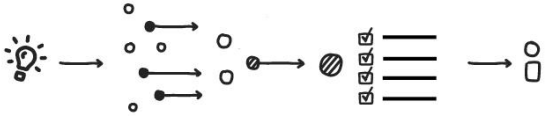
SELF SERVICE

In cloud native everyone can do their own provisioning and deployment with no handoffs between teams.



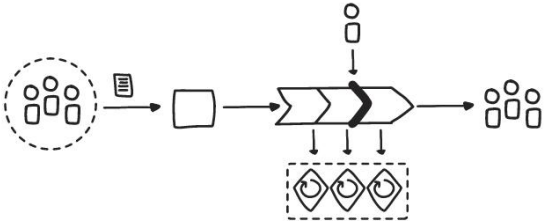
OBSERVABILITY

CN distributed systems require constant insight into the behavior of all running services in order to understand the system's behavior and to predict potential problems or incidents.



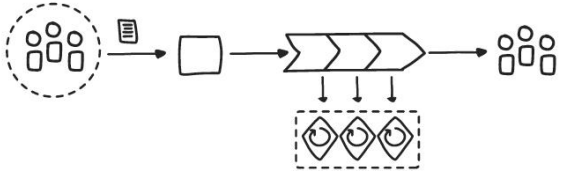
FULL PRODUCTION READINESS

Make sure your platform is fully provisioned with CI/CD, security, monitoring, observability, and other features essential to production readiness before you try to take it live.



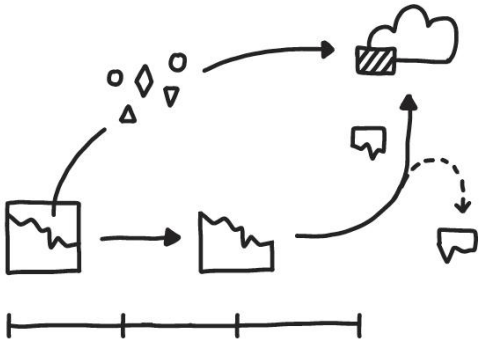
CONTINUOUS DELIVERY

Keeping a short build/test/deliver cycle means code is always ready for production and features can be immediately released to customers and their feedback quickly returned to developers.



CONTINUOUS DEPLOYMENT

Continuous Deployment automatically and seamlessly pushes code that has been accepted in the Continuous Integration/Continuous Delivery cycle into the production environment.



LIFT AND SHIFT AT THE END

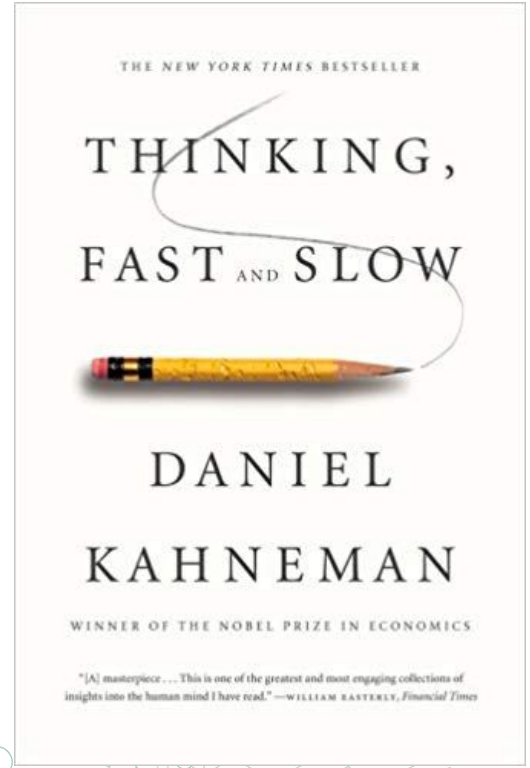
It's important not to approach a cloud native transformation by simply attempting a full "lift and shift" of your existing system onto the cloud. But it can be smart to move some intact pieces of it at the very end.

Cloud Native Pattern Language

Tool no. 3

Cognitive Biases

Cognitive Biases



Bandwagon effect

The tendency to do (or believe) things because many other people do (or believe) the same thing. Related to groupthink and herd behavior.

CN relationship: When Gartner puts certain tech on their chart, everyone decides to adopt it even without understanding how it relates to their use case.

Confirmation bias

The tendency to search for, interpret, focus on, and remember information in a way that confirms one's preconceptions.

CN relationship: Ignore all those inconvenient facts and embrace the information that supports your opinion.

Curse of knowledge

When better-informed people find it extremely difficult to think about problems from the perspective of less well-informed people.

CN relationship: Our clients' engineers frequently struggle to sell CN ideas to their managers due to this bias: they see so clearly why this is the right thing to do that they forget the managers have no background

Illusion of control

The tendency to overestimate one's degree of influence over other external events.

CN relationship: Engineers think that they know how to build microservices, managers think that they know what it takes to do DevOps. But in reality, it is only an illusion of control. Many complex and emergent processes are very difficult to even steer, much less control. Sometimes we need to embrace some uncertainty to ultimately get results.

Irrational escalation (sunk cost fallacy)

The phenomenon where people justify increased investment in a decision, based on the cumulative prior investment, despite new evidence suggesting that the decision was probably wrong.

CN relationship: If you've spent six months working on setting up an OpenShift cluster and bought all the licenses, it is very unlikely that you're going to switch to another tool even if it's proven to be superior.

Status quo bias

The tendency to like things to stay relatively the same.

CN relationship: Entire companies, and/or people within the organization, will resist moving to CN as due to this bias: everyone wants to remain comfortably right where they are right now, which is known and understood.

Hostile attribution bias

The "hostile attribution bias" is the tendency to interpret others' behaviors as having hostile intent, even when the behavior is ambiguous or benign.

