KIEF MORRIS'S

INFRASTRUCTURE AS CODE

By

Aitanjali Venkatiaman

FOREWORD

By Rachel Laycock, CTO, Thoughtworks

Infrastructure as Code (IaC) has transformed the way we build and manage technology. While it's a core practice for modern software teams, its principles and benefits extend far beyond engineers and operators. Understanding IaC is crucial for anyone involved in shaping technology strategy, making architectural decisions, or driving digital transformation.

Kief Morris' Infrastructure as Code, 3rd Edition is the definitive resource on this topic, but we recognize that not everyone has the time—or the technical background—to dive deep into its details. That's why this illustrated guide was created: to provide a lightweight, accessible way for non-technical audiences to grasp the key concepts of IaC.

Through simple explanations and engaging visuals, this guide breaks down complex ideas into practical insights. Whether you're an executive, product leader, or someone simply curious about modern infrastructure practices, this resource will help you understand why IaC matters and how it impacts the way we build resilient, scalable, and high-performing systems.

We hope this guide sparks meaningful conversations and empowers yo to make informed decisions in an increasingly automated and software-driven world.

Enjoy the read!

Rachel Laycock
Chief Technology Officer, Thoughtworks

SCOPE

PART 1 - FOUNDATIONS

INTRODUCTION

- FROM IRON AGE
- TO CLOUD AGE
- DEVOPS
- A SHIFT IN THINKING
- WHAT IS INFRASTRUCTURE?
- WHY IS INFRASTRUCTURE IMPORTANT?

WHAT IS INFRA AS CODE?

- DEFINITION
- CORE PRINCIPLES
- WHY CODE?
- CODE OR CONFIGURATION?
- WHY INFRA AS CODE?

MYTHS ABOUT AUTOMATING CHANGE

PART 2 - DESIGN

- DESIGN PRINCIPLES
- DESIGN CONTEXTS
- DESIGN FORCES
- LANGUAGE CHOICE
- GOAL OF DESIGN
- INFRASTRUCTURE COMPONENTS
- BUILD SERVERS
- DESIGN ENVIRONMENTS
- PROVIDE RUNTIME INFRA

PART 3 - DELIVERY

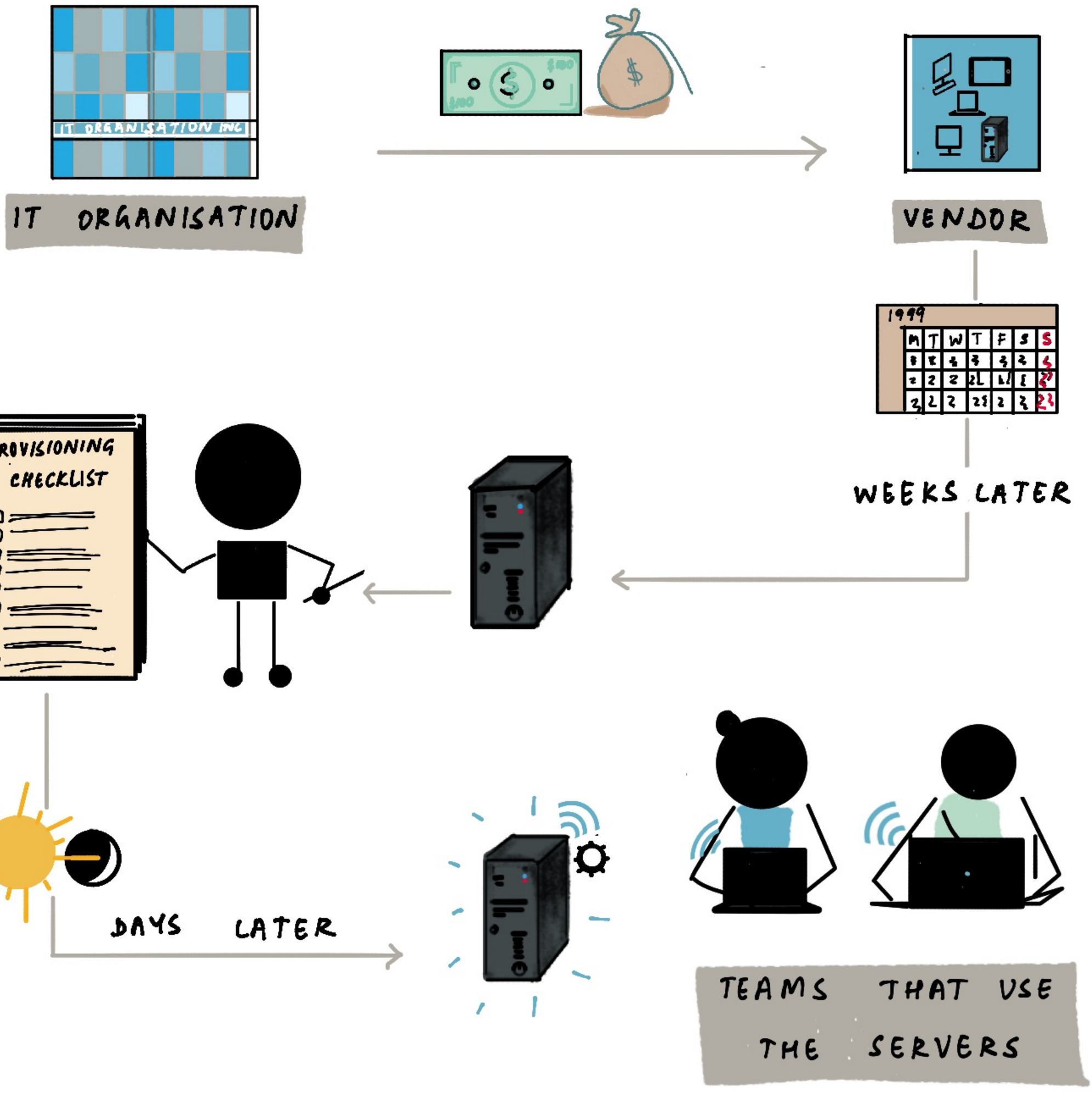
DELIVERY OF INFRASTRUCTURE AS CODE

- CORE WORKFLOWS
- BUILD INFRA AS CODE
- CREATE DELIVERY PIPELINES
- DEPLOY INFRA
- TEST INFRA CODE
- CHANGE EXISTING INFRA
- GOVERNANCE

INTRODUCTION

THE 'IRON AGE'

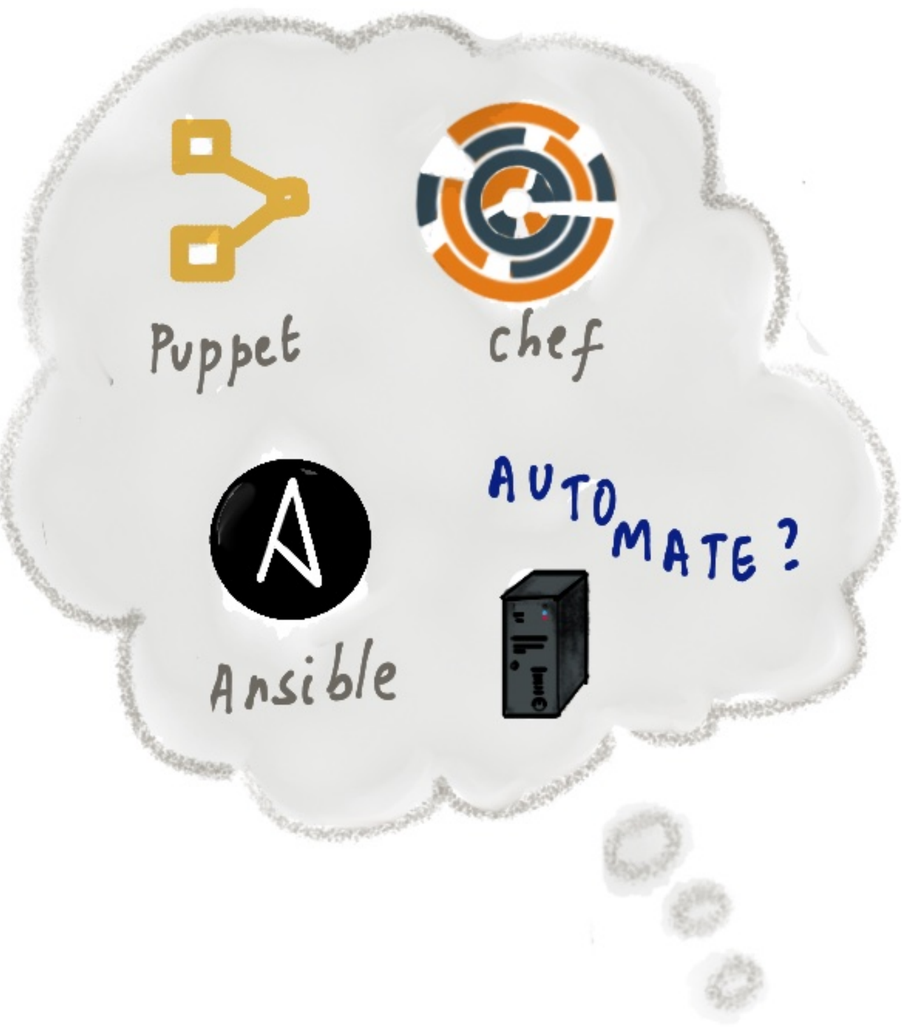
IN THE 'IRON AGE' OF SOFTWARE INFRASTRUCTURE, BEFORE WIDESPREAD CLOUD ADOPTION, DEPENDENCE ON HARDWARE PURCHASING CYCLE LOOKED LIKE

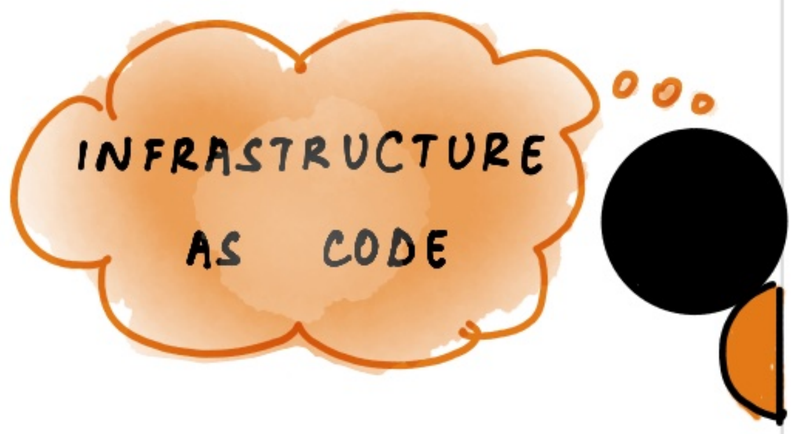


WHILE VIRTUAL MACHINES, SERVER TEMPLATES, AND CLONING HELPED SPEED THINGS UP, THERE WAS A NEED TO CONTINUOUSLY UPDATE THE MANY MANY SERVERS AT WORK.

TOWARDS THE CLOUD AGE

AFTER 2005

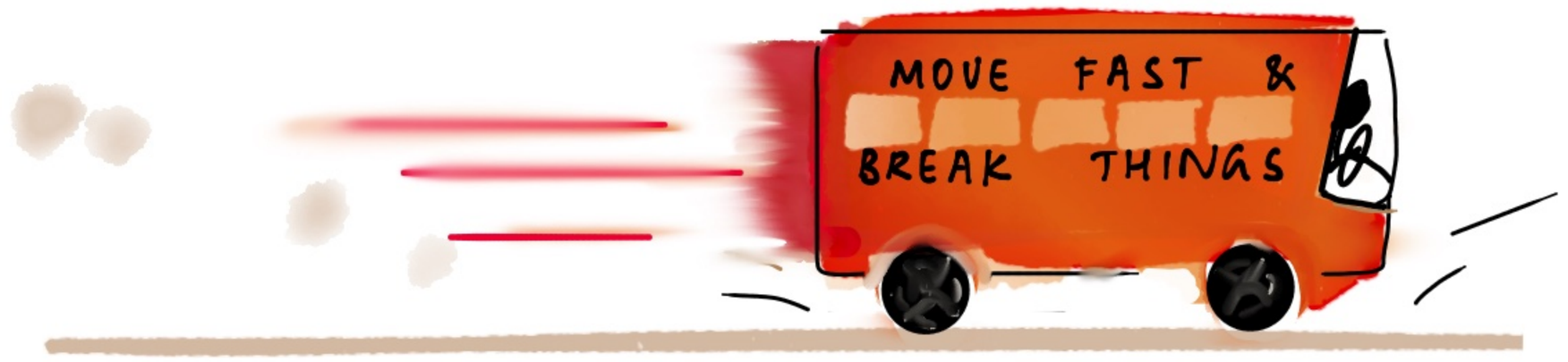




THE CONCEPTS RELATED TO DEVOPS, CLOUD ETC WERE APPEARING.

'SHADOW AGE' OF IT

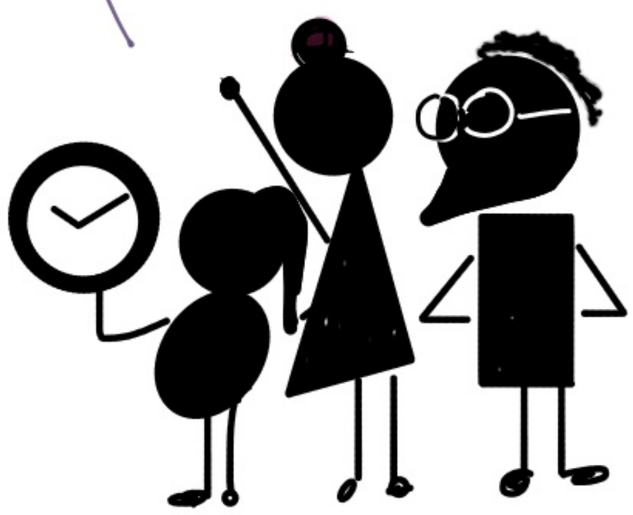
IN THE MID 2010s



STARTUPS!

NEW IDEAS!

DISREGARD THE PROCESS!

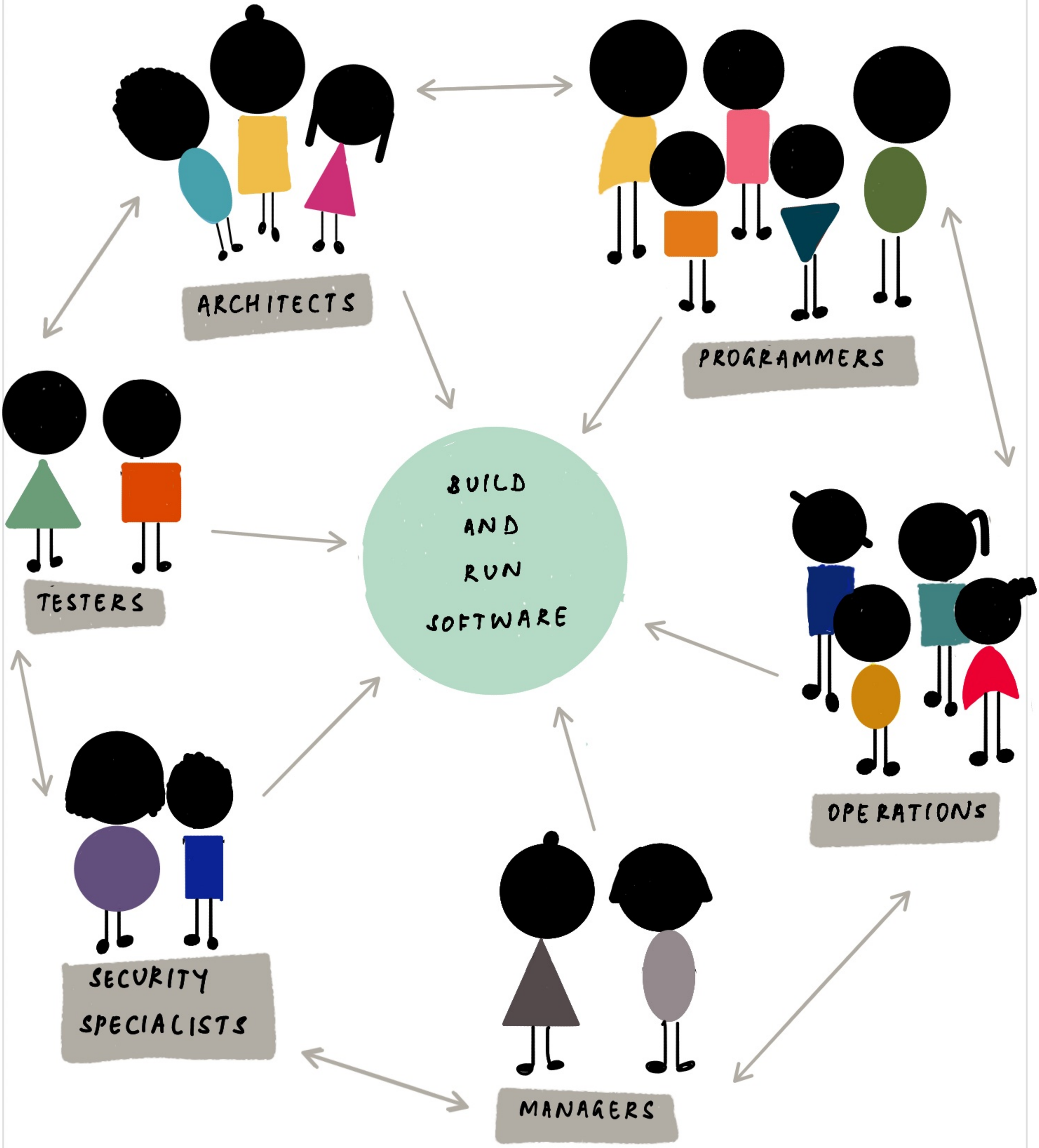


'DIGITAL HIPSTERS'

MAINLY HAPPENED OUTSIDE THE REMIT OF IT ORGANISATIONS AS A WAY TO GET AROUND MORE FORMAL POLICIES

A NOTE ON DEVOPS

DEVOPS IS ABOUT PEOPLE, CULTURE AND WAYS OF WORKING

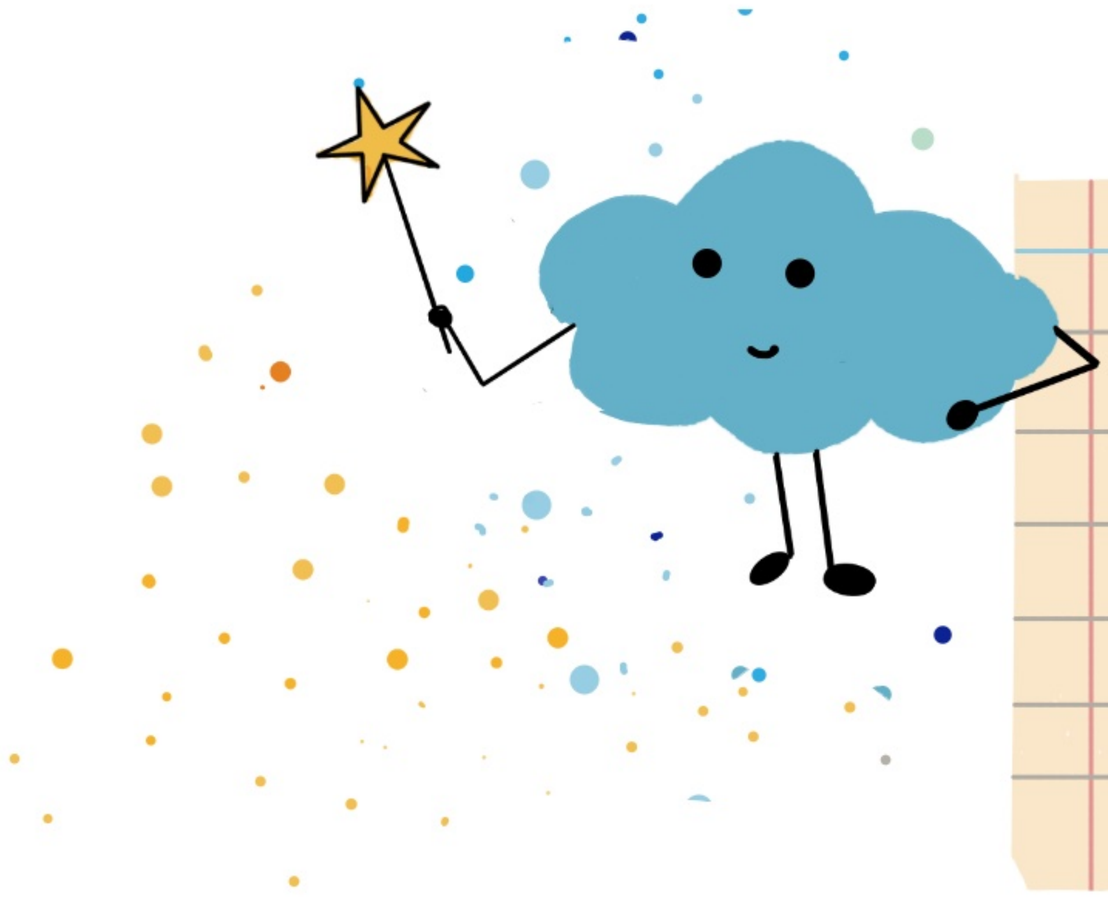


INFRASTRUCTURE AS CODE IS A PRACTICE THAT IS ONE OF THE MANY WAYS TO TO BRIDGE GAPS AND IMPROVE COLLABORATION.

THE CLOUD AGE

THE LATE 2010s

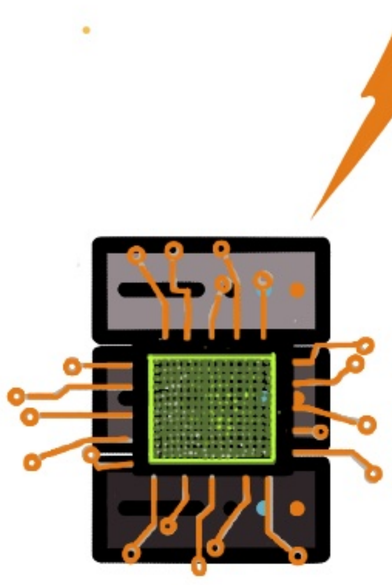
AGE OF SPRAWL



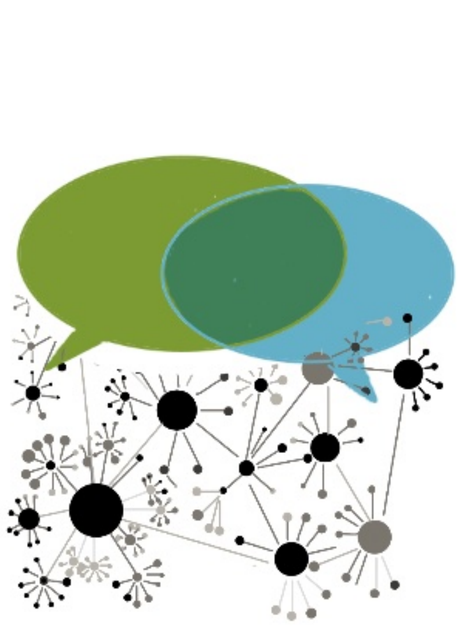
BENEFITS

- Automated ✓
- Fast ✓
- Flexible ✓
- scalable ✓
- Responsive ✓

STORAGE



COMPUTE

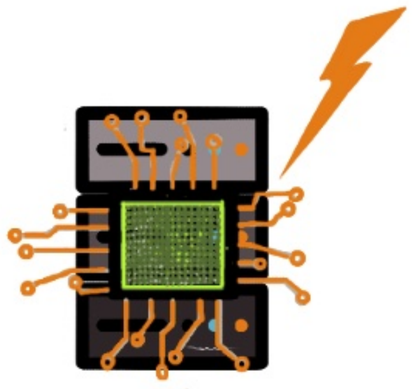


NETWORKING

CLOUD COMPUTING, WHICH WE NOW TAKE FOR GRANTED, HAS CHANGED HOW BUSINESSES AND IT VIEW INFRASTRUCTURE.

