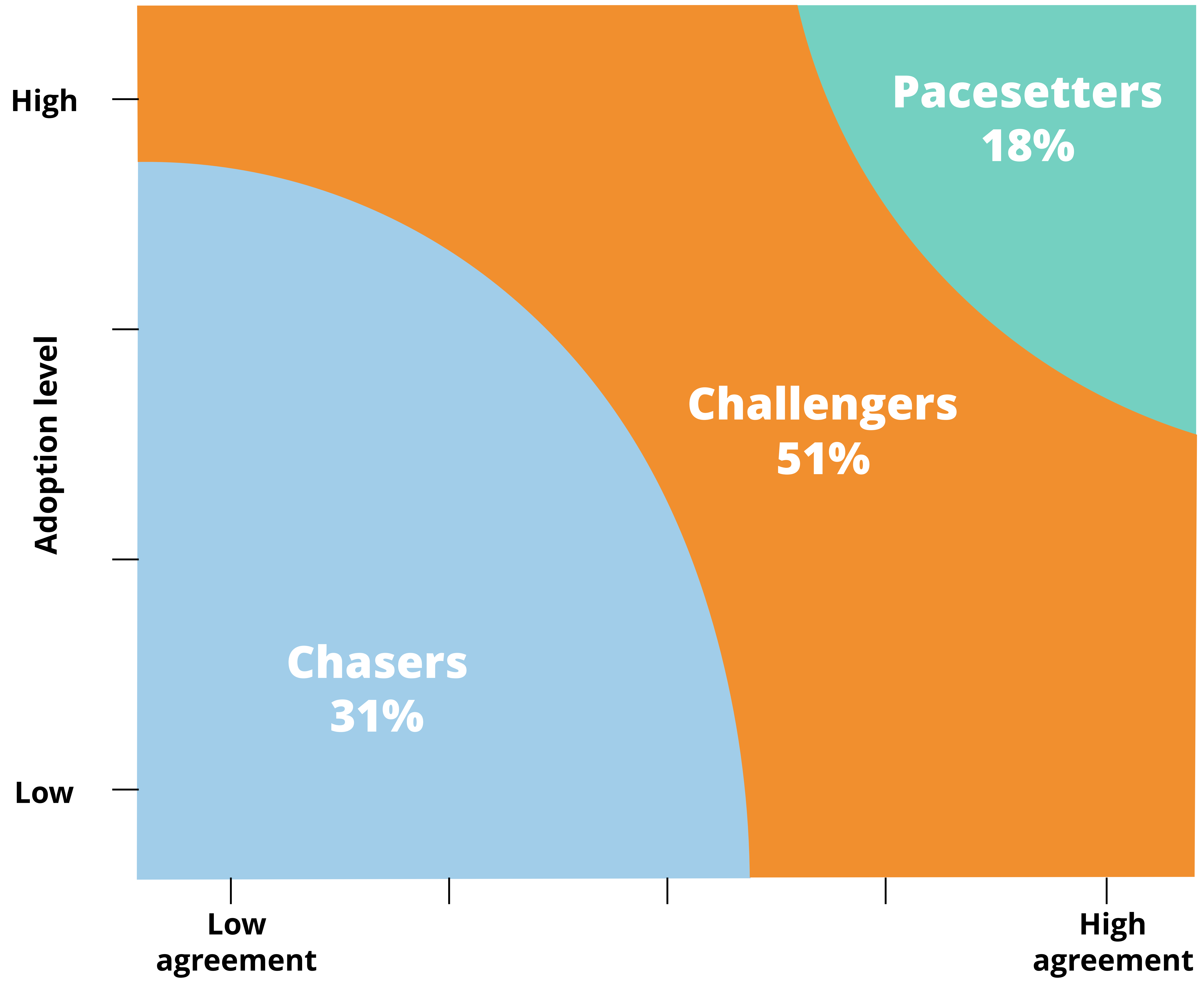
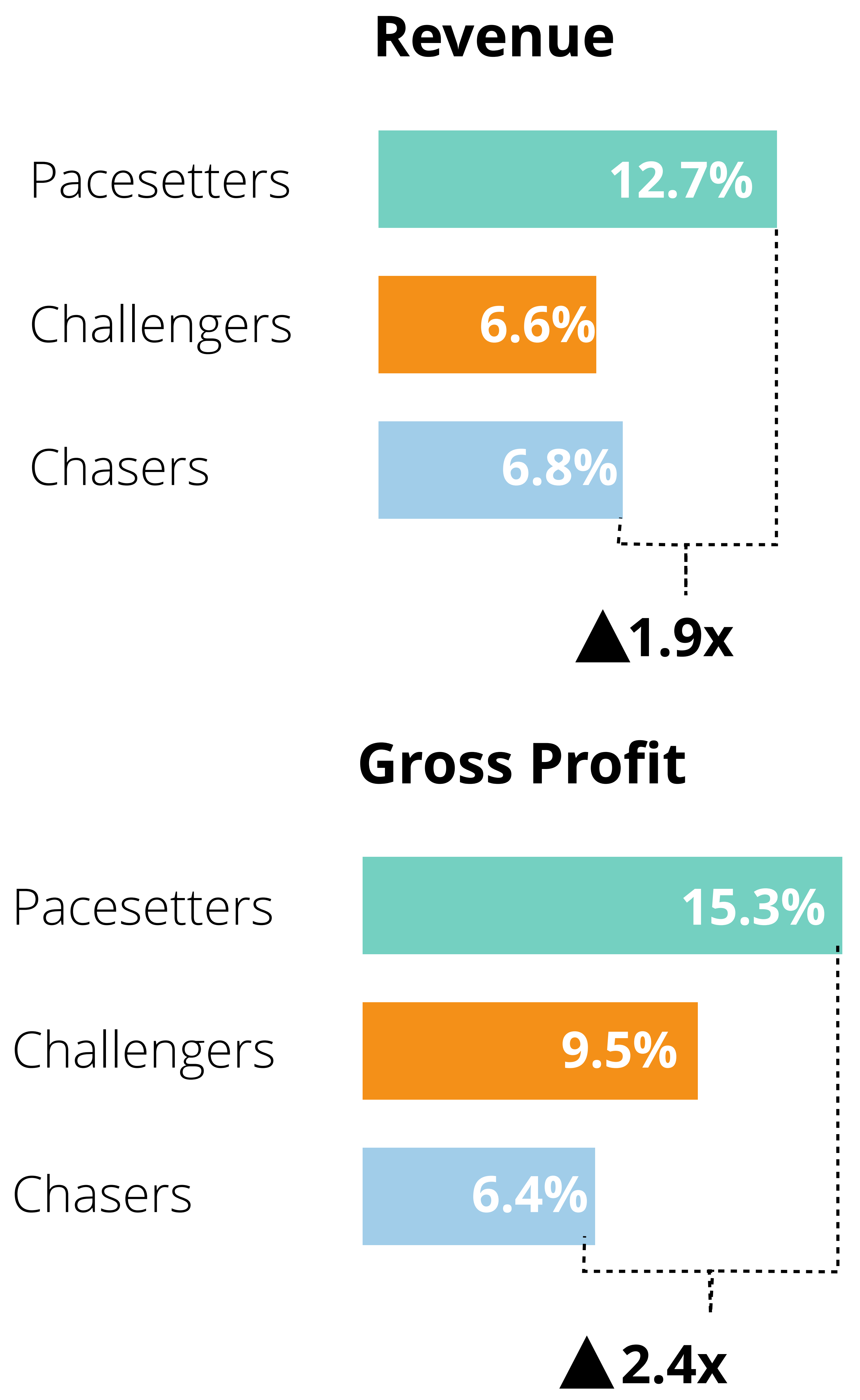
ThoughtWorks®

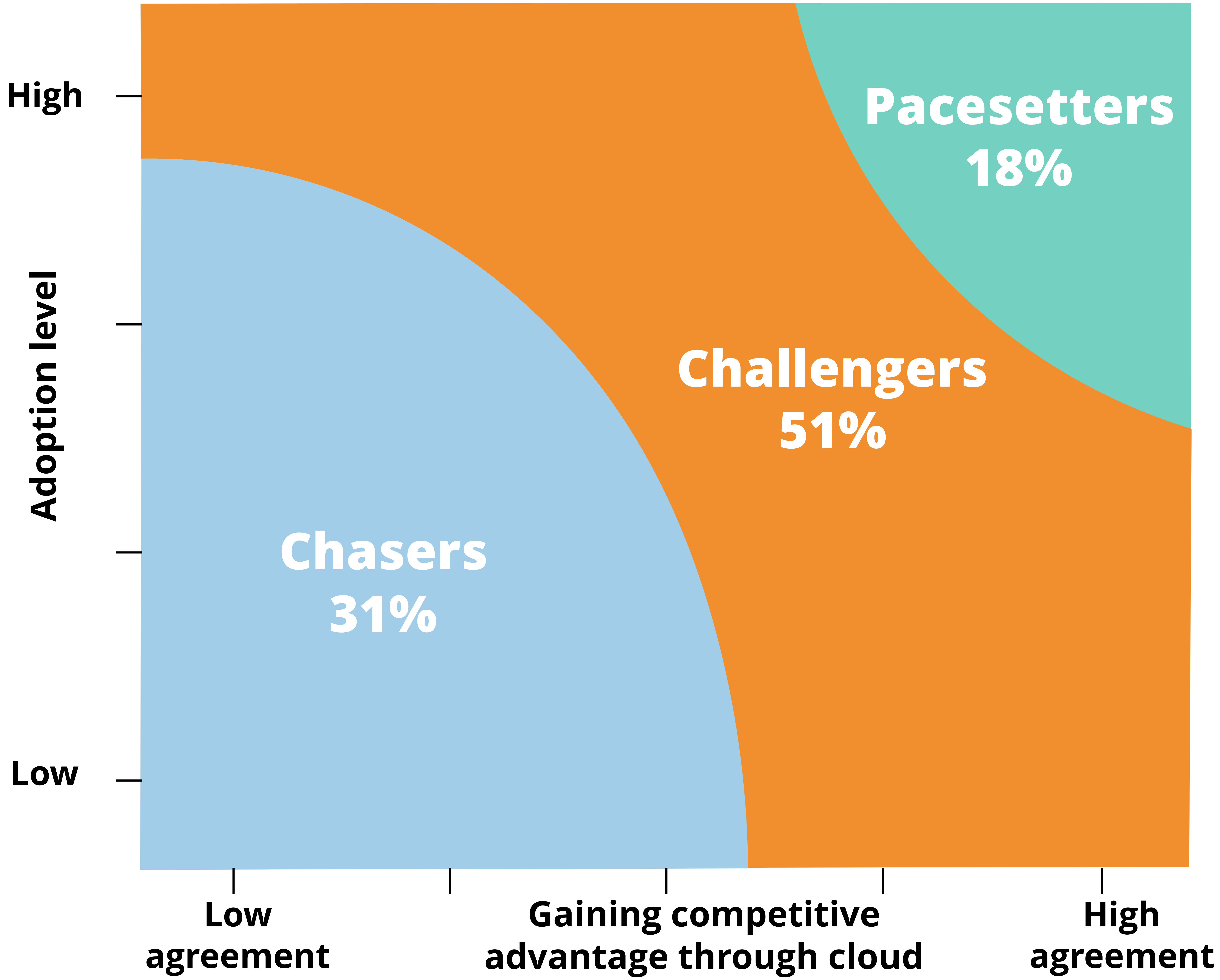
Improve 30% Productivity by Cloud Enabled Stack Management