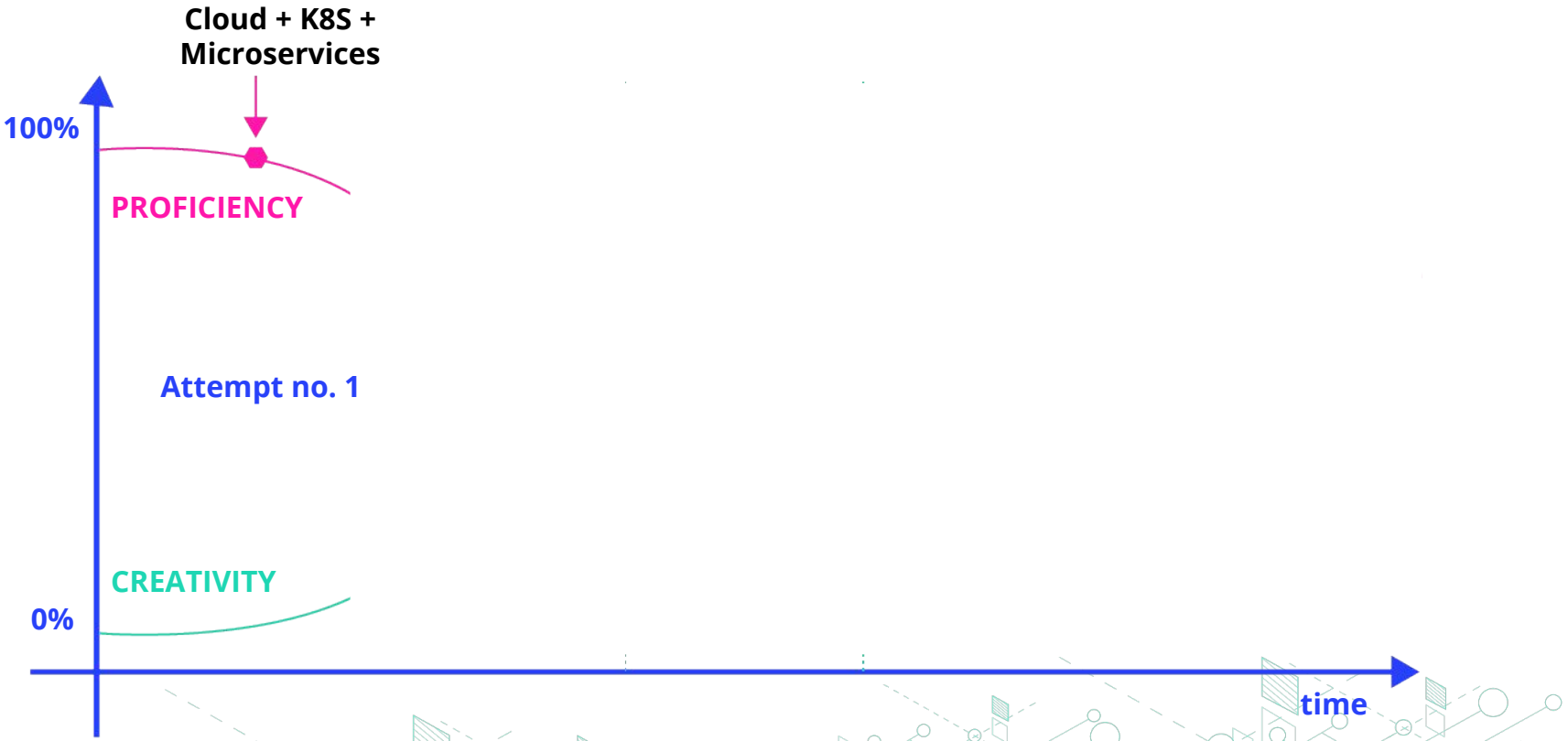
CN relationship: We meet people who think that we're there to destroy their work and wreck their company. We shouldn't think that this is their real opinion, as it is a normal human bias arising from the fact that change frequently creates anxiety in those poised to undergo it.

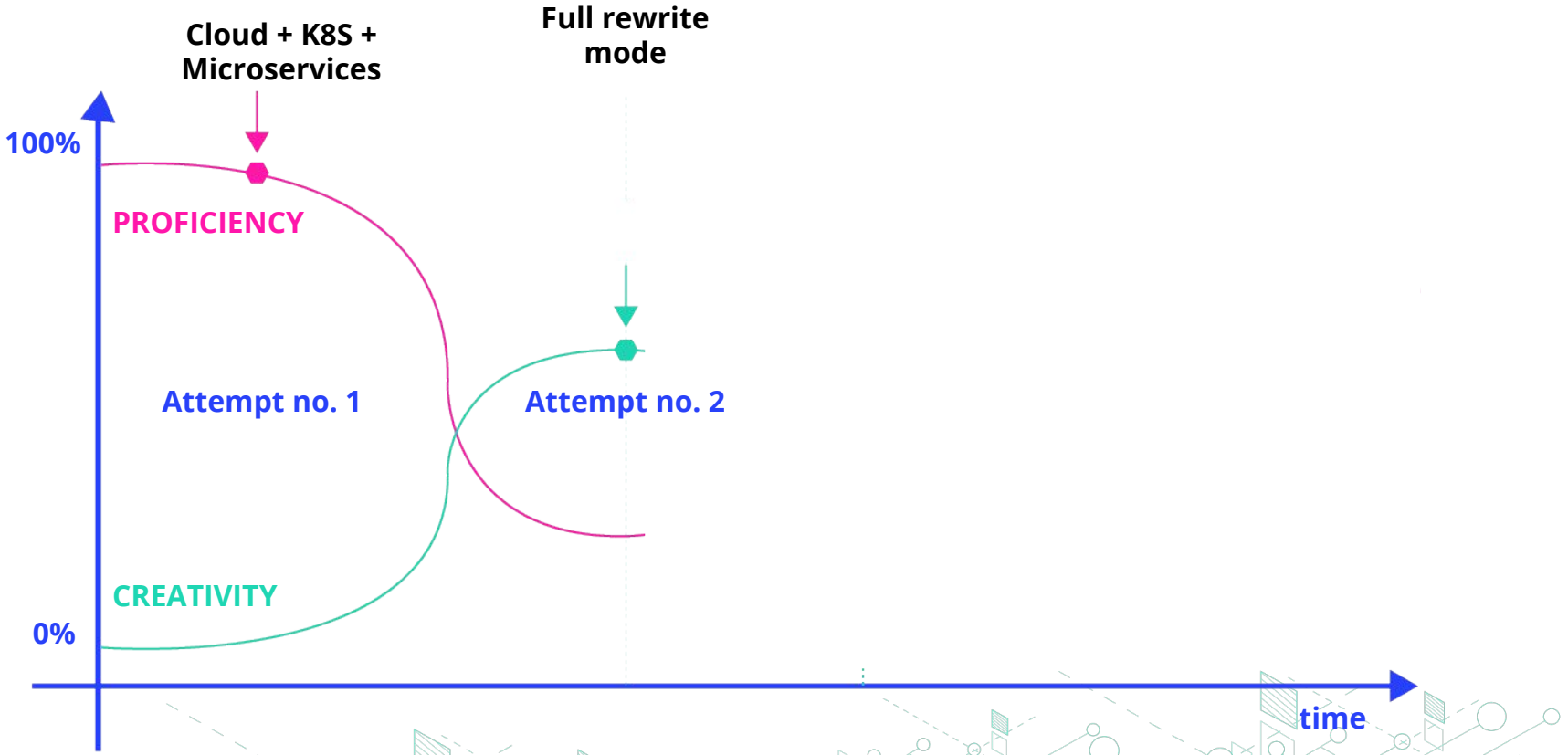
IKEA effect

The tendency for people to place a disproportionately high value on objects that they partially assembled themselves, such as furniture from IKEA, regardless of the quality of the end result.

CN relationship: This one can be used positively, as a nudge, and explains why we have to involve everyone in the client's organization in every stage of planning and executing a transformation. Whoever is involved is biased to like and support the solution.

What happened so far?

