» Skan.ai - chief Architect

» Ai.robotics - chief Architect

» Genpact - solution Architect

» Welldoc - chief Architect

» Microsoft

» Mercedes

» Siemens

» Honeywell

Mubarak

- **Cloud Computing**
- **Micro services**
- **Data Science**
- **ML**
- **DevSecOps**

- Years of experience
- Role
- Expectations


E Commerce
Customer
Admin

System Decomposition

Client Application

UI Application
Native Application
* WPF
* Swing
* Android
# Browser - Native Application
* Flash
* Activex
* Applet
# Browser - Server Pages
* JSP
* Php
# Browser - SPA
* Angular
* React

Background Application
* Windows Service
* Daemon

Low Level Application
* Device Driver

Server Application

Foreground Application
(Request/response)
* API Application

Background Application
Event Based Application
Scheduled Application

Client-Server
Applications

Two-tier Client Server
Multi(N)-tier client-server

Application Decomposition

Transformations at Similar levels of abstraction

1

Pipes and filter

Transformations at Different levels of abstraction

2

Layered
Hexagonal

Decomposition based on core and Variance

3

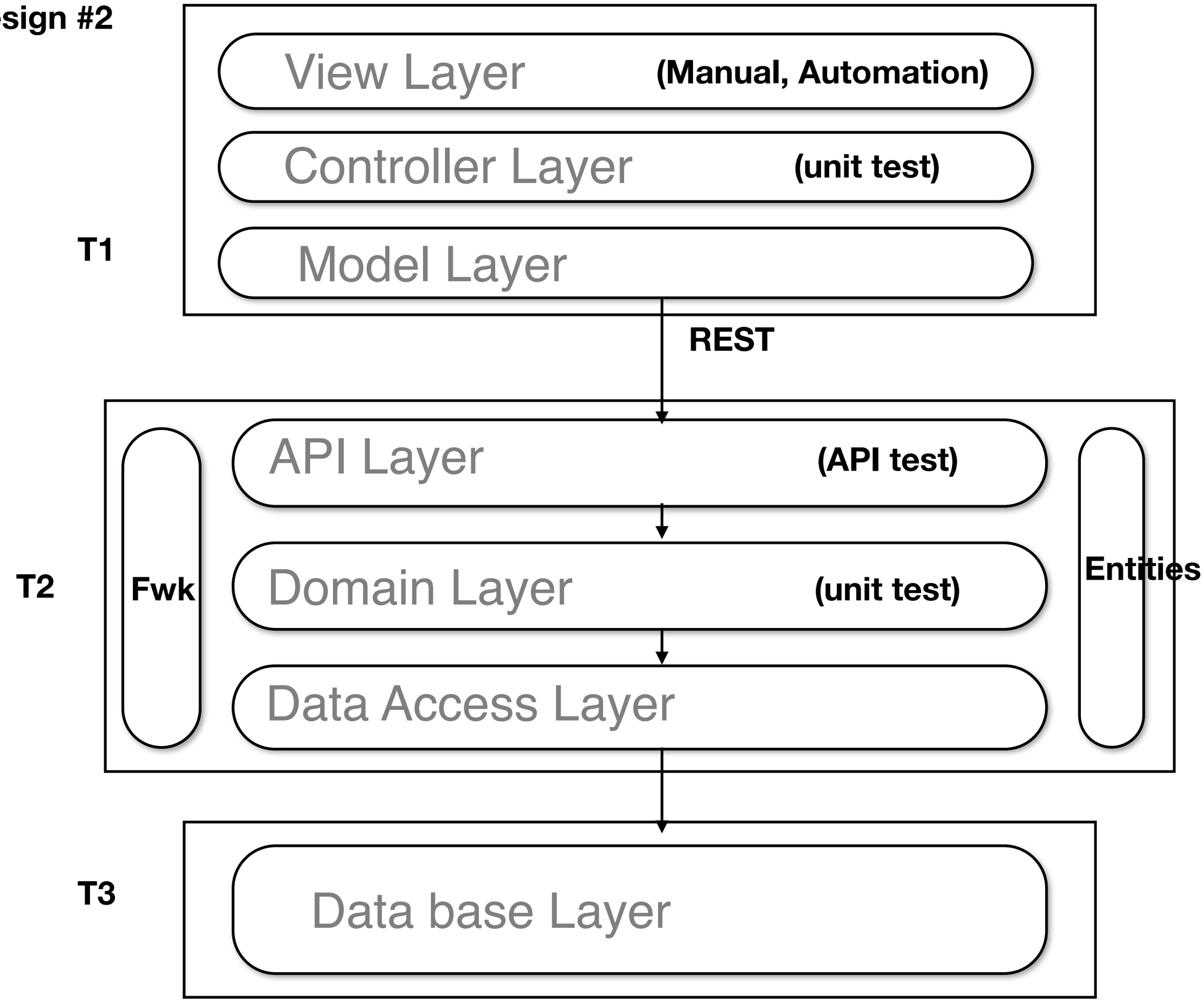
Micro Kernel

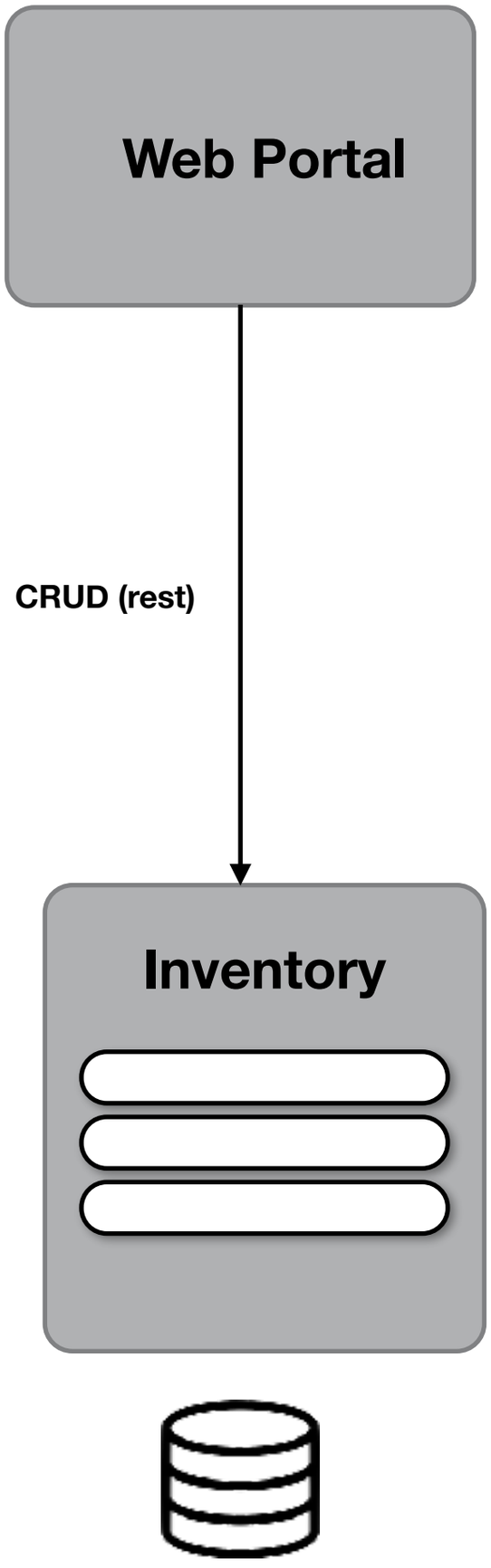
Decompose based on business capabilities (features)

4

Component based
Distributed component
Microservice
SOA

Design #2


View Layer
(Manual, Automation)
Controller Layer
(unit test)
Model Layer
T1
REST
API Layer
(API test)
Domain Layer
(unit test)
Data Access Layer
Fwk
Entities
T2
Data base Layer
T3


Web Portal
CRUD (rest)
Inventory
RDMBS

3rd normal

Pupil Table			
Pupil ID	**First Name**	**Last Name**	**ClassID**
1	Bob	Jones	1
2	Bill	Jones	2
3	Fred	Jones	1

ClassTable		
Class ID	**Class Name**	**Room**
1	CompSci 101	S16
2	English 101	M42

 # write friendly
 # transaction friendly

Denormalized Form

Weather			
city	**state**	**high**	**low**
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flagstaff	Arizona	88	69
San Diego	California	77	60
Albuquerque	New Mexico	80	72

< 10 TB

Binary data
(videos, images, audio, ..)

Lob data

Rest API Access
(access anywhere)

Cloud Object Storage

S3, Azure blob

File protocol Support

Cloud File Storage

EFS, Azure Files

Persistent
access from VM

Managed Disk

EBS

Ephemeral (short)
access from VM

Unmanaged Disk

Transactional
(ACID)