7 May 2016



Data from ibm



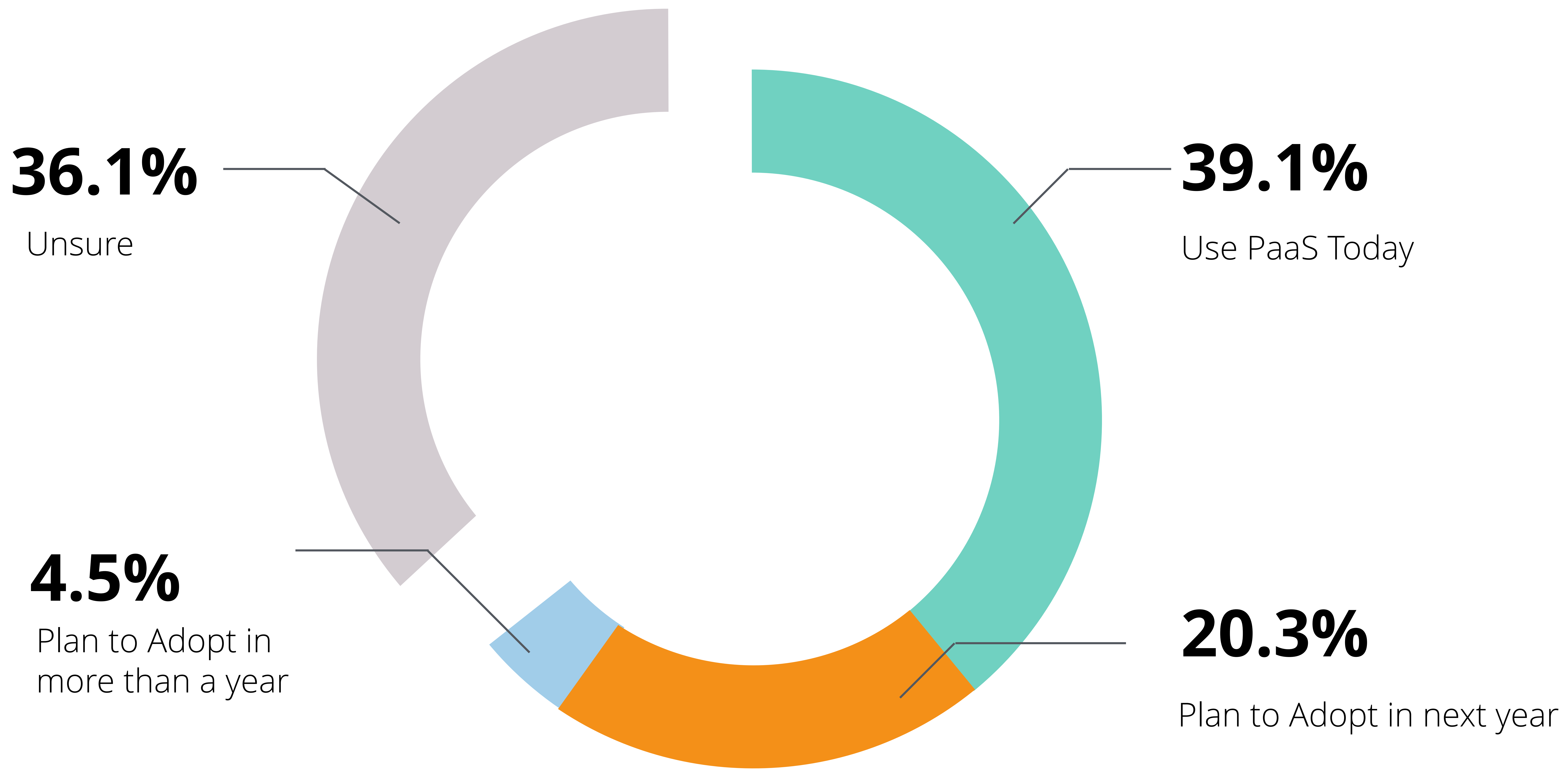


Data from ibm

Competitive advantage through cloud

		Chasers	Challengers	Pacesetters	%Pacesetters surpass Chasers
Strategic reinvention	Reinvent customer relationship	25%	46%	59%	+136%
	Innovate products/services rapidly	30%	51%	52%	+73%
	Build new/improved business models	30%	44%	51%	+70%
Better decisions	Use analytics extensively to derive insights from big data	20%	44%	54%	+170%
	Share data seamlessly across applications	27%	51%	59%	+119%
	Make data-driven, evidence-based decisions	30%	62%	65%	+117%
Deep collaboration	Easier to locate and leverage knowledge of experts anywhere in ecosystem	34%	51%	61%	+79%
	Improve integration between development and operations	34%	49%	59%	+74%
	Collaborate across organization and ecosystem	34%	45%	58%	+71%

Data from ibm



Data from Engine Yard

ThoughtWorks®

TECHNOLOGY RADAR *APRIL '16*

Our thoughts on the technology and trends that are shaping the future

thoughtworks.com/radar

WHAT'S NEW?

Here are the themes highlighted in this edition:

OPEN SOURCE AS A VIRTUOUS BY-PRODUCT

Some of the most influential software appearing on our radar comes from companies whose first mandate isn't to create software tools. Several of our radar entries come from Facebook, not considered a traditional software development toolmaker. Unlike in the past, today many companies open source their important software assets—to attract new recruits and credentialize themselves. This creates a virtuous feedback loop: Innovative open source attracts good developers who are in turn more likely to innovate. As a side effect, these companies' frameworks and libraries are some of the most influential in the industry. This represents a big shift in the software development ecosystem and is further proof of the efficacy of open source software … in the right context (our advice about Web Scale Envy still stands).

PARSING THE PAAS PUZZLE

Many large organizations see the Cloud and Platform as a Service (PaaS) as an obvious way to standardize infrastructure, ease deployment and operations, and make developers more productive. But it's still early days, the definition of PaaS remains nebulous, and many PaaS approaches are incomplete or suffer from the immaturity of supporting frameworks and tools. Some PaaS solutions make it harder to do things more easily done with plain Infrastructure as a Service (IaaS), such as using a custom Service Locator or complex network topology, and the jury is still out on whether a "Containers as a Service" approach will provide similar value with more flexibility. We see many companies implementing an off-the-shelf PaaS or gradually building their own, with varying degrees of success. We suspect that any PaaS built today will not be an end state but rather part of an evolutionary path. Enterprise migration to Cloud and PaaS, while bringing many benefits, has difficulties and challenges, particularly around overall pipeline design and tooling. Consumers of these technologies should seek the inflection point that indicates "ready for prime time" for their context and should avoid coupling too tightly to the implementation details of their PaaS.

DOCKER, DOCKER, DOCKER!

Containerization, and Docker in particular, has proven hugely beneficial as an application-management technique, rationalizing deployment between environments and simplifying the "it works here but not there" class of problems. We see a significant amount of energy focused on using Docker—and, particularly, the ecosystem surrounding it—beyond dev/test and all the way into production. Docker containers are used as the "unit of scaling" for many PaaS and "data center OS" platforms, giving Docker even more momentum. As it matures as both a development and production environment, people are paying more attention to containerization, its side effects and its implications.

OVER-REACTIVE?

Reactive programming—where components react to changes in data that are propagated to them rather than use imperative wiring—has become extremely popular, with reactive extensions available in almost all programming languages. User interfaces, in particular, are commonly written in a reactive style, and many ecosystems are settling on this paradigm. While we like the pattern, overuse of event-based systems complicates program logic, making it difficult to understand; developers should use this style of programming judiciously. It is certainly popular: We added a significant number of reactive frameworks and supporting tools on this Radar.