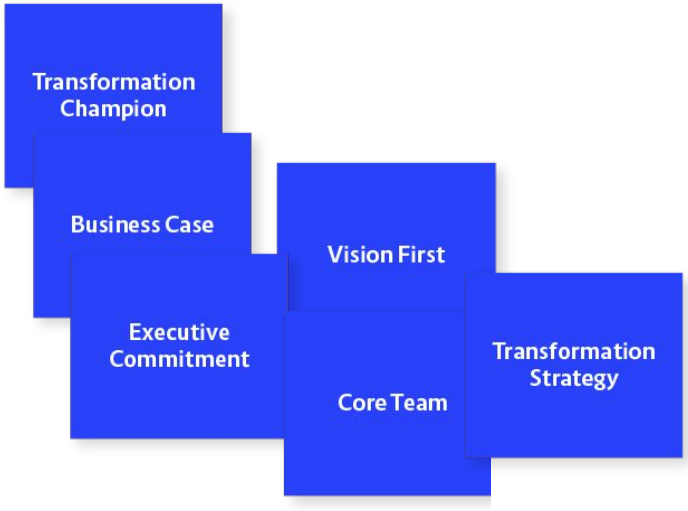

Cloud + K8S +
Microservices
100%
PROFICIENCY
Attempt no. 1
CREATIVITY
0%
time

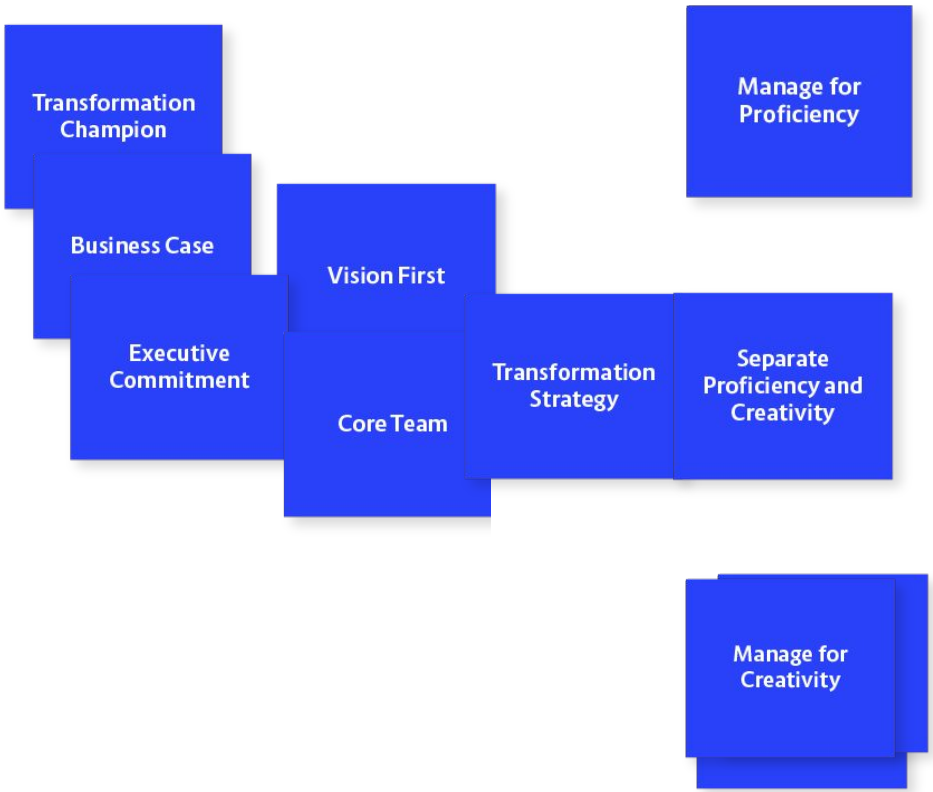

Cloud + K8S +
Microservices
Full rewrite
mode
100%
PROFICIENCY
Attempt no. 1
Attempt no. 2
CREATIVITY
0%
time