RDBMS

Complex
Relationships

Graph

Semi Structured
Data

Document

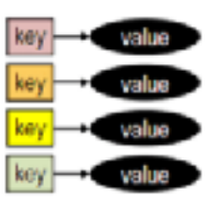
Simple
structure

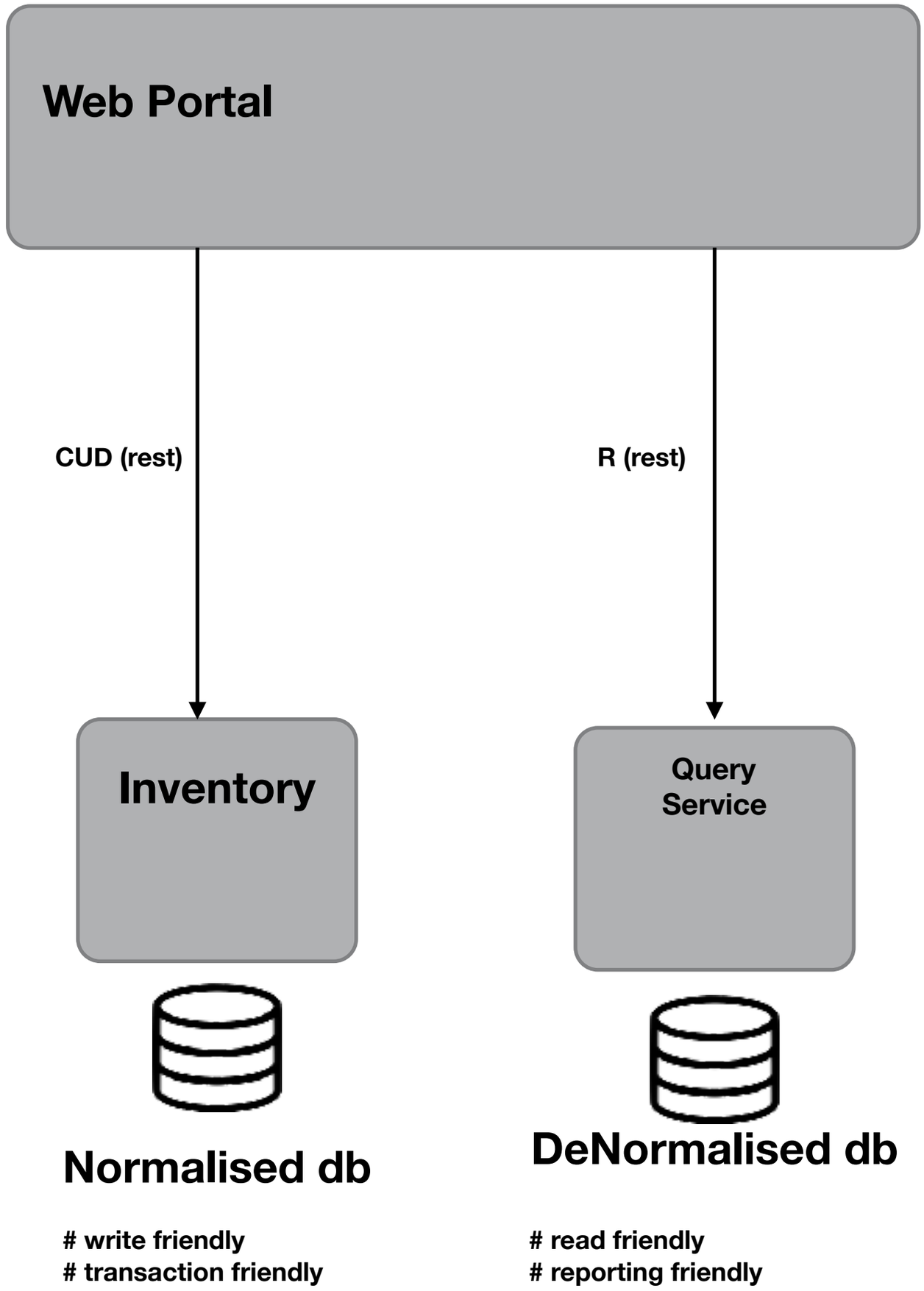
Persistent

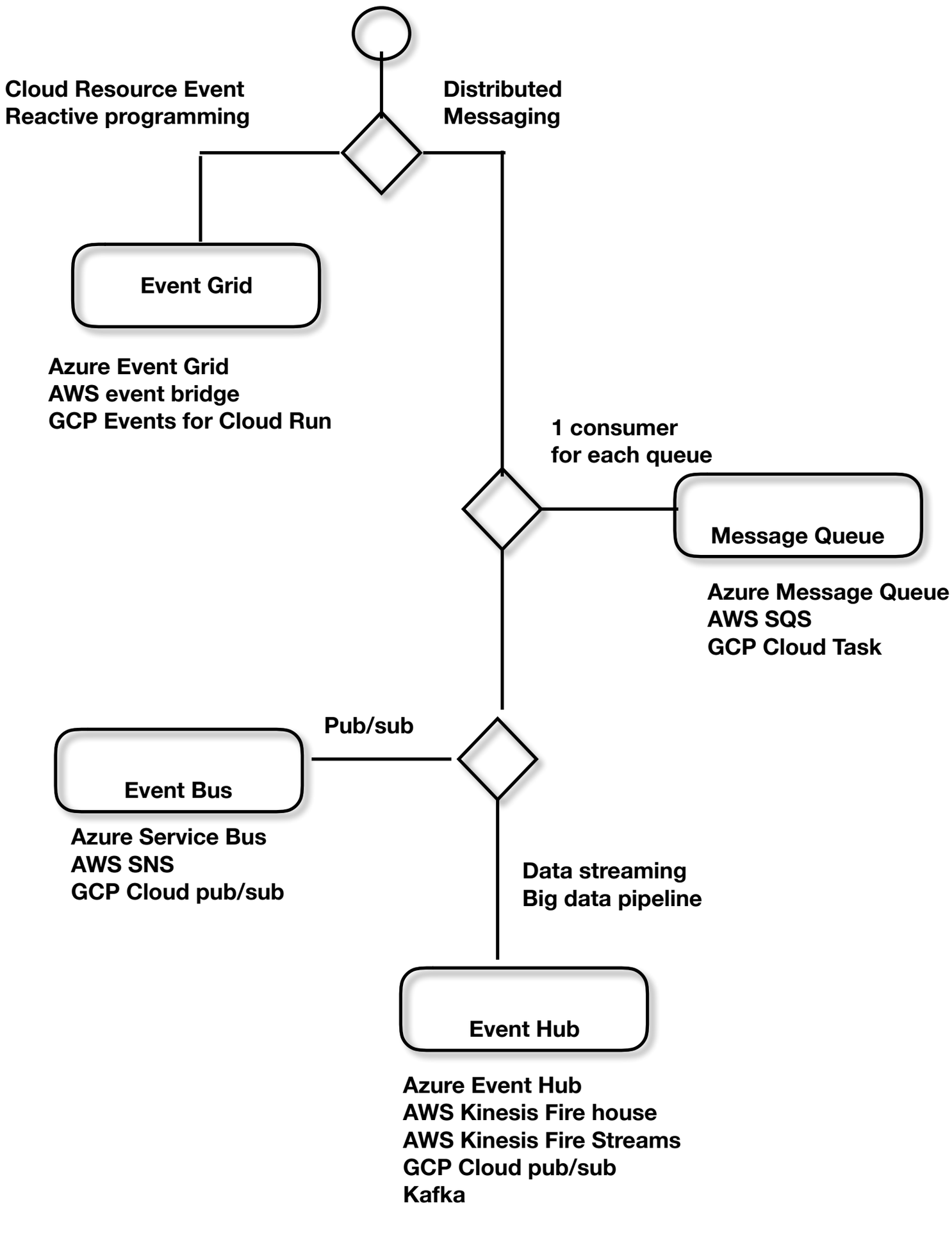
Key Value Store

Transient

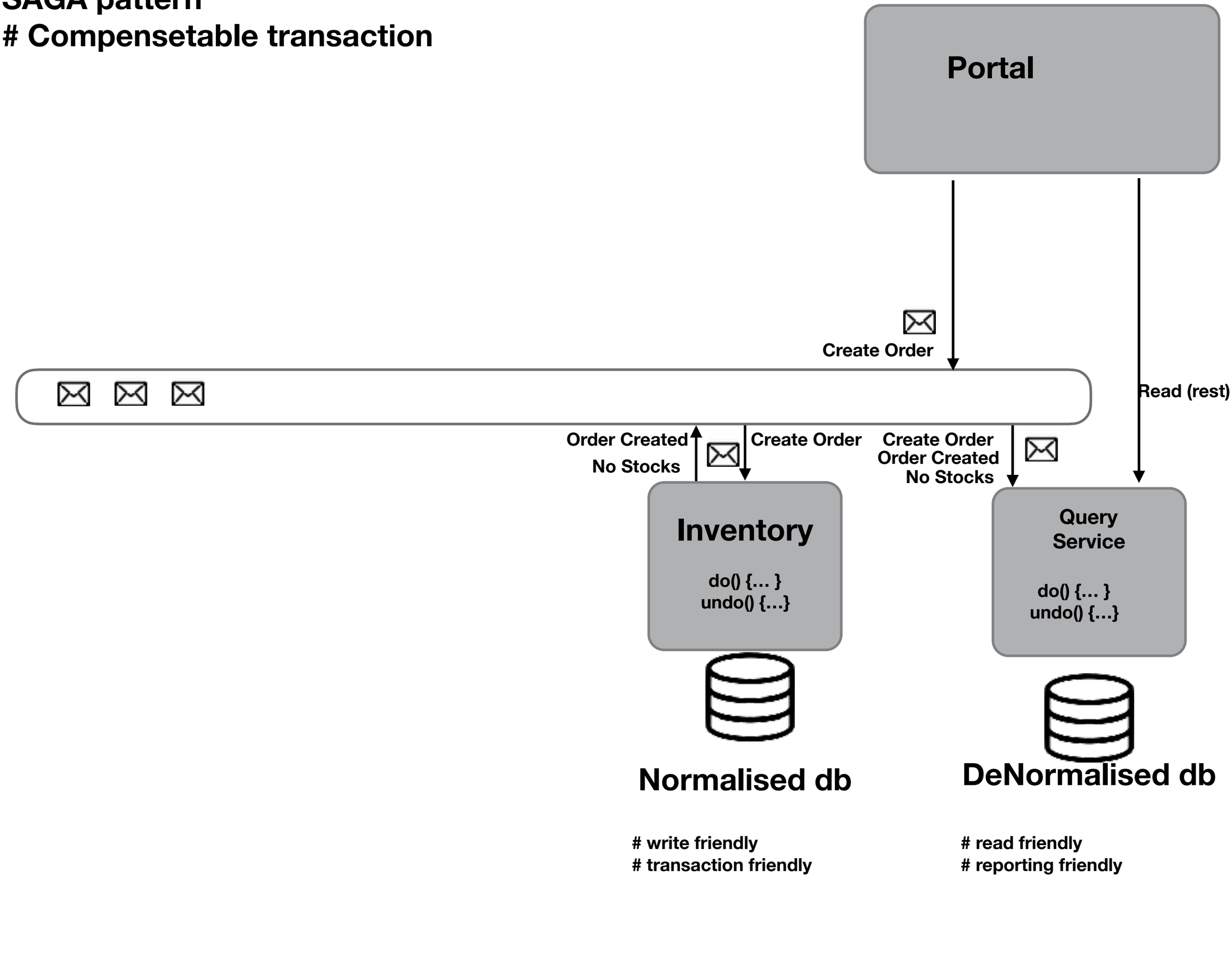
Key Value Cache

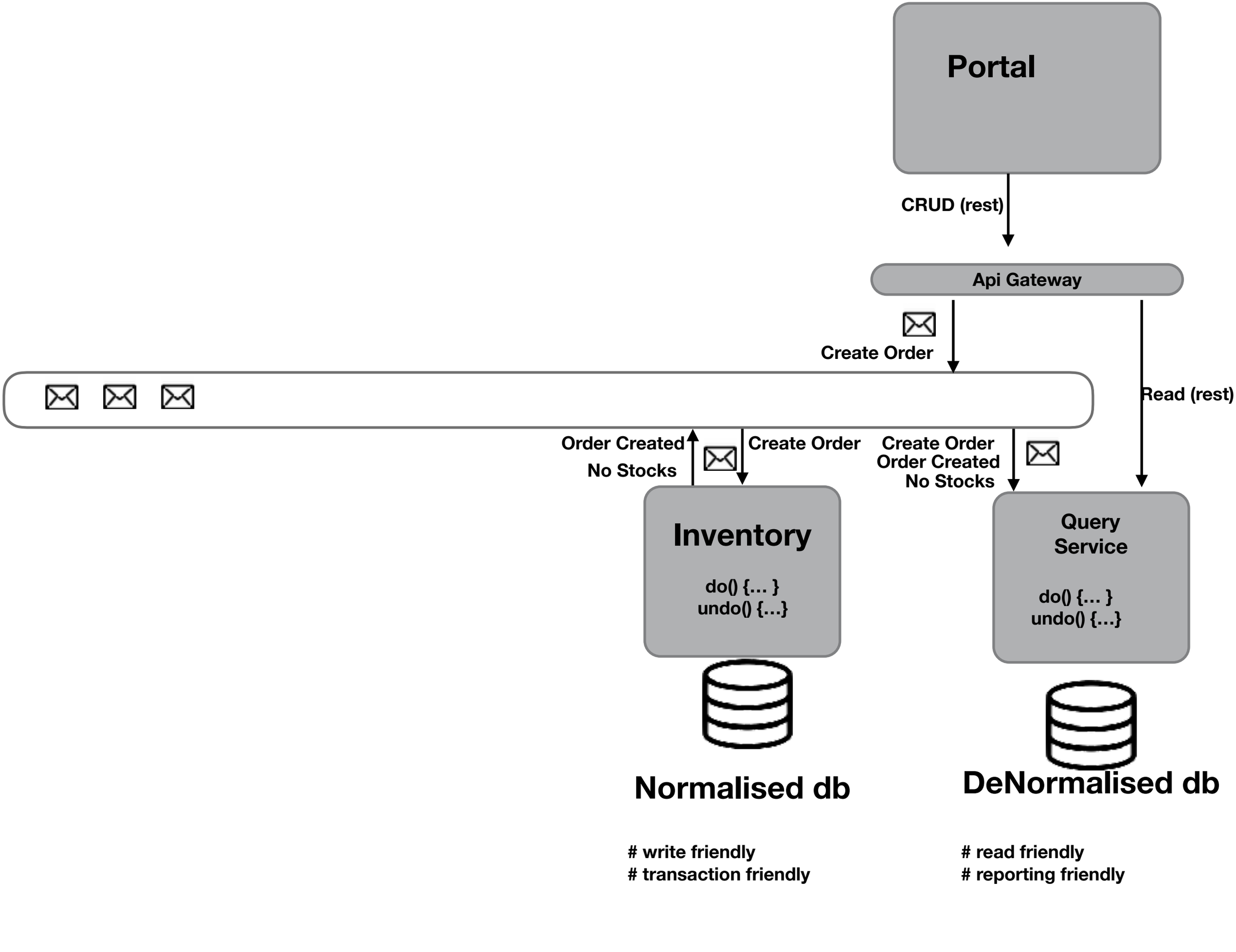



Web Portal
CUD (rest)
R (rest)
Inventory
Query
Service
Normalised db
DeNormalised db
write friendly
transaction friendly
read friendly
reporting friendly


Cloud Resource Event
Reactive programming
Distributed
Messaging
Event Grid
Azure Event Grid
AWS event bridge
GCP Events for Cloud Run
1 consumer
for each queue
Message Queue
Azure Message Queue
AWS SQS
GCP Cloud Task
Pub/sub
Event Bus
Azure Service Bus
AWS SNS
GCP Cloud pub/sub