CLOUD COMPUTING

CLOUD DECOUPLES

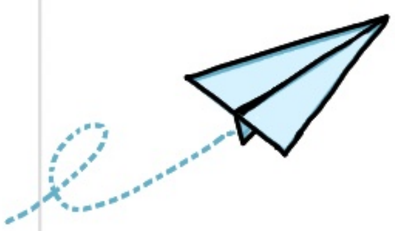


COMPUTING RESOURCES

FROM THE

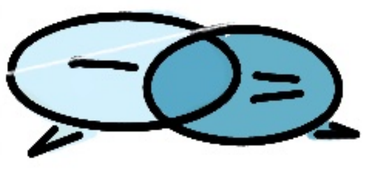
PHYSICAL HARDWARE

SO MUCH SO THAT IT IS NOW POSSIBLE TO DYNAMICALLY



DEFINE INFRASTRUCTURE RESOURCES

SUPPLY LIBRARIES, TOOLS & DEPENDENCIES

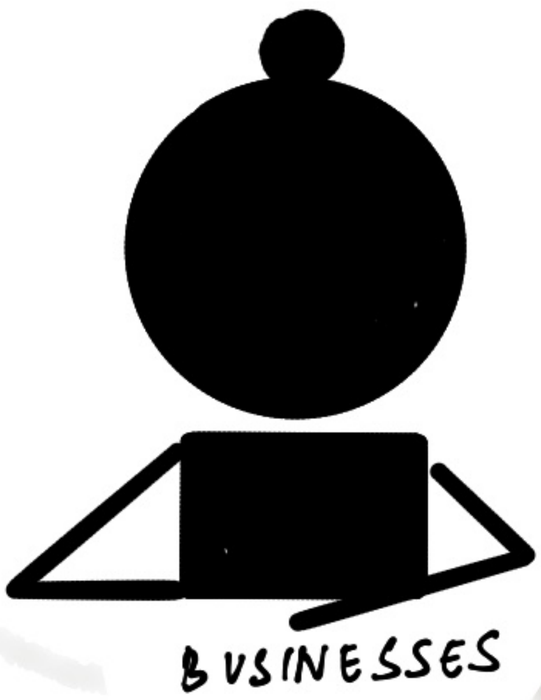


AUTHENTICATE & LET SERVICES COMMUNICATE

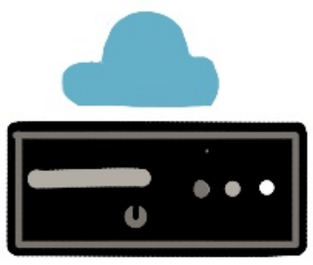
BUSINESSES NOW THINK

WHAT IS INFRASTRUCTURE DELIVERY LIFECYCLE?

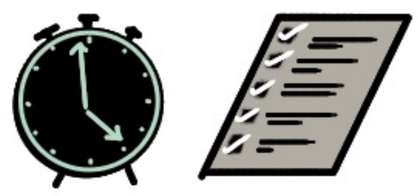
WHAT ARE OUR DAY 2 REQUIREMENTS?



BUSINESSES



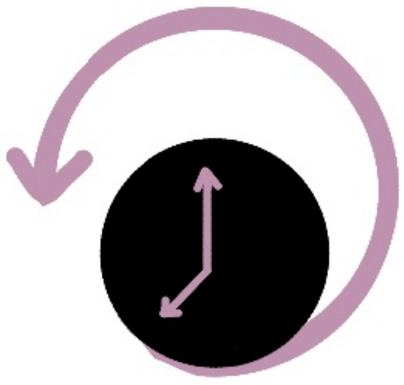
WHEN WOULD IT BE BETTER TO HAVE PHYSICAL HARDWARE?



CHANGES CAN BE MADE ... QUICKLY? OFTEN?

NO NEED TO WAIT WEEKS TO PROVISION

CHANGE OF PLAN IS NOT A FAILURE, IT IS LEARNING



DOES CHANGE CAUSE A FAILURE - AND CAN WE SAFELY ROLLBACK?

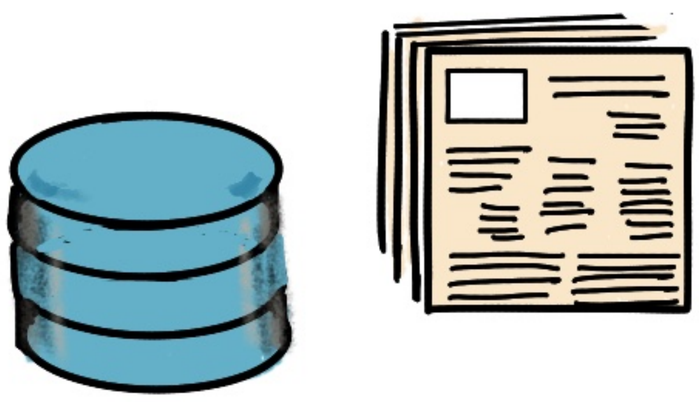
SUCCESS IS WHEN WE CAN RAPIDLY RESTORE SERVICE FROM OUTAGE

WHAT IS INFRASTRUCTURE?

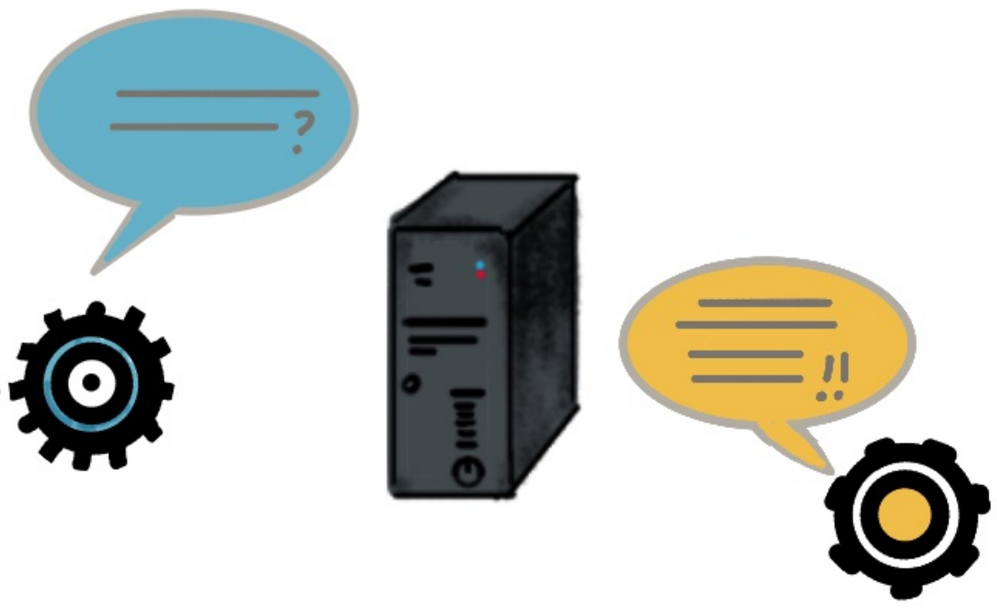
INFRASTRUCTURE COULD BE

INFRASTRUCTURE IS WHAT SYSTEMS RELY ON
FOR SETUP AND OPERATION

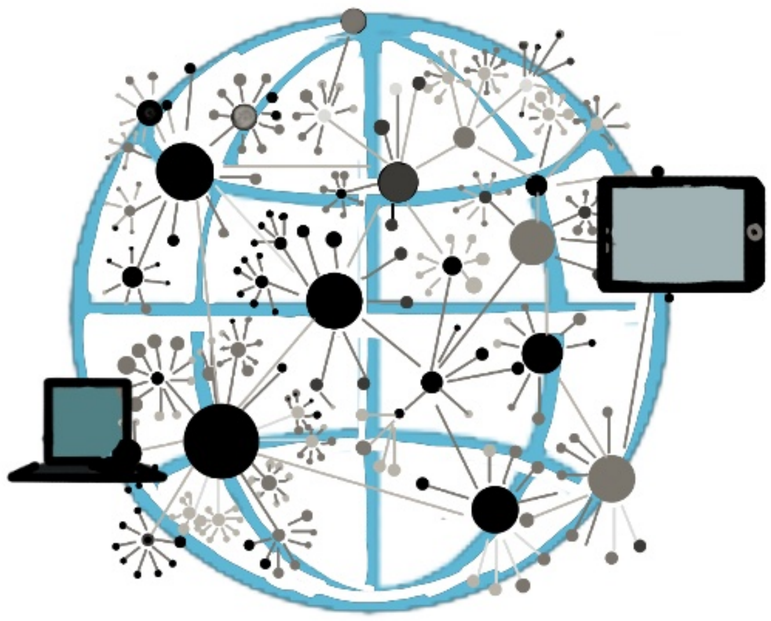
OPERATING SYSTEMS



DATA STORAGE



MESSAGING SERVICES



NETWORK CONNECTIVITY

SCOPE OF INFRASTRUCTURE

PEOPLE DEFINE THE SCOPE OF INFRASTRUCTURE IN MANY WAYS.

TO PUT IT IN CONTEXT OF A WHOLE SYSTEM VISUALISED HERE:

APPLICATION LAYER

USER SERVICES

ENGINEERING PLATFORM LAYER

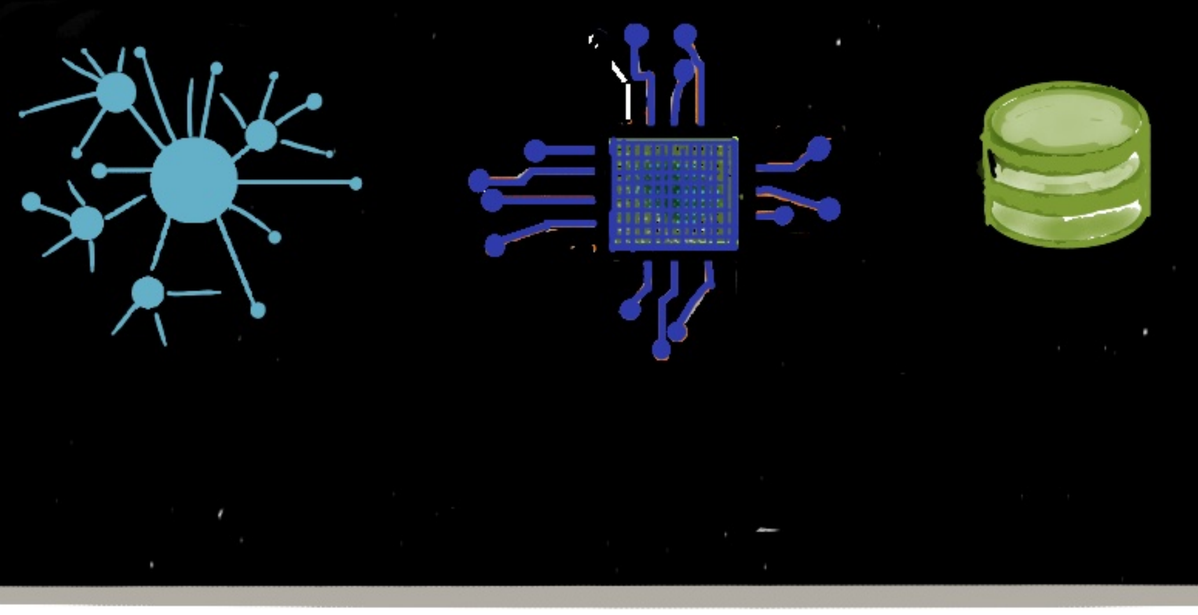
DELIVERY SERVICES

APPLICATION RUNTIME SERVICES

OPERATIONAL SERVICES

PLATFORM SERVICES

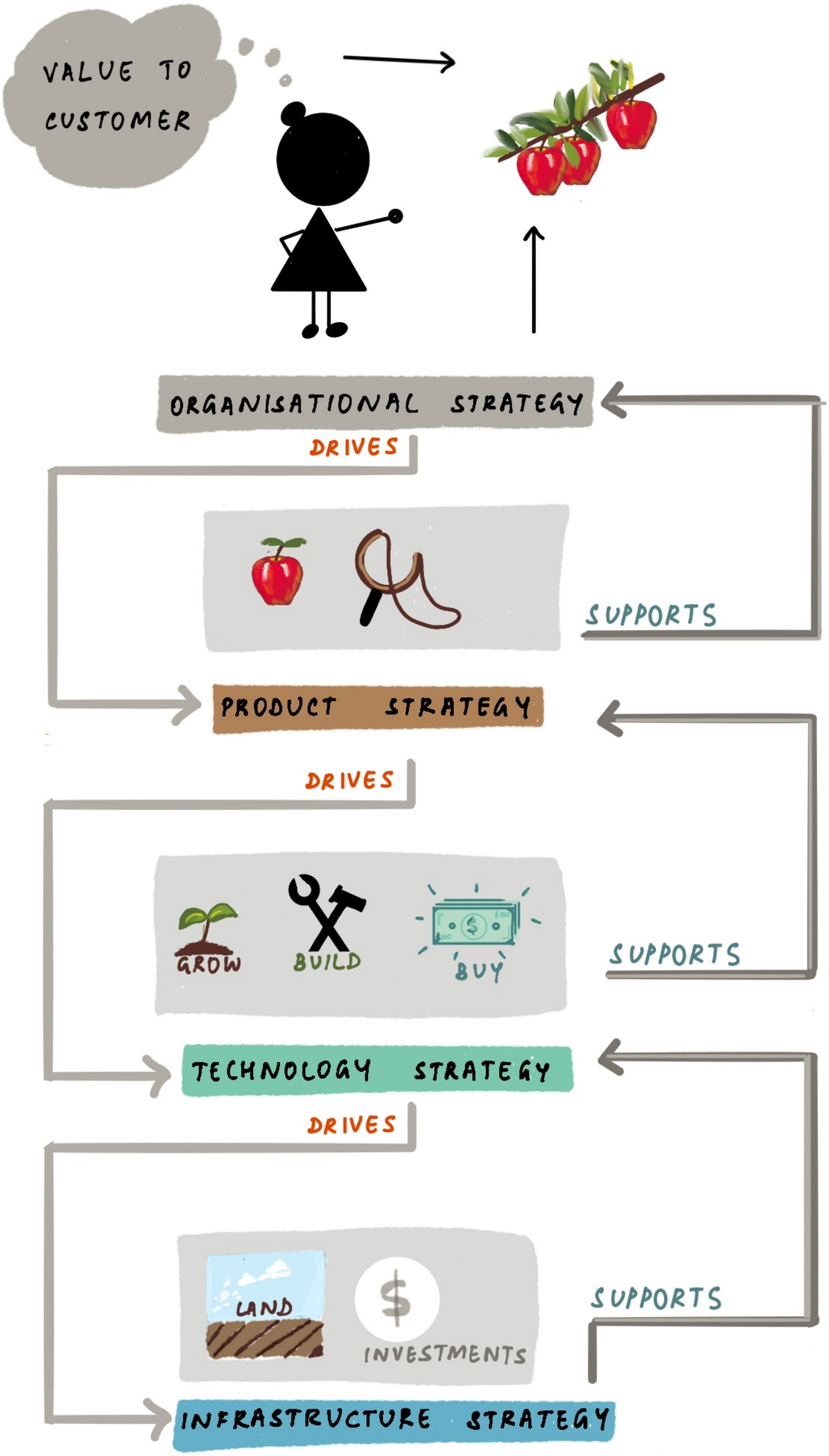
INFRASTRUCTURE PLATFORM LAYER



THIS BOOK USES 'INFRASTRUCTURE' TO DESCRIBE THE RESOURCES PROVIDED BY THE INFRASTRUCTURE PLATFORM THAT CAN BE DEFINED, SETUP AND CONFIGURED USING CODE.

WHY IS INFRASTRUCTURE IMPORTANT?

INFRASTRUCTURE & ORG GOALS



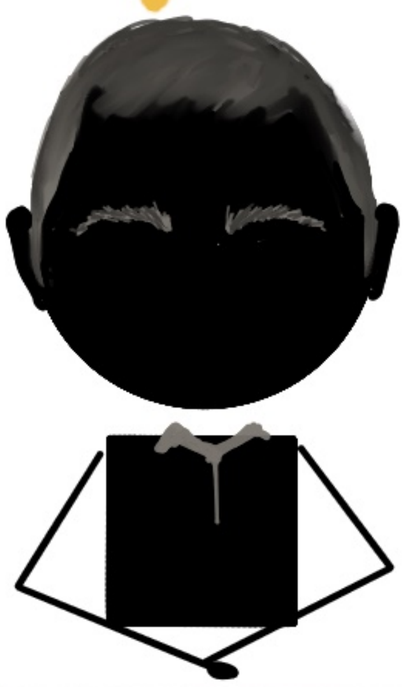
WHAT IS INFRASTRUCTURE AS CODE?

DEFINITION

WHAT IS INFRASTRUCTURE AS CODE?

THE PRACTICE OF PROVISIONING AND MANAGING INFRASTRUCTURE USING CODE

. . . INSTEAD OF USING A GUI OR A COMMAND LINE INTERFACE



KIEF MORRIS

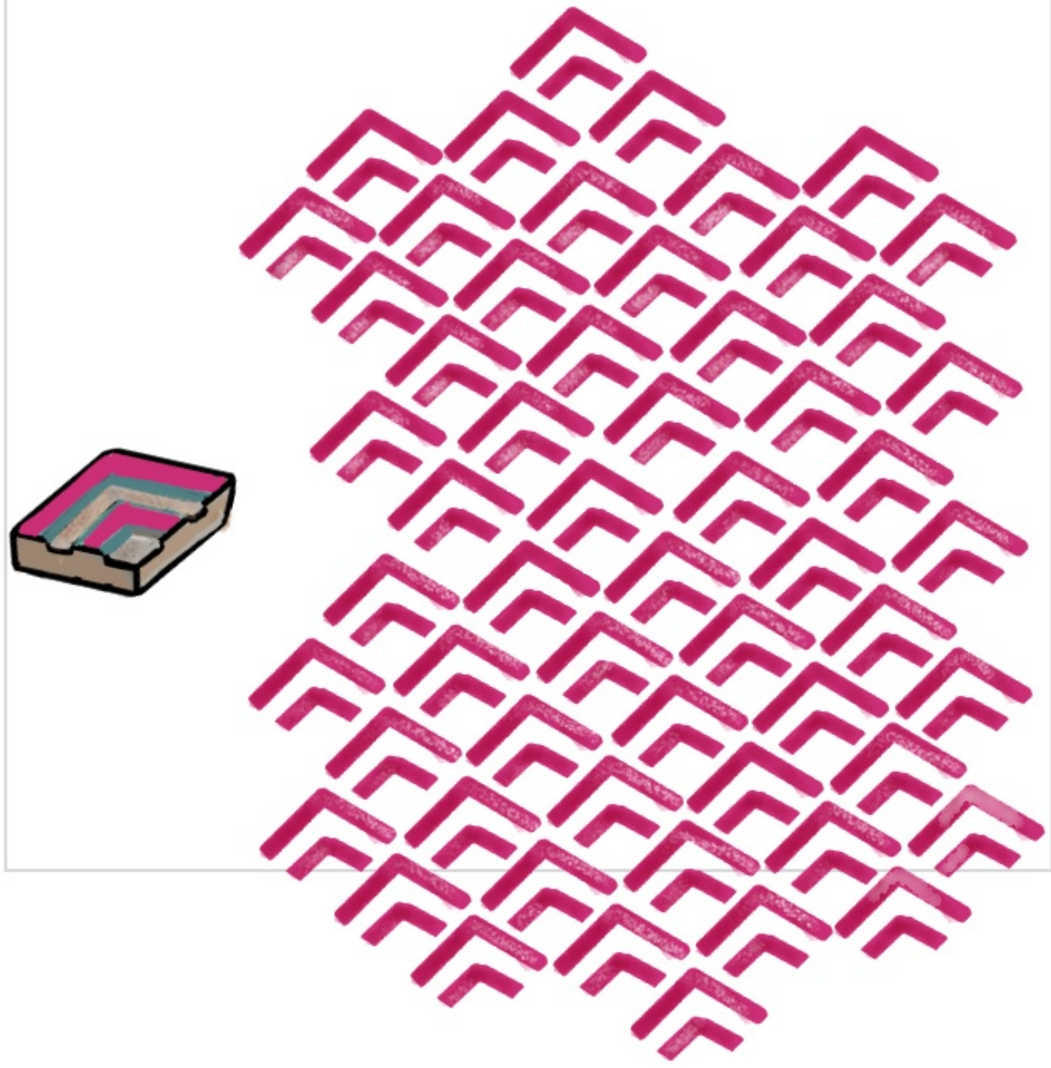
3 CORE PRACTICES

1. Define everything as code
2. Continuously test and deliver all work in progress
3. Build small, simple pieces that can be changed independently

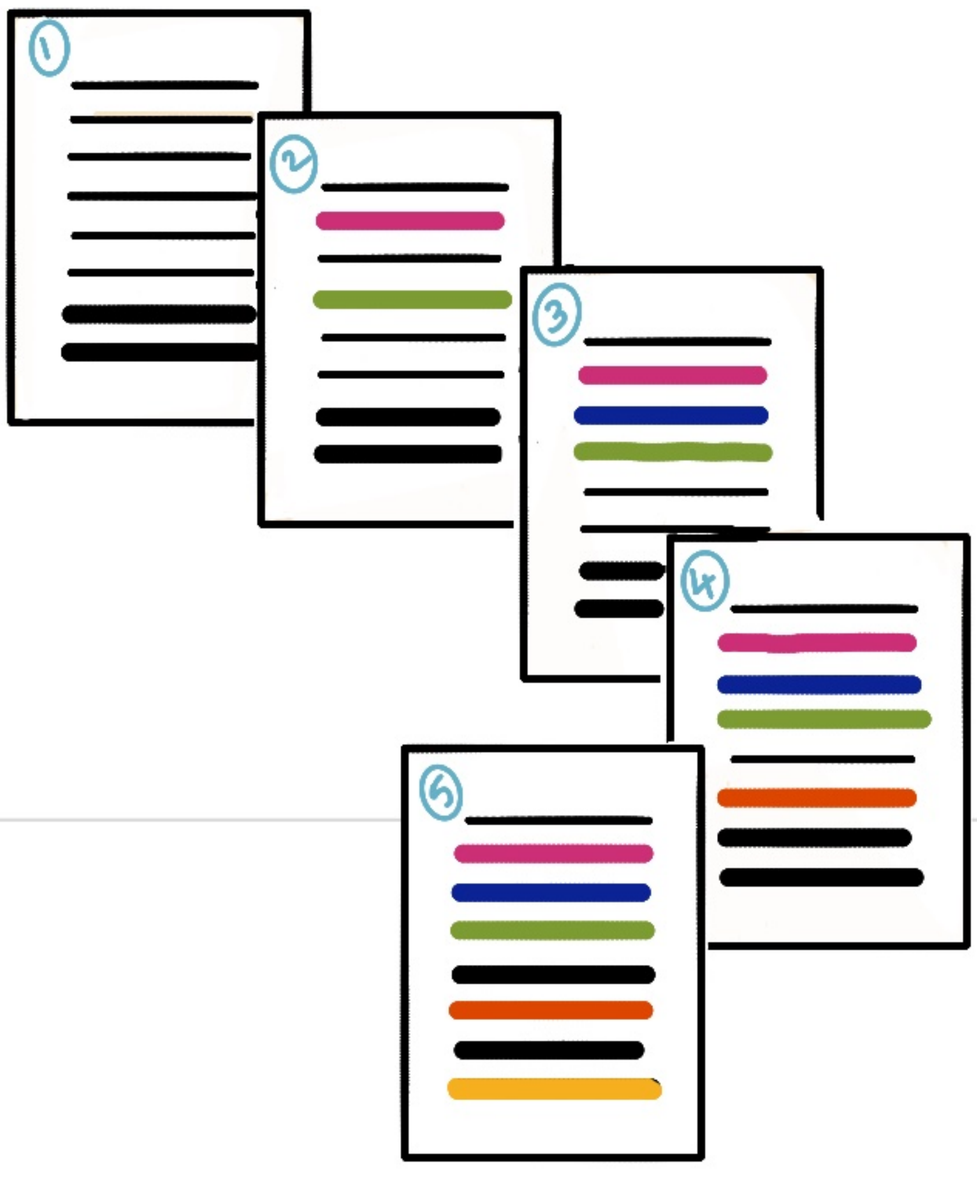
CODE - BECAUSE ...