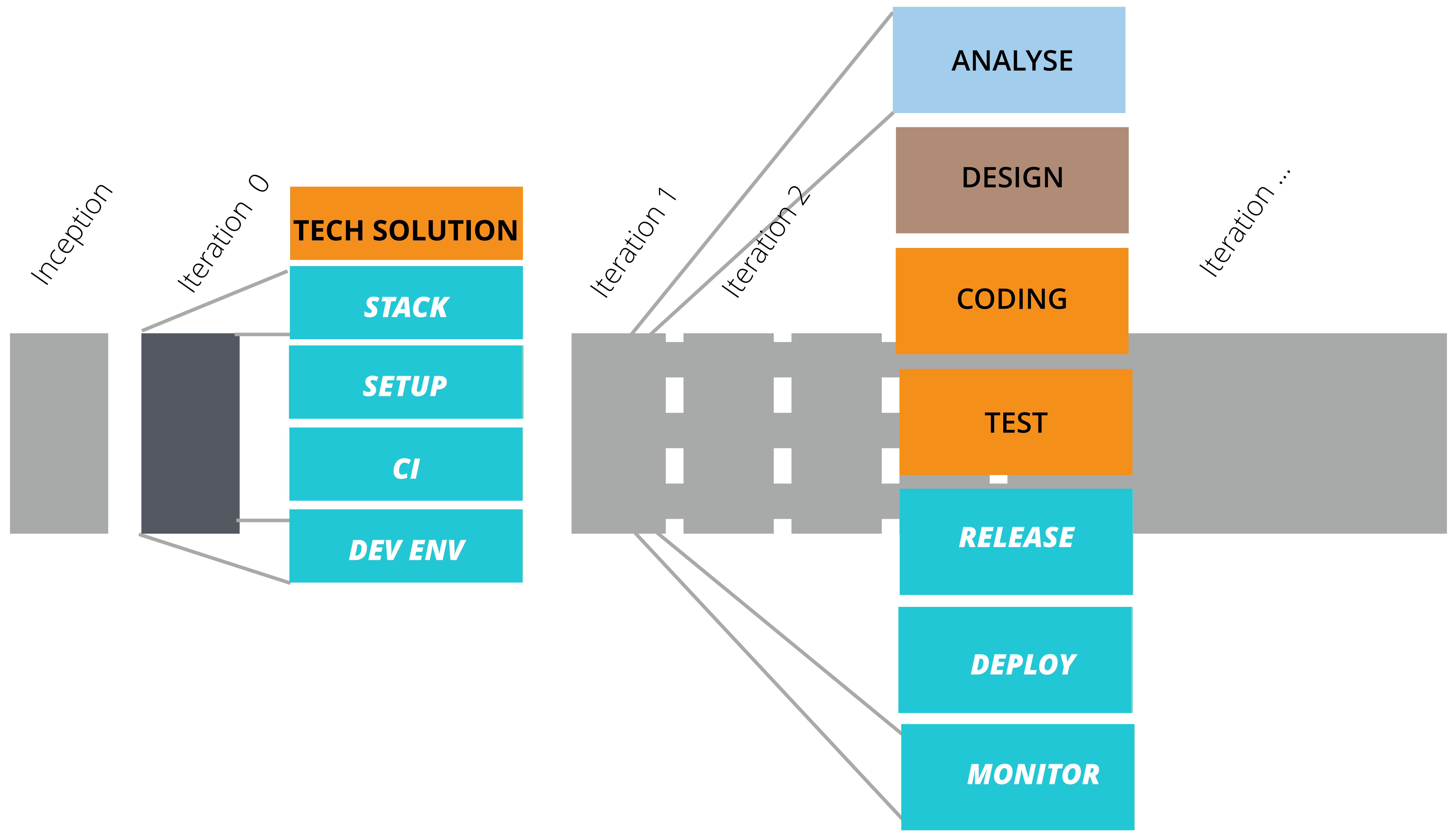
PARSING THE PAAS PUZZLE

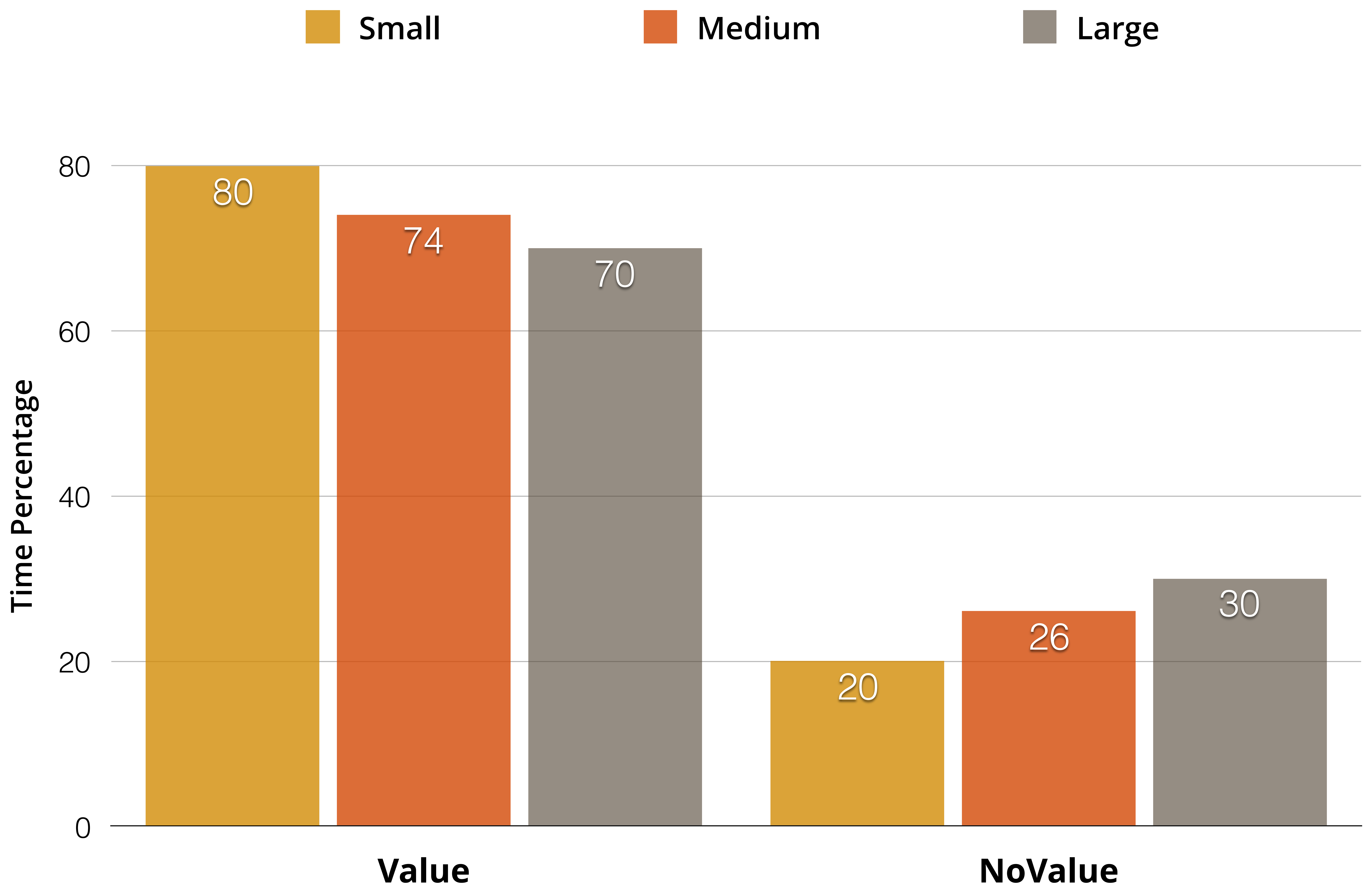
Many large organizations see the Cloud and Platform as a Service (PaaS) as an obvious way to standardize infrastructure, ease deployment and operations, and make developers more productive. But it's still early days, the definition of PaaS remains nebulous, and many PaaS approaches are incomplete or suffer from the immaturity of supporting frameworks and tools. Some PaaS solutions make it harder to do things more easily done with plain Infrastructure as a Service (IaaS), such as using a custom Service Locator or complex network topology, and the jury is still out on whether a "Containers as a Service" approach will provide similar value with more flexibility. We see many companies implementing an off-the-shelf PaaS or gradually building their own, with varying degrees of succes We suspect that any PaaS built today will not be an end state but rather part of an evolutionary path. Enterprise migration to Cloud and PaaS, while bringing many benefits, has difficulties and challenges, particularly around overal pipeline design and tooling. Consumers of these technologies should seek the inflection point that indicates "ready for prime time" for their context and should avoid coupling too tightly to the implementation details of their PaaS.

PARSING THE PAAS PUZZLE

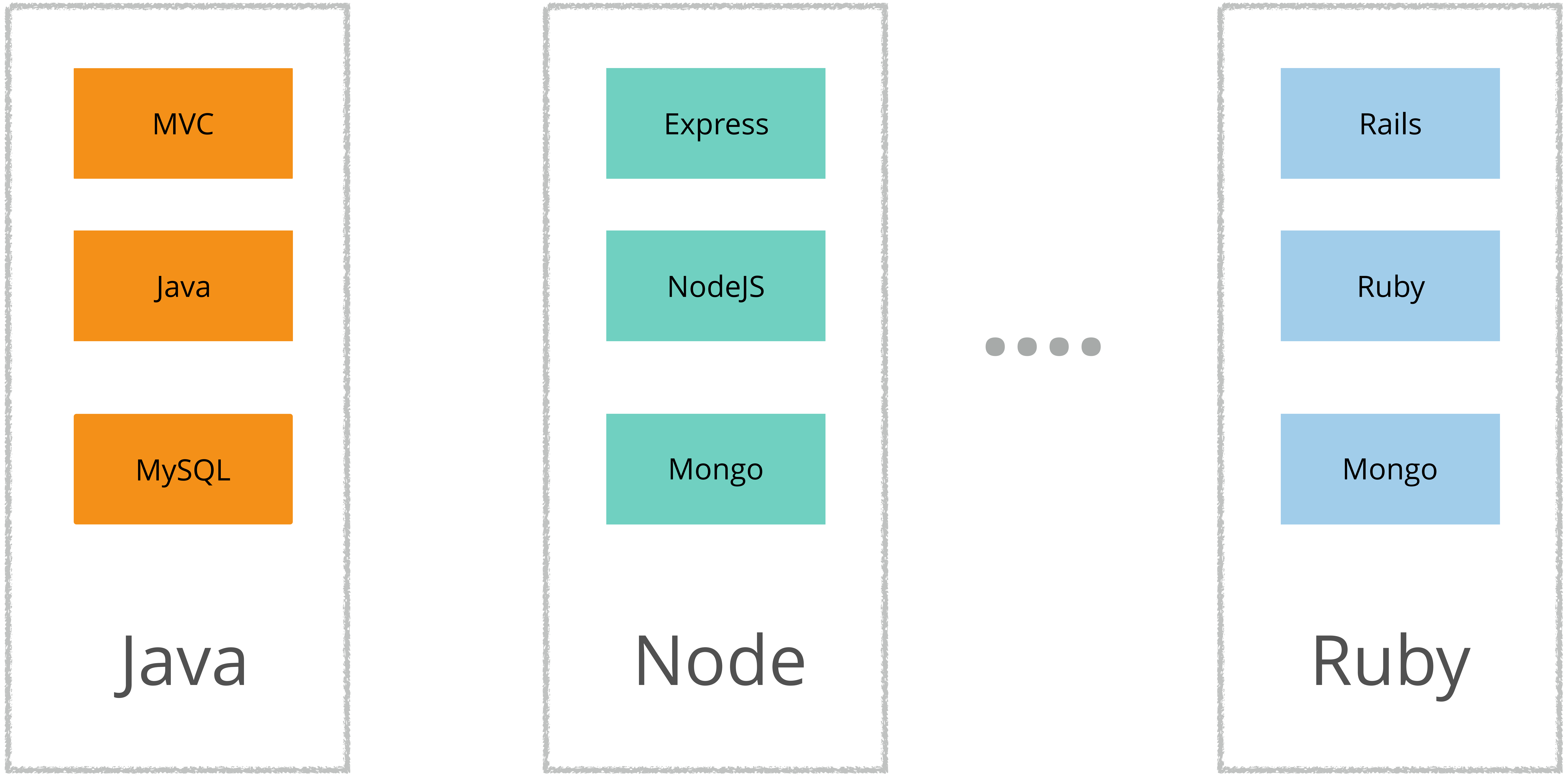
Many large organizations see the Cloud and Platform as a Service (PaaS) as an obvious way to **standardize infrastructure, ease deployment and operations, and make developers more productive.** But it's still early days, the definition of PaaS remains nebulous, and many PaaS **approaches are incomplete** or suffer from the **immaturity of supporting frameworks and tools**. Some PaaS solutions make it harder to do things more easily done with plain Infrastructure as a Service (IaaS), such as using a custom Service Locator or complex network topology, and the jury is still out on whether a "Containers as a Service" approach will provide similar value with more flexibility. We see many companies implementing an off-the-shelf PaaS or gradually building their own, with varying degrees of success. We suspect that any PaaS built today will not be an end state but rather part of an evolutionary path. Enterprise migration to Cloud and PaaS, while bringing many benefits, has **difficulties and challenges, particularly around overall pipeline design and tooling**. Consumers of these technologies should seek the inflection point that indicates "ready for prime time" for their context and should avoid **coupling too tightly** to the implementation details of their PaaS.

BACKGROUND

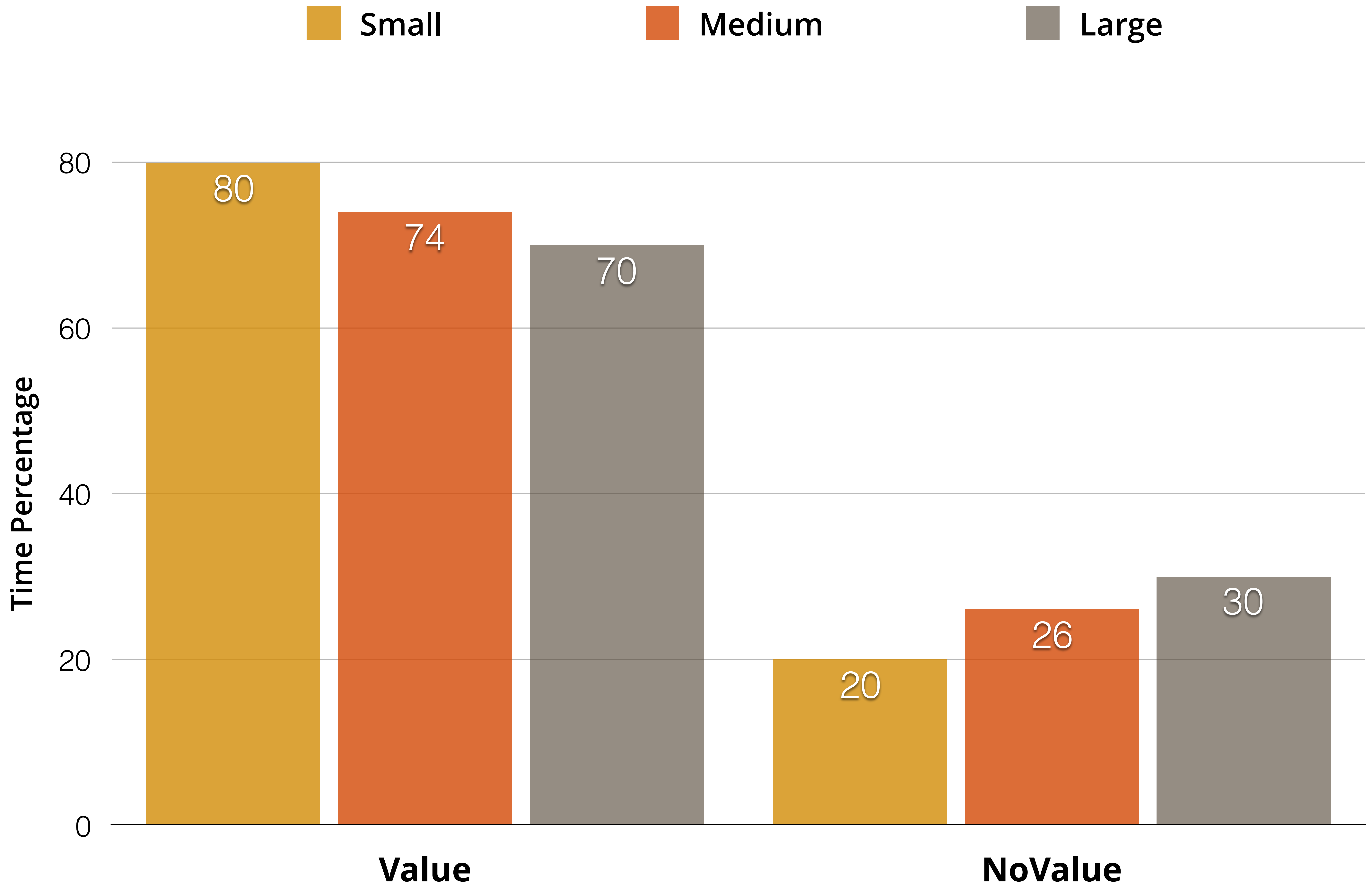

Inception
Iteration 0
TECH SOLUTION
STACK
SETUP
CI
DEV ENV
Iteration 1
Iteration 2
ANALYSE
DESIGN
CODING
TEST
RELEASE
DEPLOY
MONITOR
Iteration ...