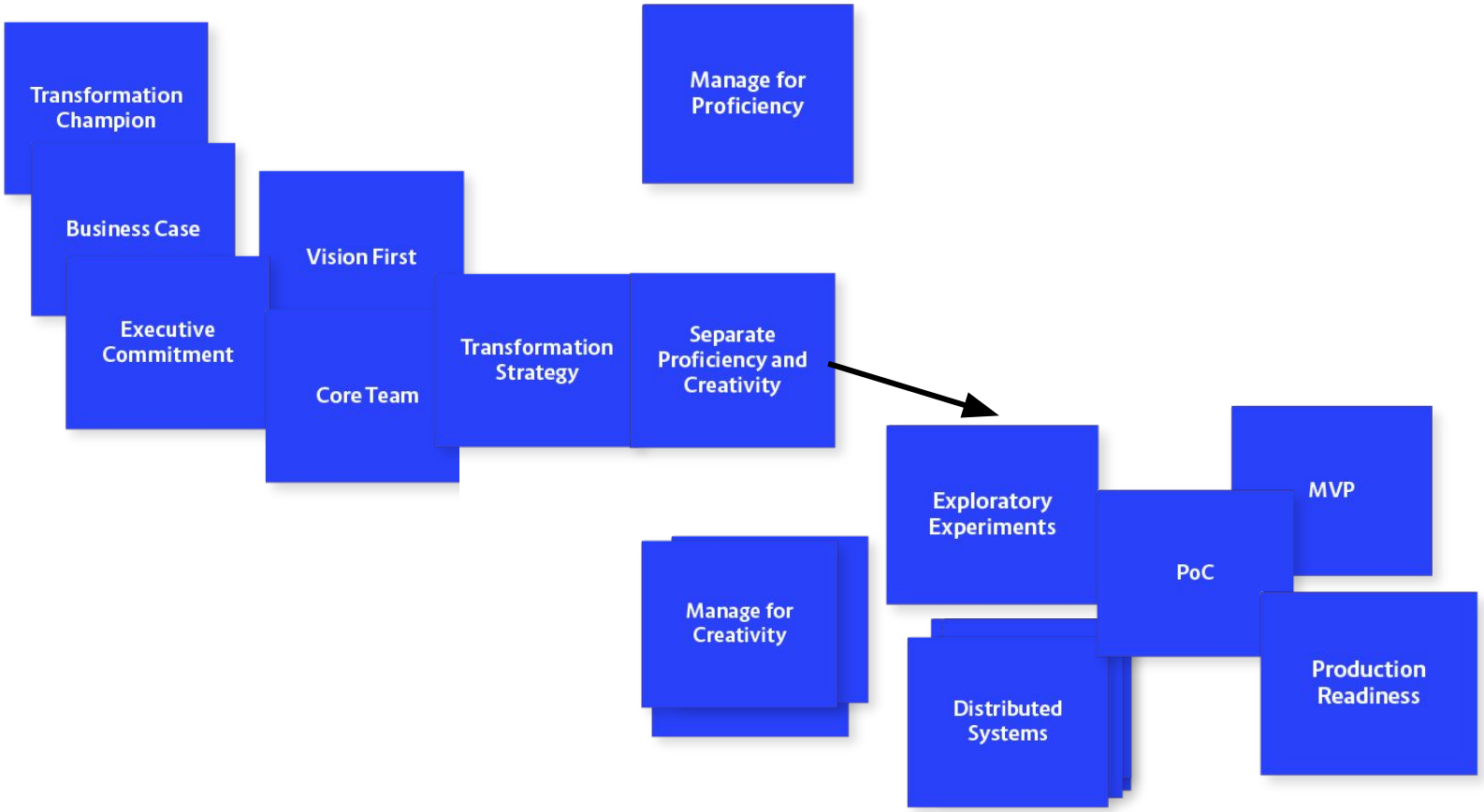
Design the Transformation

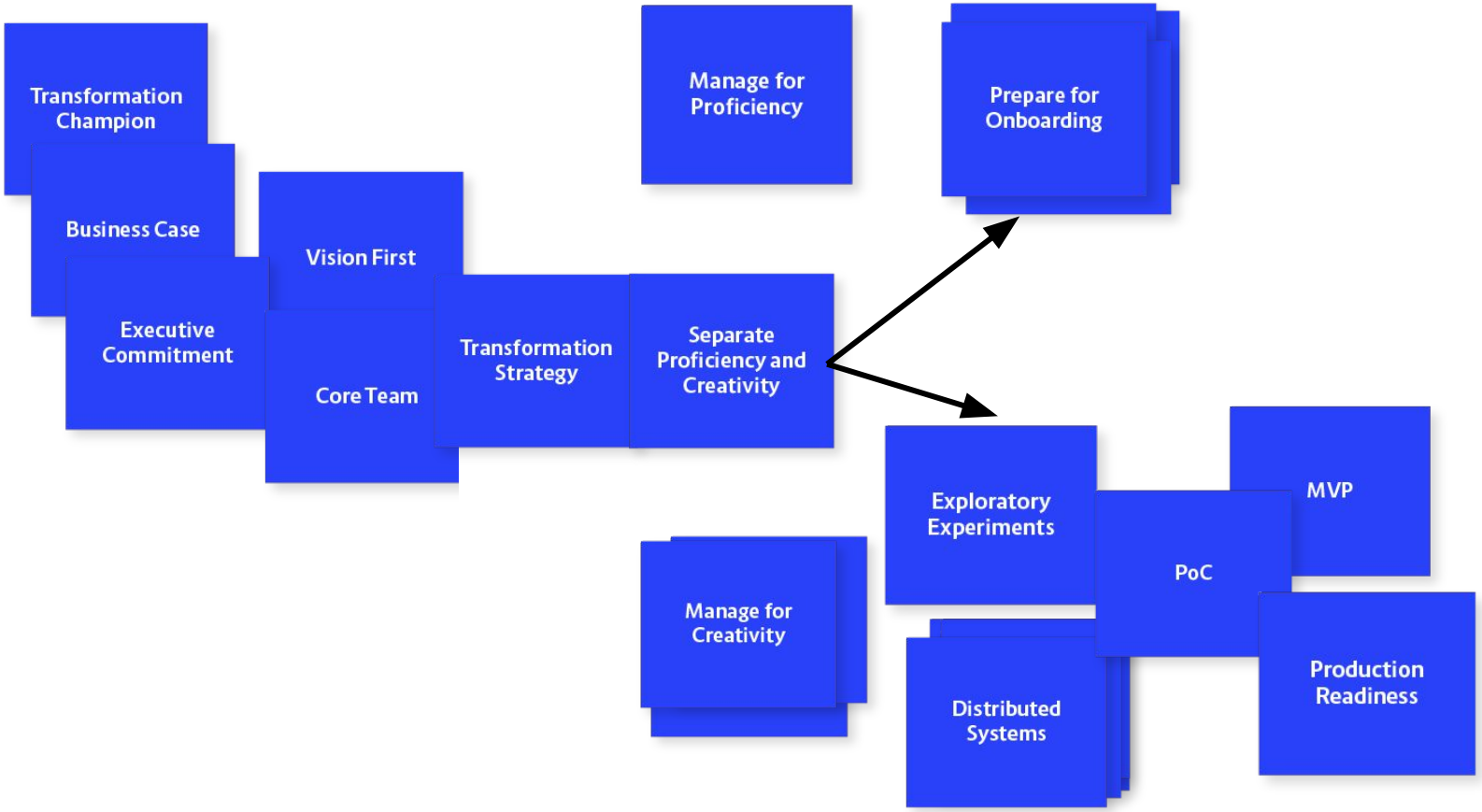
By using Cloud Native Patterns Language

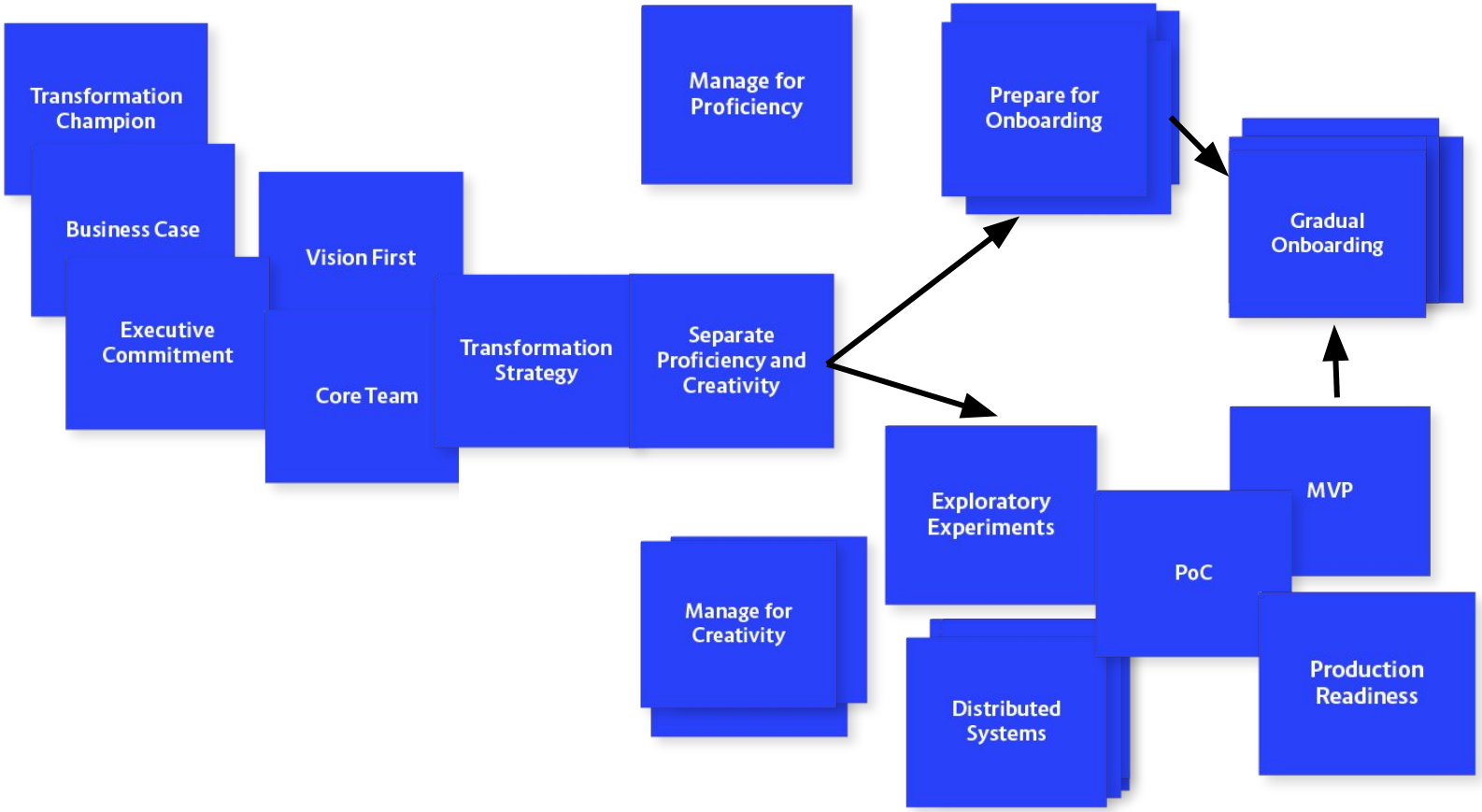