REUSABILITY

CONSISTENCY

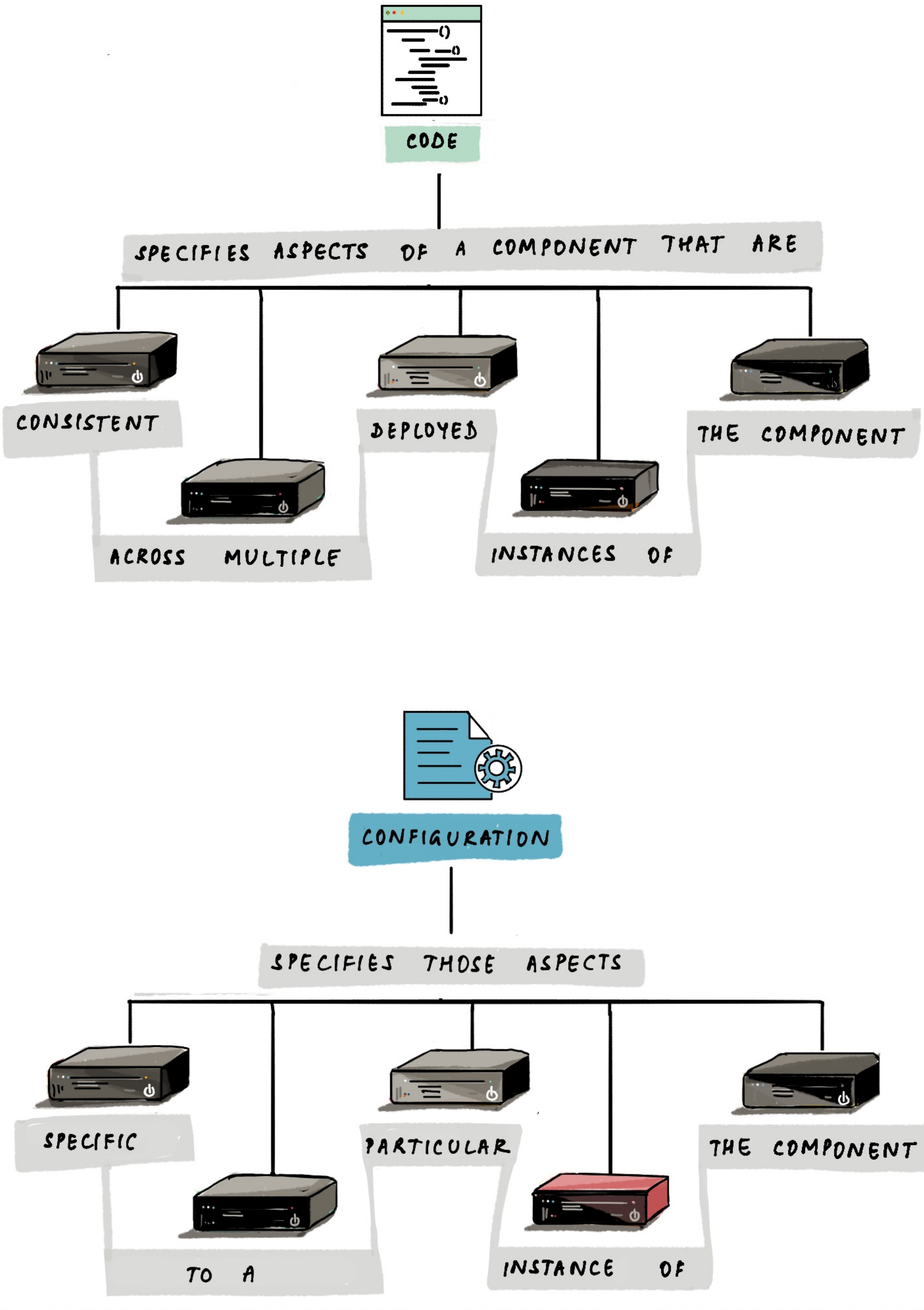


VERSION CONTROL

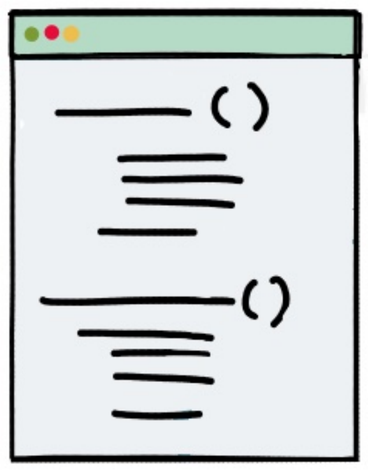


ABILITY TO ANALYSE

CODE OR CONFIGURATION ?



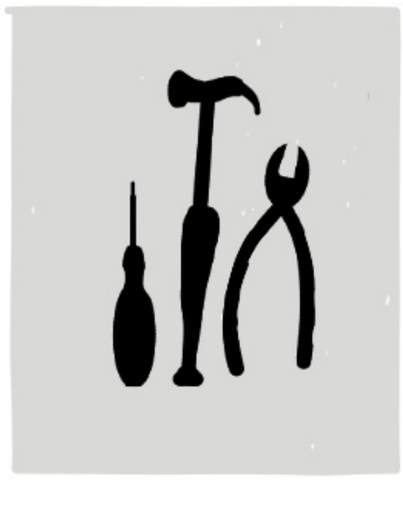
FROM CODE TO ENVIRONMENT



INFRASTRUCTURE

CODE

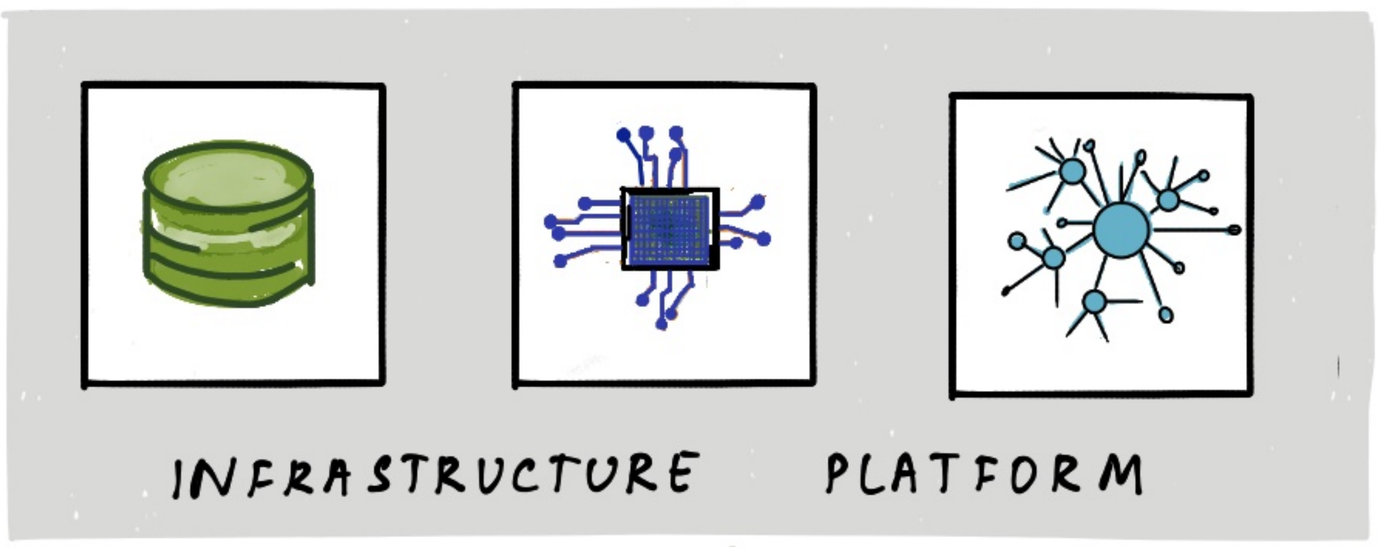
IS READ BY



INFRASTRUCTURE

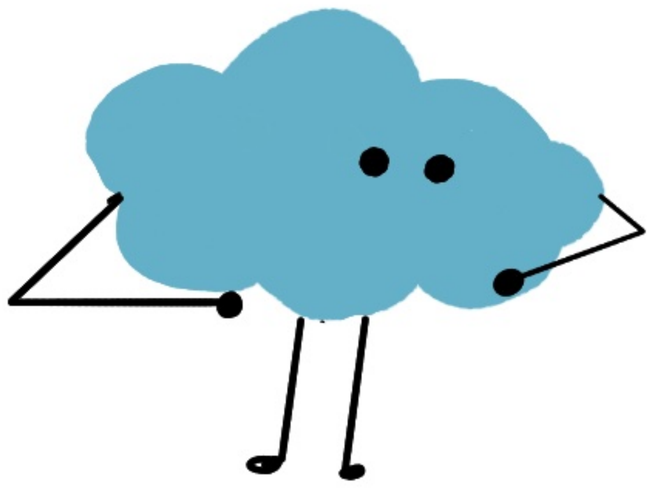
CODE TOOL

CONNECTS TO API



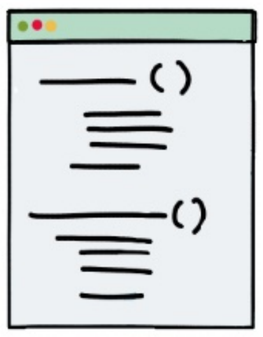
PROVISIONS INFRASTRUCTURE

WHY ADOPT INFRA AS CODE?

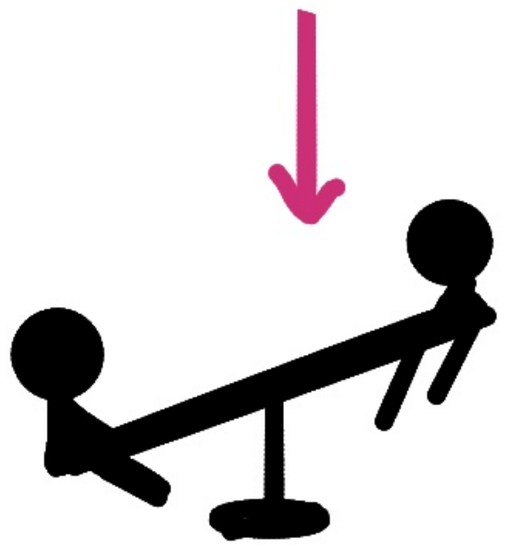


CLOUD AND INFRASTRUCTURE AUTOMATION

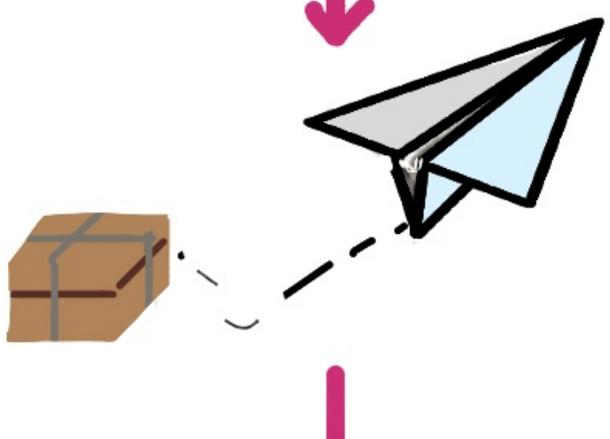
ENABLE ORGANISATIONAL CHANGE



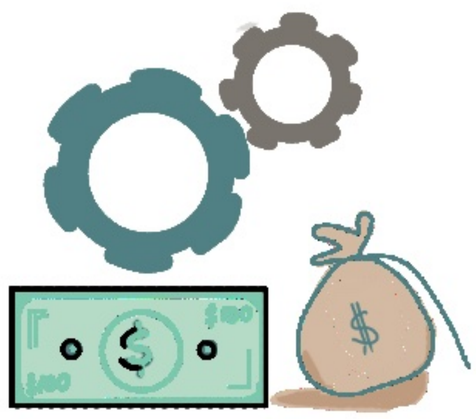
INFRASTRUCTURE AS CODE



OPTIMISES THE PROCESS FOR MAKING CHANGES TO IT SYSTEMS



DELIVERING CHANGES RAPIDLY & RELIABLY



TO CREATE RESILIENT, HIGHLY-AVAILABLE VALUABLE SYSTEMS

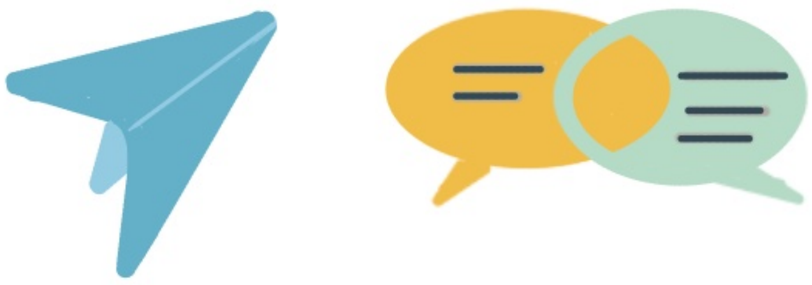
MYTHS ABOUT AUTOMATING CHANGE

MYTH - 1

MYTH : INFRASTRUCTURE DOESN'T CHANGE VERY OFTEN.

REALITY: VERY FEW SYSTEMS STOP CHANGING BEFORE THEY ARE RETIRED.

EXAMPLES OF CHANGE BEING CONTINUOUS IN A SYSTEM



UPGRADE TO A NEW MESSAGING SERVICE

NEW SERVICE FOR DATA PROCESSING

PATCH CLUSTERS AGAINST A SECURITY VULNERABILITY

REDEPLOY APP TO IMPROVE PERFORMANCE

HEAVY WEIGHT CHANGE PROCESS

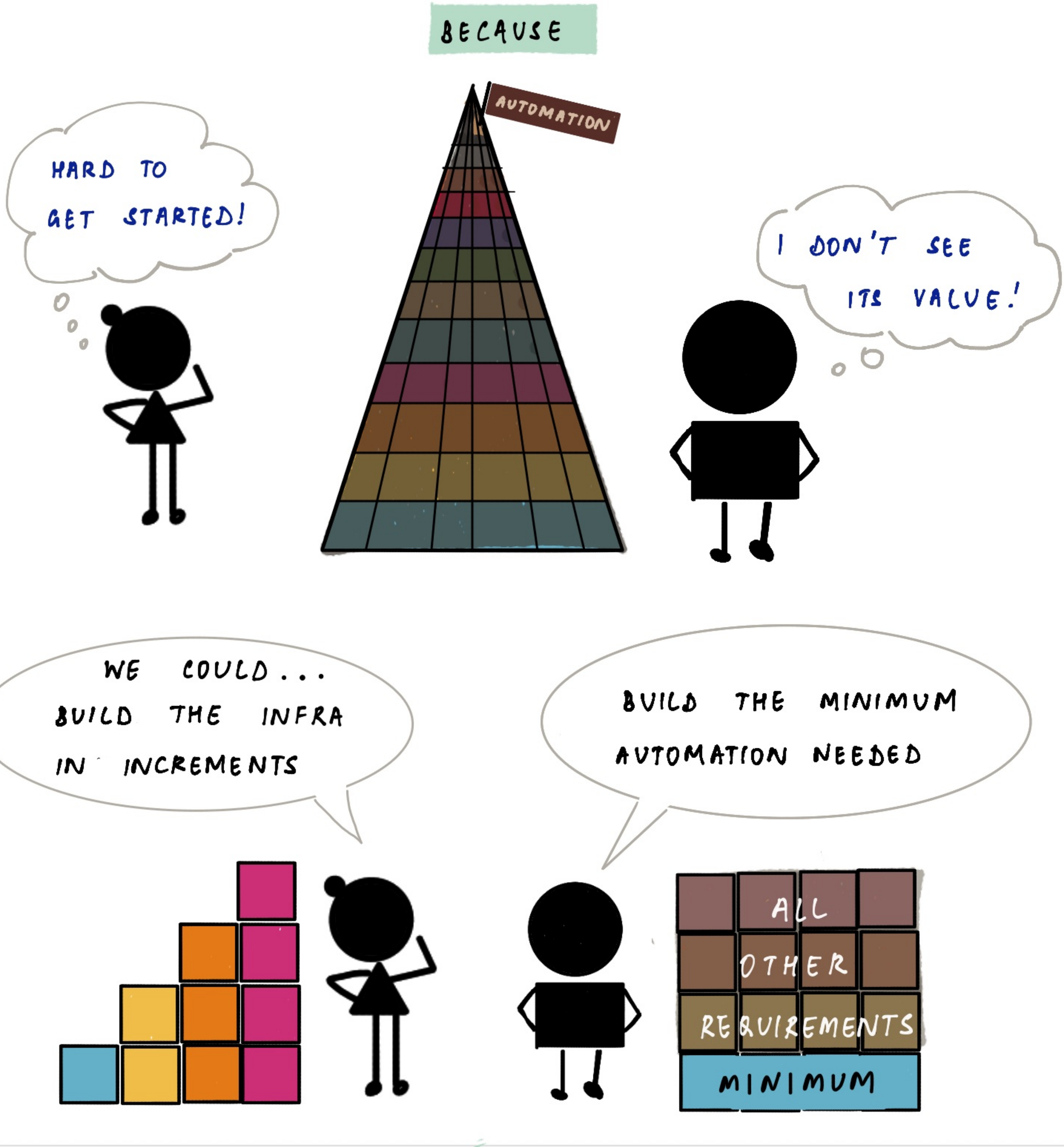
BACKLOG ACCUMULATES

ORGANISATION'S ABILITY TO ADAPT IMPACTED

MYTH - 2

MYTH : WE CAN BUILD THE INFRASTRUCTURE FIRST AND AUTOMATE IT LATER

REALITY: AUTOMATING AN EXISTING SYSTEM IS HARD. IT IS EASIER TO AUTOMATE AS WE BUILD

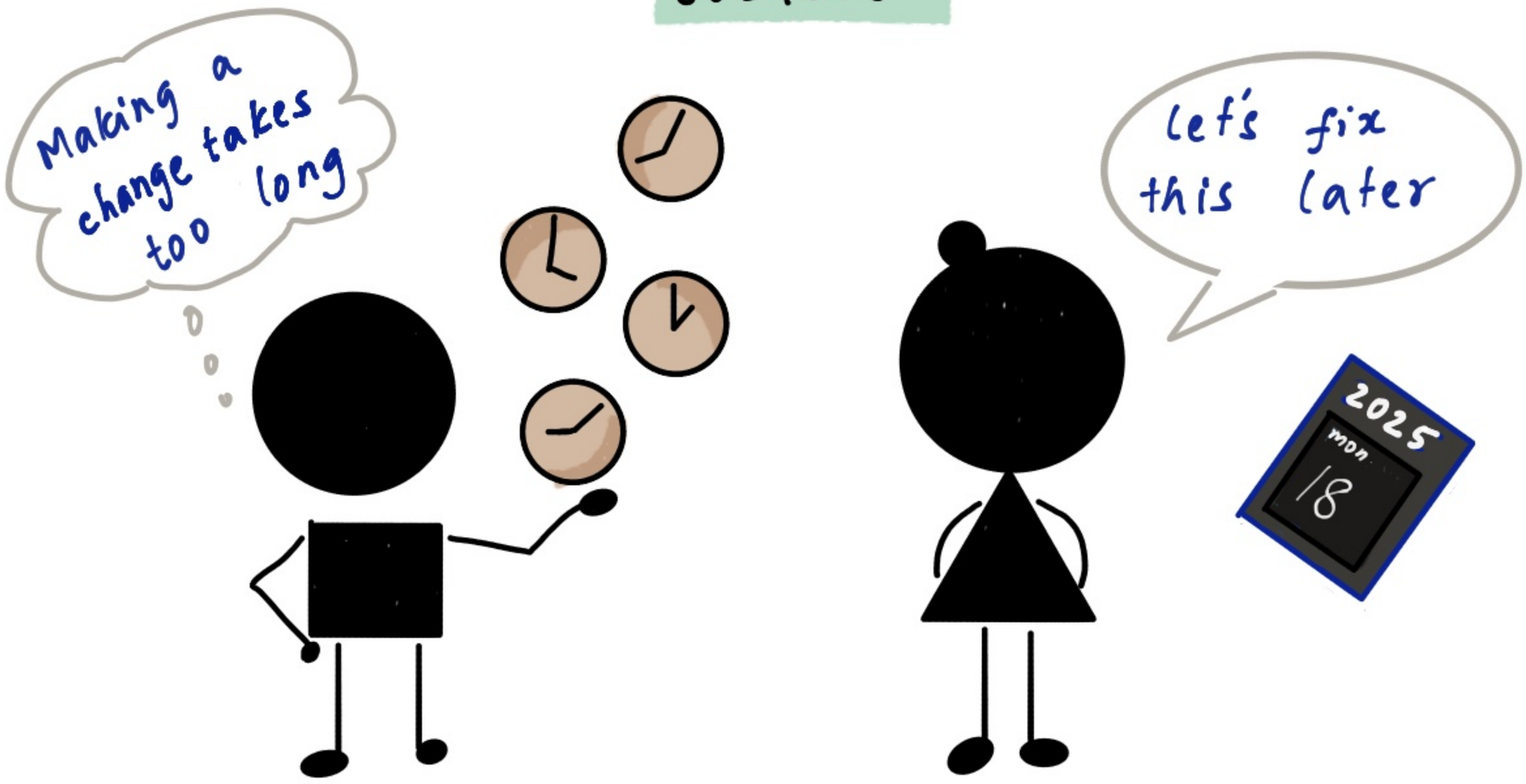


MYTH - 3

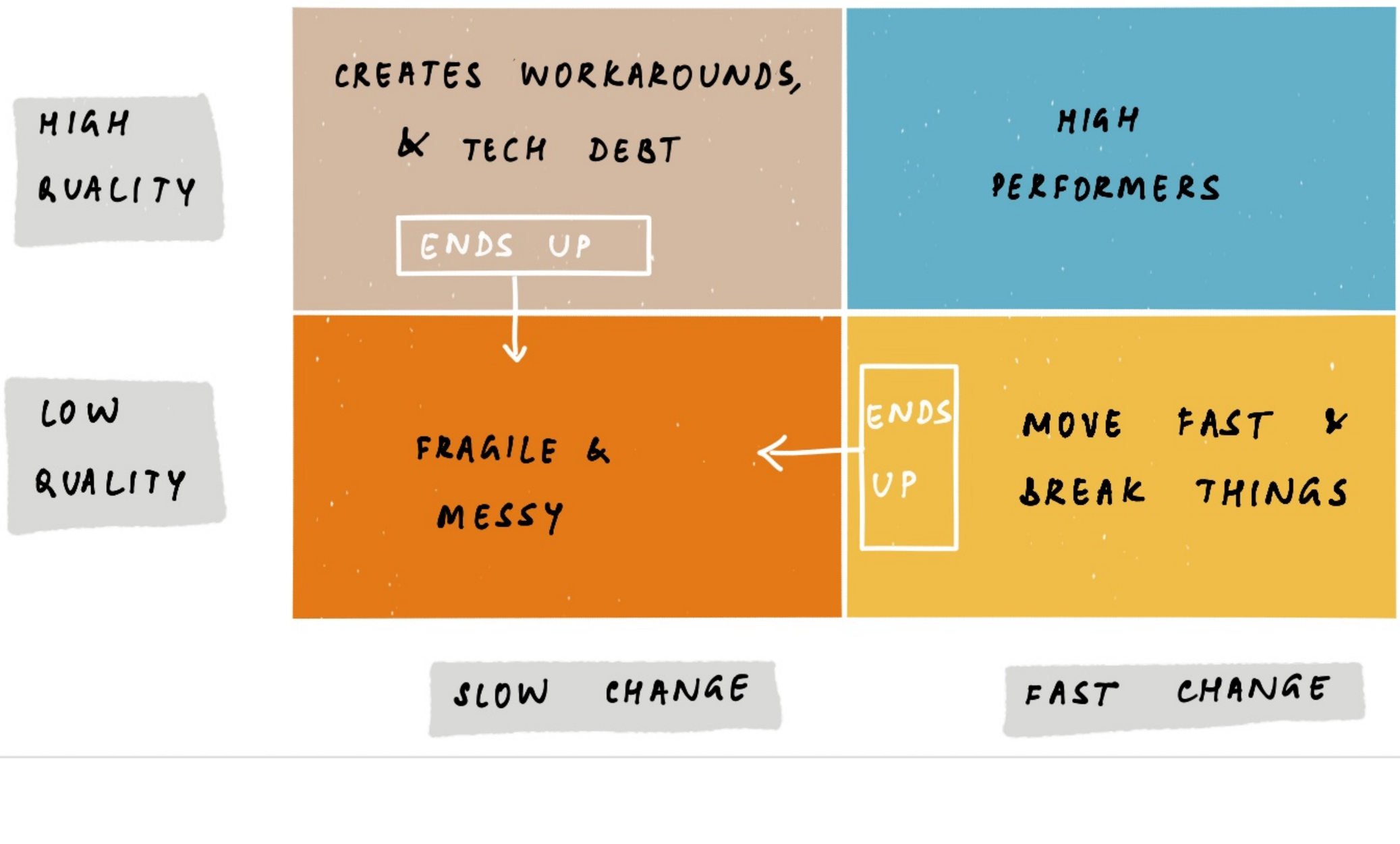
MYTH — QUALITY COMES FROM MOVING SLOWLY

REALITY — IT IS NECESSARY TO BE GOOD AT CHANGE TO CREATE STABILITY.