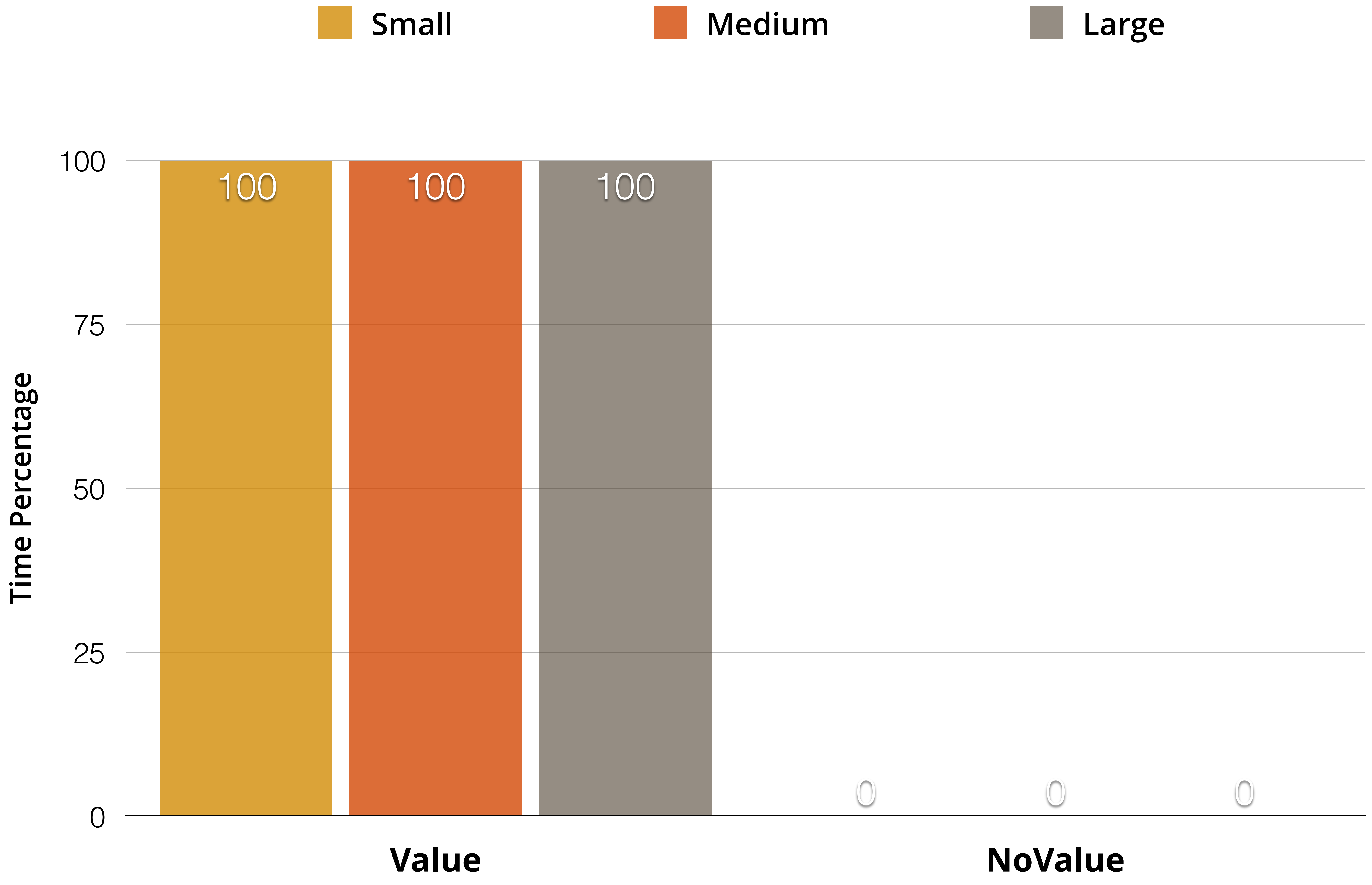

Small
Medium
Large
Time Percentage
80
60
40
20
0
80
74
70
20
26
30
Value
NoValue

Enumerable stacks



WHAT CAN WE DO

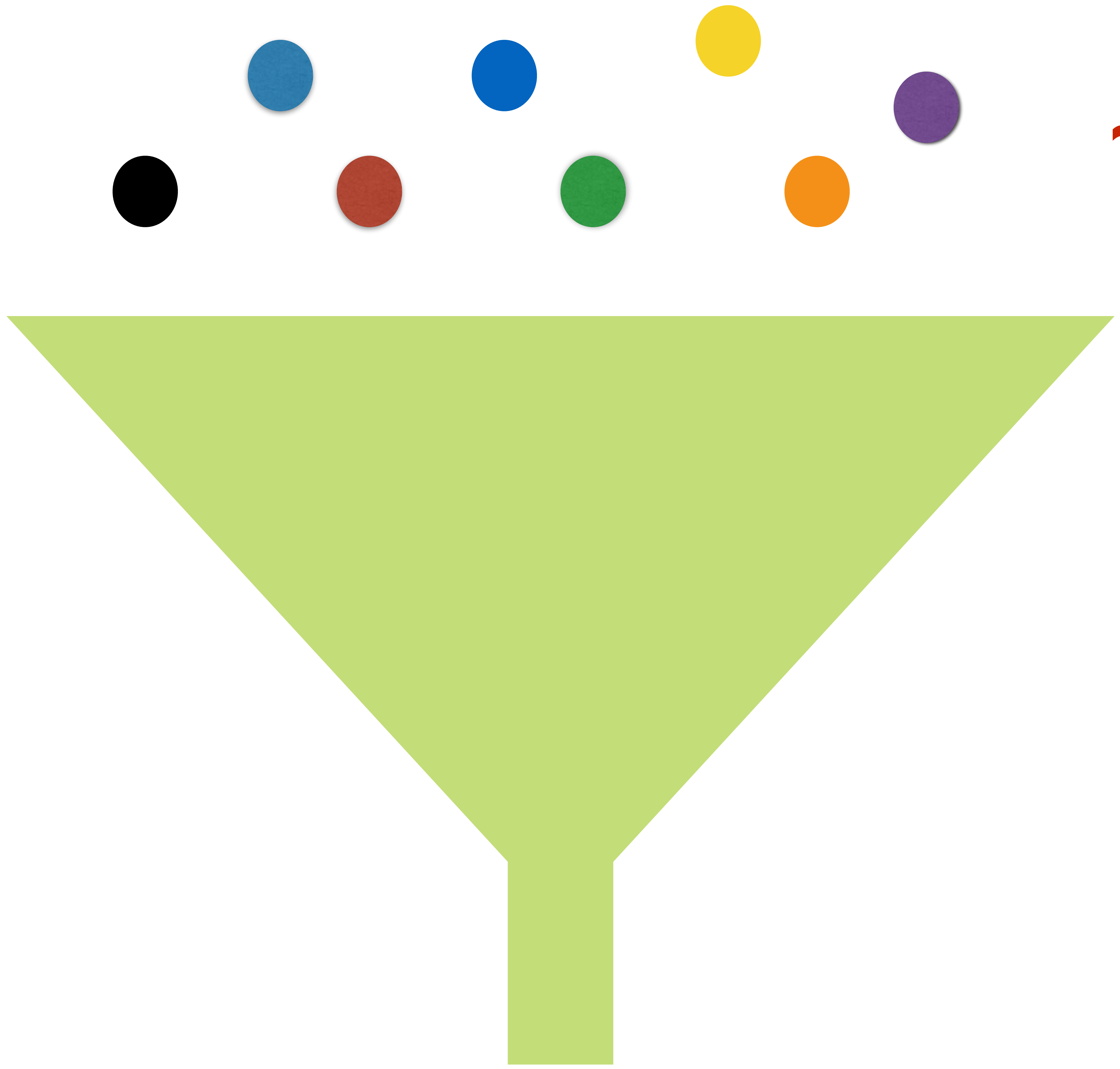

Small
Medium
Large
Time Percentage
80
60
40
20
0
80
74
70
20
26
30
Value
NoValue


Small
Medium
Large
Time Percentage
100
75
50
25
0
100
100
100
0
0
0
Value
NoValue

STACK MANAGEMENT

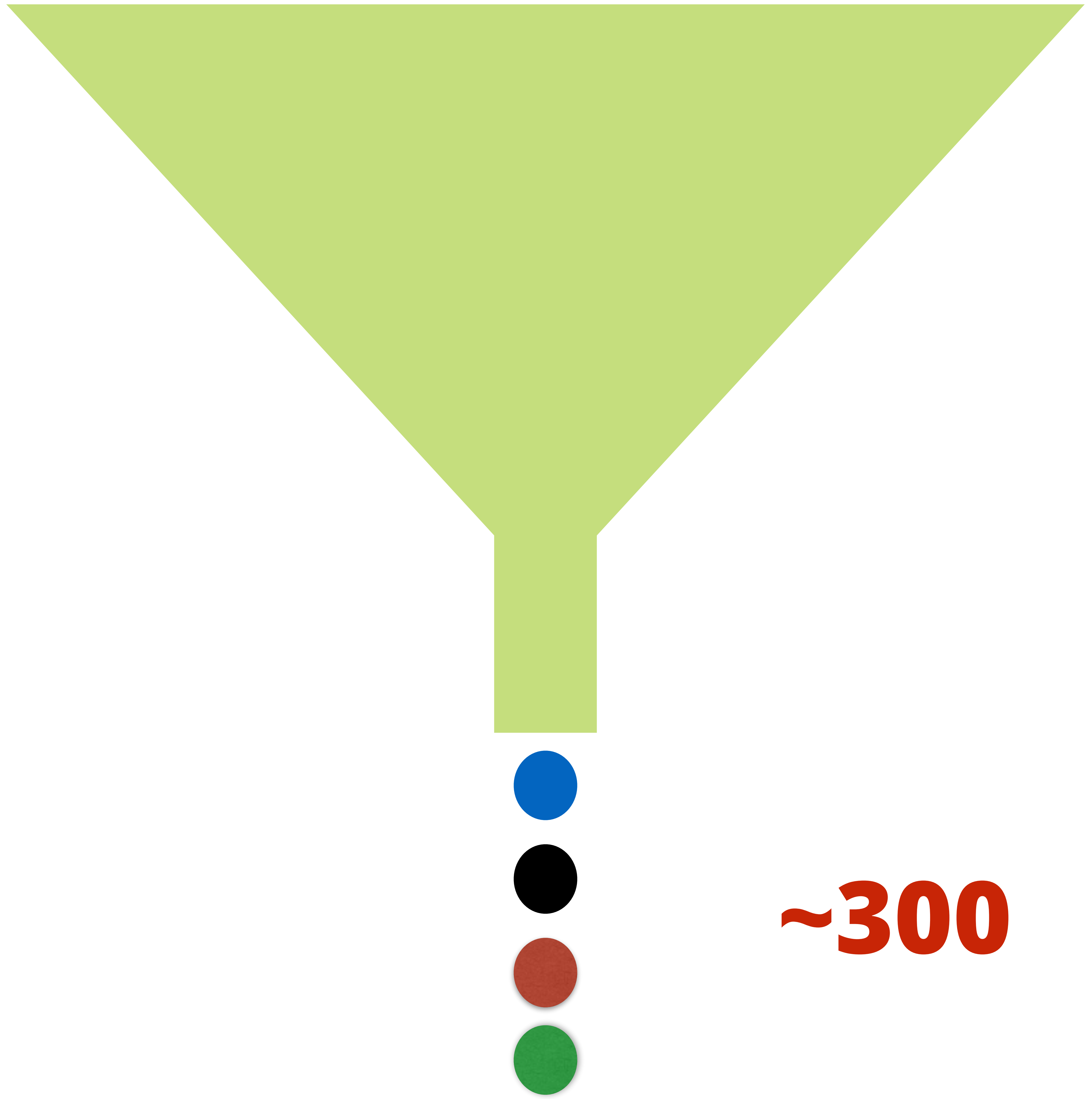
Z
W
P
S
W
X

STILL WORKS?