Data streaming
Big data pipeline
Event Hub
Azure Event Hub
AWS Kinesis Fire house
AWS Kinesis Fire Streams
GCP Cloud pub/sub
Kafka

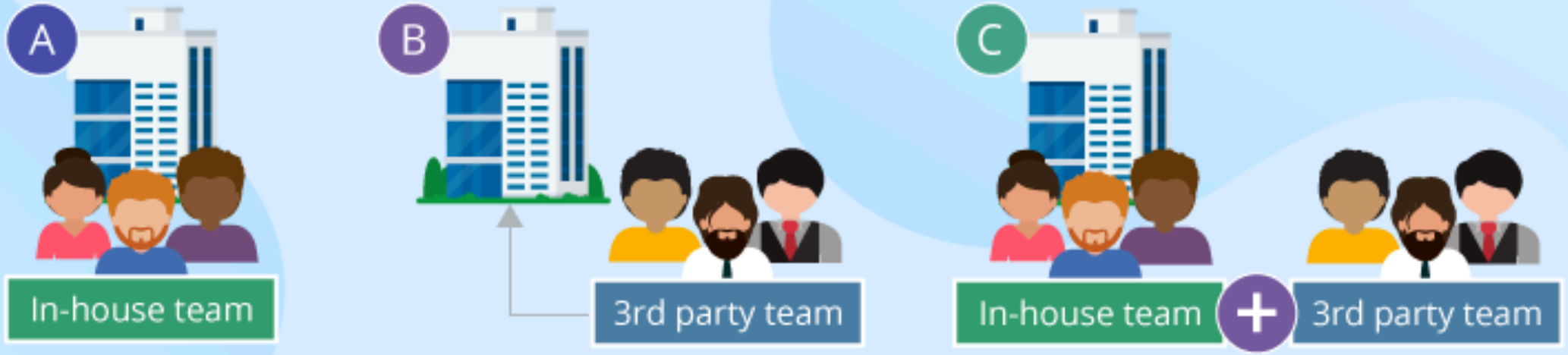
SAGA pattern
Compensetable transaction



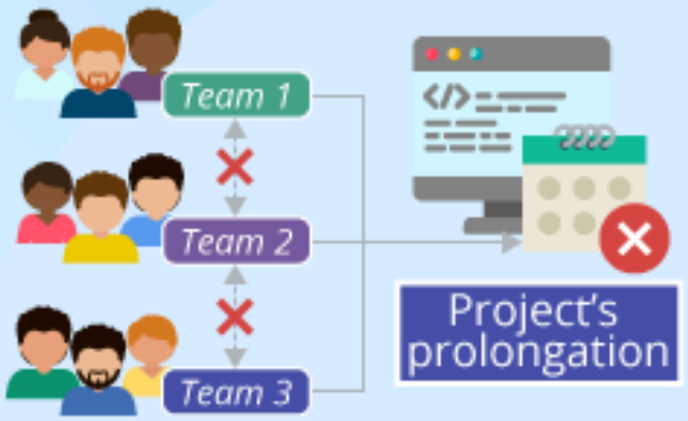

Portal
CRUD (rest)
Api Gateway
Create Order
Read (rest)
Order Created
No Stocks
Create Order
Create Order
Order Created
No Stocks
Inventory
do() {... }
undo() {...}
Query
Service
do() {... }
undo() {...}
Normalised db
DeNormalised db
write friendly
transaction friendly
read friendly
reporting friendly

pre-DevOps

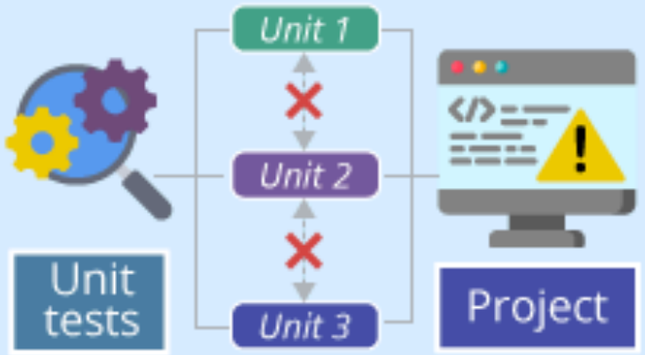
1. In-house, outsourced, or partially outsourced software development



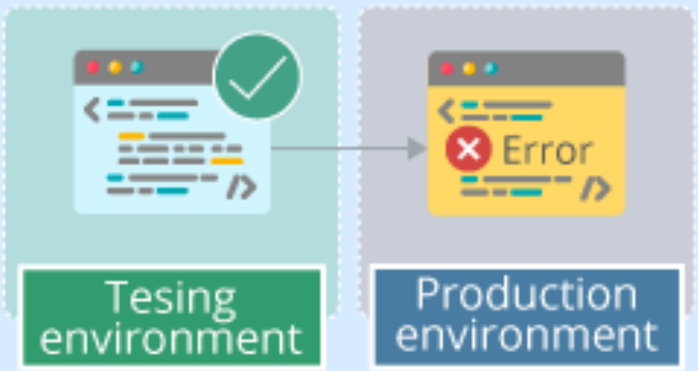
2. Strict segregation of duties between the departments