BECAUSE



IT IS MORE HELPFUL TO SEE QUALITY & SPEED AS A QUADRANT



DESIGN PRINCIPLES

A WELL-DESIGNED SYSTEM

A case for good design

GIVEN THAT INFRASTRUCTURE PHYSICAL ATTRIBUTES ARE VARIABLE,

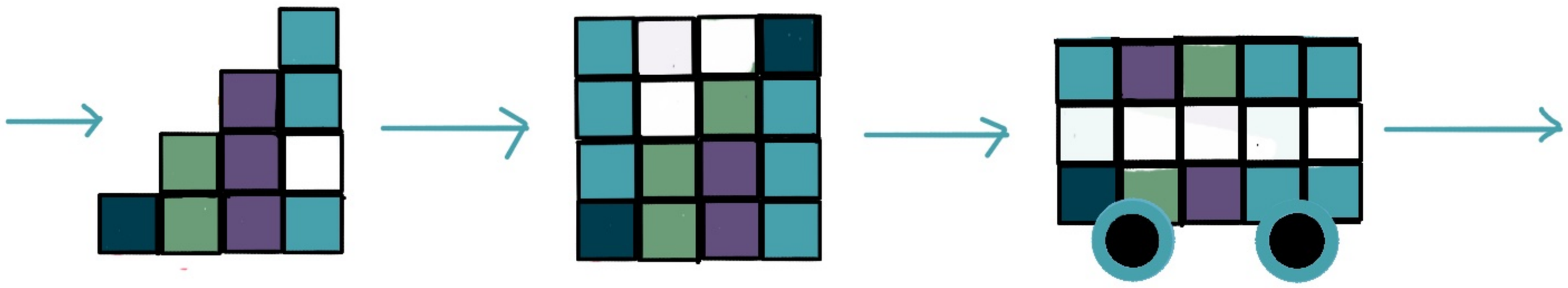
AS A TEAM MEMBER,

I WANT TO

- ALTER SYSTEM INSTANCES OR COMPONENTS
- KEEP THE SYSTEM QUALITY

SO THAT WE CAN SCALE EASILY & RAPIDLY

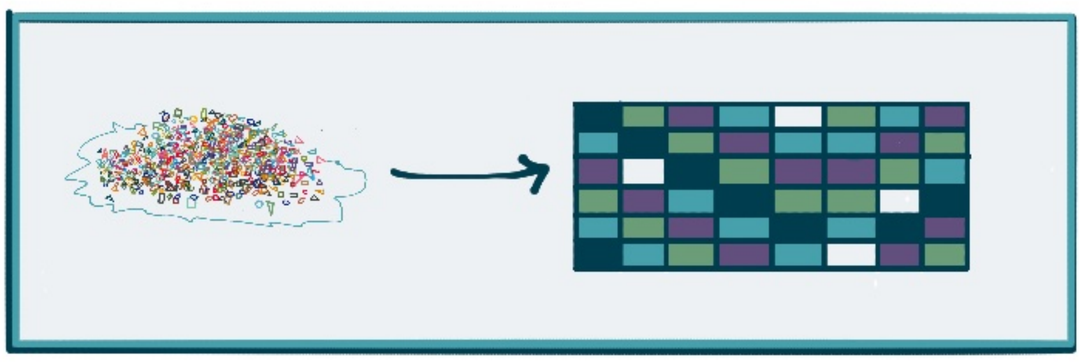
GOOD SYSTEM DESIGN



A GOOD SYSTEM DESIGN LEADS TO IMPLEMENTATION THAT WORKS WELL AND CAN EVOLVE CONTINUOUSLY.

A DESIGN THAT TRIES TO ANTICIPATE ALL PRESENT AND FUTURE NEEDS ENDS UP BEING FAR TOO COMPLEX.

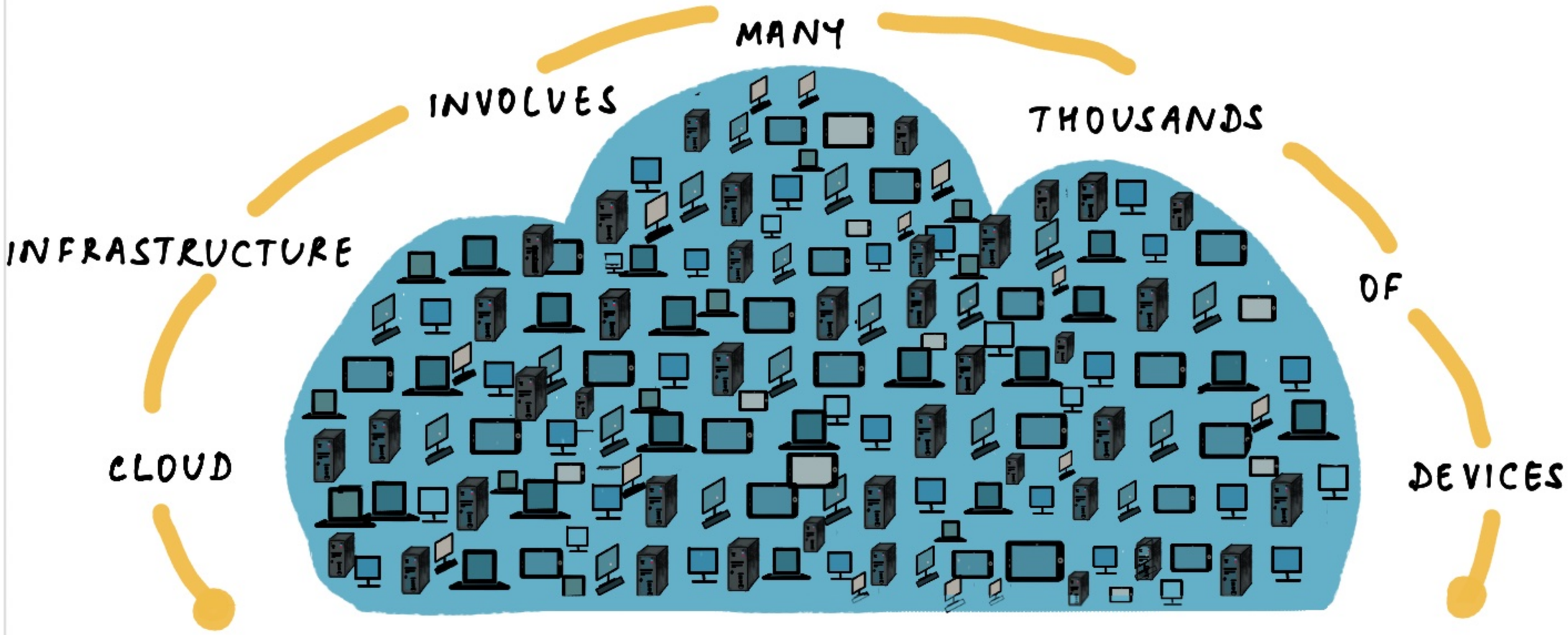
THE GOAL OF INFRASTRUCTURE DESIGN THEN SHOULD BE



SIMPLIFYING MAINTENANCE AND CHANGES.

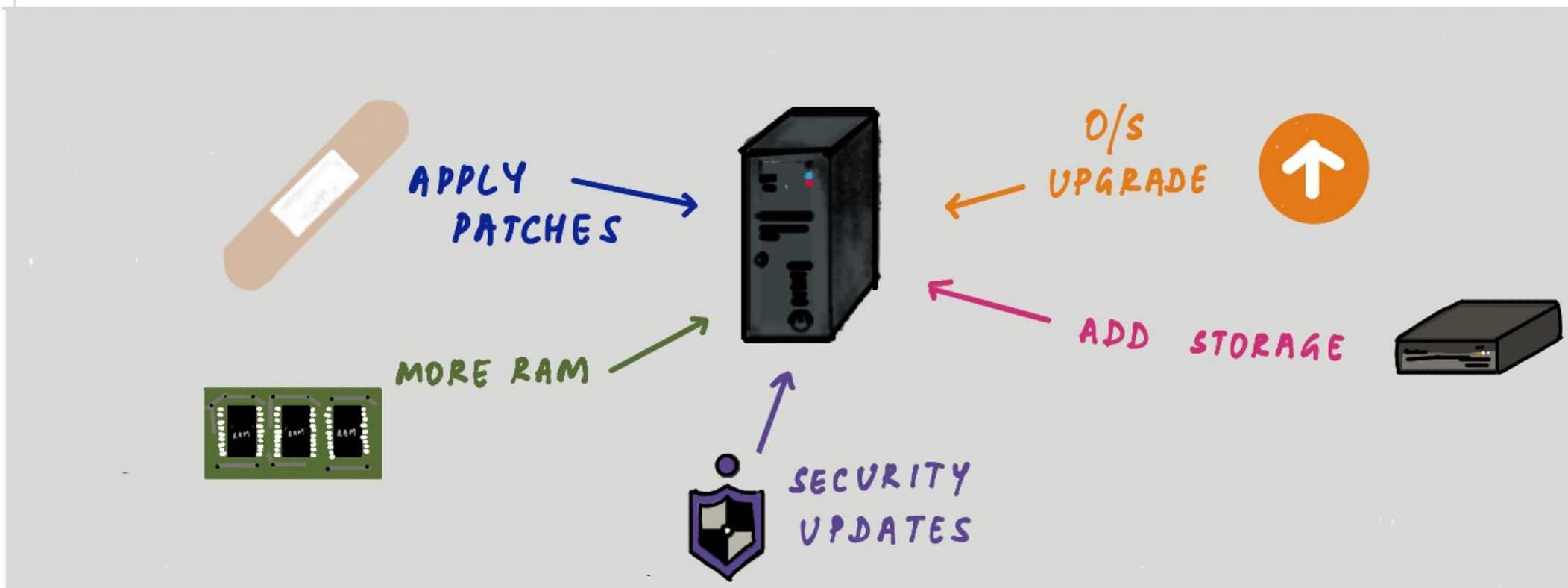
MUCH LIKE THE PRINCIPLES OF GOOD SOFTWARE DESIGN.

ASSUME SYSTEMS ARE UNRELIABLE



THESE REQUIRE FREQUENT MAINTENANCE -

OFTEN REQUIRING SYSTEMS TO GO OFFLINE.

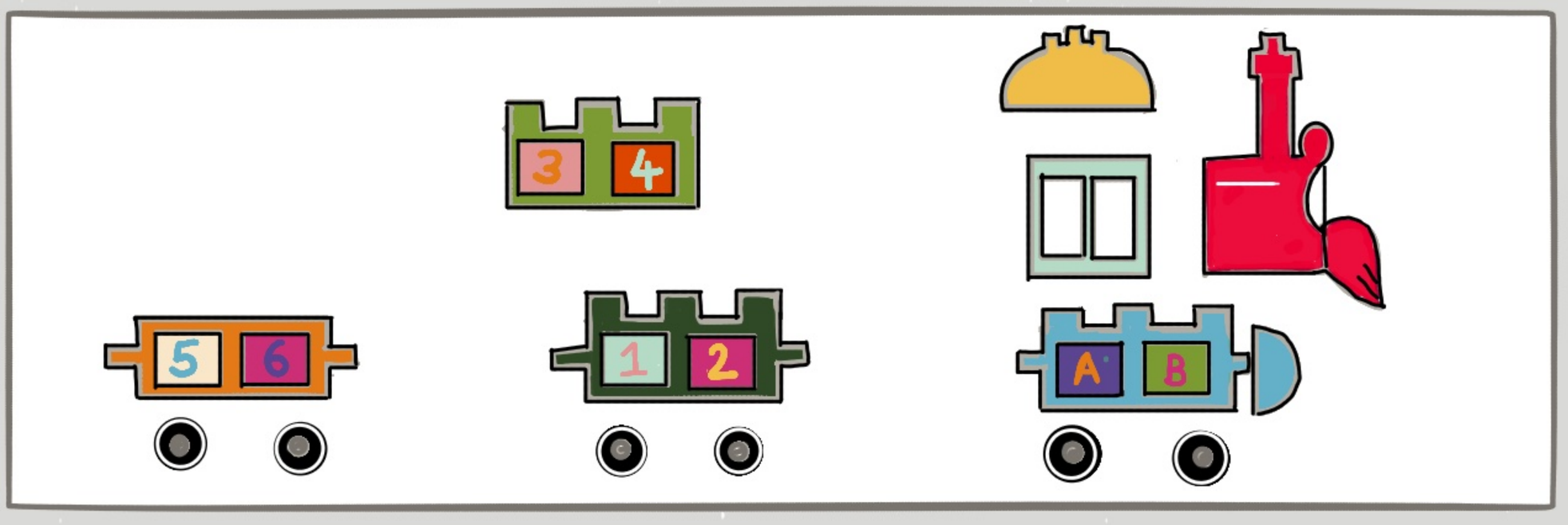


GOING OFFLINE MIGHT MEAN TAKING A BUSINESS OFFLINE.

TO AVOID THIS, WE MUST DESIGN FOR UNINTERRUPTED SERVICE.

i.e. COPE WITH DYNAMIC INFRASTRUCTURE.

MAKE EVERYTHING REPRODUCIBLE



A SYSTEM IS **RESILIENT** WHEN WE CAN REBUILD PARTS OF IT EFFORTLESSLY AND RELIABLY AS WELL AS ROLL BACK TO AN OLDER VERSION.

BENEFITS

- ✔ KEEP ENVIRONMENTS CONSISTENT
- ✔ REPLICATE WHOLE SYSTEMS
- ✔ ADD INSTANCES
  - TO COPE WITH DEMAND
  - FOR INDIVIDUAL CUSTOMERS

REMEMBER TO KEEP SYSTEM-GENERATED DATA, CONTENT & LOGS ALSO AVAILABLE

THE OPPOSITE OF A **RESILIENT** SYSTEM IS A **SNOWFLAKE** SYSTEM

THIS REPRODUCIBILITY REDUCES RISK AND FEAR OF CHANGE.

IT ALSO ENABLES SPEEDY ENVIRONMENT PROVISIONING

CREATE DISPOSABLE THINGS

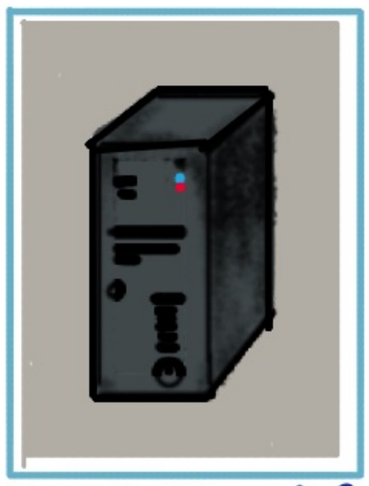
CONSIDER BUILDING A SYSTEM THAT IS ITSELF DYNAMIC

Svr2 Intel Core 2 Quad 2.4Hz 4 core 16GB

95W Peak/125W Idle/65W

$ 1400

− 1 +



Svr4 ICQ1G9

INFO

Status

Response 31ms

Connections 7

Server SLOW

SERVICE

Start

Stop

FEATURES

Edit

Move

SET UP AN ALERT WHEN A SERVER IS REPLACED

1

COPE WELL WITH A DISAPPEARING SYSTEM

SET UP AN ALERT IF SOMETHING LOOPS REBUILDING ITSELF

1

BEING ABLE TO START, STOP, EDIT AND MOVE PARTS OF A SYSTEM CREATES FLEXIBILITY AND DERISKS CHANGES.

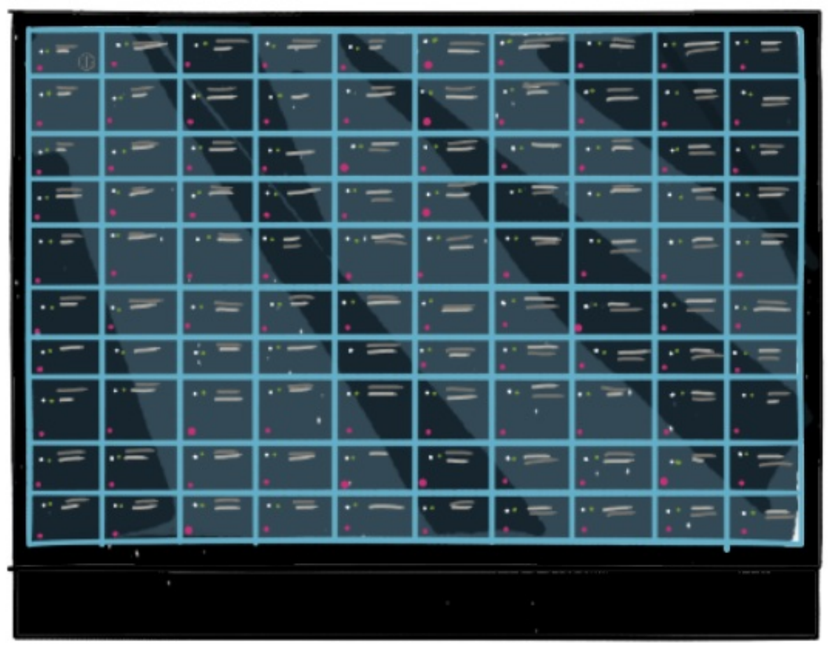
MINIMISE VARIATION

IT IS EASIER TO MANAGE

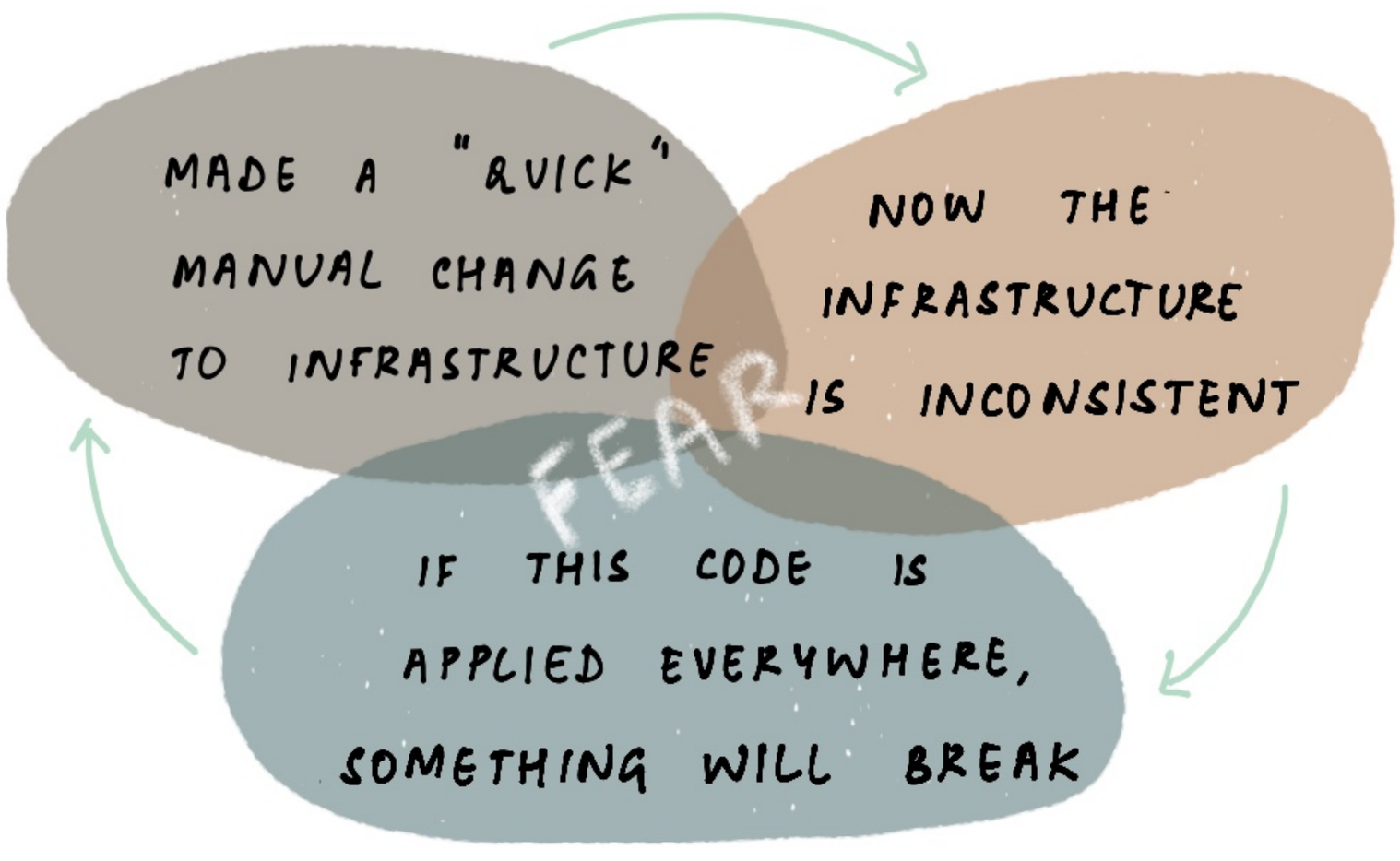
A 100 IDENTICAL SERVERS

THAN

5 DIFFERENT ONES



WHAT MIGHT VARY
OPERATING SYSTEMS
KUBERNETES DISTRIBUTIONS
SOFTWARE VERSIONS
PACKAGE VERSIONS



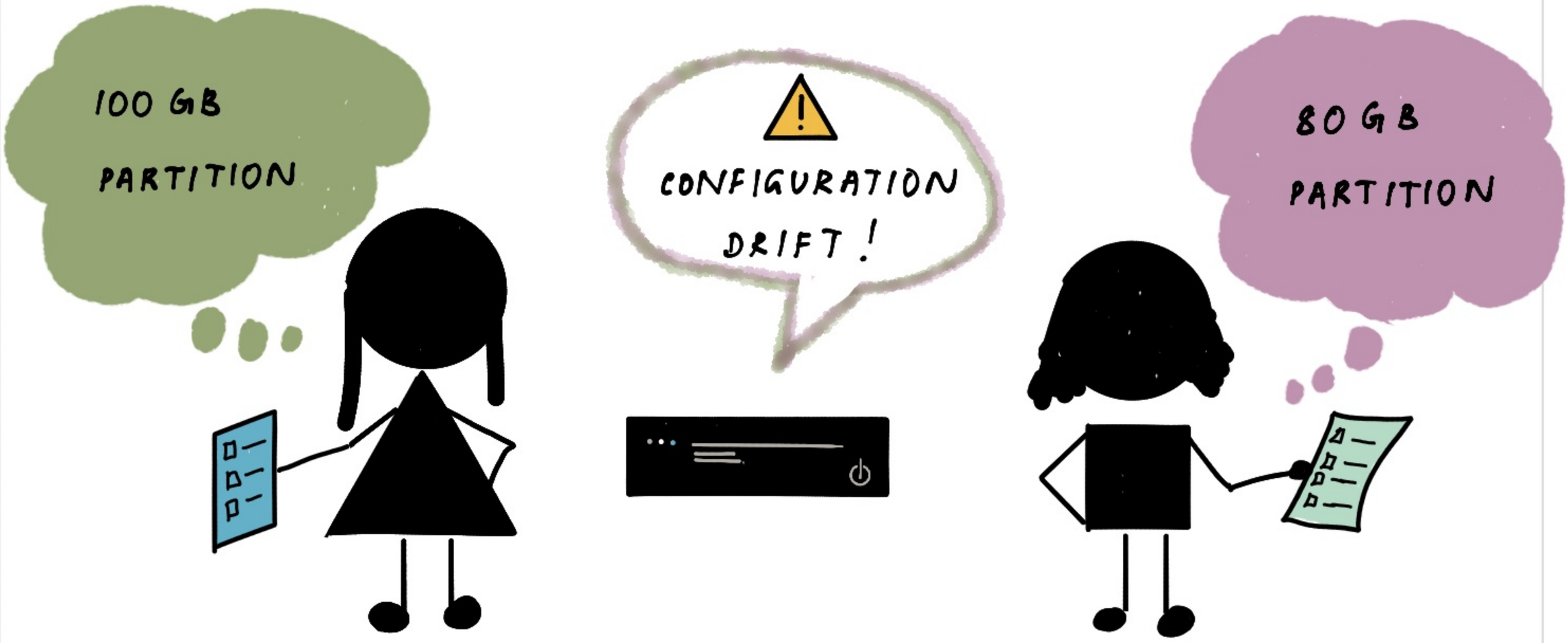
KEEP AWARE OF THE NECESSARY VARIATIONS AND THE UNINTENDED VARIATIONS IN YOUR SYSTEMS.