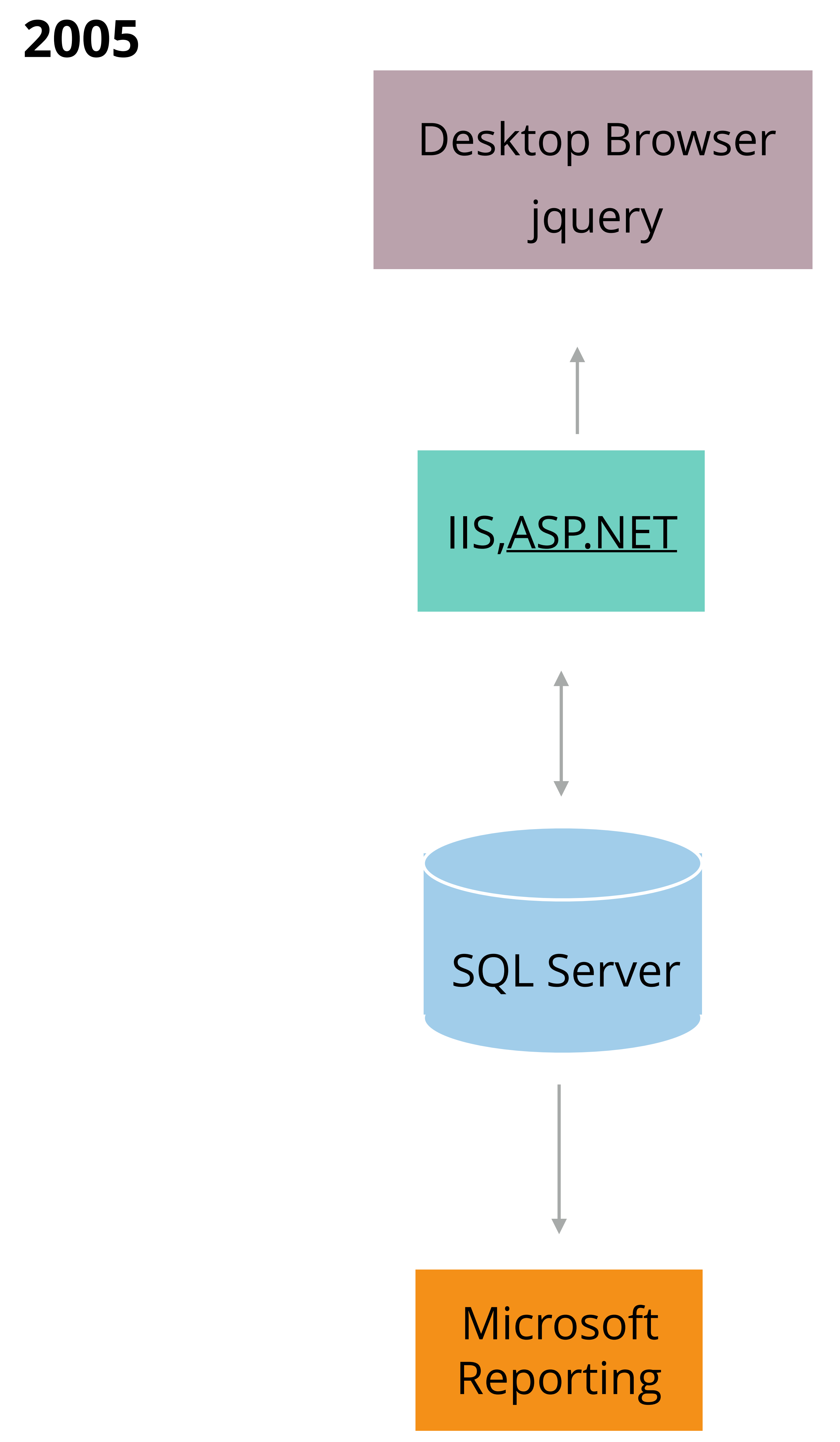


~2000

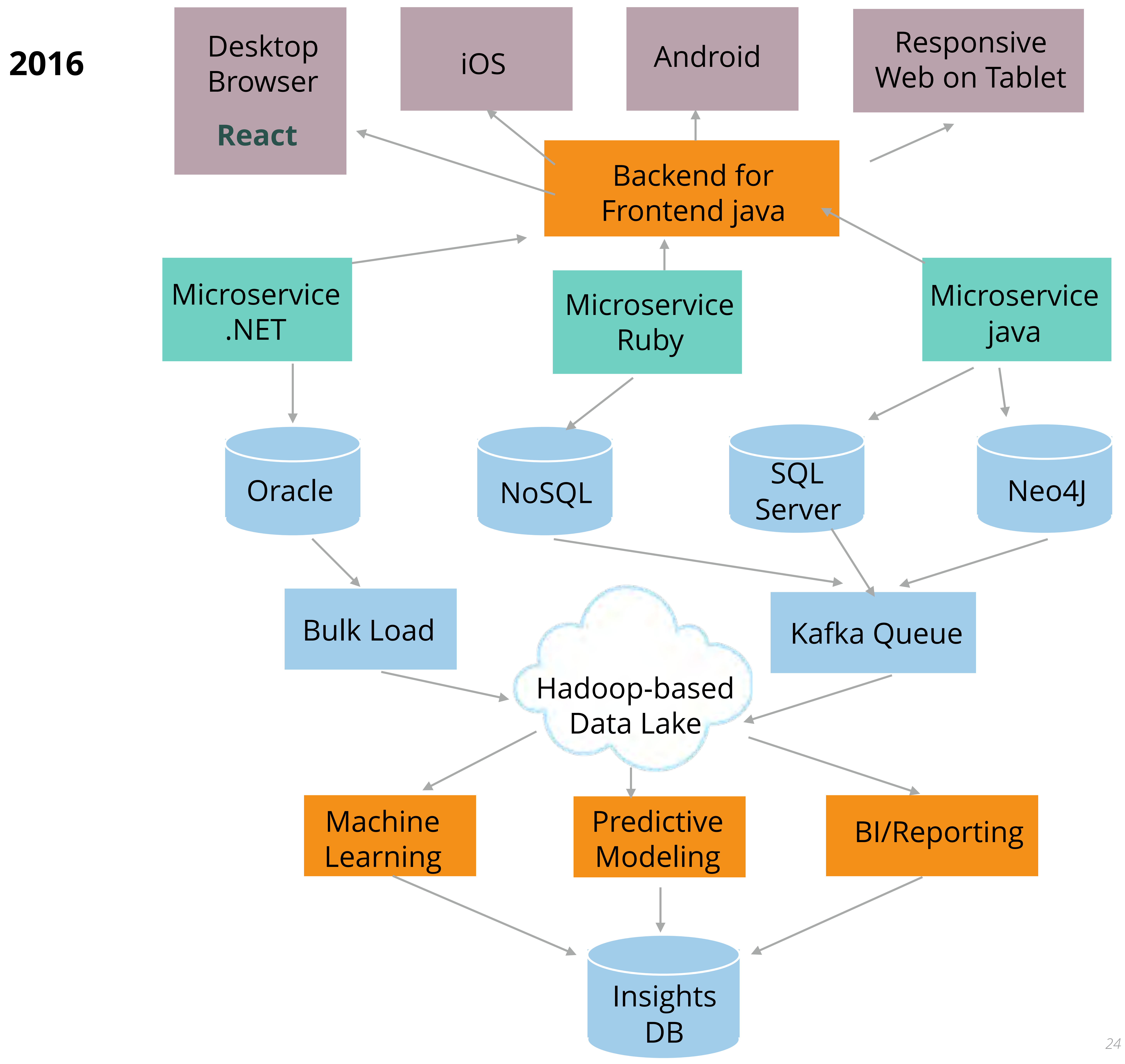
~2000



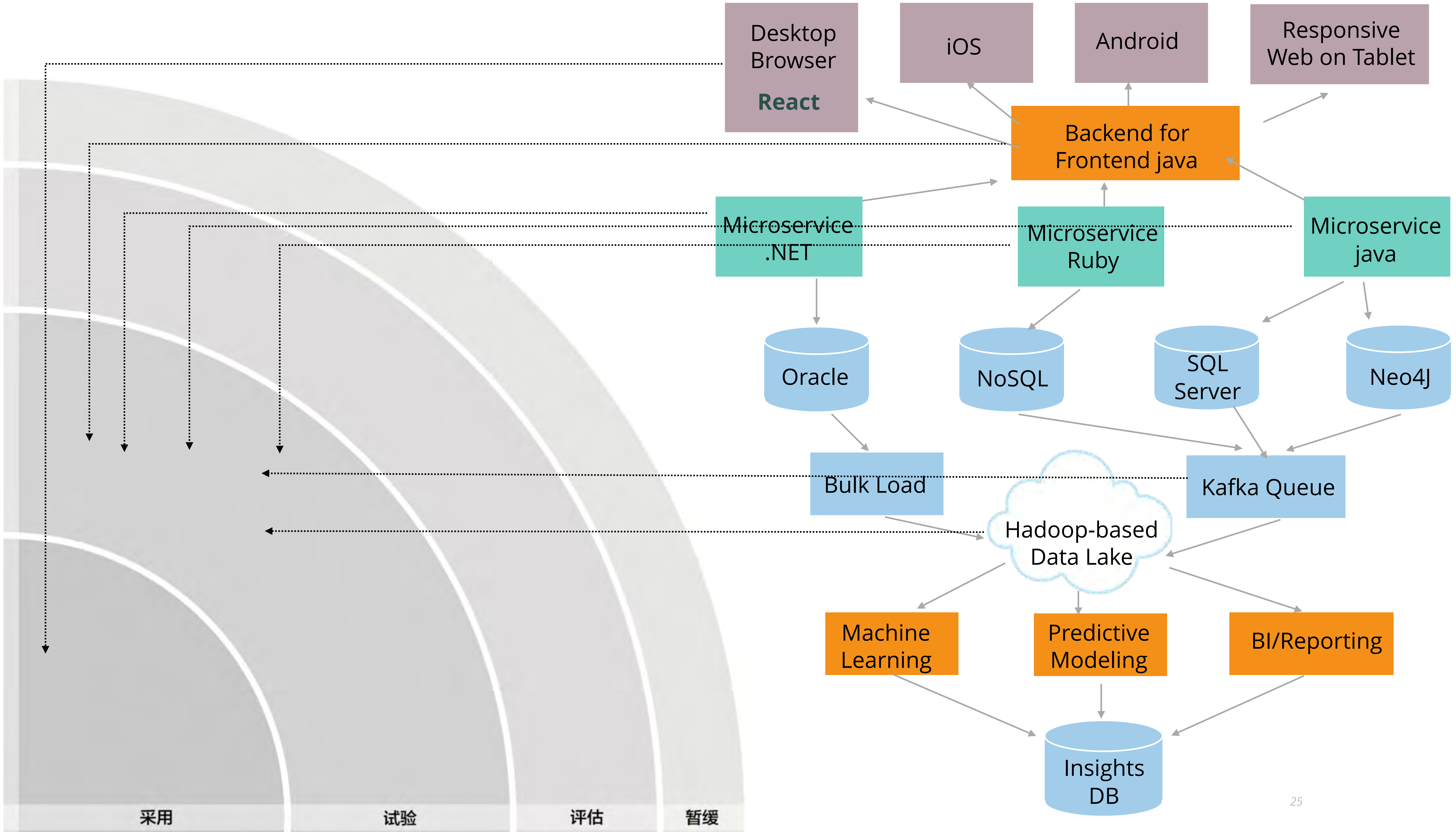
~300



2005
Desktop Browser
jquery
IIS,ASP.NET
SQL Server
Microsoft Reporting



2016
Desktop Browser
React
iOS
Android
Responsive Web on Tablet
Backend for Frontend java
Microservice .NET
Microservice Ruby
Microservice java
Oracle
NoSQL
SQL Server
Neo4J
Bulk Load
Kafka Queue
Hadoop-based Data Lake
Machine Learning
Predictive Modeling
BI/Reporting
Insights DB


Desktop Browser
React
iOS
Android
Responsive Web on Tablet
Backend for Frontend java
Microservice .NET
Microservice Ruby
Microservice java
Oracle
NoSQL
SQL Server
Neo4J
Bulk Load
Kafka Queue
Hadoop-based Data Lake
Machine Learning
Predictive Modeling
BI/Reporting
Insights DB
采用
试验
评估
暂缓