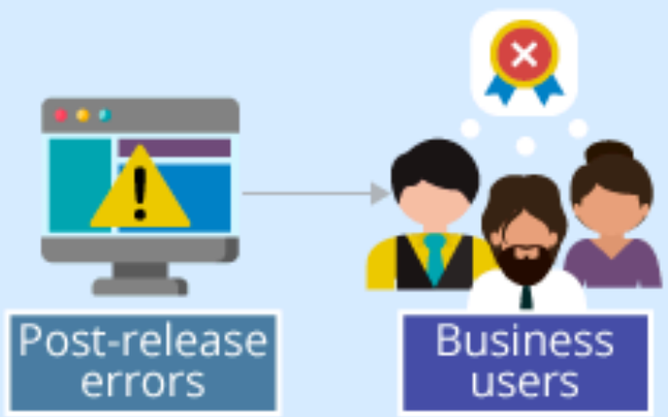
3. Insufficient coverage by tests



4. High probability of post-release errors

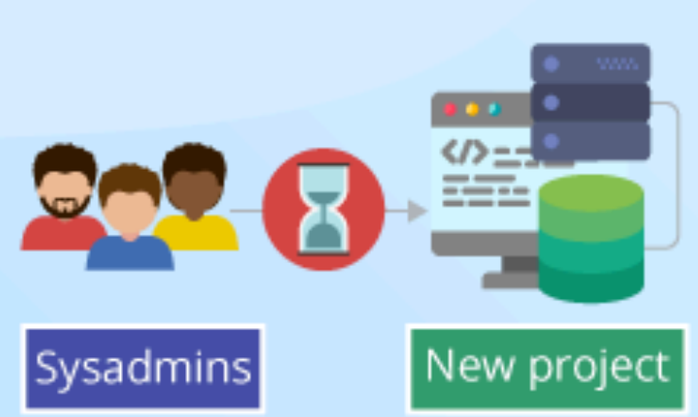


5. Lack of users' trust in software quality



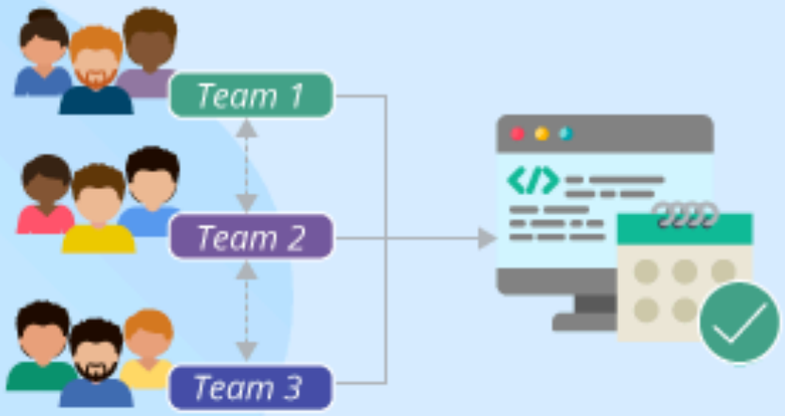
6. Weeks for updates and fixes

7. Time-consuming deployment of the infrastructure



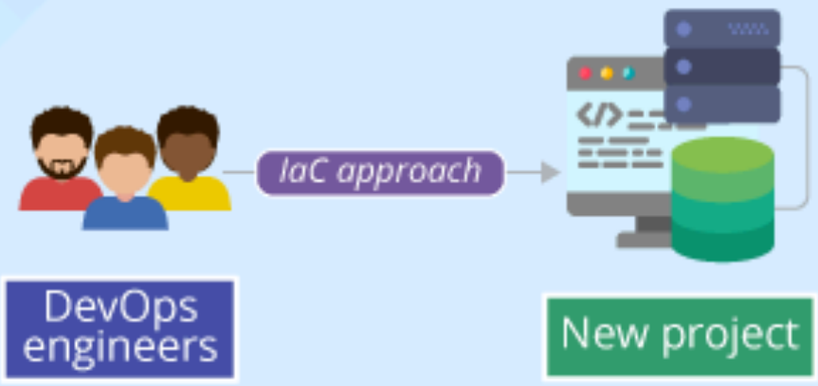
DevOps

1. Constant communication between the teams engaged in software development

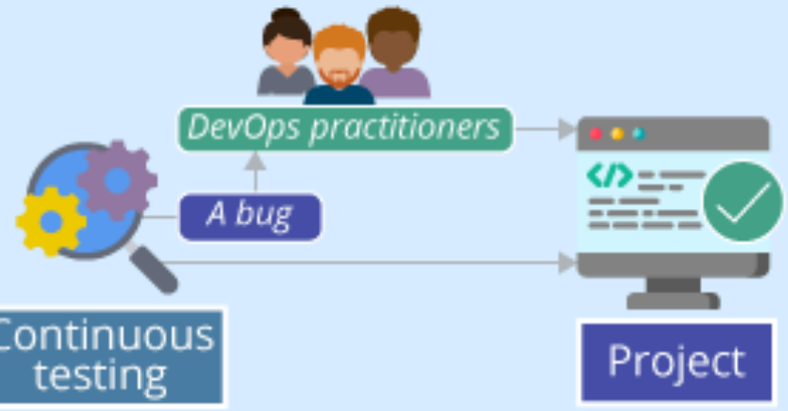


2. Fewer software failures caused by the differences in infrastructure configurations

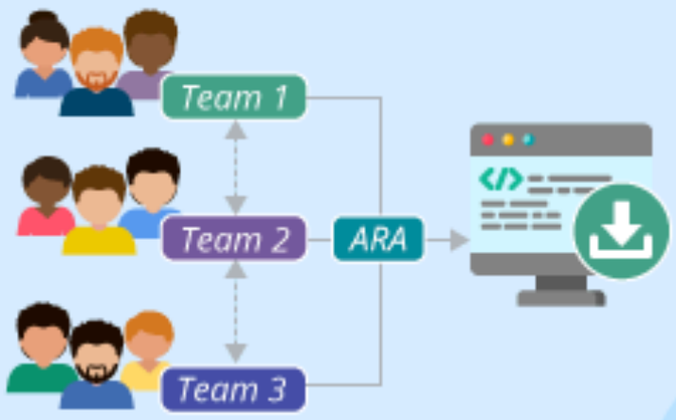
3. Fast provision of new infrastructure



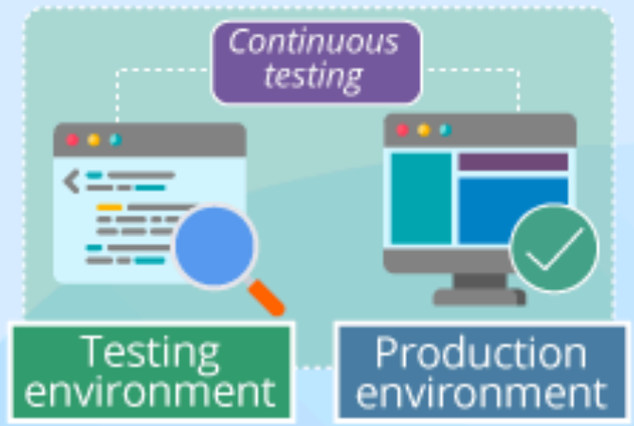
4. The increased amount of test automation



5.Quick and reliable delivery of application updates

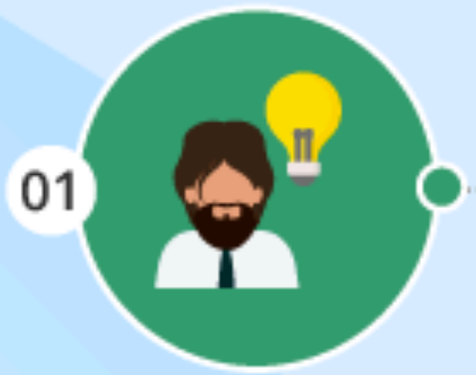


6. Fewer post-release errors



7. Improved users' trust

DEVOPS IMPLEMENTATION ROADMAP



01

Organizing a DevOps initiative

02

Building the DevOps strategy

03

Containerizing

04

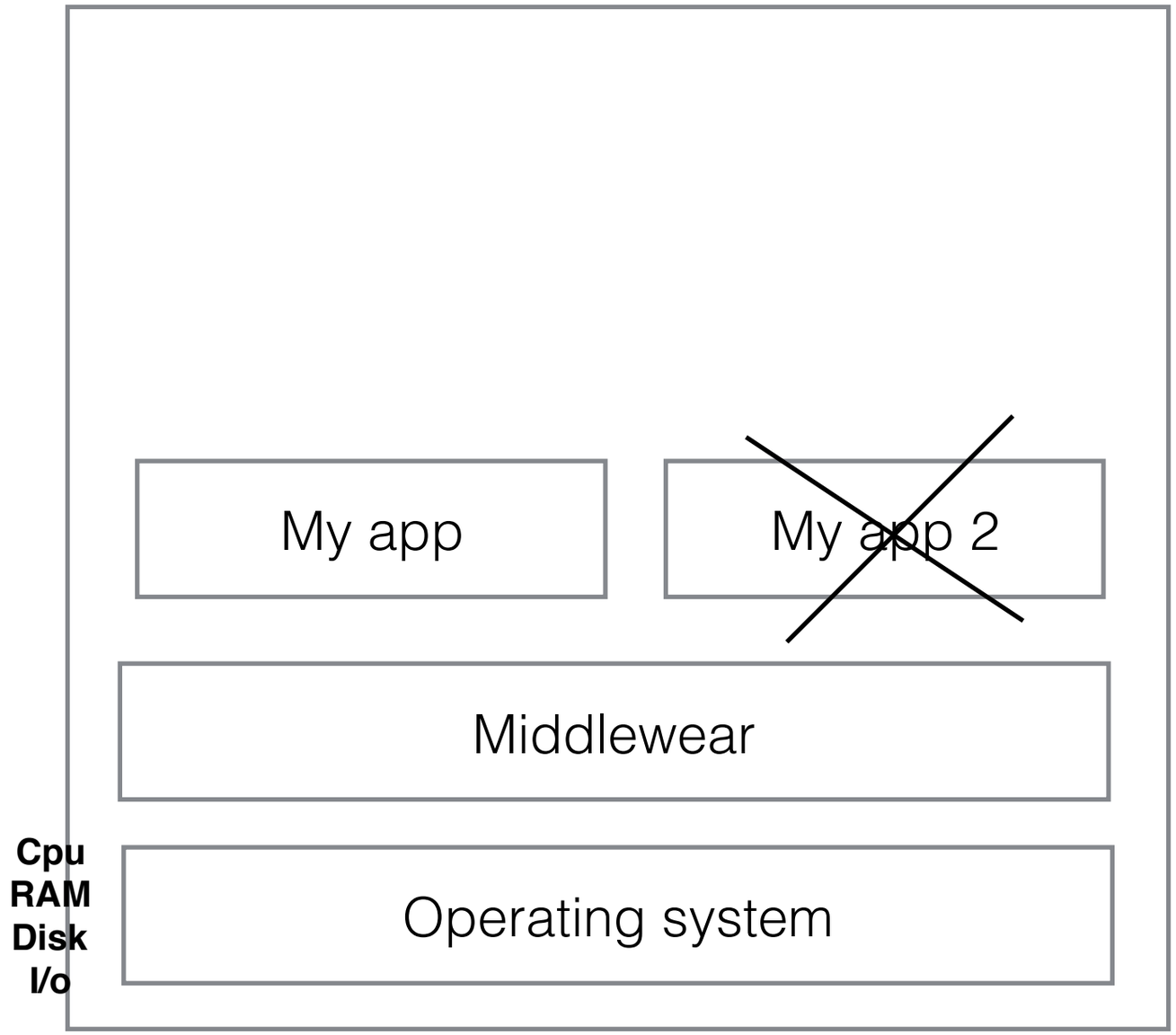
Integrating infrastructure automation with CI/CD tools