ENSURE THAT ANY PROCEDURE CAN BE REPEATED

JUST LIKE THE REPRODUCIBILITY PRINCIPLE,

IT SHOULD BE EASY TO

REPEAT ANY ACTION RELIABLY



IF POSSIBLE

SCRIPT THE TASK

ELSE

SIMPCIFY THE TASK

SCRIPT THE TASK

ELSE

BREAKDOWN TASK

SCRIPT EACH TASK

APPLY SOFTWARE DESIGN PRINCIPLES TO INFRASTRUCTURE

THIS PRINCIPLE NEEDS TO BE TREATED WITH CARE

AS THERE ARE DIFFERENCES BETWEEN

INFRASTRUCTURE AND SOFTWARE CODE

THEY ARE AS RELEVANT FOR INFRASTRUCTURE CODE

AS THEY ARE FOR GENERAL SOFTWARE DESIGN

CUPID PROPERTIES CAN BE ASSESSED

TO UNDERSTAND TRADEOFFS

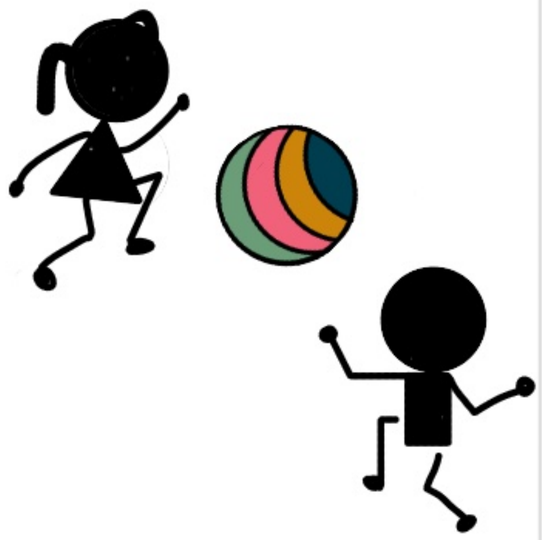
COHESION & COUPLING ARE KEY ATTRIBUTES TO SEE

HOW WELL A DESIGN HAS GROUPED ELEMENTS INTO COMPONENTS

CUPID PROPERTIES FOR DESIGN

COMPOSABLE

PLAYS WELL WITH OTHERS



EASY TO

- WORK ON
- TEST
- CHANGE IN ISOLATION

UNIX PHILOSOPHY

DOES ONE THING WELL

EACH COMPONENT EMBODIES A SINGLE PURPOSE

PREDICTABILE

DOES WHAT YOU EXPECT

EACH COMPONENT IS DETERMINISTIC & OBSERVABLE

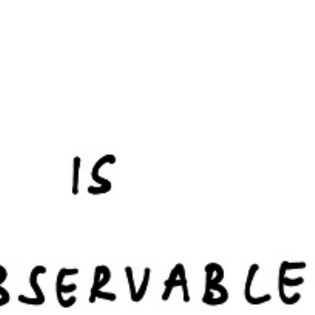
IDIOMATIC

FEELS NATURAL

EACH COMPONENT'S USE FEELS OBVIOUS TO A TRAINED USER

DOMAIN-BASED

SOLUTION MODELS PROBLEM DOMAIN



INFRA CODE IS ORGANISED AROUND PURPOSE WHY RATHER THAN HOW

THESE PROPERTIES

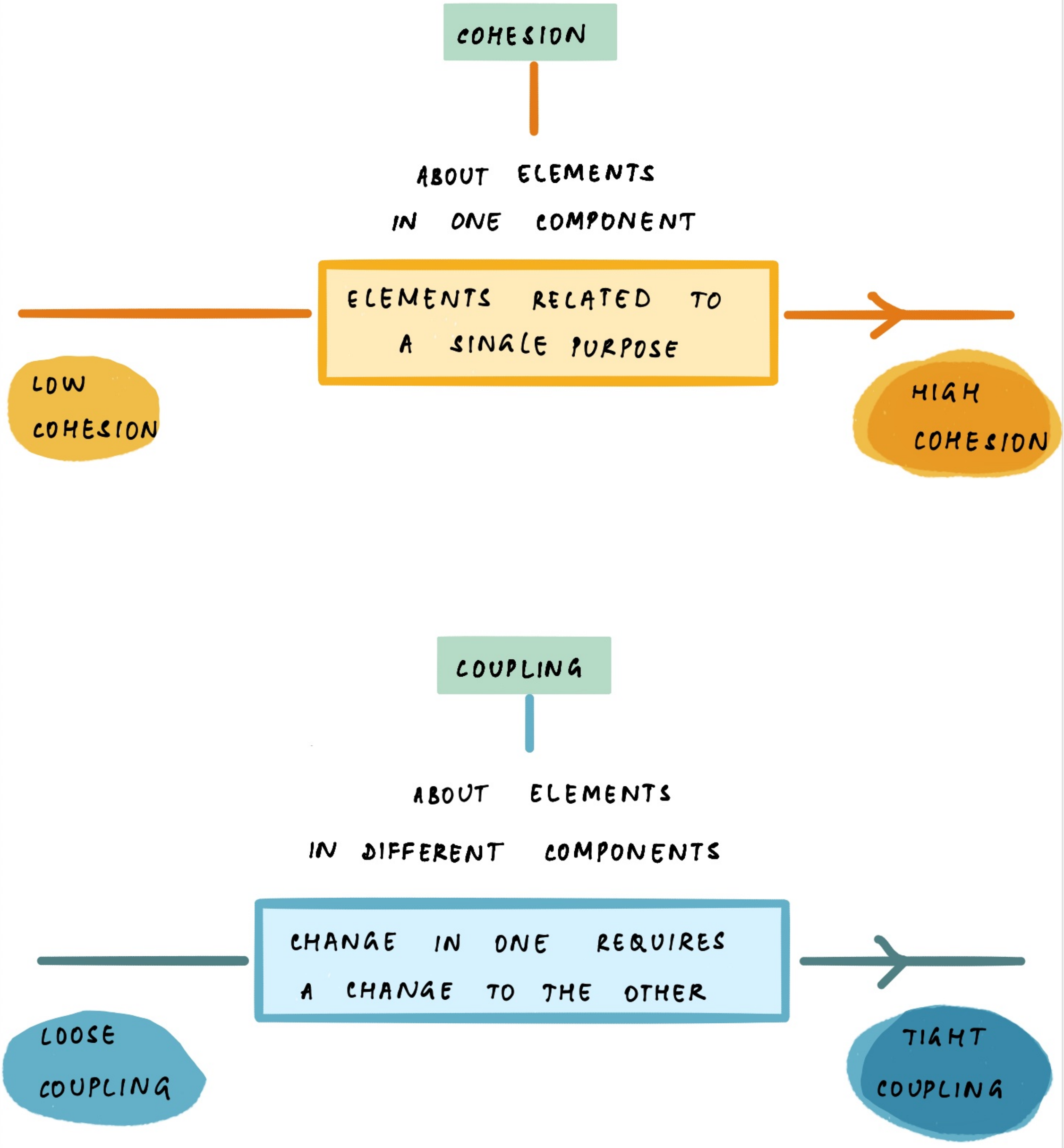
MAKE SOFTWARE

"A JOY TO WORK WITH"

-DANIEL TERHORST-NORTH

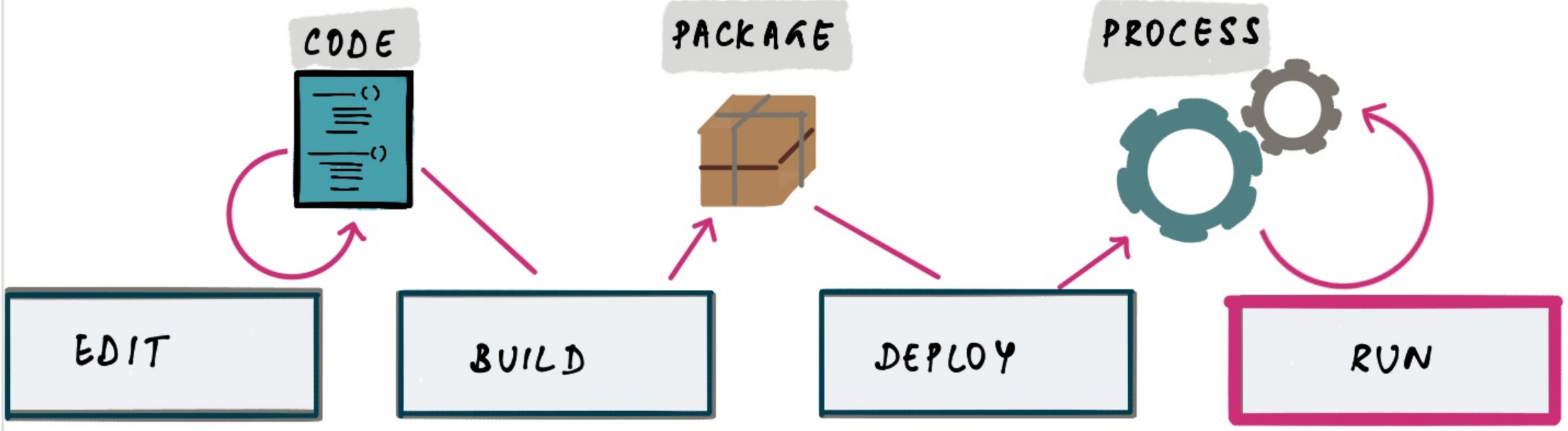
COHESION & COUPLING

STUDYING COHESION AND COUPLING HELPS US UNDERSTAND HOW WELL ELEMENTS HAVE BEEN GROUPED INTO COMPONENTS.



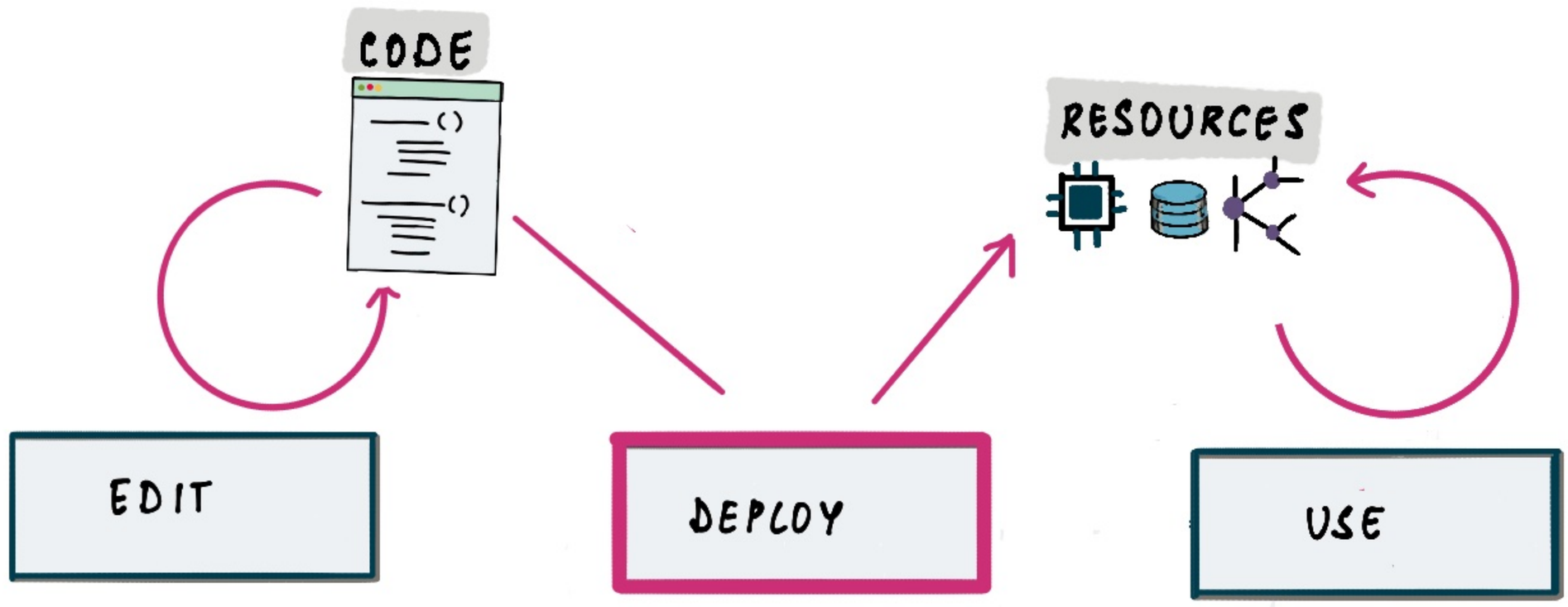
CODE PROCESSING WORKFLOW

APPLICATION CODE WORKFLOW



APPLICATION CODE EXECUTES AFTER IT IS DEPLOYED

INFRASTRUCTURE CODE WORKFLOW



INFRASTRUCTURE CODE EXECUTES IN THE DEPLOYMENT CONTEXT

IT CREATES, MODIFIES, REMOVES INFRASTRUCTURE RESOURCES DEPLOYED

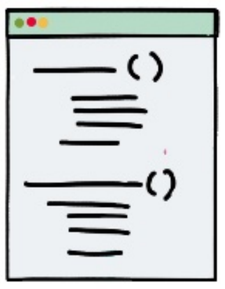
THIS HAS IMPLICATIONS FOR UNIT TESTS & COMPILED CODE

DESIGN CONCERNS

THE DESIGN CONCERNS IN

DIFFERENT PARTS OF THE CODE LIFECYCLE

SOURCE CODE



WRITE AND TEST INFRASTRUCTURE CODE

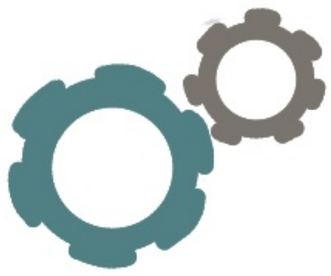
CONCERN: COLLABORATING

PACKAGE

MAKE CODE AVAILABLE TO DEPLOY

CONCERN: GETTING RELIABLE FEEDBACK

DEPLOYMENT



CODE PROVISIONS RESOURCES

CONCERN: SPEEDY/RELIABLE DEPLOYMENTS

LIVE RESOURCES

APPS USE THE INFRASTRUCTURE

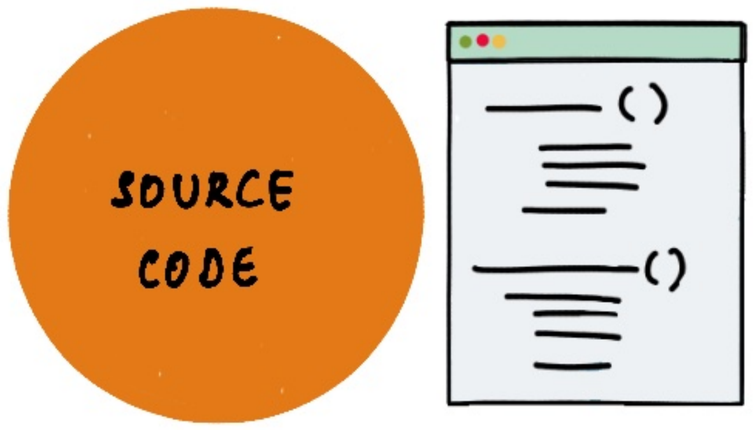
CONCERN: TROUBLE SHOOTING SUPPORT

DESIGN FORCES

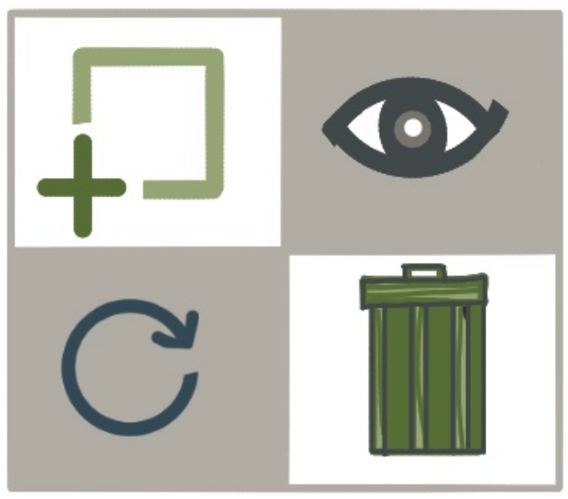
DESIGN FORCES ARE CONSTRAINTS, REQUIREMENTS AND INFLUENCES THAT GUIDE DECISIONS.

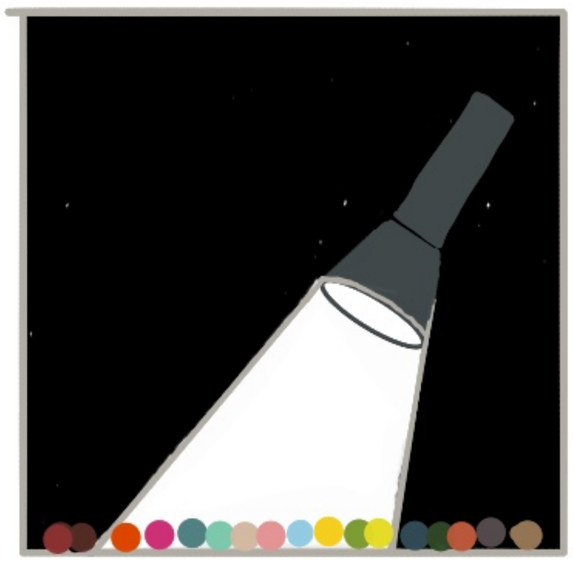
THEY ARE RELEVANT IN EACH INFRASTRUCTURE CODE DESIGN CONTEXTS DESCRIBED

DESIGN CONTEXT - 1



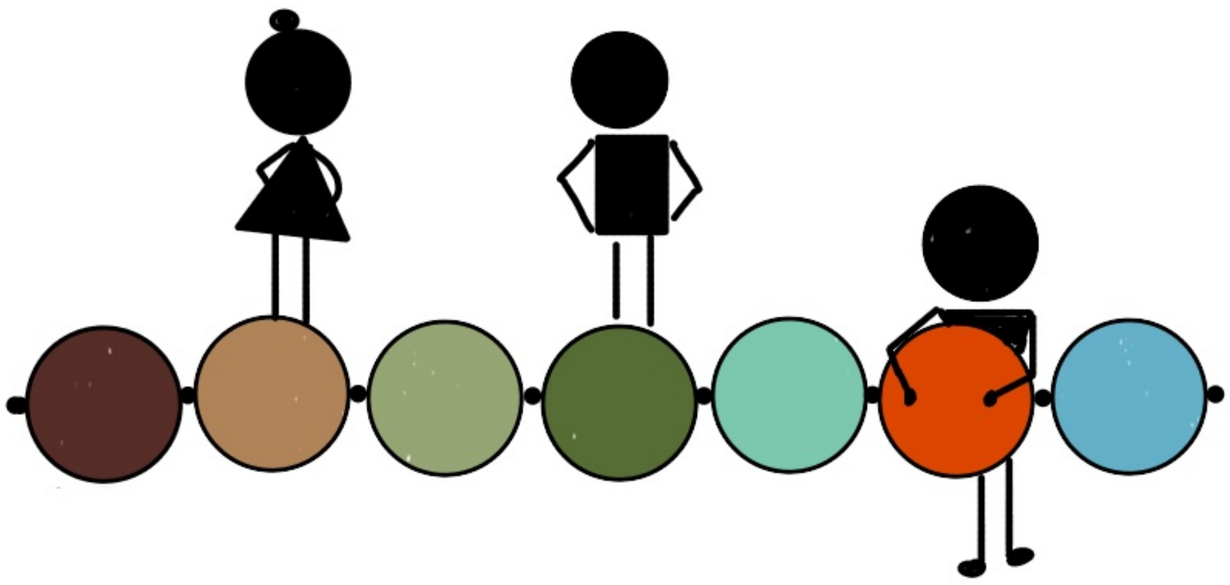
HOW DOES A TEAM ORGANISE THEIR INFRASTRUCTURE ACROSS SOURCE CODE REPOSITORIES ?





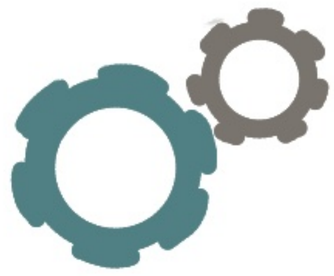
WHO OWNS CODE & CAN VIEW/MODIFY IT

DELIVERY SCOPE OF EACH UPDATED COMPONENT



WHAT CODE/COMPONENT SUBSETS ARE PEOPLE WORKING ON ?

DESIGN CONTEXT - 2



TO GET FROM CODE TO DEPLOYMENT, THE EFFECTIVENESS OF INFRASTRUCTURE CODE DESIGN IS MEASURED USING THE DORA METRICS

- TIME
- FREQUENCY
- FAILURE RATE OF DELIVERING CHANGES
- TIME TO RECOVER FROM FAILURE

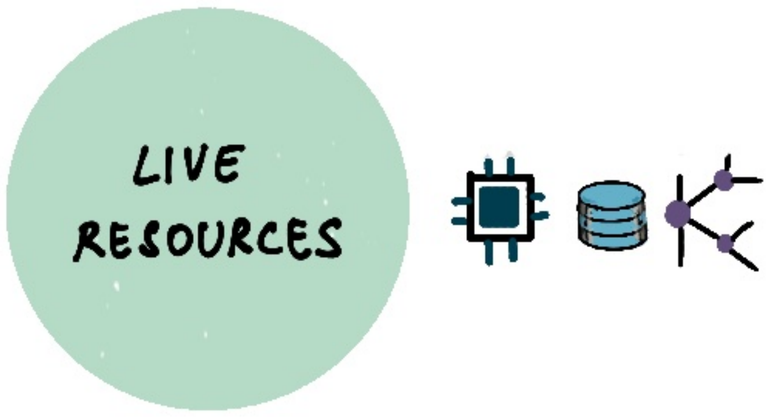
DIFFERENT FORCES COME INTO PLAY FOR OPTIMISING FOR THESE MEASURES.

MANAGING DOWNTIME

WORKLOAD ALIGNMENT

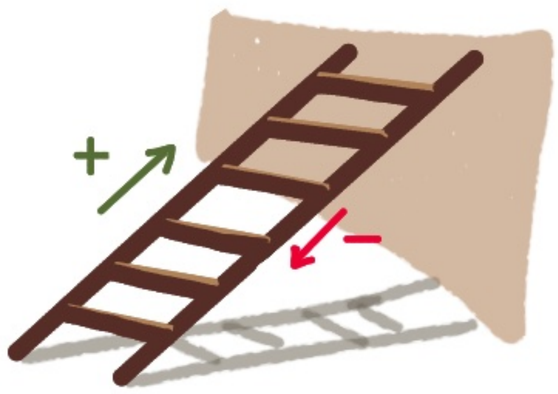
COMPLIANCE

DESIGN CONTEXT - 3



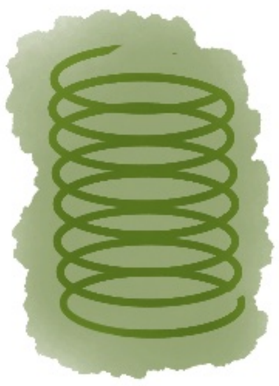
AMONG THE GOALS OF AN INFRASTRUCTURE SYSTEM ARE : PERFORMANCE, RELIABILITY, COST OF OWNERSHIP ETC.