Docker

Consul

Nomad

Apache Mesos

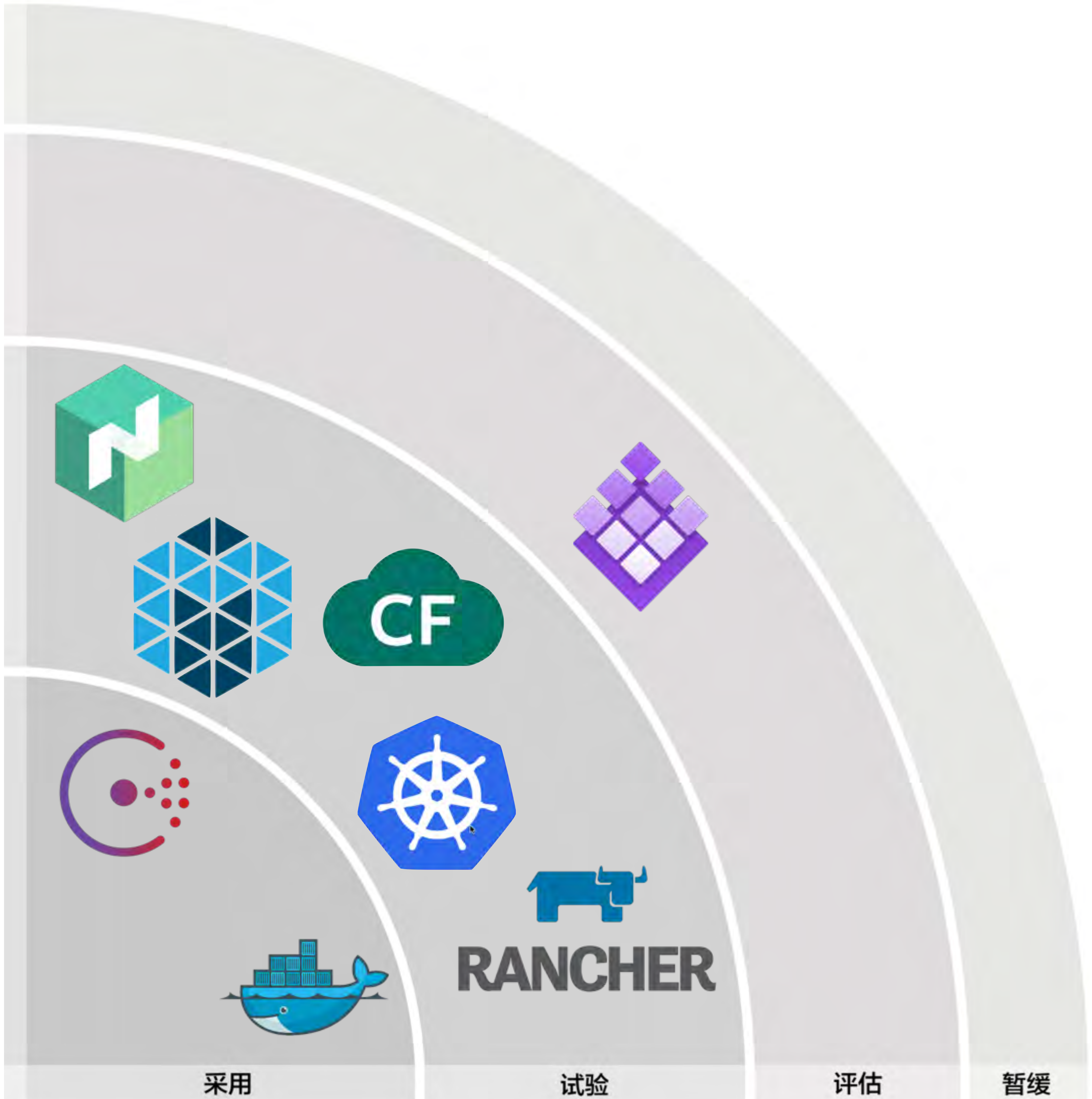
Pivotal Cloud Foundry

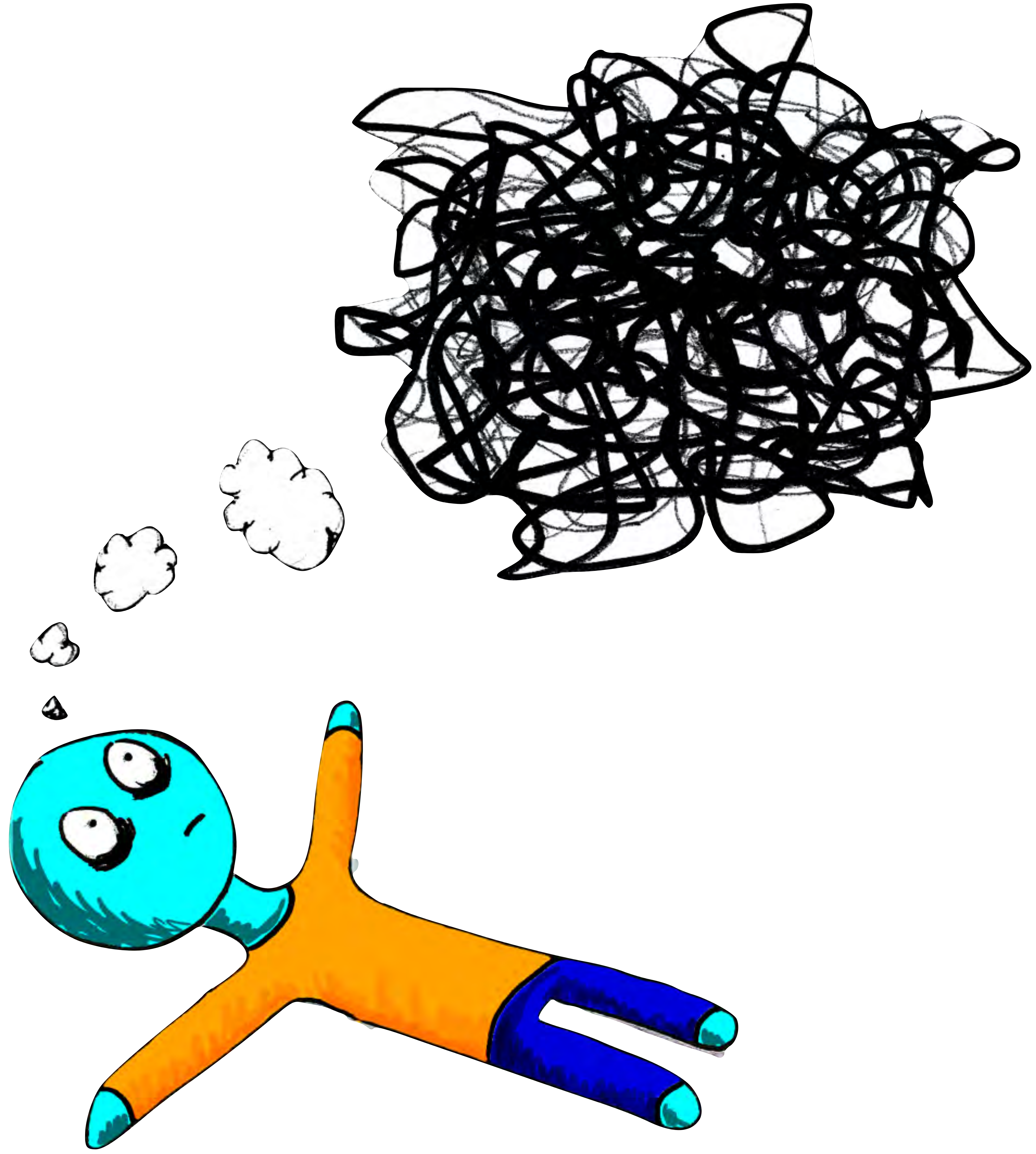
Kubernetes

Rancher

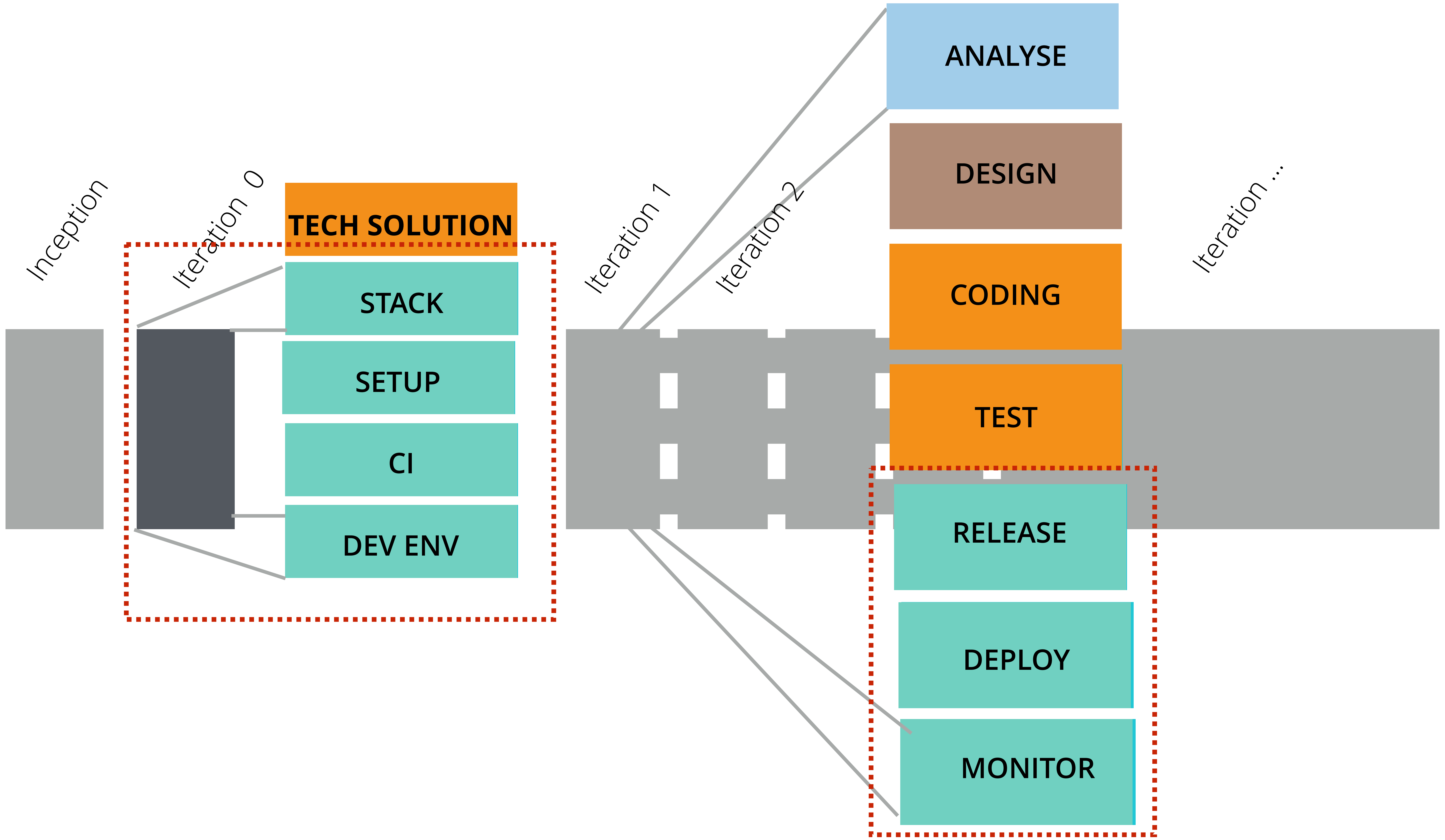
Mesosphere DCOS

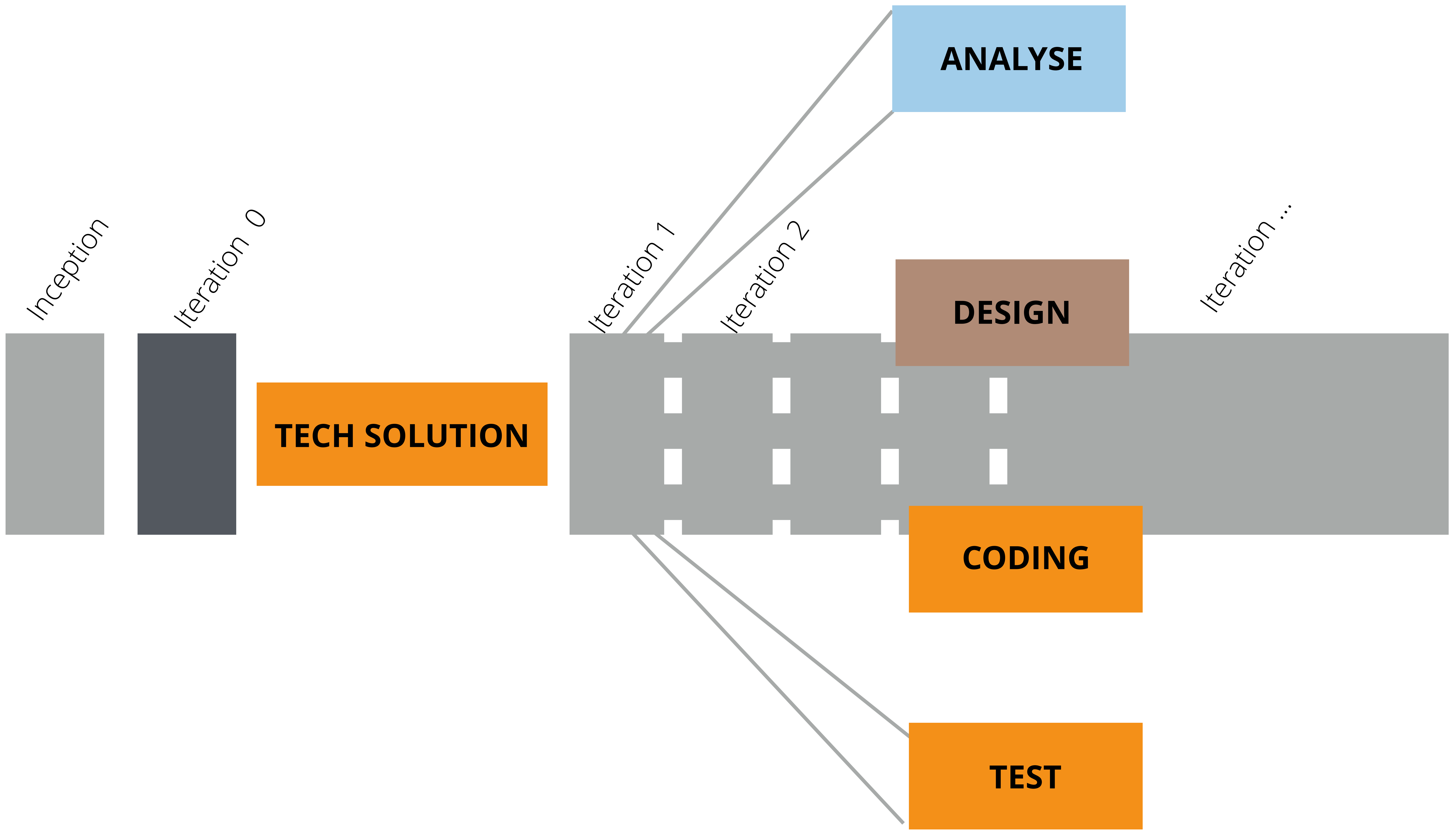