05

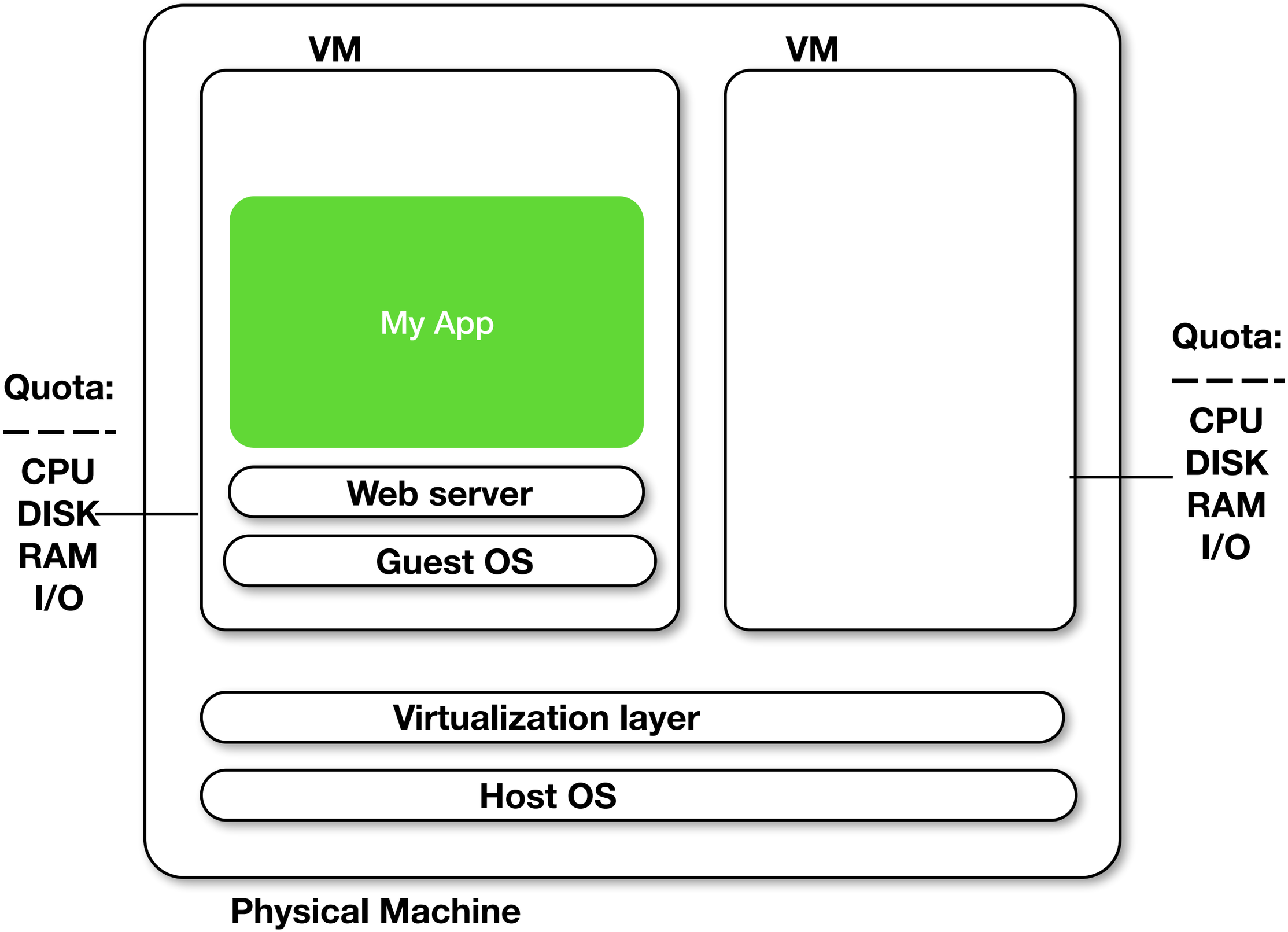
Increasing the amount of test automation and aligning QA with development

06

Ensuring total application performance monitoring



Flask + apache
Flask + gunicorn,
tomcat,
jboss,
Iis,
Express + nodejs,


VM
VM
My App
Web server
Guest OS
Quota:
CPU
DISK
RAM
I/O
Quota:
CPU
DISK
RAM
I/O
Virtualization layer
Host OS
Physical Machine

Design #2

VM1

VM2

My app 1

My app 3

My app 2

Middlewear

Middlewear

Unix Guest Os

Win Guest Os

Cpu
RAM
Disk
I/o

Cpu
RAM
Disk
I/o

Virtualisation layer

Host Os

Node

Physical Machine

VM

Container

Container

Quota:

CPU
DISK
RAM
I/O

My App

Web server

Containerisation layer

Guest OS

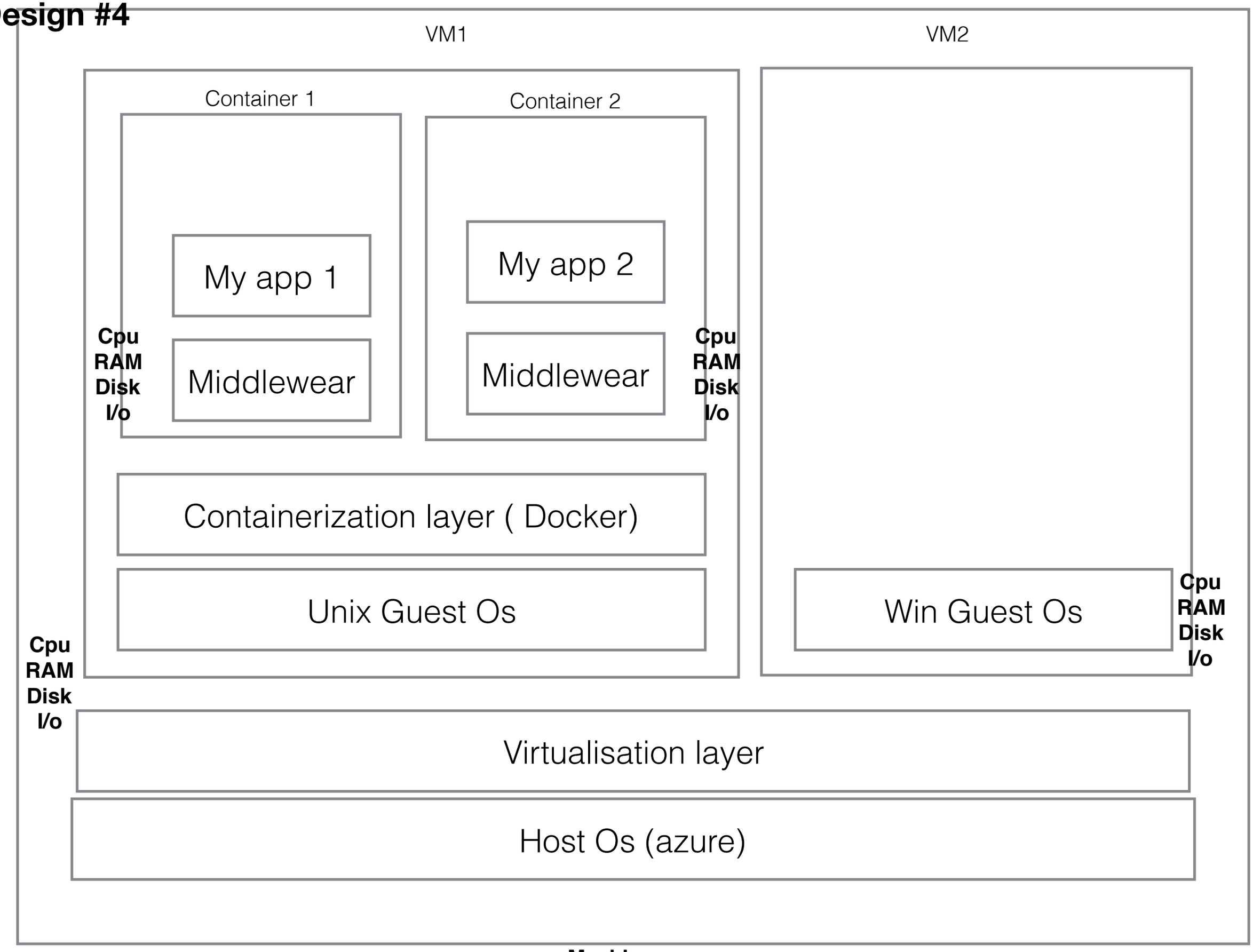
Quota:

CPU
DISK
RAM
I/O

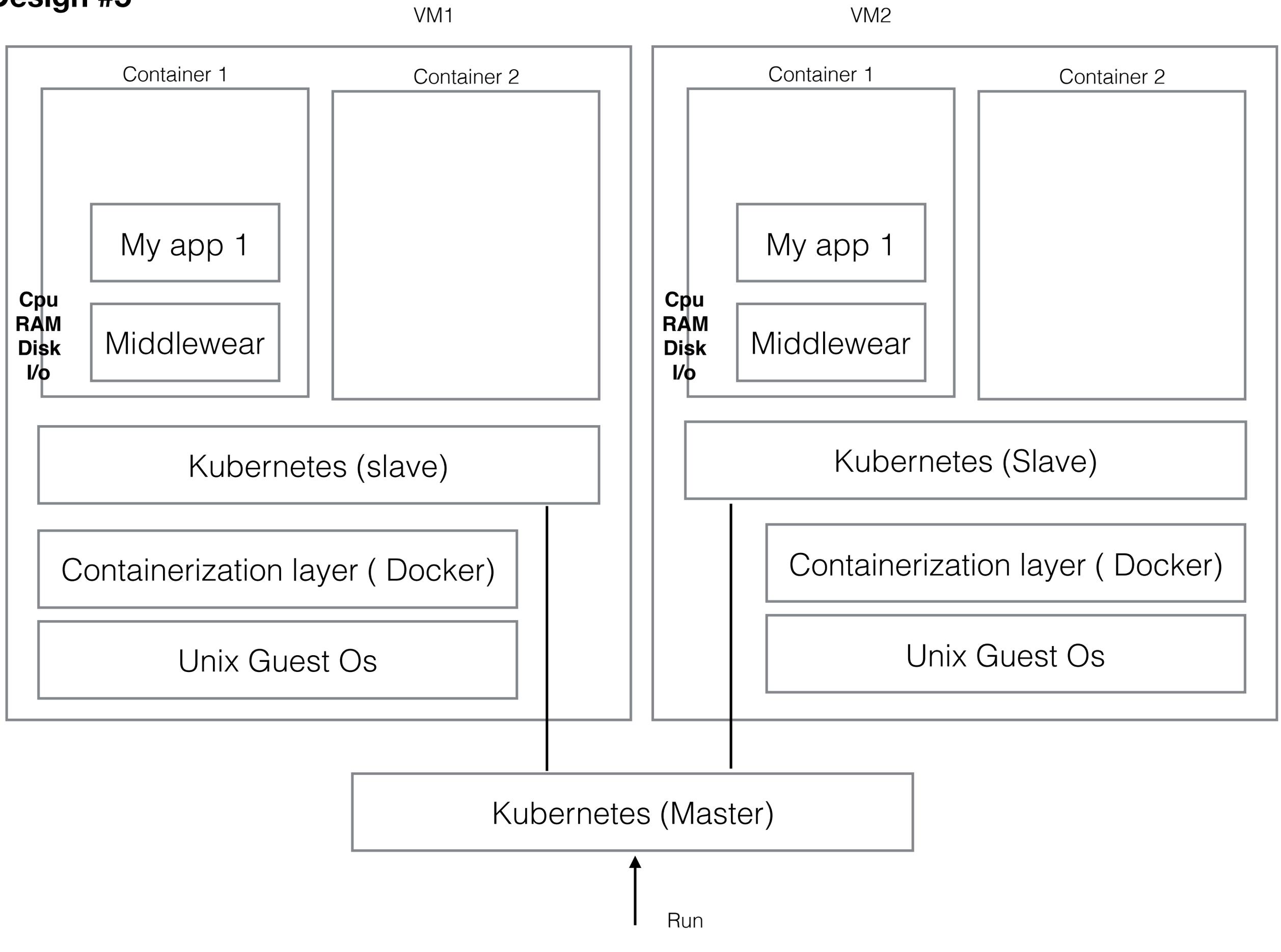
Virtualization layer

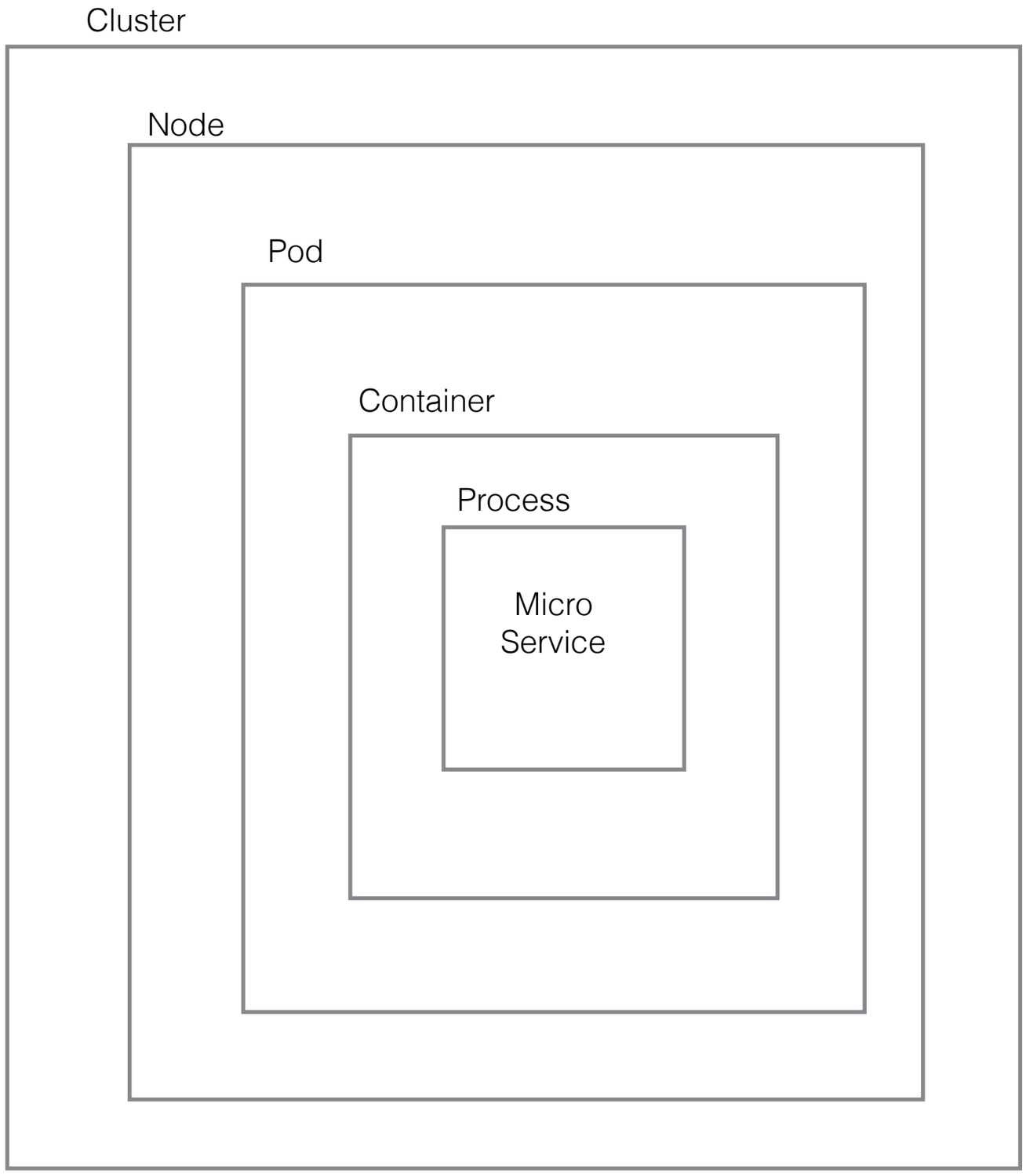
Host OS

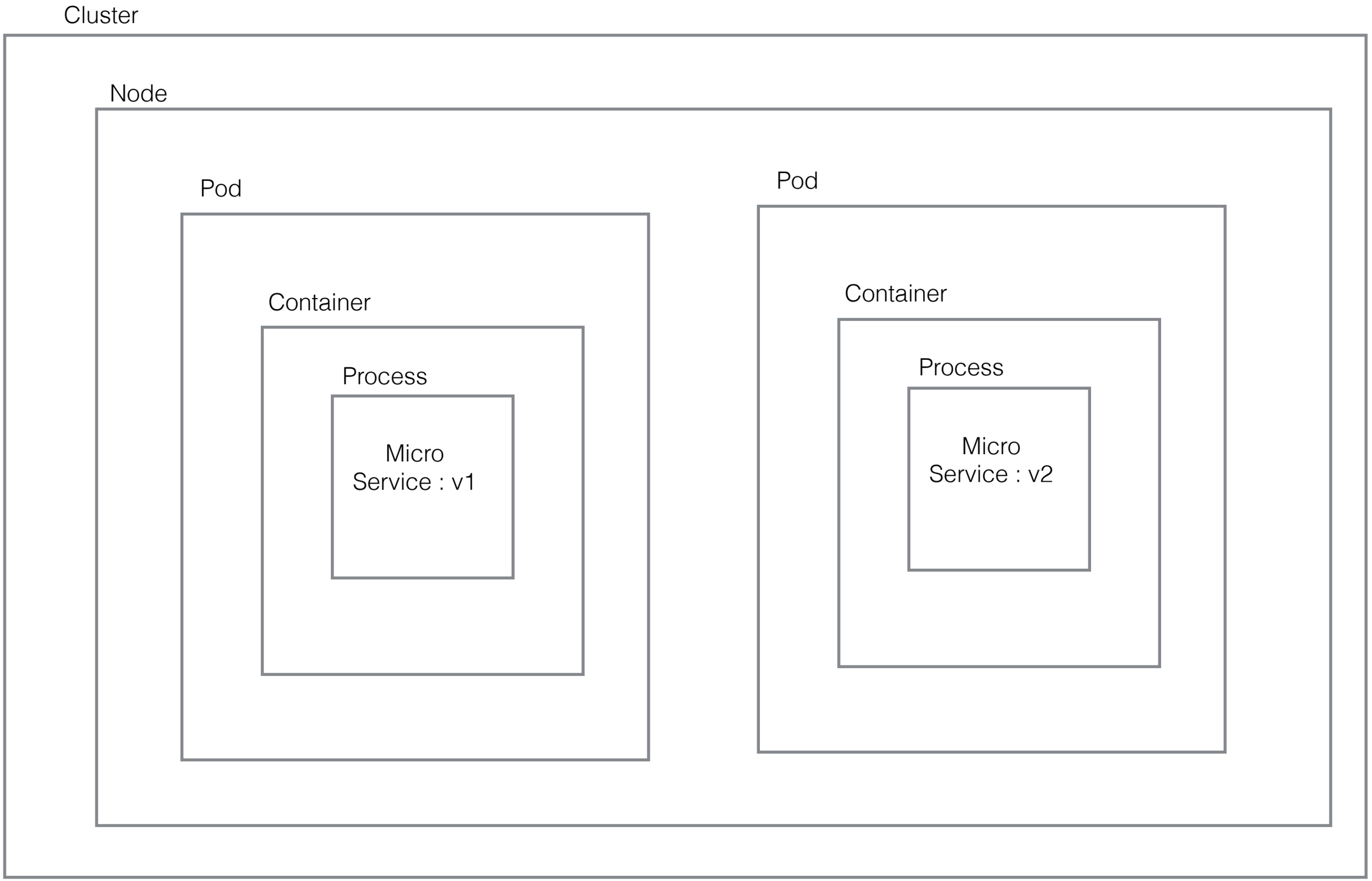
Design #4

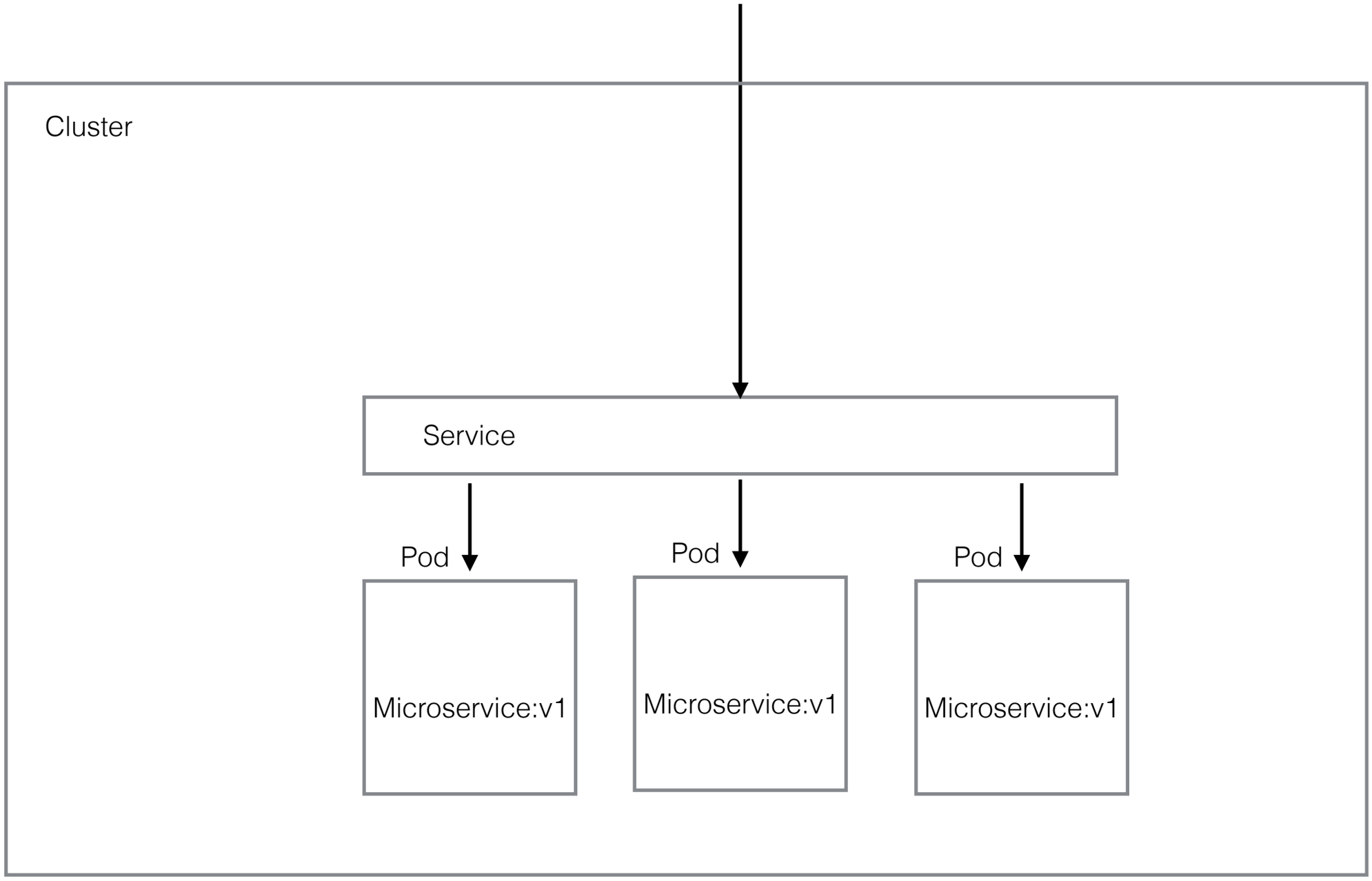

VM1
VM2
Container 1
Container 2
My app 1
My app 2
Middlewear
Middlewear
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Containerization layer (Docker)
Unix Guest Os
Win Guest Os
Cpu
RAM
Disk
I/o
Cpu
RAM