FOR ACHIEVING THESE, DESIGN FORCES AFFECT DECISIONS ABOUT WHERE TO SPLIT INFRASTRUCTURE INTO COMPONENTS.



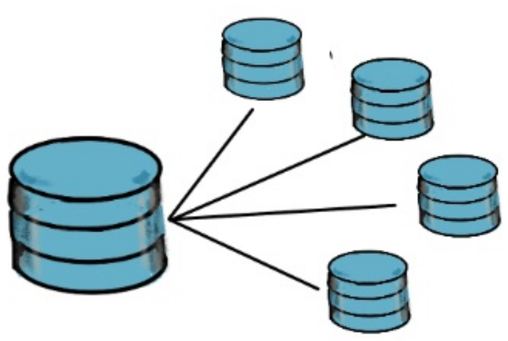
SCALING : USAGE LEVELS AND RESOURCE CONSUMPTION

GEOGRAPHICAL DISTRIBUTION OF RESOURCES



RESILIENCE

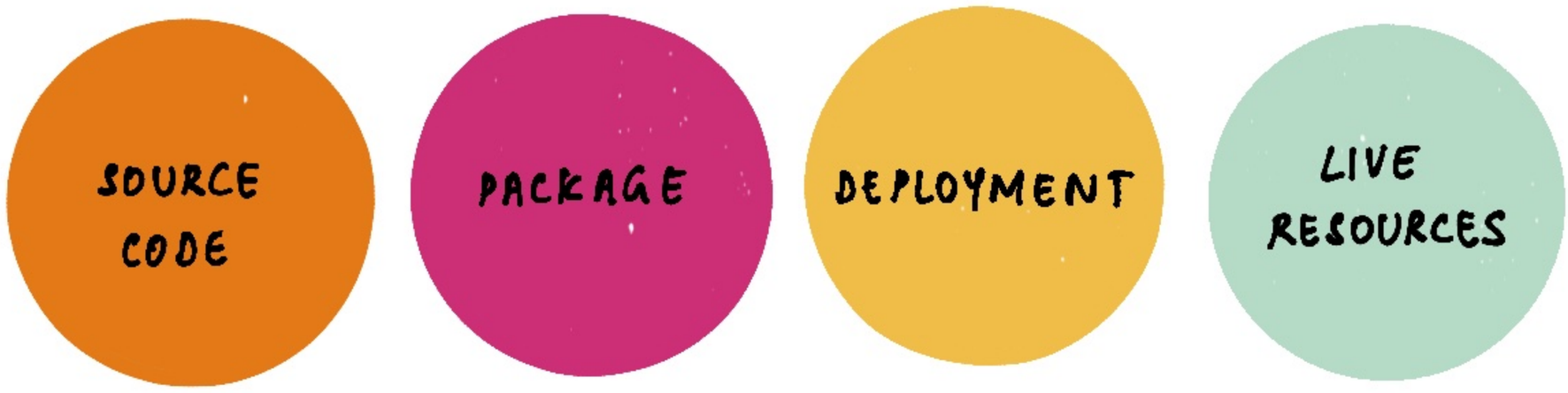
DATA REGULATIONS



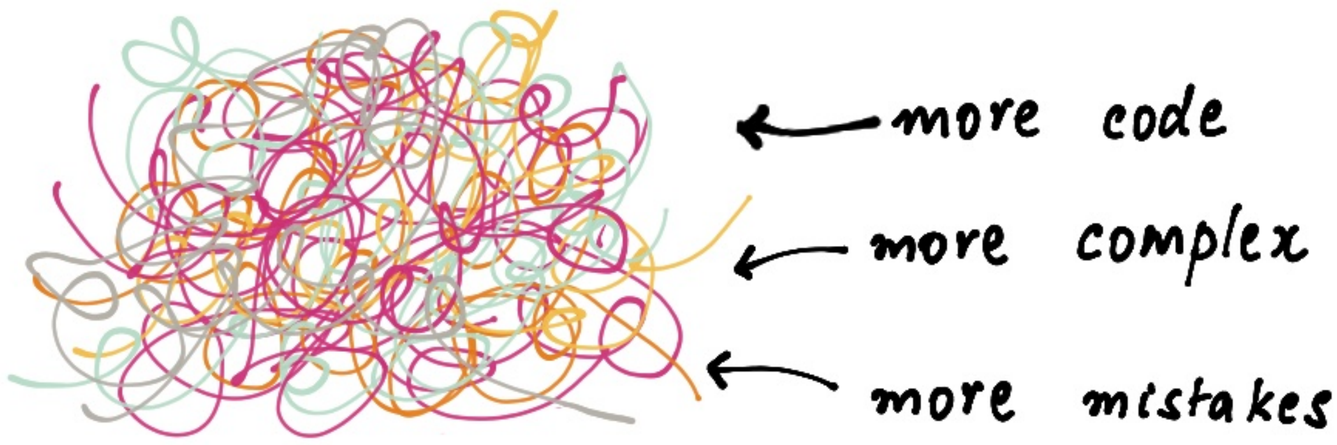
DATA PARTITIONING

HOSTING COSTS

DESIGN CONTEXT - 4

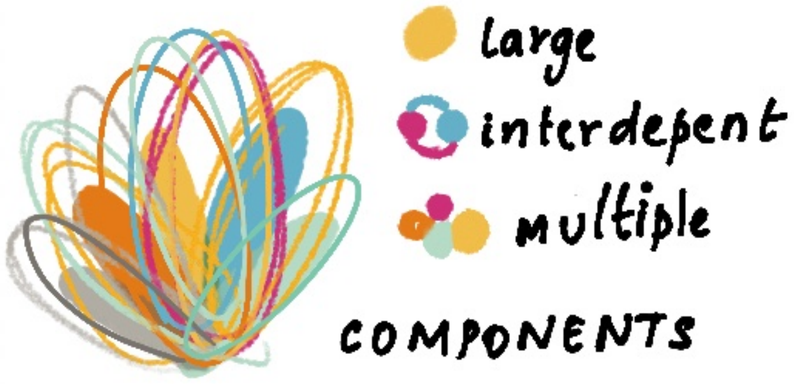


DECISIONS ARE NEVER MADE IN ISOLATION. HERE ARE SOME INFLUENCES ACROSS CONTEXTS



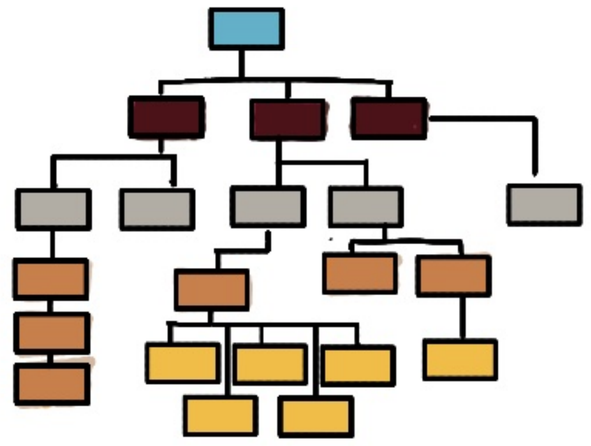
COGNITIVE SIZE OF CODEBASE

CHANGES TO



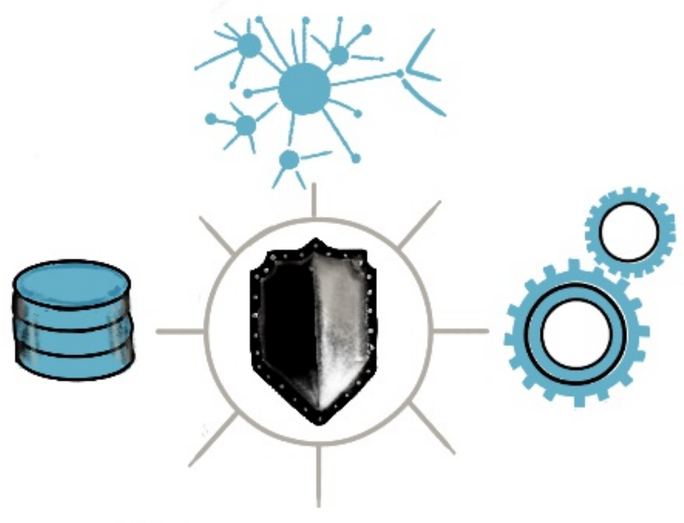
COMPONENTS

SCOPE OF CHANGE



ORGANISATIONAL STRUCTURE

COST OF OWNERSHIP



SECURITY

LANGUAGE CHOICES FOR INFRA AS CODE

THE BEST LANGUAGE

WILL VARY FOR DIFFERENT

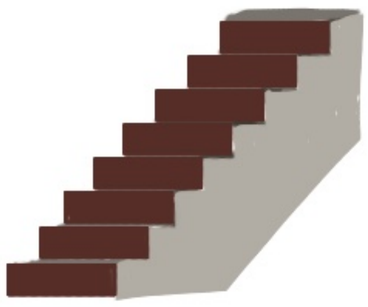
SYSTEMS,

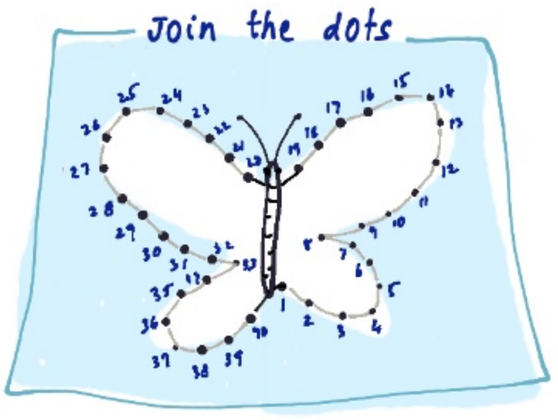
PARTS &

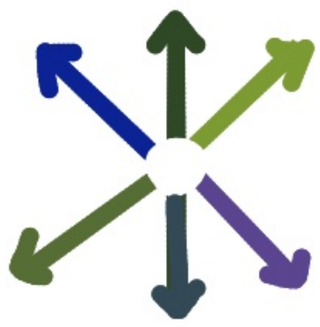
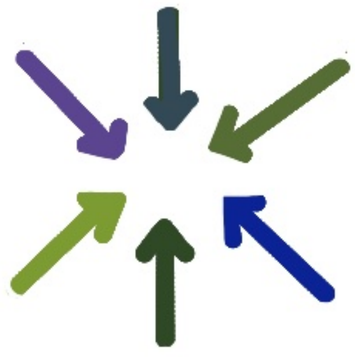
PEOPLE

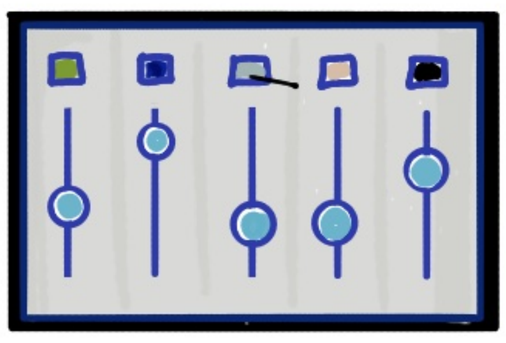
THIS IS A HIGHLY DEBATED TOPIC. THE NEXT SECTION DESCRIBES THE 4 GROUPS OF LANGUAGE TYPES AND THEIR OPPOSING CHARACTERISTICS TO GIVE CONTEXT.

LANGUAGES

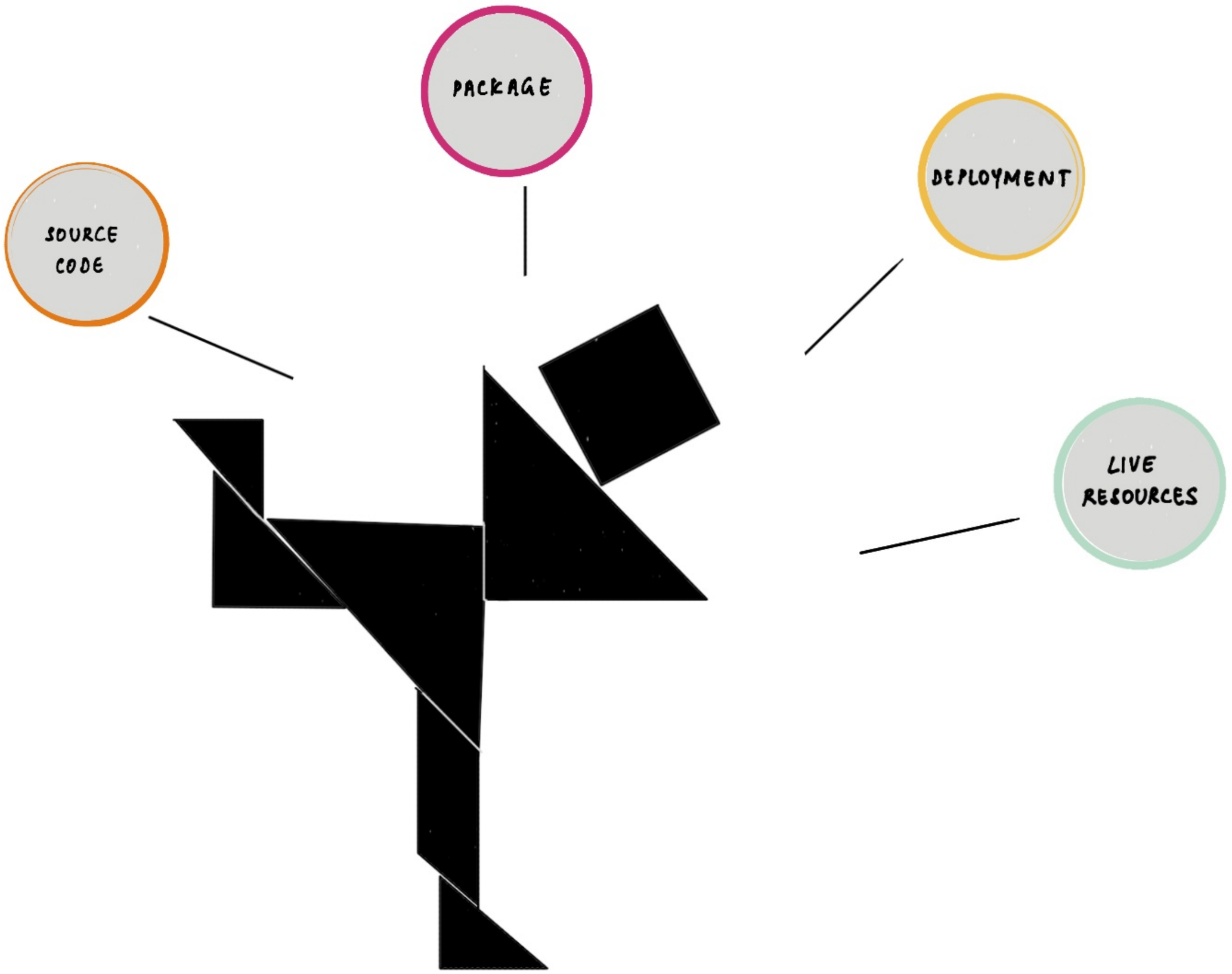
	PROCEDURAL	IDEMPOTENT
FOCUS	STEPS TO BUILD 	DESIRED END-STATE 

	DECLARATIVE	IMPERATIVE
DESCRIBE	WHAT YOU WANT 	HOW TO ACHIEVE 

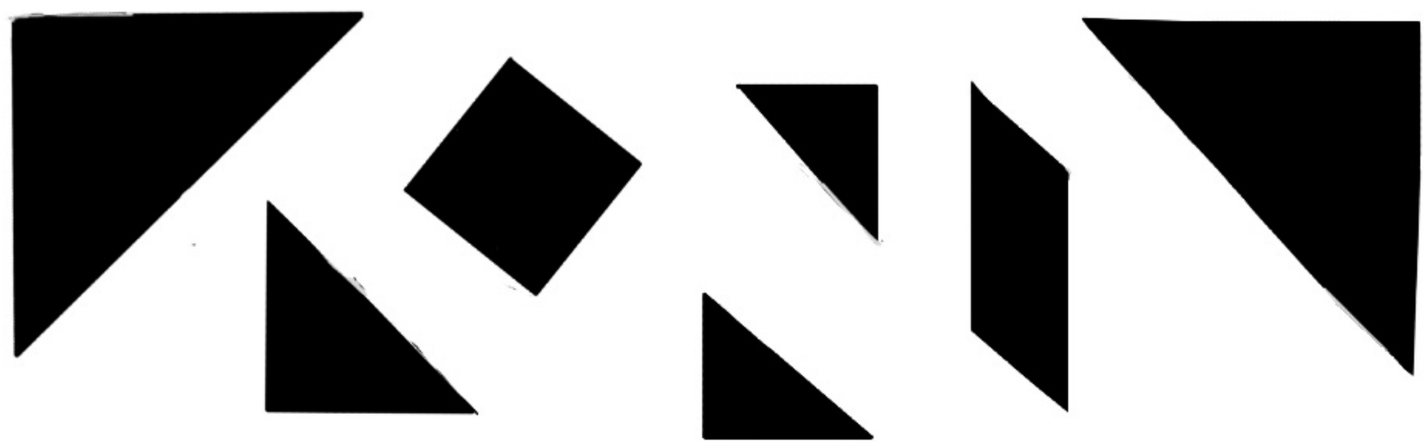
	GENERAL PURPOSE	DOMAIN-SPECIFIC
ARE	 VERSATILE	TAILORED FOR  INFRASTRUCTURE

	LOW-LEVEL	HIGH LEVEL
OFFER	 FINE-GRAINED CONTROL	 ABSTRACTIONS

GOAL OF DESIGN



✓ DESIGN FORCES BALANCED



⊘ NOT TO BREAK INFRASTRUCTURE INTO THE SMALLEST DEPLOYABLE PIECES

(ART BASED ON PUZZLE TANGRAM)

PATTERNS

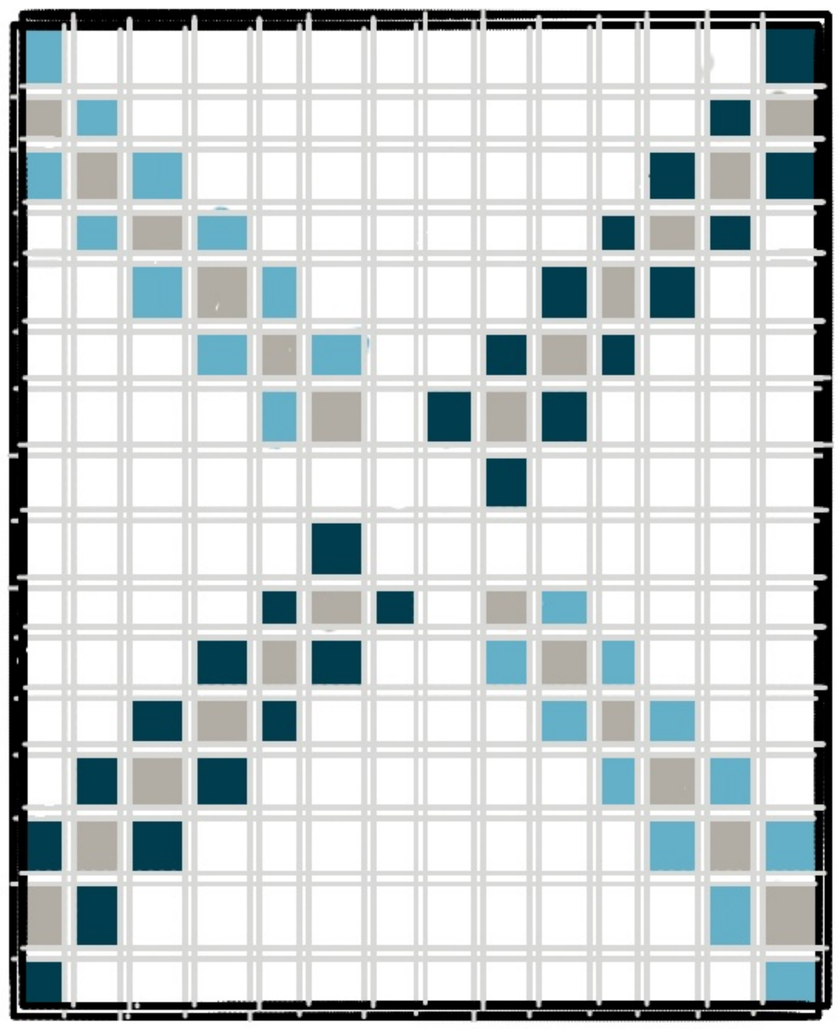
A DESIGN PATTERN

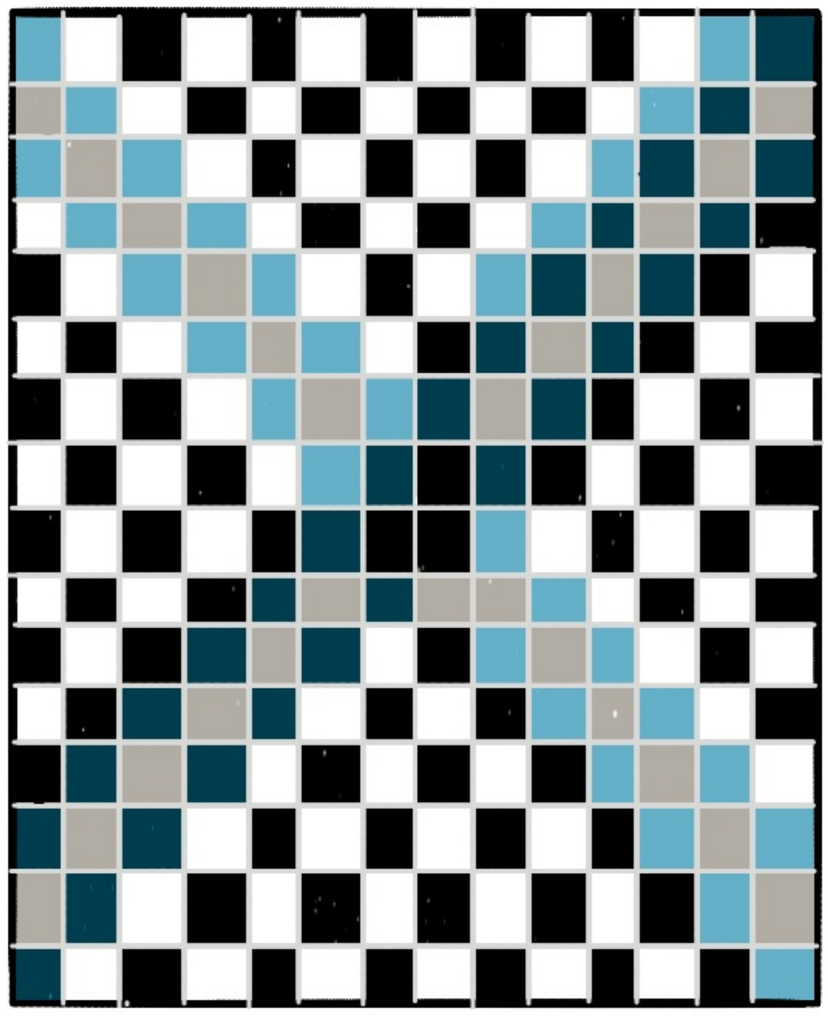
IS A

TYPICAL SOLUTION

TO A

COMMON PROBLEM





ANTIPATTERNS

PATTERNS AID

IN CLARIFYING

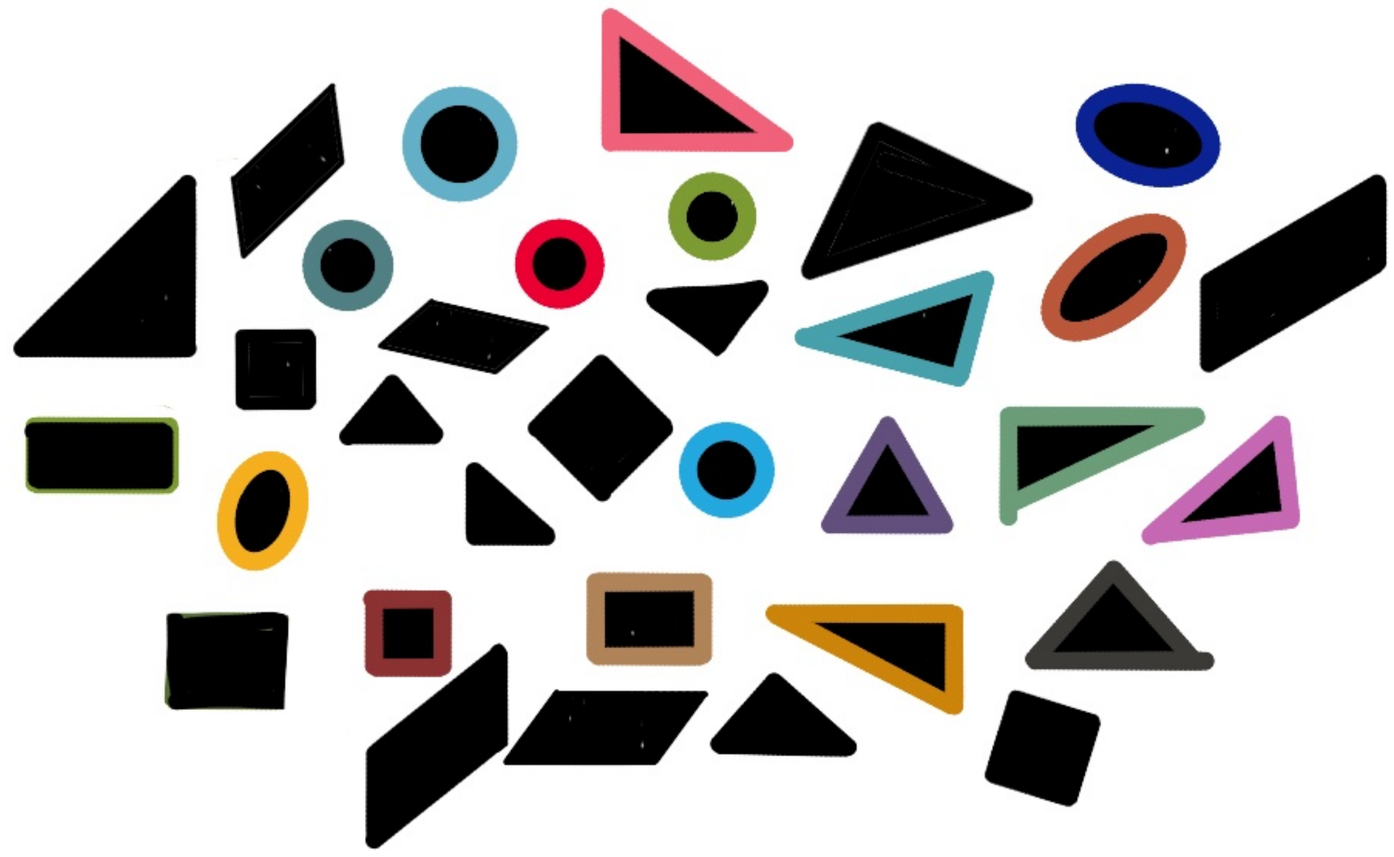
APPROACHES & TRADEOFFS

AN ANTIPATTERN IS A TYPICAL, BUT PROBLEMATIC SOLUTION TO A COMMON DESIGN PROBLEM

THROUGH THE BOOK, KIEF DISCUSSES MULTIPCE APPROACHES TO DESIGN, CONFIGURE, INTEGRATE AND DEPLOY EACH COMPONENT.