...





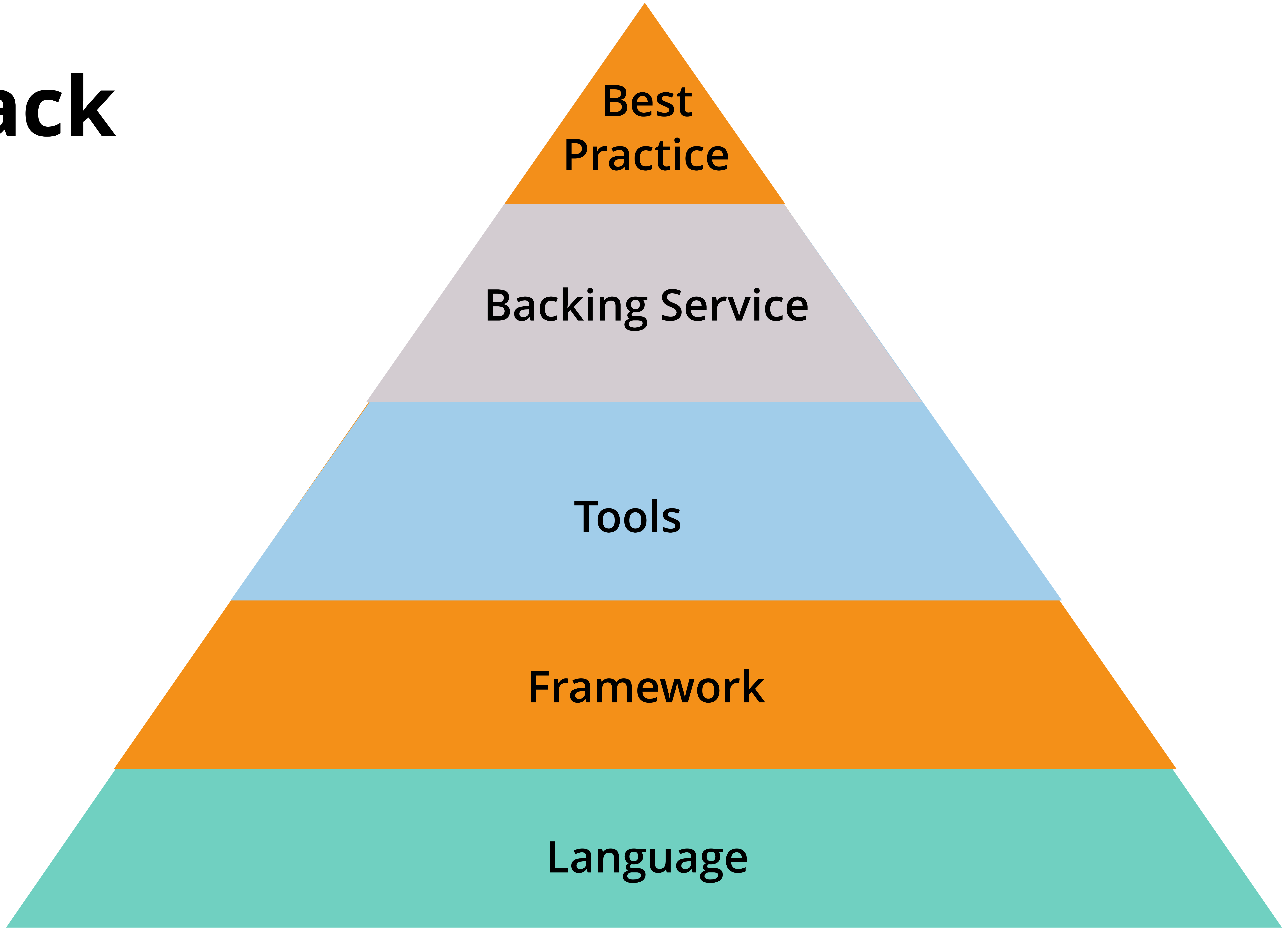
HOW TO MANAGE

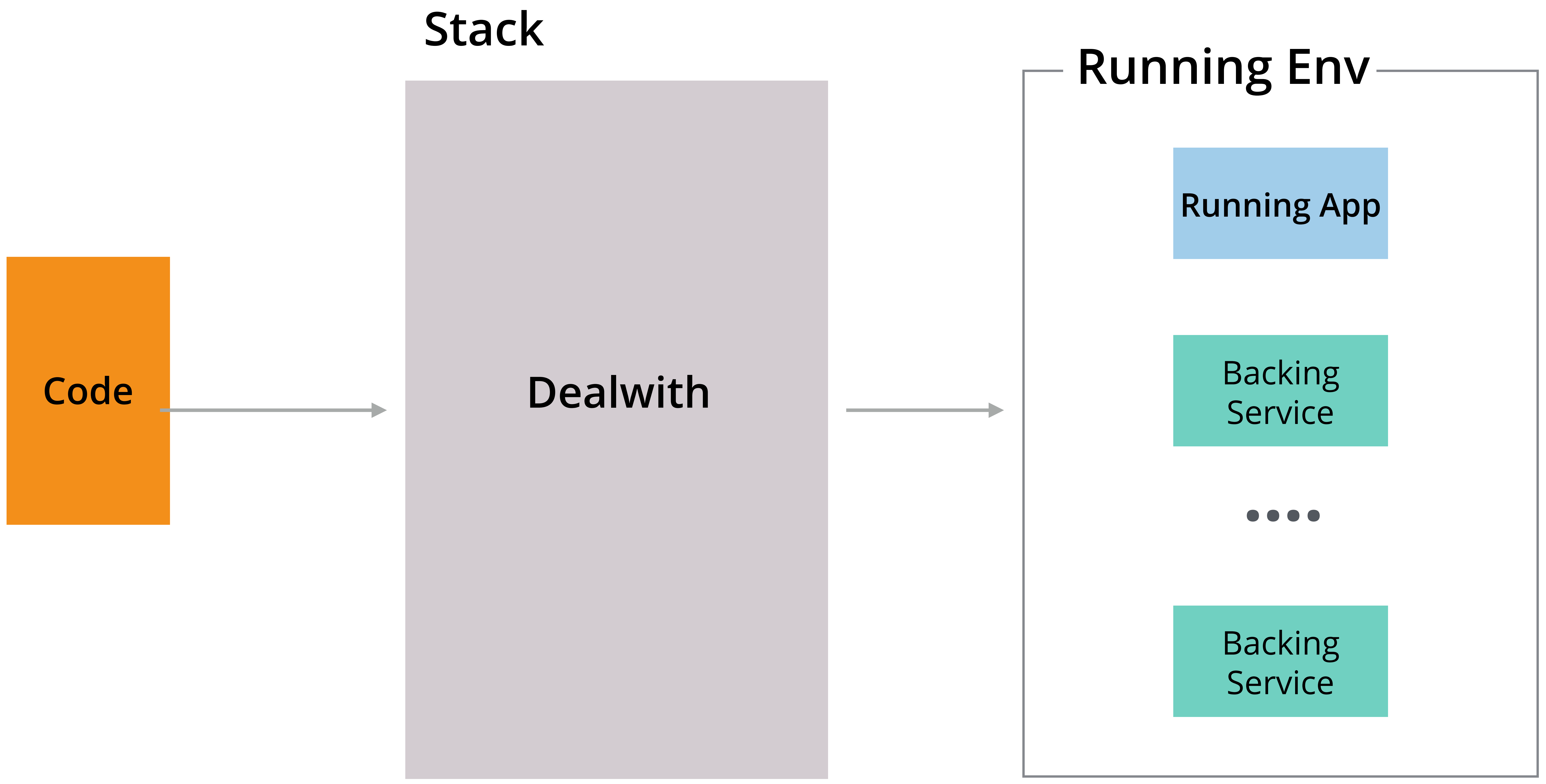

Inception
Iteration 0
TECH SOLUTION
STACK
SETUP
CI
DEV ENV
Iteration 1
Iteration 2
ANALYSE
DESIGN
CODING
TEST
RELEASE
DEPLOY
MONITOR
Iteration ...