Disk
I/o
Virtualisation layer
Host Os (azure)
Machine

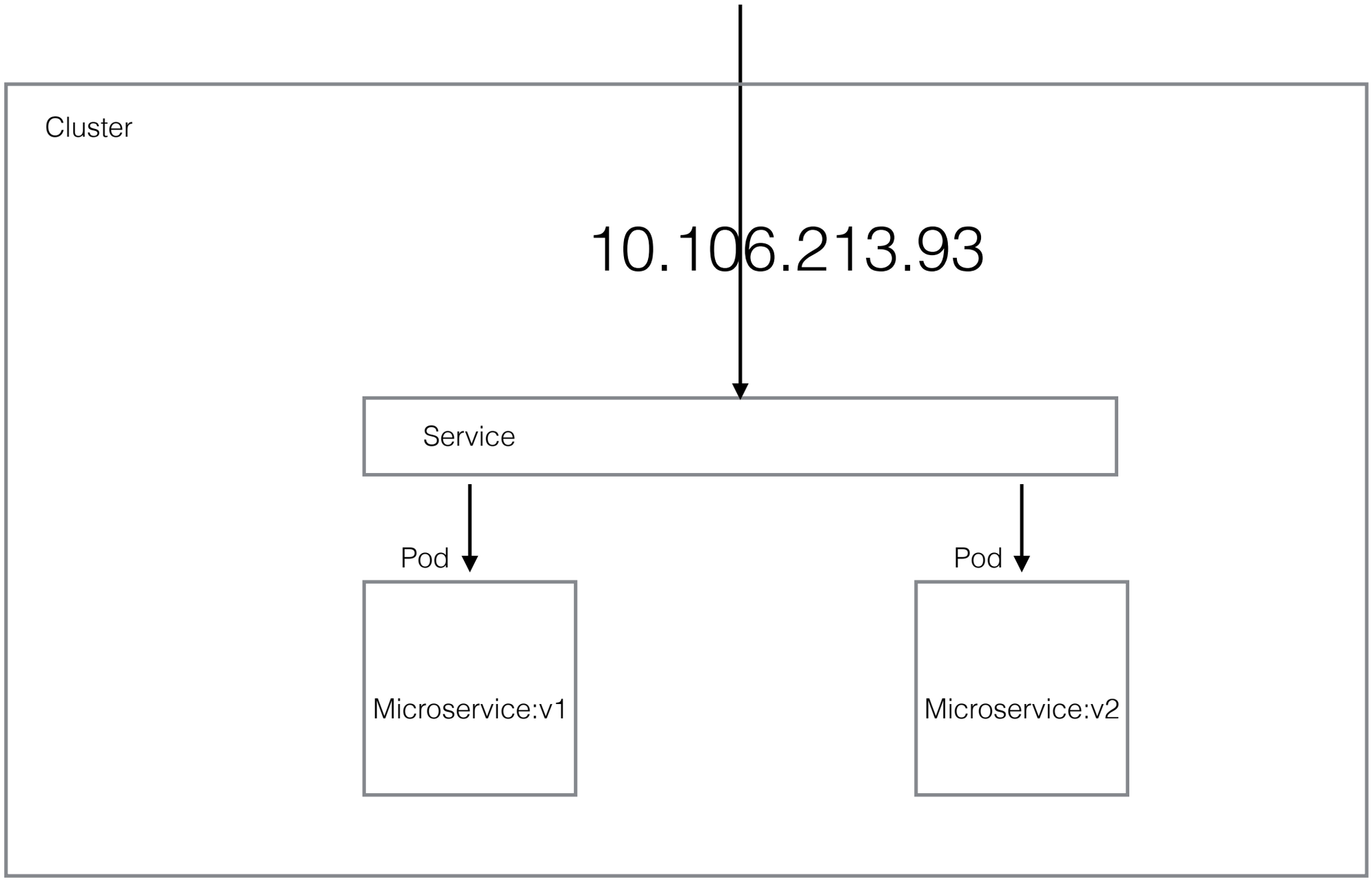
Design #5


VM1
VM2
Container 1
Container 2
My app 1
Middlewear
Cpu
RAM
Disk
I/o
Kubernetes (slave)
Containerization layer (Docker)
Unix Guest Os
Kubernetes (Slave)
Kubernetes (Master)
Run


Cluster
Node
Pod
Container
Process
Micro
Service


Cluster
Node
Pod
Container
Process
Micro
Service : v1
Pod
Container
Process
Micro
Service : v2