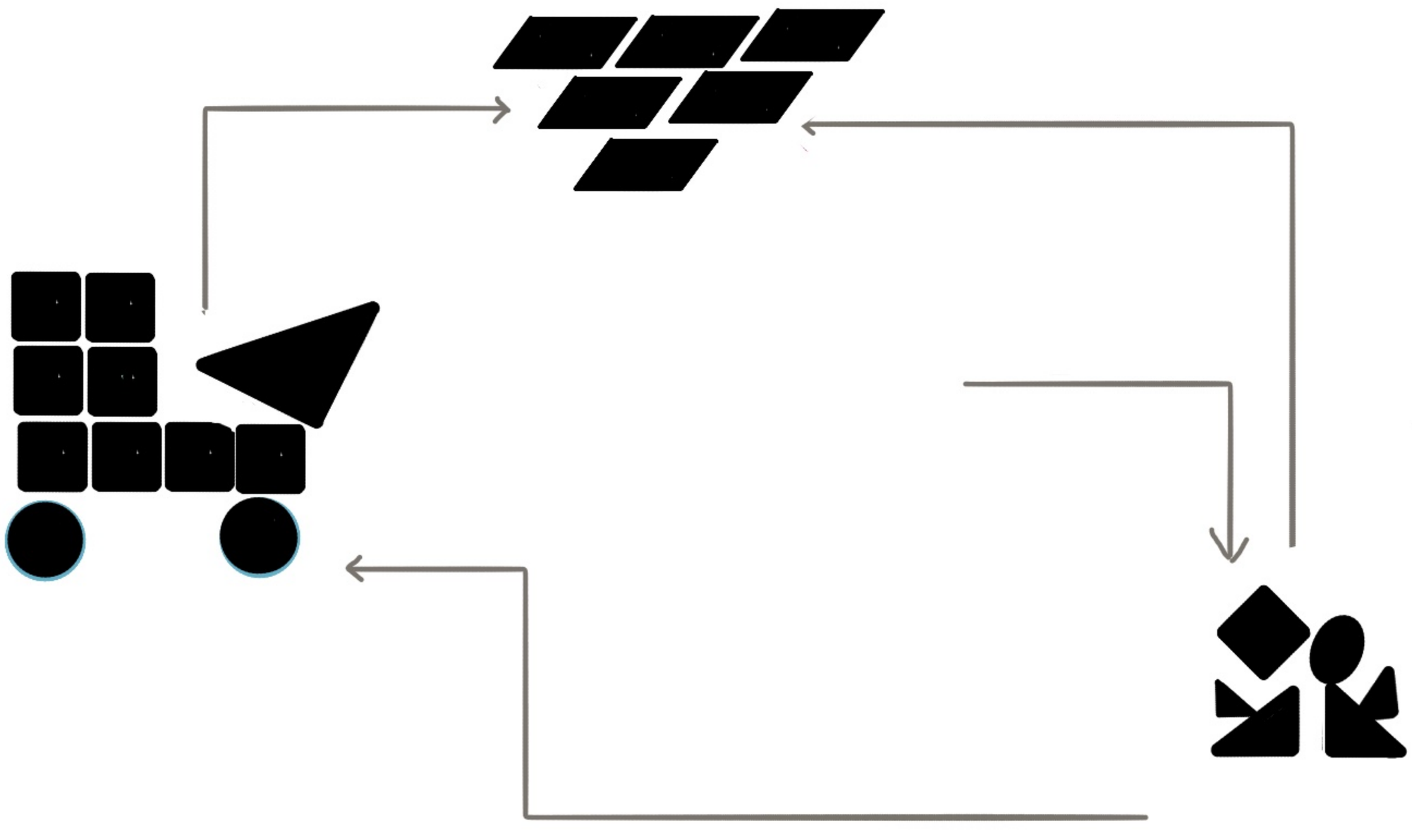
SYSTEM DESIGN INVOLVES

TAKING ELEMENTS



GROUPING THEM

INTO COMPONENTS



AND DEFINING RELATIONSHIPS BETWEEN THE GROUPS

EXAMPLES OF COMPONENTS: APPLICATIONS, MICROSERVICES, LIBRARIES, CLASSES

INFRASTRUCTURE COMPONENTS

THERE ARE NO WIDELY AGREED DEFINITIONS FOR INFRASTRUCTURE AS CODE COMPONENTS. SO, HERE ARE 4 TYPES DEFINED FOR THE PURPOSES OF THIS BOOK.

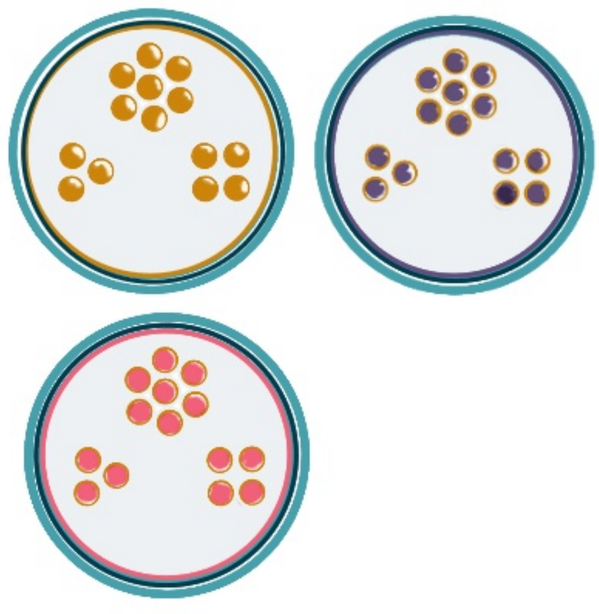
THESE TERMS, ESPECIALLY 'STACK', ARE WIDELY USED, EVEN IF THEY ARE NOT UNIVERSAL.

IAAS RESOURCE

CODE LIBRARY

INFRASTRUCTURE DEPLOYMENT STACK

INFRASTRUCTURE COMPOSITION



INFRASTRUCTURE COMPONENTS

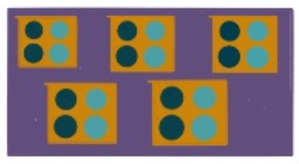
IAAS RESOURCE

SMALLEST UNIT OF INFRASTRUCTURE DEFINED & PROVISIONED INDEPENDENTLY.

CODE LIBRARY

INFRASTRUCTURE RESOURCES GROUPED BY CODE SHARING & REUSE ACROSS STACKS

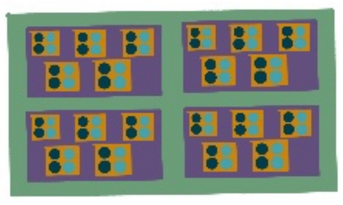
INFRASTRUCTURE DEPLOYMENT STACK



COLLECTION OF INFRASTRUCTURE RESOURCES DEPLOYED AS A UNIT.

FOR THE CONVENIENCE OF PROVISIONING

INFRASTRUCTURE COMPOSITION

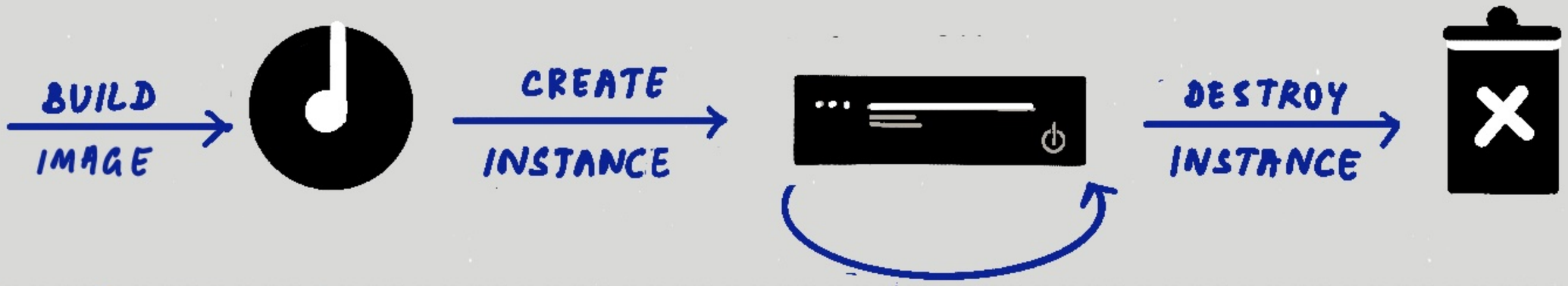


COLLECTION OF DEPLOYABLE STACKS. DEFINES DEPENDENCIES & INTEGRATIONS BETWEEN STACKS

FOR THE CONVENIENCE OF TEAMS RESPONSIBLE FOR THE APPS/SERVICES THAT USE THE COMPOSITION.

BUILD SERVERS AS CODE

DESPITE THE POPULARITY OF CLOUD NATIVE AND SERVERLESS APPS, SERVER AS CODE IS STILL VERY RELEVANT

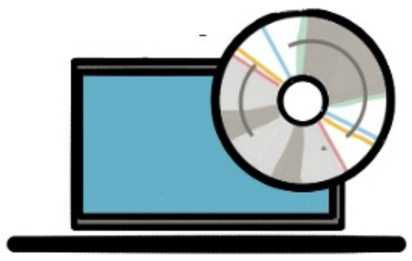


SERVER LIFECYCLE

A SERVER

CONTAINS

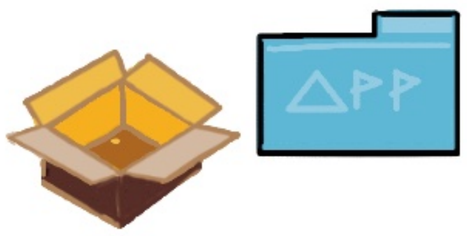
- SOFTWARE
- CONFIGURATION
- DATA



OPERATING SYSTEM IMAGE

IS BUILT FROM

INFRASTRUCTURE CODE



APP INSTALLATION PACKAGES

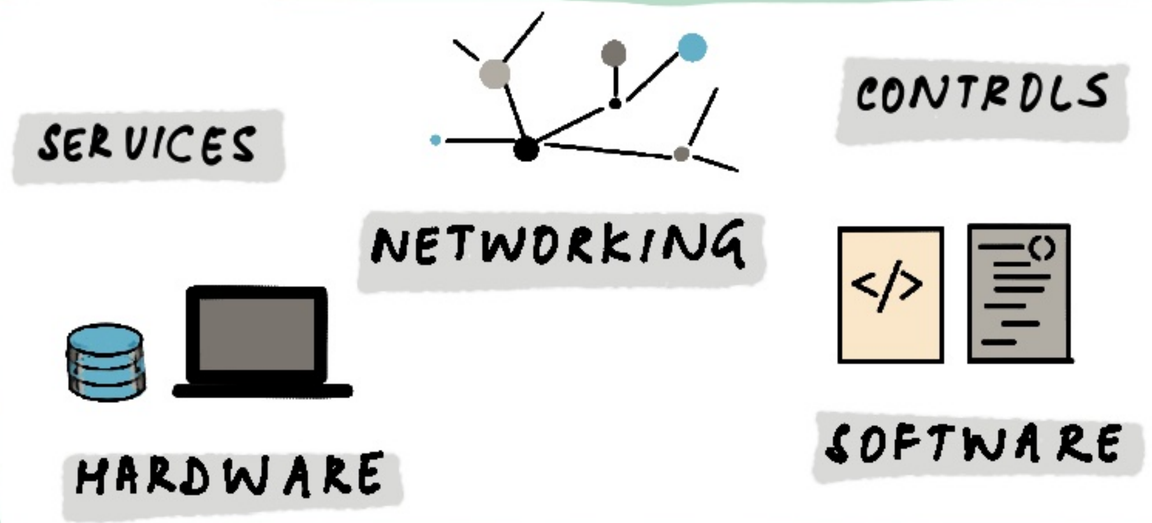
INFRASTRUCTURE AS CODE EMERGED AS A WAY TO CONFIGURE SERVERS

THE BOOK DISCUSSES PATTERNS TO USE FOR EACH STAGE OF A SERVER'S LIFECYCLE & TO MAKE SERVER A REUSABLE COMPONENT

DESIGN ENVIRONMENTS

AN ENVIRONMENT PROVIDES A SERVICE

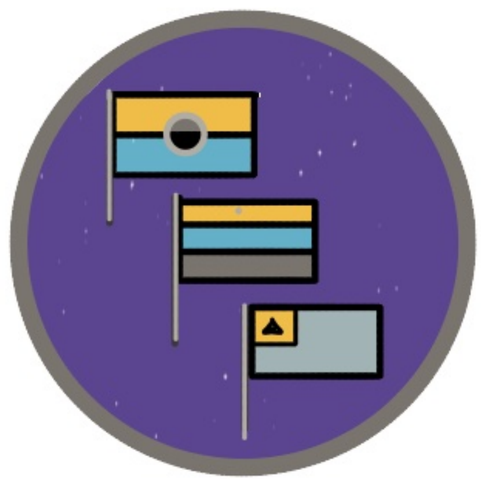
FROM AN INFRASTRUCTURE AS CODE VIEWPOINT, ENVIRONMENT IS A LOGICAL GROUPING OF DEPLOYED INFRASTRUCTURE



ENVIRONMENTS MAY BE USED

TO MANAGE CHANGE

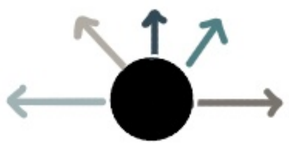
TO BE OWNED BY DIFFERENT GROUPS



TO CUSTOMISE BY REGION / BRAND

CONSIDERATIONS FOR DESIGNING AN ENVIRONMENT

DRAW BOUNDARIES BETWEEN ENVIRONMENTS



SHARE RESOURCES & INFRASTRUCTURE

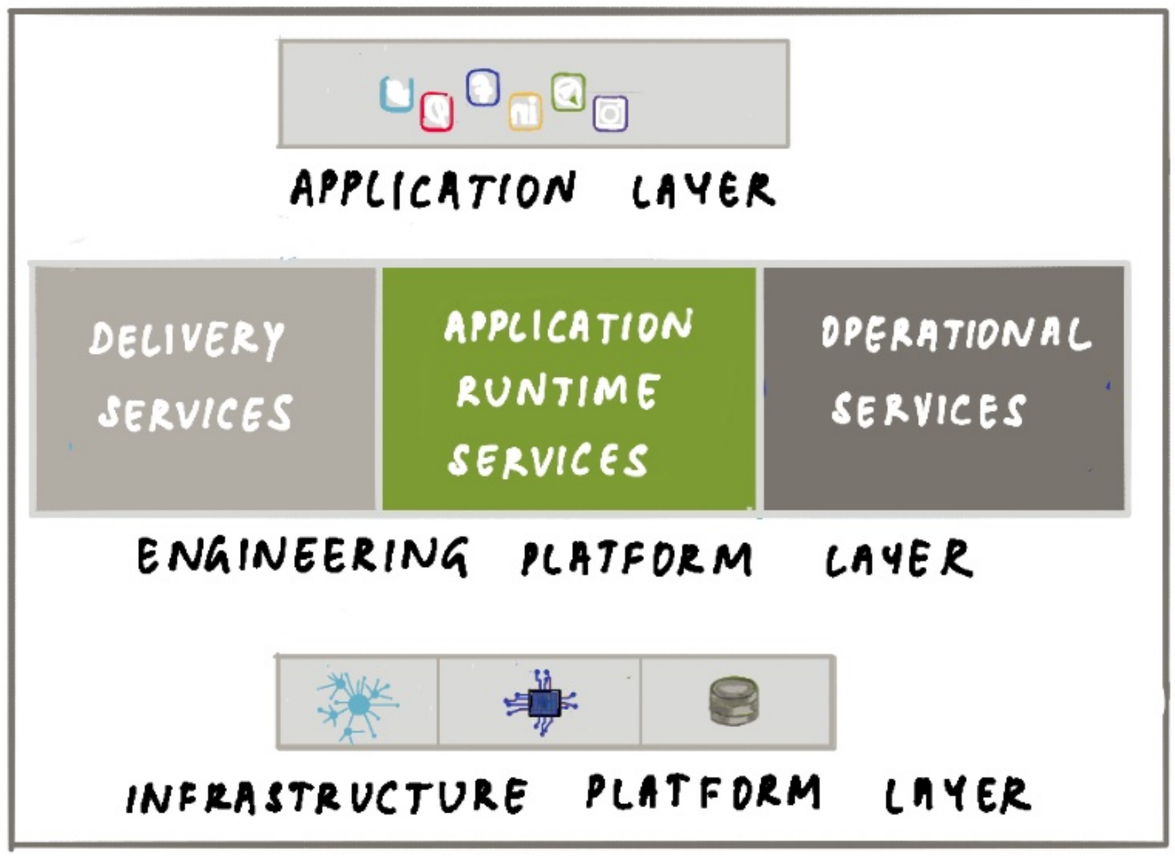
REDUCE COUPLING AND INCREASE COHESION

ENABLE BETTER GOVERNANCE

BUILD IN LAYERS — PHYSICAL, VIRTUAL & CONFIG

TEST & MAKE CHANGES TO ENVIRONMENTS

PROVIDE RUNTIME INFRASTRUCTURE



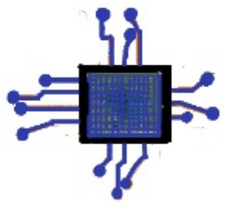
THE FOCUS HERE IS THE **APPLICATION RUNTIME SERVICES**

- THAT HOST RUNNING WORKLOADS

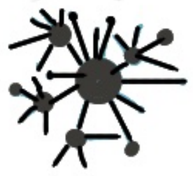
THIS INVOLVES PROVISIONING

SERVERS

DATABASE INSTANCES



COMPUTE CLUSTERS



NETWORK STRUCTURES

TO START PLANNING THE INFRASTRUCTURE,

- EXAMINE THE WORKLOADS
- IDENTIFY THE CAPABILITIES THEY NEED

THEN, DELVE INTO THE DETAIL OF

- SERVERS AS CODE
- CLUSTERS AS CODE
- SERVERLESS INFRASTRUCTURE
- SEPARATION OF BUSINESS LOGIC & INFRASTRUCTURE

DELIVERY OF INFRASTRUCTURE AS CODE

CORE DELIVERY WORKFLOWS

EVERYTHING INFRASTRUCTURE TO BE AUTOMATED

INFRASTRUCTURE RESOURCES

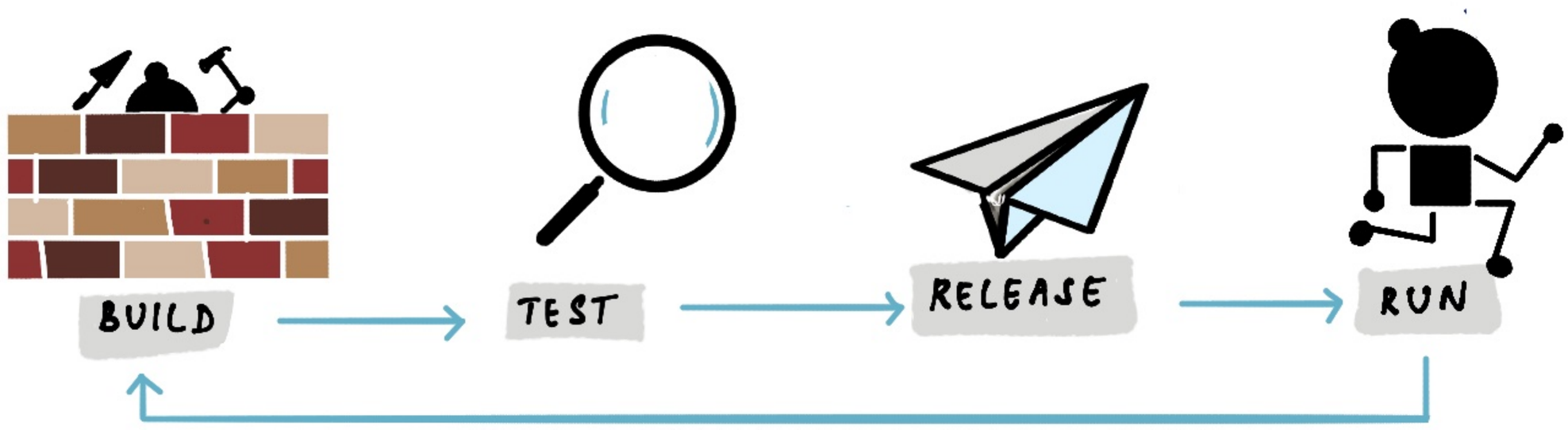
DEPLOYMENT PROCESS

PIPELINE STAGES

DEVELOPER SETUP

TESTS & MONITORING

IN CONTINUOUS DELIVERY SYSTEMS, THE SOFTWARE DELIVERY WORKFLOW IS



FOR EACH STAGE THAT SOFTWARE IS DEPLOYED

AUTOMATED TEST ENVIRONMENT

STAGING ENVIRONMENT

PRODUCTION ENVIRONMENT

INFRASTRUCTURE NEEDS TO BE IN PLACE, UPDATED

THE BOOK DISCUSSES

- WORKFLOWS FOR DELIVERING INFRASTRUCTURE
- OPTIONS FOR TEAM TOPOLOGIES

BUILD & DEPLOY INFRASTRUCTURE AS CODE

INFRASTRUCTURE COMPONENTS ARE BUILT SEPARATELY

DEPLOYING INFRASTRUCTURE INVOLVES THREE STEPS

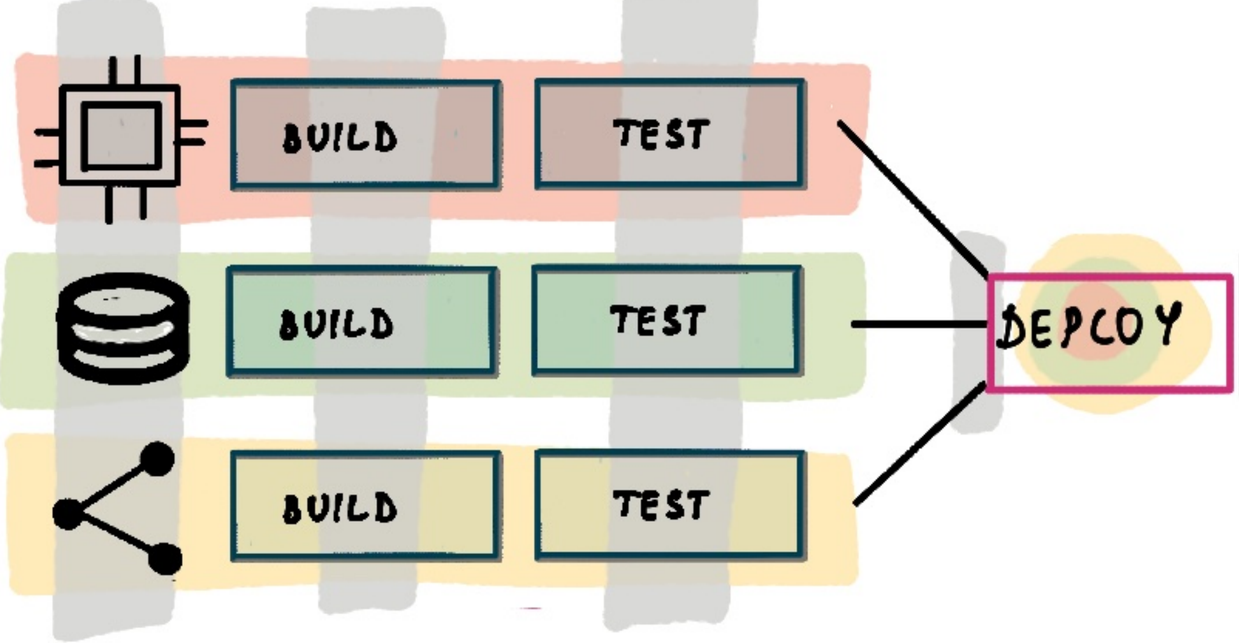
- ASSEMBLE — GENERATES A BUILD
- COMPILE — RESOLVES DEPENDENCIES
- EXECUTE — PROVISIONS ENVIRONMENT

CONSIDER THE CONDITIONS FOR DELIVERING AN

INTEGRATED INFRASTRUCTURE COMPONENT

VS

INTEGRATING DURING DEPLOYMENT?