Inception
Iteration 0
TECH SOLUTION
Iteration 1
Iteration 2
ANALYSE
DESIGN
Iteration ...
CODING
TEST

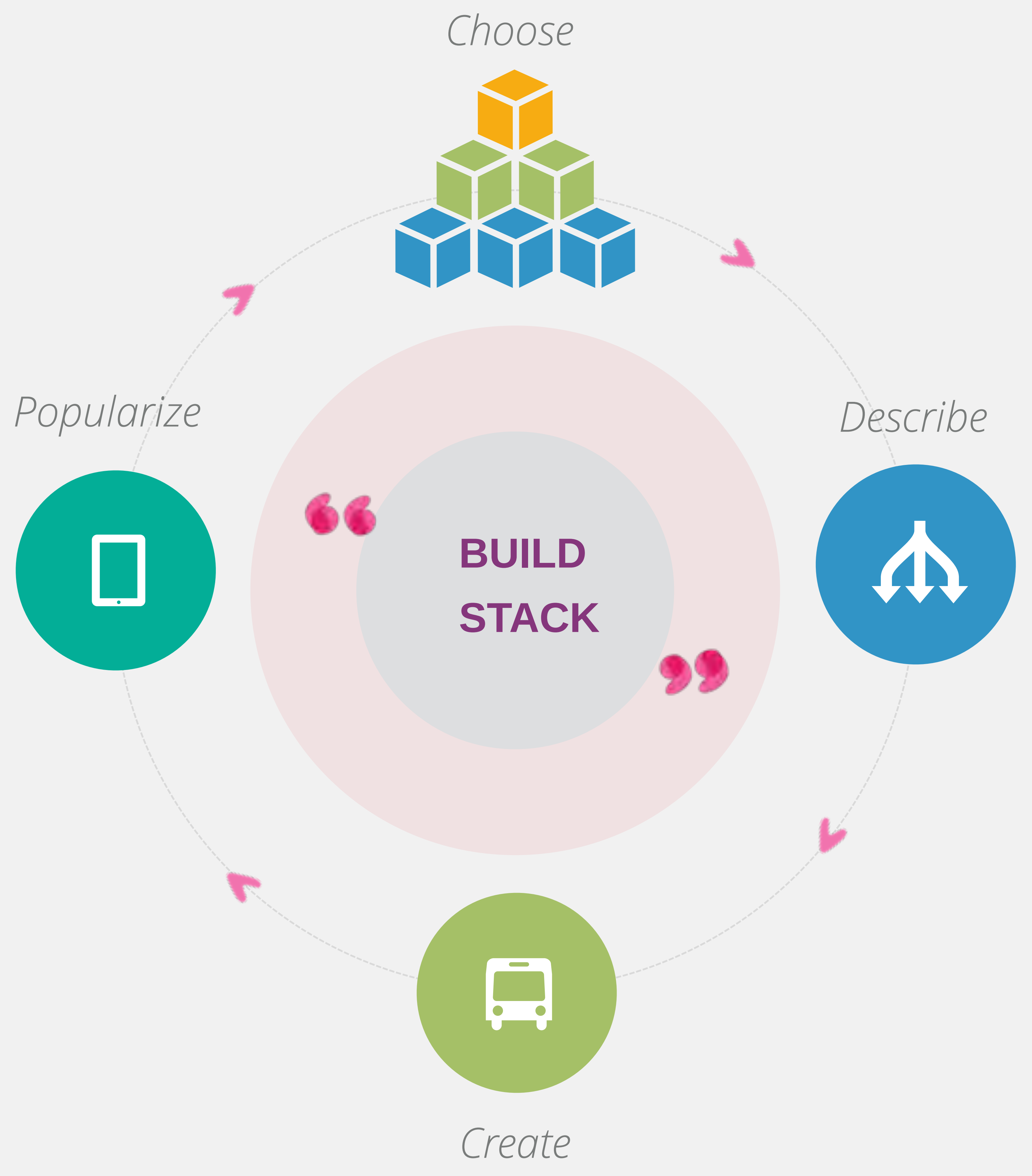
BUILD STACK ONCE AND USE ANYWHERE

Stack

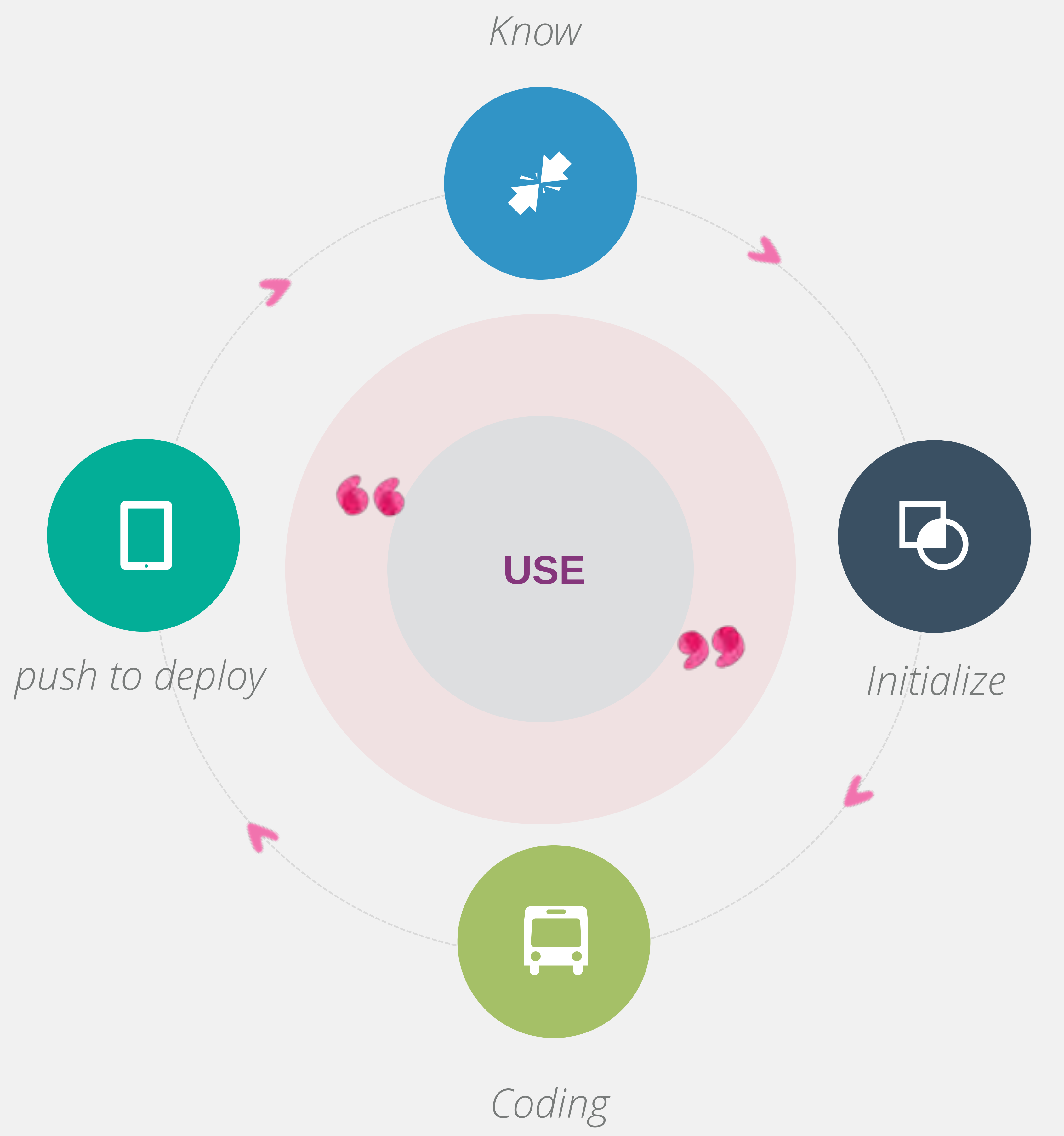



Stack
Code
Dealwith
Running Env
Running App
Backing Service
••••
Backing Service

Define



Use



DEFINE STACK

STACK AS CODE

```
name: "javajersey"
description: "A sample java jersey stack"
template:
  type: "git"
  uri: "https://github.com/aisensiy/
javajersey_api.git"
tags:
    - "java"
```

Language

```
languages:
  - name: "java"
    version: "1.8"
  - name: "xml"
    version: "4.0"
```

Framework

```
frameworks:
  - name: "jersey"
    version: "2.17"
  - name: "mybatis"
    version: "3.3"
```

Tools

```
tools:
  - name: "gradle"
    version: "2.8"
```

```
services:
  web:
    build:
      image: "hub.deepi.cn/javajersey-build"
      mem: 512
      cpus: 0.5
```

BEST PRACTICE AS CODE

```yaml
name: "javajersey"
description: "A sample java jersey stack"
template:
  type: "git"
  uri: "https://github.com/aisensiy/javajersey_api.git"
tags:
    - "java"
languages:
  - name: "java"
    version: "1.8"
  - name: "xml"
    version: "4.0"
frameworks:
  - name: "jersey"
    version: "2.17"
  - name: "mybatis"
    version: "3.3"
tools:
```

Best Practice

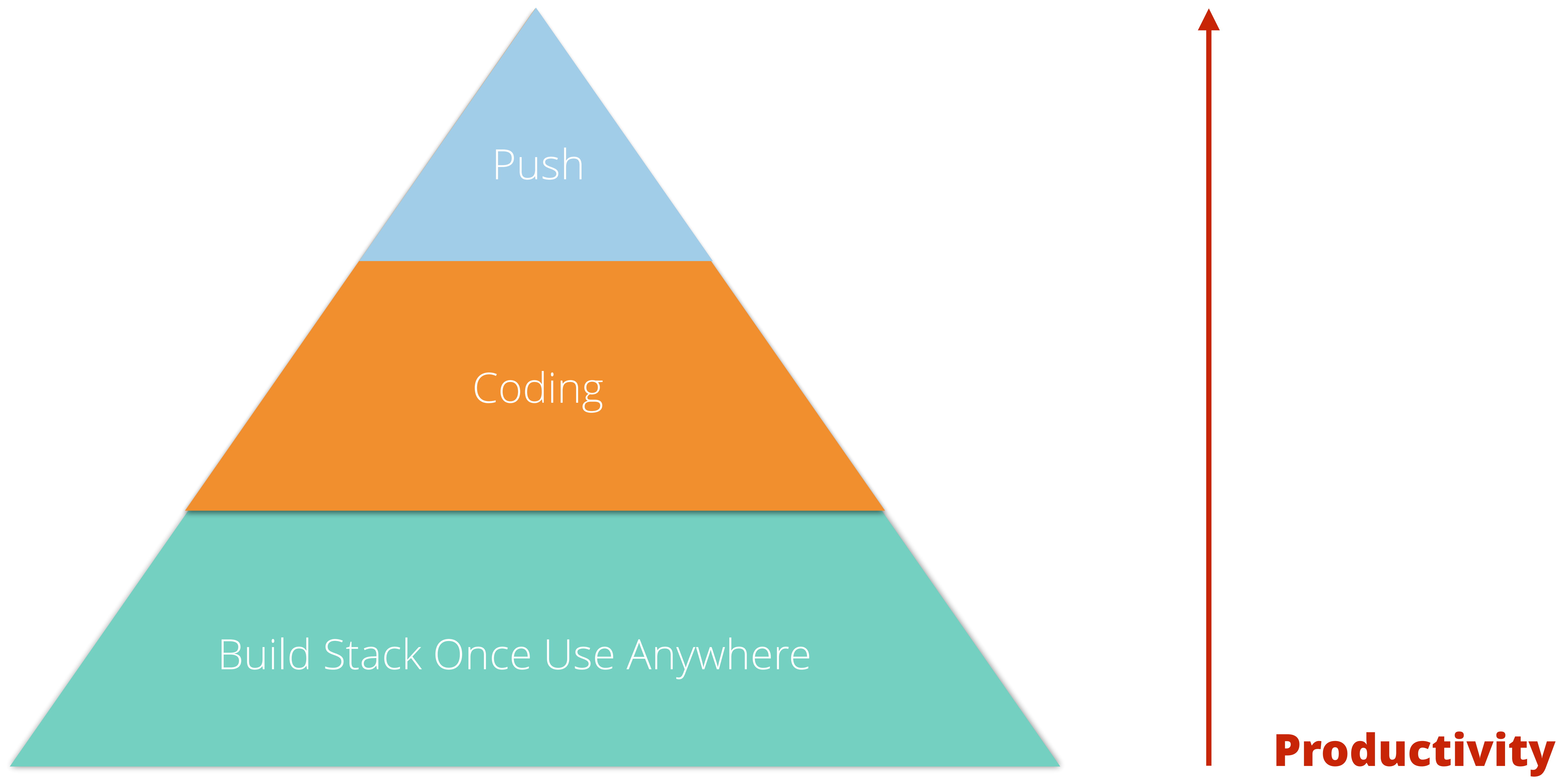
USE STACK

SUMMARY

STACK AS CODE BEST PRACTICE AS CODE

↓

BUILD STACK ONCE USE ANYWHERE


Push
Coding
Build Stack Once Use Anywhere
Productivity

THANK YOU

孙建康
jksun@thoughtworks.com