Transformation Champion
Business Case
Executive Commitment

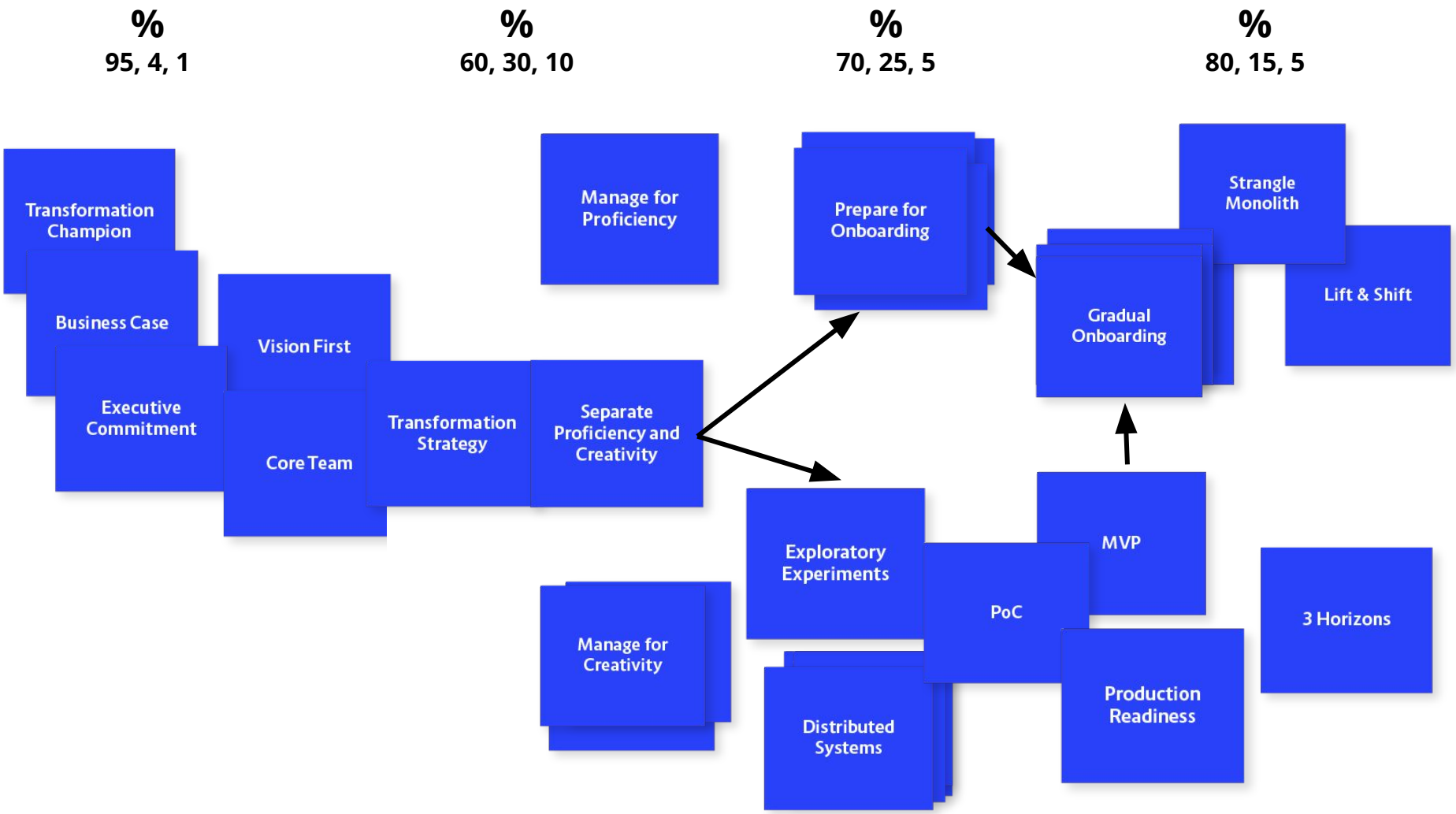

Transformation Champion
Business Case
Vision First
Executive Commitment
Core Team
Transformation Strategy