AUTOMATED TEST ENVIRONMENT

STAGING ENVIRONMENT

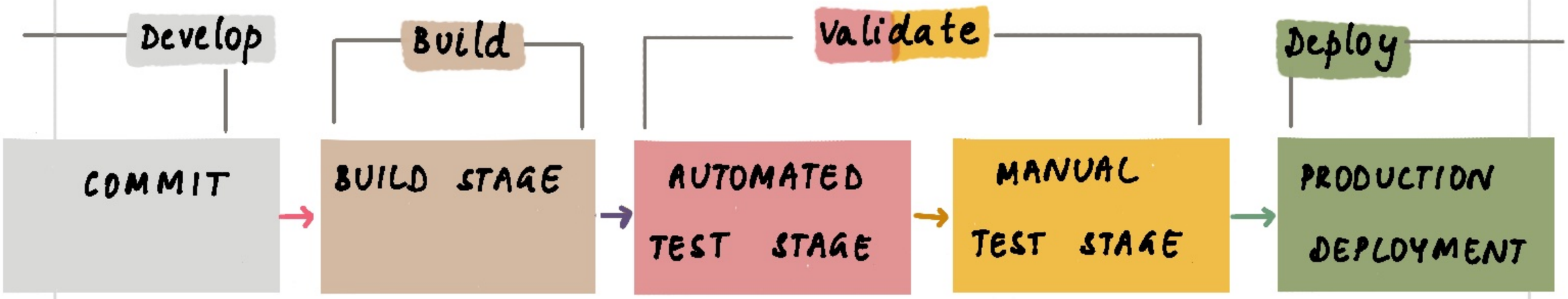
PRODUCTION ENVIRONMENT

THE BOOK DISCUSSES ACTIONS TO BE TAKEN TO ENSURE THAT INFRASTRUCTURE DEPLOYED ACROSS ENVIRONMENTS STAYS CONSISTENT.

IMPLEMENT INFRASTRUCTURE DELIVERY WITH PIPELINES

AN INFRASTRUCTURE DELIVERY PIPELINE AUTOMATES THE WORKFLOW THAT BUILDS, DELIVERS AND DEPLOYS INFRASTRUCTURE.

HERE IS A PICTURE OF PIPELINE STAGES AS AN EXAMPLE



EACH STAGE

- TAKES INPUT, MAKES OUTPUT — DEPENDING ON SCOPE
- PROGRESSES THE WORKFLOW — PERFORMS AN ACTION
- ACTS IN A CONTEXT — OFFLINE / DEPENDENCIES

WE NEED OTHER CAPABILITIES TO IMPLEMENT PIPELINES - SUCH AS:

SOURCE CODE REPOSITORY	ARTIFACT REPOSITORY
PIPELINE ORCHESTRATION	DEPLOYMENT SERVICE

TEST INFRA CODE-1

TIGHT FEEDBACK LOOPS ARE THE ESSENCE OF CONTINUOUS TESTING

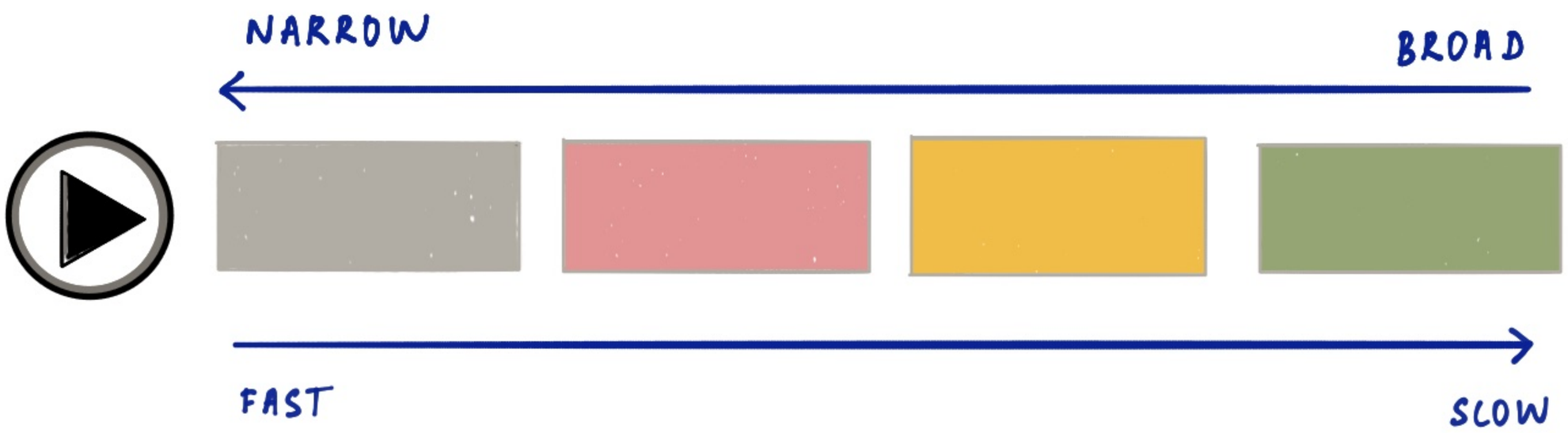
WHAT CAN BE TESTED

- CODE QUALITY
- FUNCTIONACITY
- SECURITY
- COMPLIANCE
- LIBRARIES / OTHER CODE
- PERFORMANCE
- SCALABICITY
- AVAILABILITY
- OPERABILITY

CHALLENGES

- BENEFITS FROM TESTING DECCARATIVE CODE
- SPEED OF TESTING
- DEPENDENCIES
- TESTING IN PRODUCTION

A GUIDING PRINCIPLE FOR STAGES & SCOPE OF TESTING

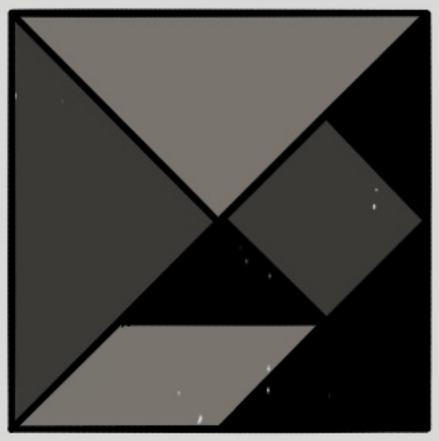


THE BOOK GOES ON TO DISCUSS SOLUTIONS WITH SOME EXAMPLES

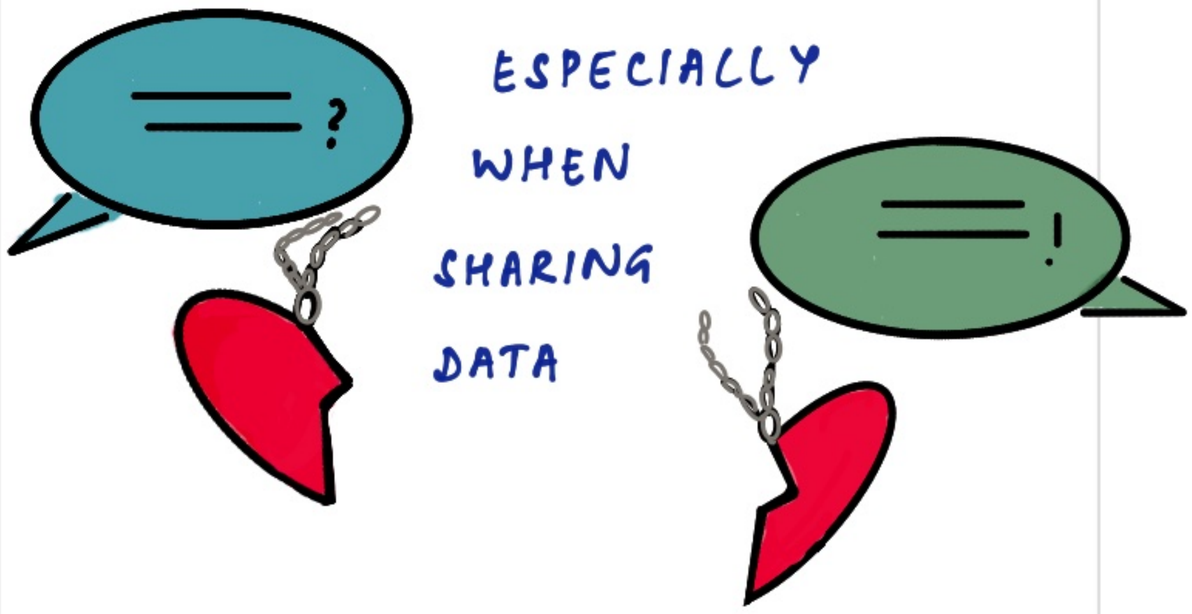
TEST INFRA CODE - 2

DIVIDE INFRASTRUCTURE INTO MORE TRACTABLE PIECES

FASTER, EASIER TO PROVISION TEST AND MAINTAIN

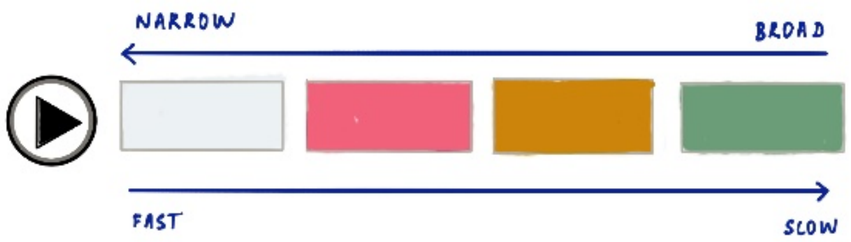


CLARIFY DEPENDENCIES



ESPECIALLY WHEN SHARING DATA

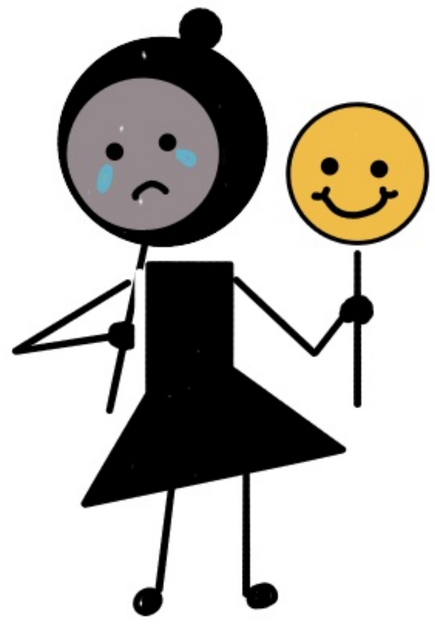
PROGRESSIVE TESTING



SLOWER TESTS RUN ONLY AFTER POSITIVE FEEDBACK FROM FASTER TESTS

LOCAL EMULATORS AND TEST-DOUBLES

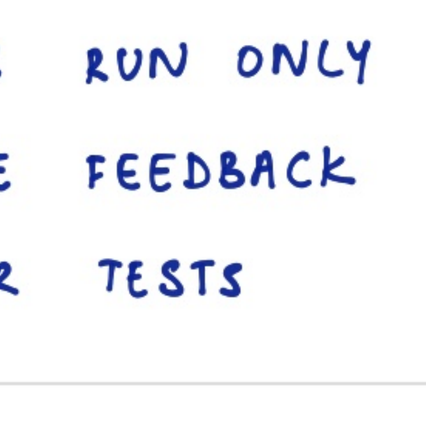
USE TEST DOUBLES IN CASE OF DEPENDENCIES



TEST OFFLINE

OFFLINE TESTS RUN FASTER

USE WITH TEST-DOUBLES

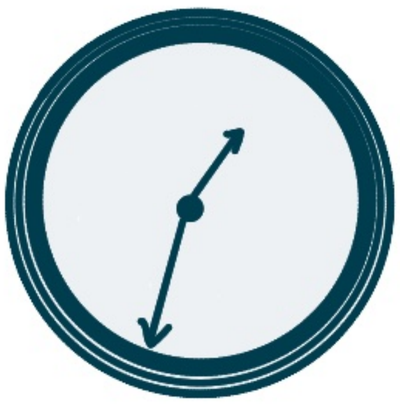


USE PERSISTENT INSTANCES OF THE ENVIRONMENT

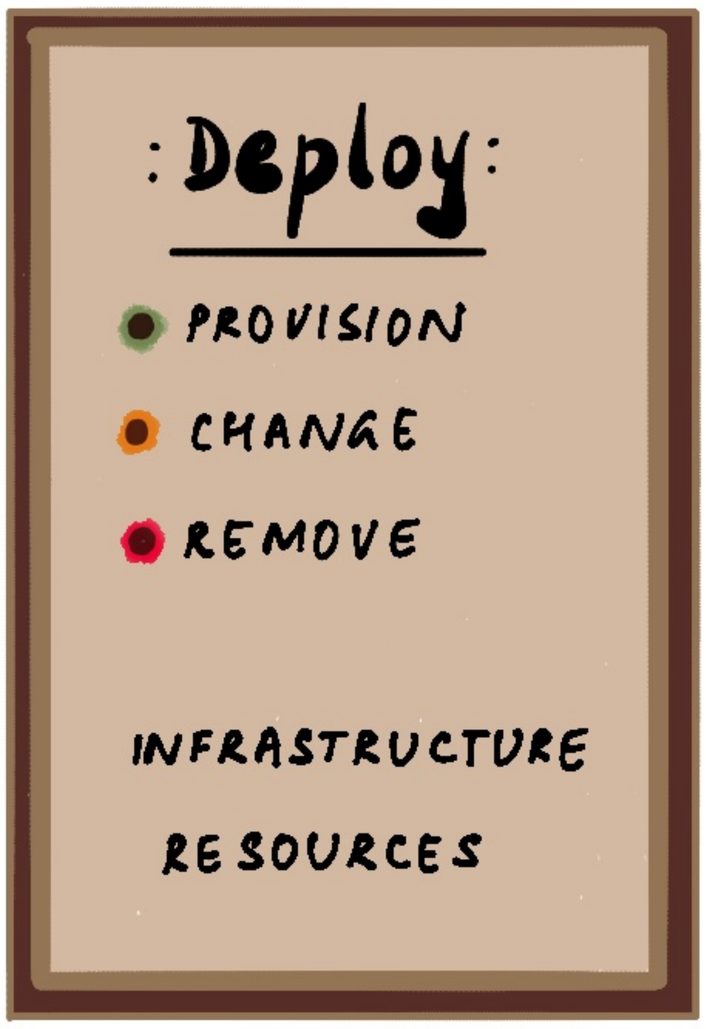
SAVES ON TEST RUN TIME

BUILDS UP

- INCONSISTENCIES
- COST



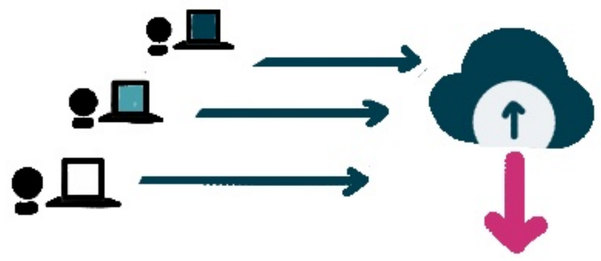
DEPLOY INFRASTRUCTURE



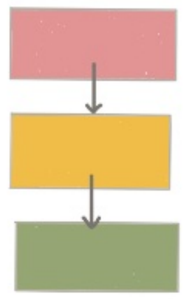
UNDERSTANDING SOFTWARE DEPLOYMENT STRATEGIES
HELPS DEVELOP STRATEGIES FOR INFRASTRUCTURE DEPLOYMENT

WHERE TO DEPLOY FROM?

MY MACHINE



A CENTRAL SERVICE

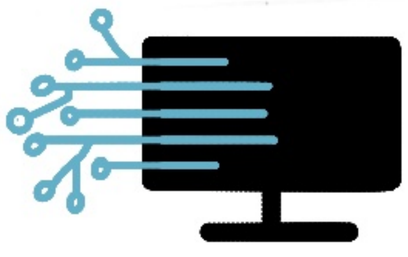


A DELIVERY PIPELINE

AND MORE!

TRIGGERING INFRASTRUCTURE DEPLOYMENTS

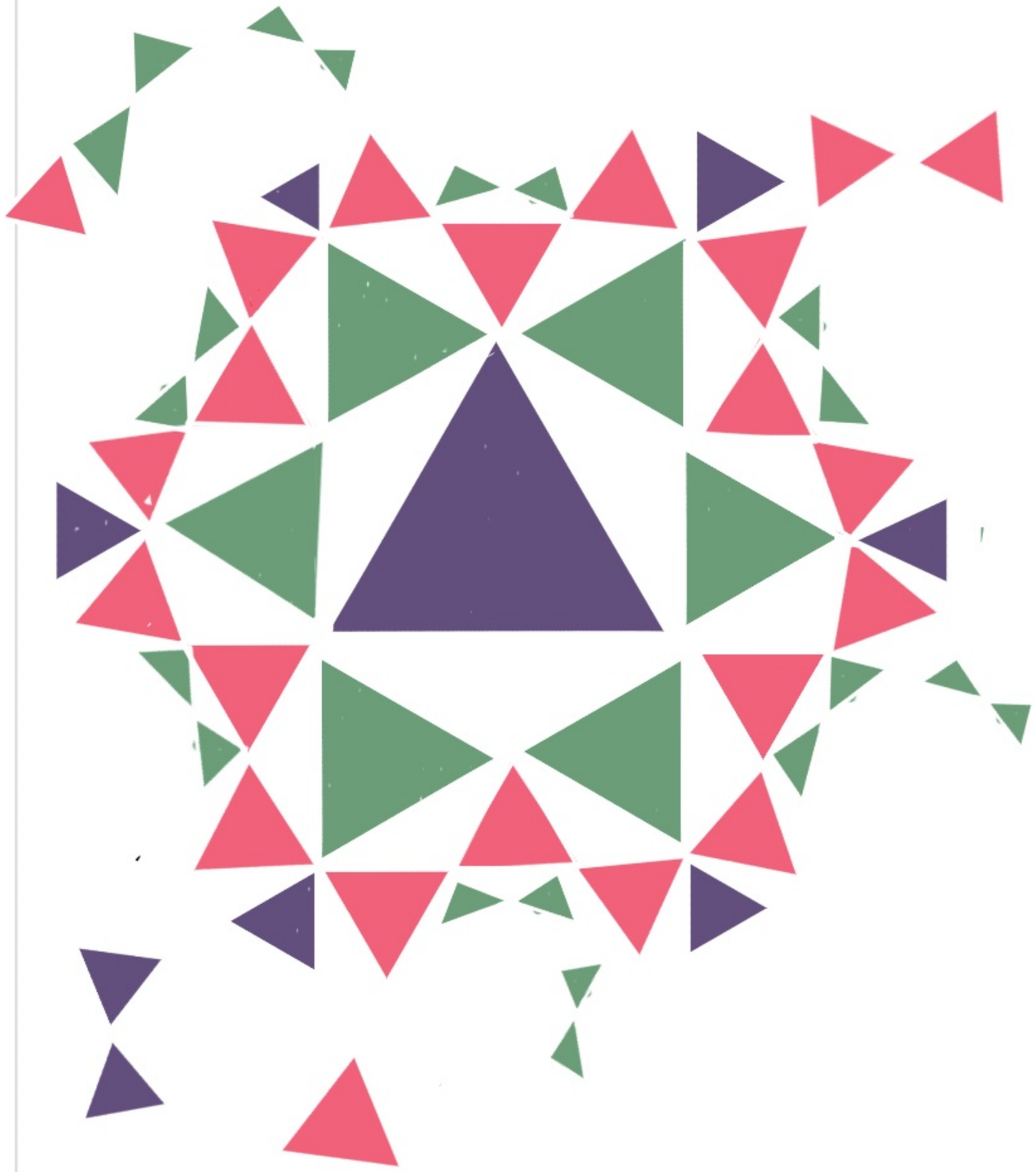
MANUAL



AUTOMATIC

AND THE MANY WAYS TO IMPLEMENT EITHER APPROACH

CHANGE EXISTING INFRASTRUCTURE



DELIVER CHANGES IN SMALL INCREMENTS

SAFELY CHANGE LIVE INFRASTRUCTURE

MINIMISE DOWNTIME DURING DEPLOYMENTS

RETAIN DATA WHEN CHANGING INFRASTRUCTURE

THE BOOK DISCUSSES THE ISSUES INVOLVED AND THE TECHNIQUES THAT GIVE CONTROL WHILE MAKING CHANGES.

GOVERNANCE

GOVERNANCE IS MAKING SURE THE SYSTEM IS BUILT RIGHT

THROUGH

- CONTINUOUS VALIDATION
- VISIBILITY INTO KEY OPERATIONS
- SAFE, ROUTINE CHANGE - PROCESSES
- MAINTAIN RECORDS

INCLUDES

- QUALITY
- PERFORMANCE
- SAFETY OF USER ASSETS
- RELIABILITY
- CORRECTNESS
- COST
- CAPACITY
- SCALING
- PRIVACY

REFERENCE

INFRASTRUCTURE AS CODE - KIEF MORRIS, O'REILLY MEDIA