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


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


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

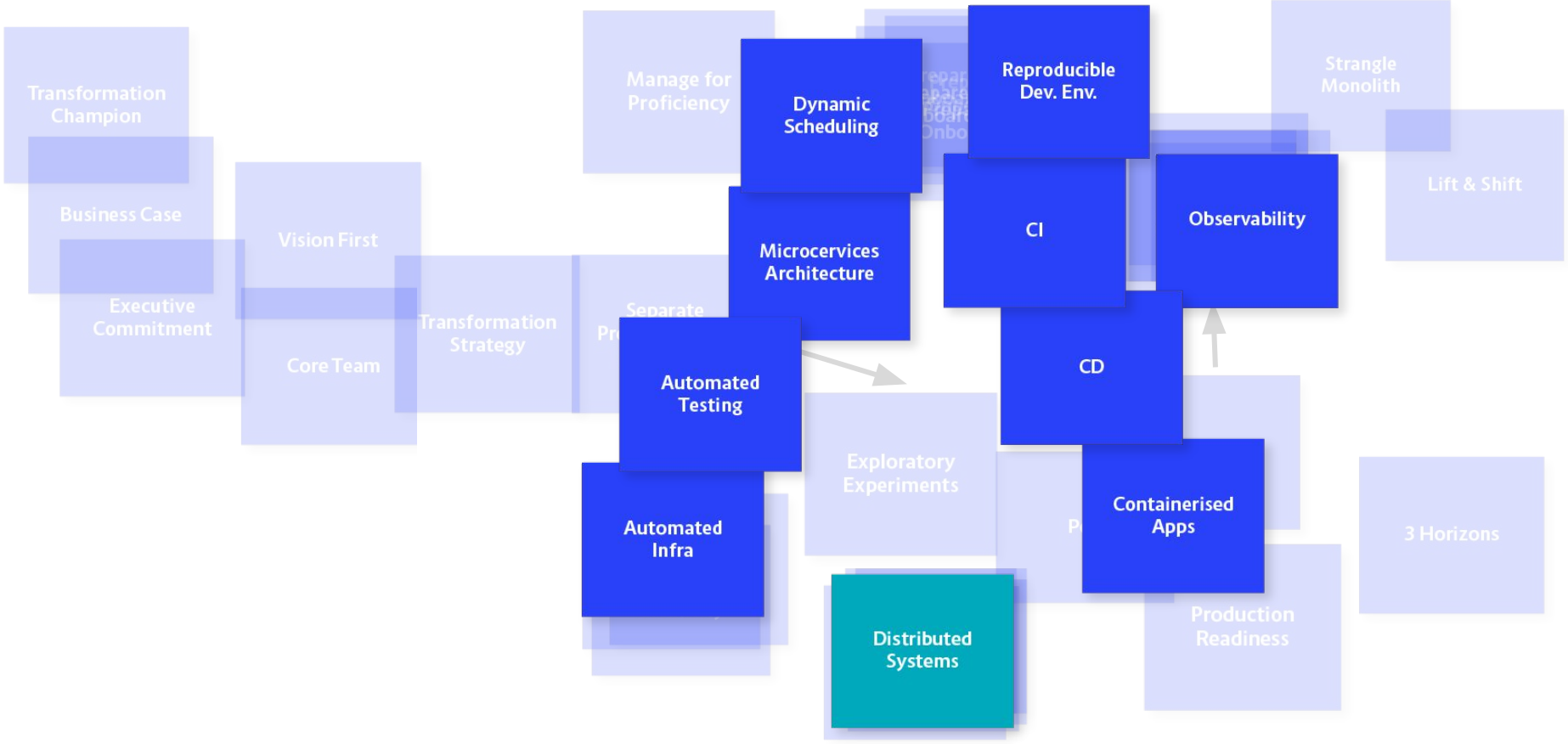

%
95, 4, 1
%
60, 30, 10
%
70, 25, 5
%
80, 15, 5
Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Separate Proficiency and Creativity
Manage for Proficiency
Prepare for Onboarding
Gradual Onboarding
Strangle Monolith
Lift & Shift
Manage for Creativity
Exploratory Experiments
PoC
MVP
Distributed Systems
Production Readiness
3 Horizons

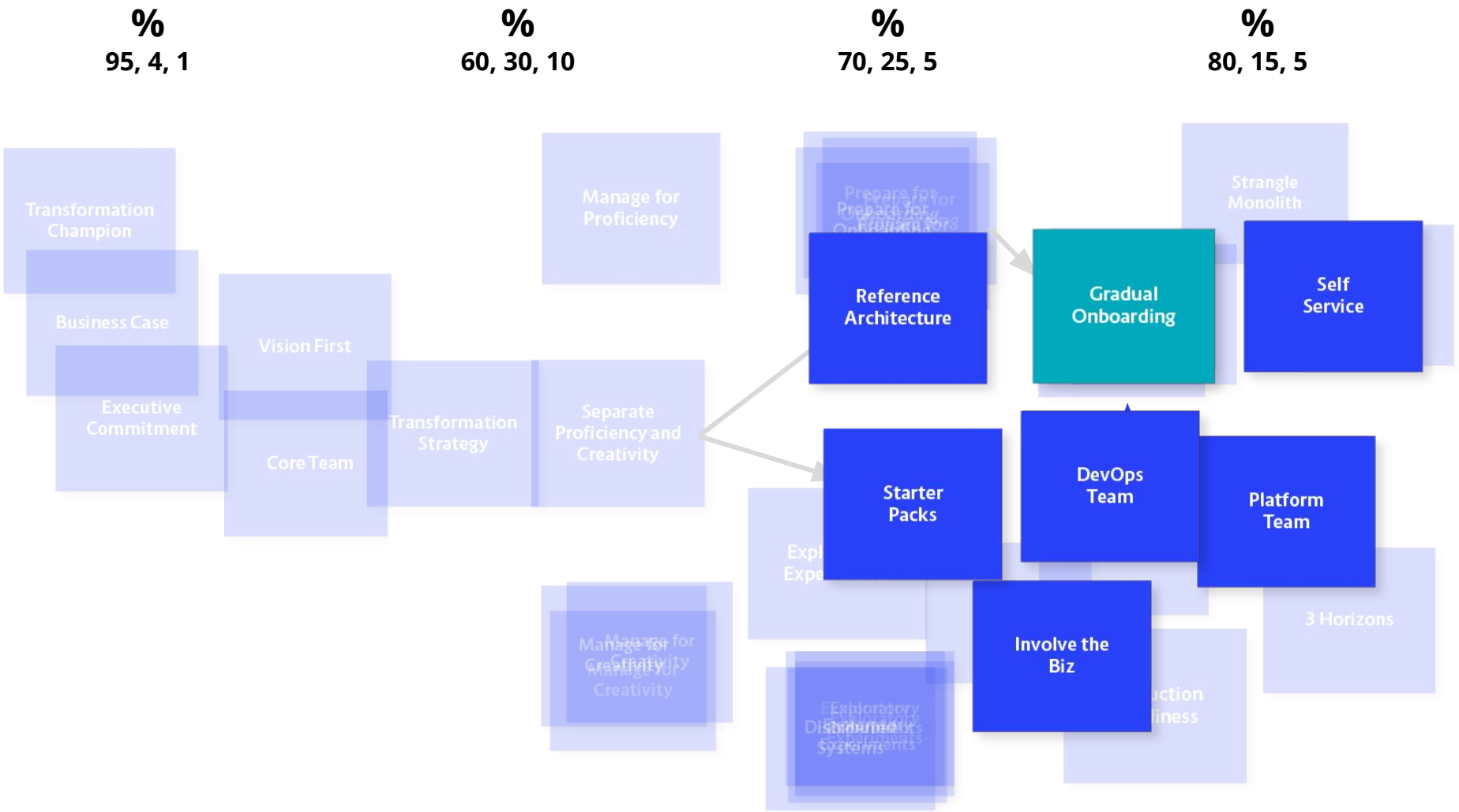
%
95, 4, 1

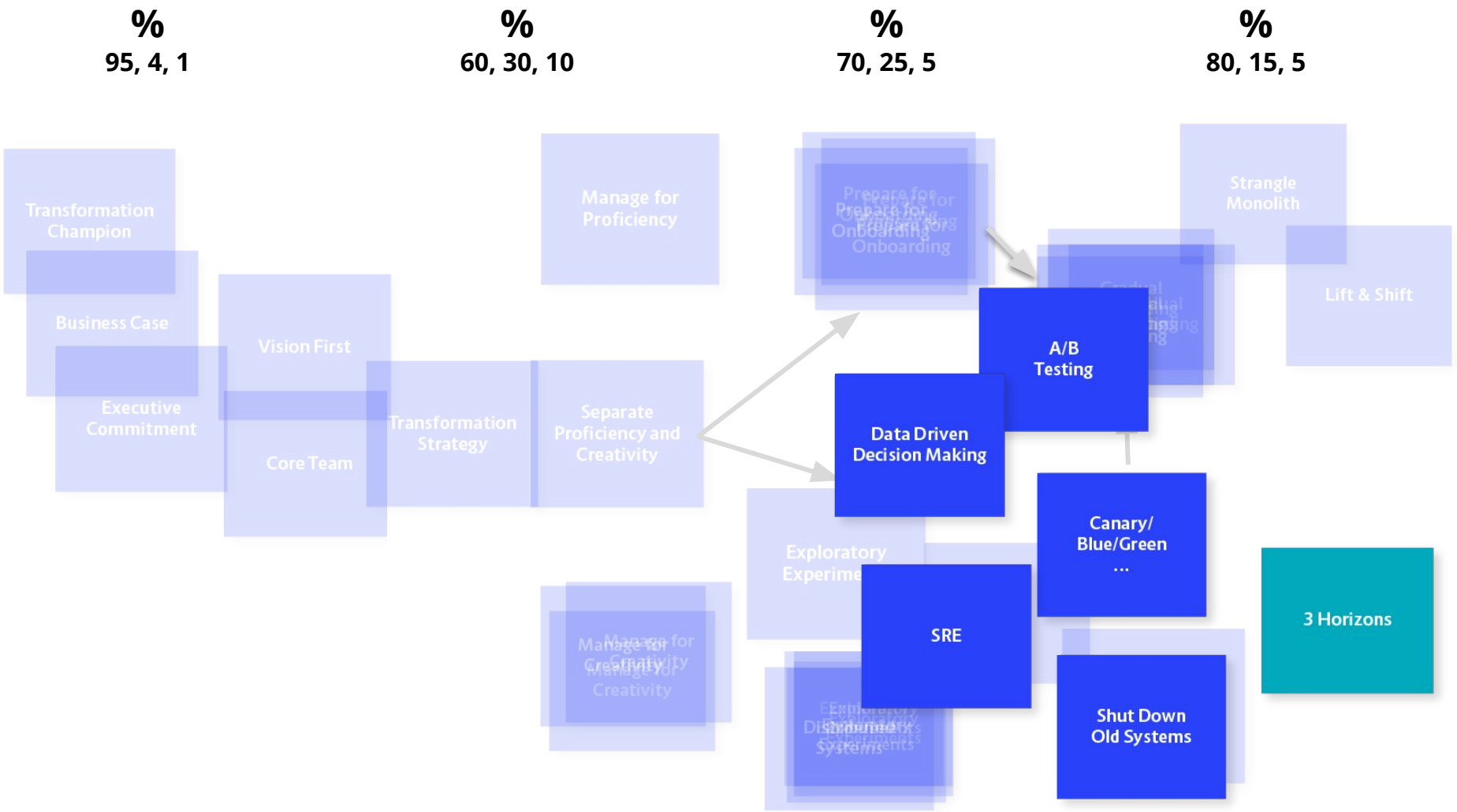
%
60, 30, 10

%
70, 25, 5

%
80, 15, 5



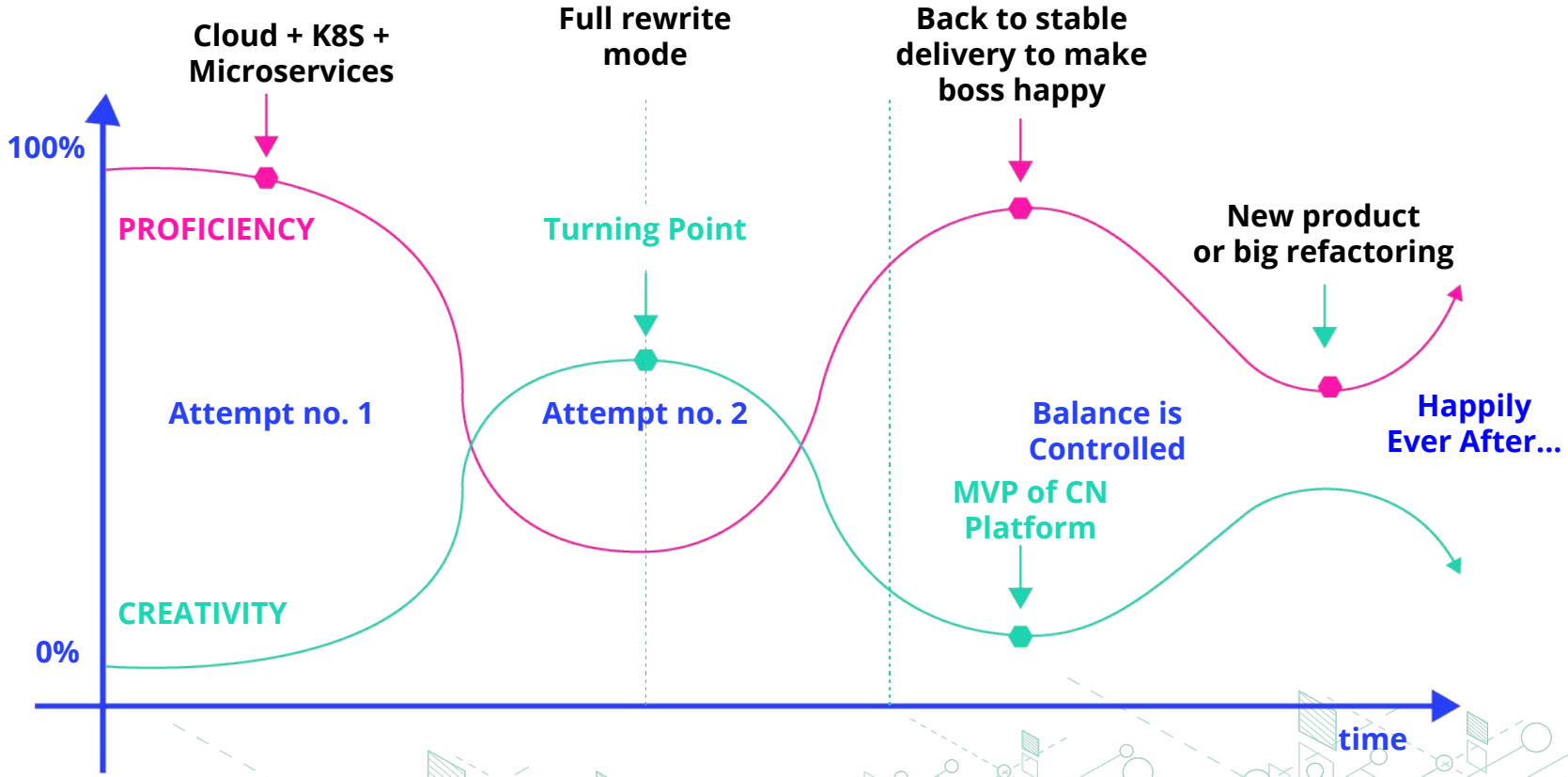

%
95, 4, 1
%
60, 30, 10
%
70, 25, 5
%
80, 15, 5
Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Manage for Proficiency
Separate Proficiency and Creativity
Reference Architecture
Gradual Onboarding
Strangle Monolith
Self Service
Starter Packs
DevOps Team
Platform Team
Involve the Biz
3 Horizons


%
95, 4, 1
%
60, 30, 10
%
70, 25, 5
%
80, 15, 5
Transformation Champion
Business Case
Executive Commitment
Vision First
Core Team
Transformation Strategy
Manage for Proficiency
Separate Proficiency and Creativity
Prepare for Onboarding
Strangle Monolith
Lift & Shift
A/B Testing
Data Driven Decision Making
Canary/ Blue/Green ...
Exploratory Experime
SRE
3 Horizons
Shut Down Old Systems

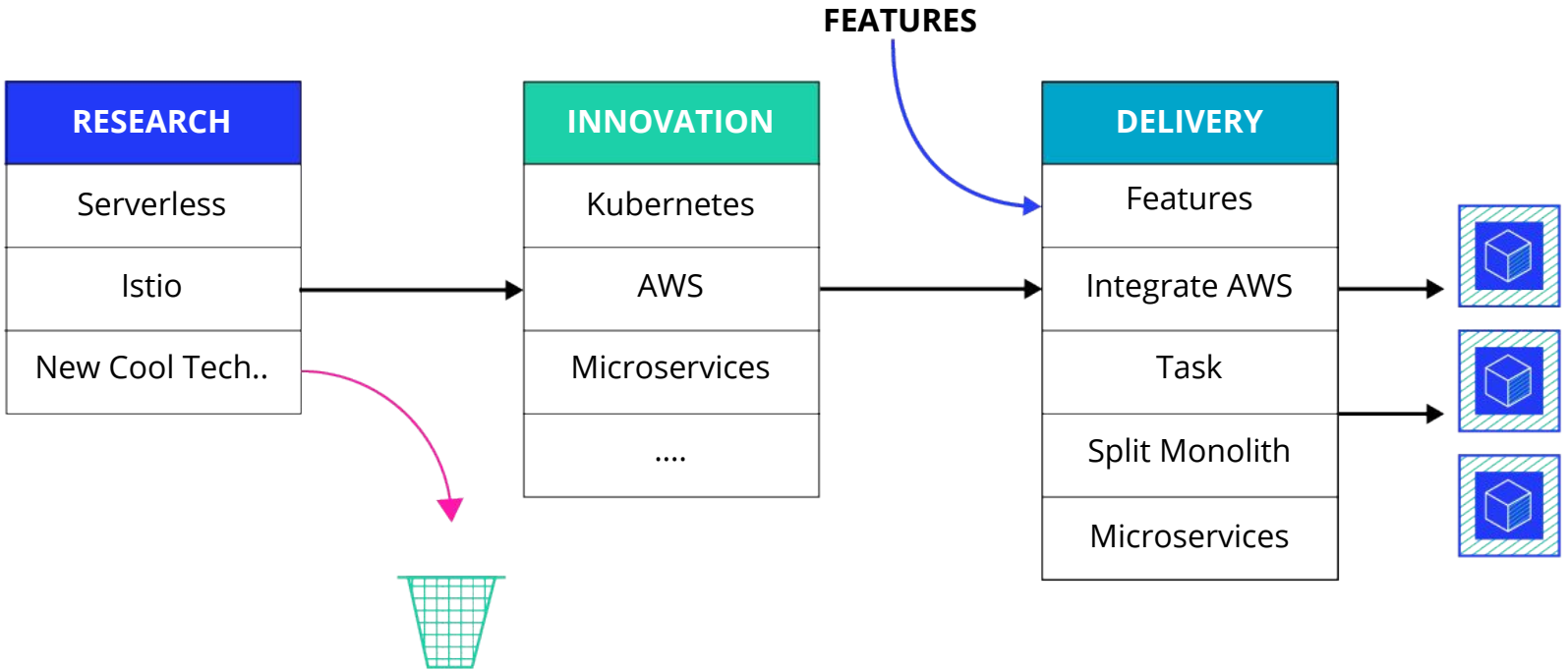
Culture Patterns

“Culture is a set of living relationships working toward a shared goal. It’s not something you are. It’s something you do.”

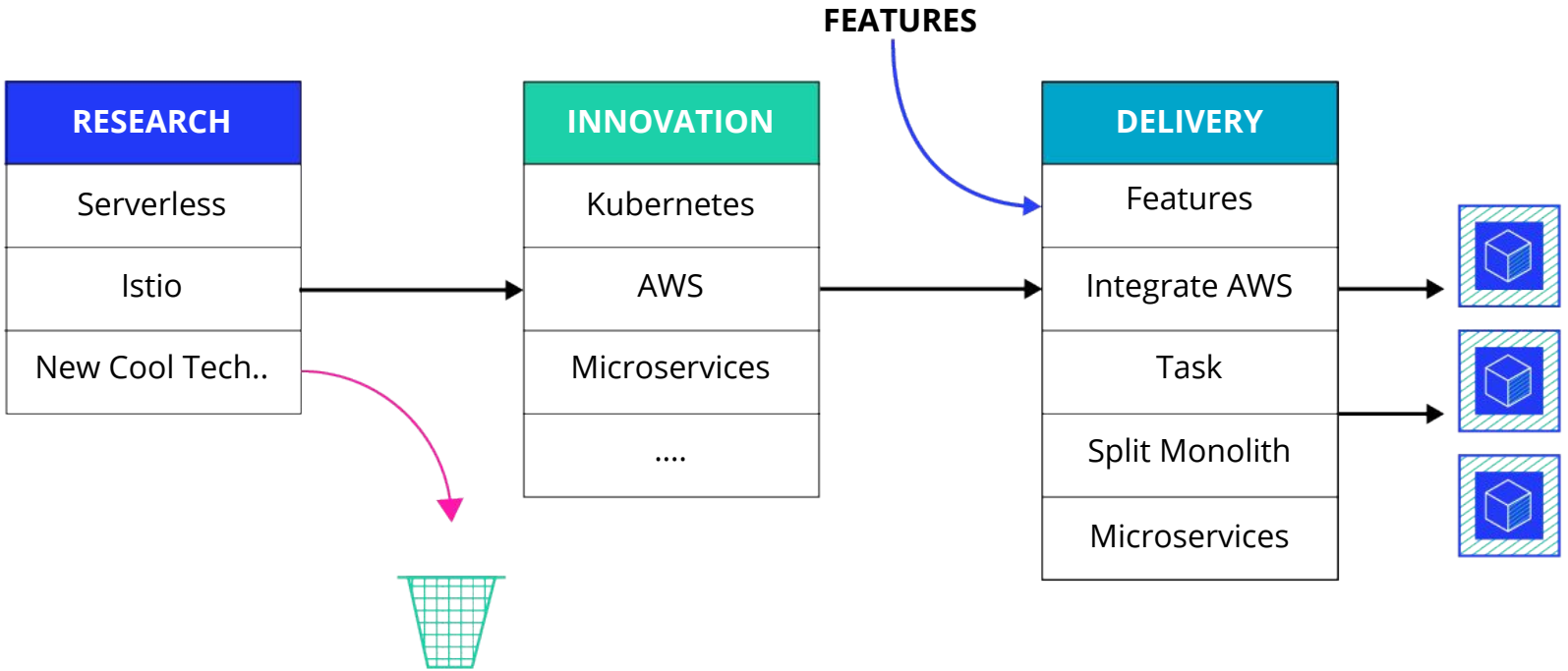
The Culture Code
Daniel Coyle


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

Cloud Native Innovation

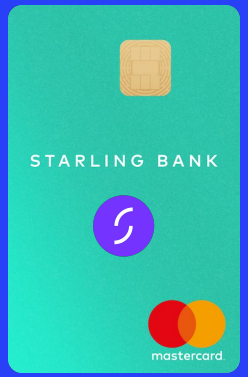


~~Cloud Native~~ Innovation

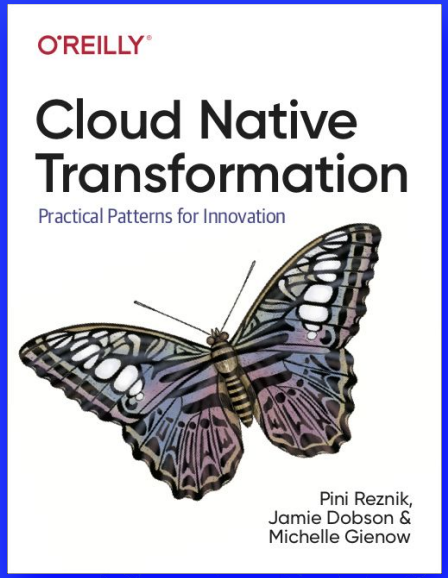


The Stranger is Coming...

... you are ready now!



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



www.container-solutions.com/learn-with-cs/cloud-native