Cluster
Service
Pod
Pod
Pod
Microservice:v1
Microservice:v1
Microservice:v1


Cluster
10.106.213.93
Service
Pod
Pod
Microservice:v1
Microservice:v2

Cluster

10.106.213.94

Service B

XX.XX.XX.XX

Pod

Microservice B

XX.XX.XX.XX

Pod

Microservice B

10.106.213.93

Service A

XX.XX.XX.XX

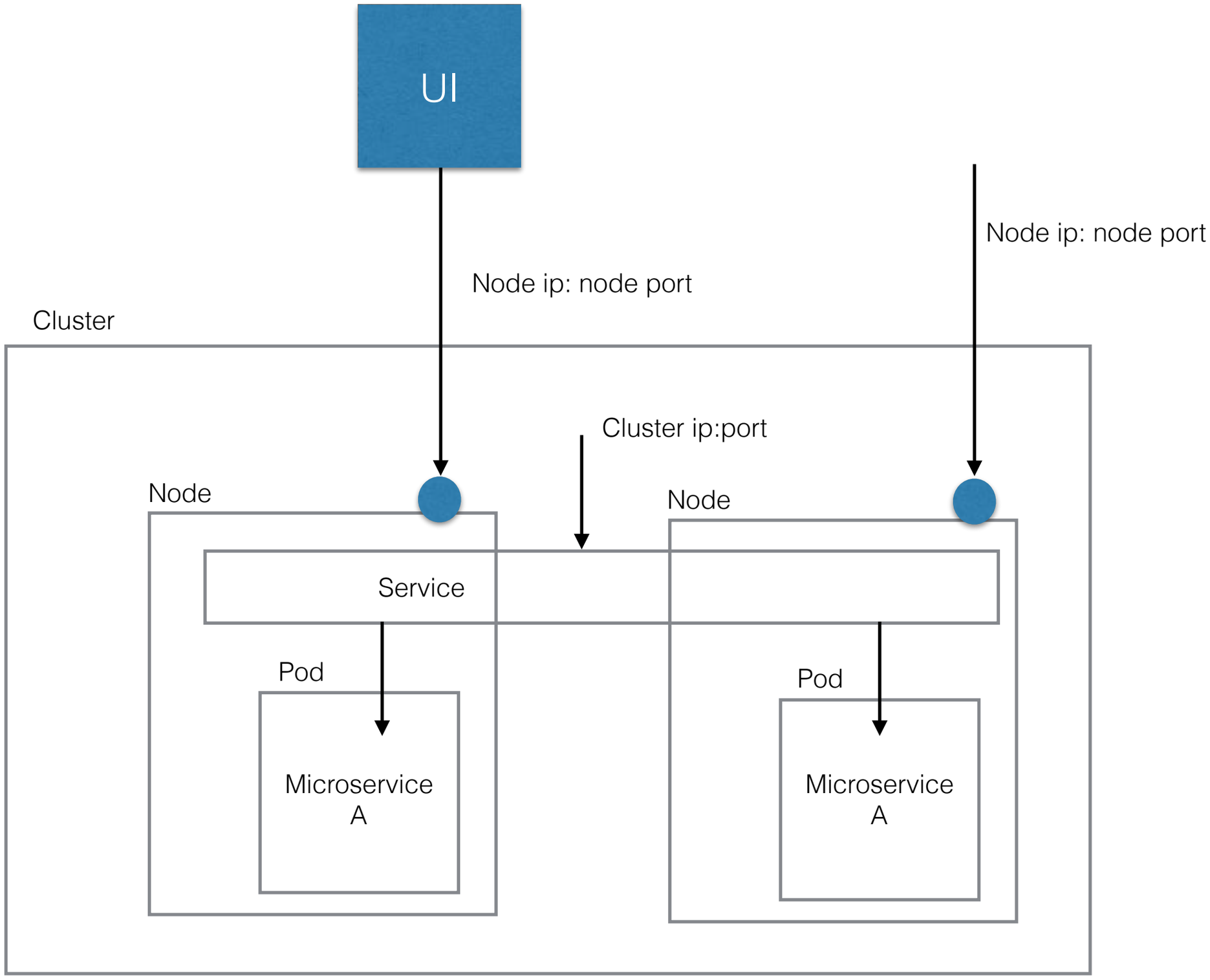
Pod

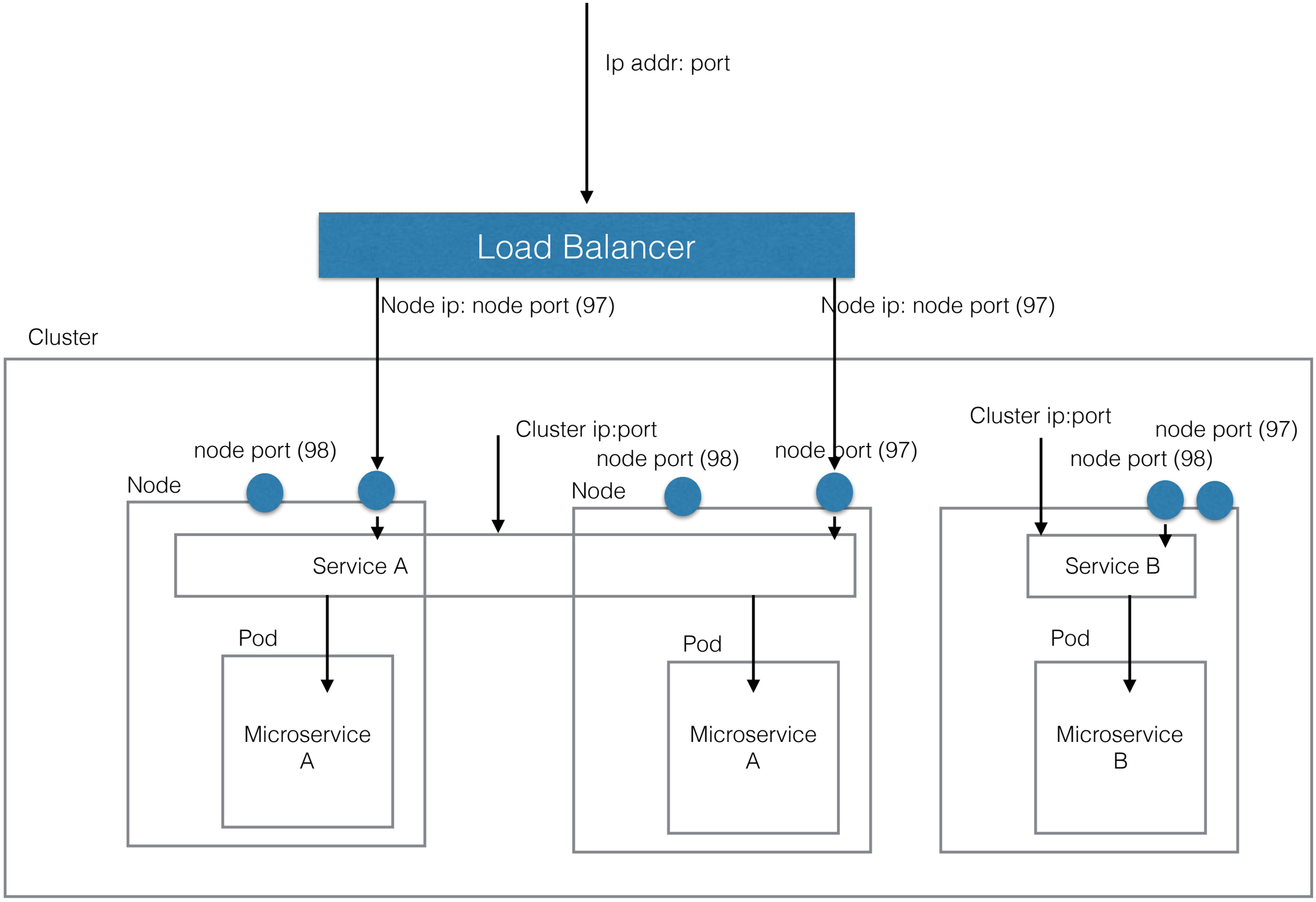
Microservice A

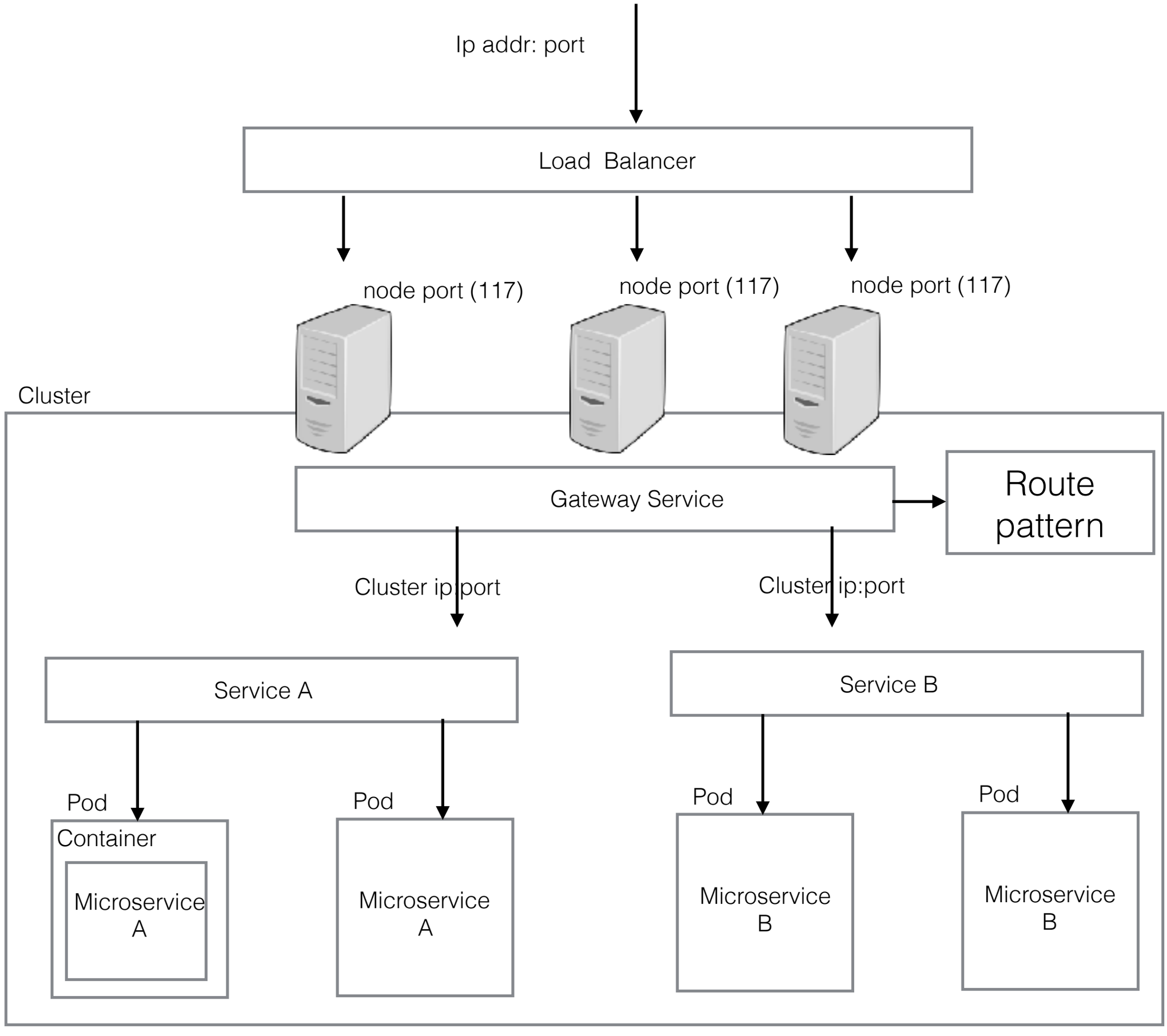
XX.XX.XX.XX

Pod

Microservice A


UI
Node ip: node port
Node ip: node port
Cluster
Cluster ip:port
Node
Node
Service
Pod
Pod
Microservice
A
Microservice
A


Ip addr: port
Load Balancer
Node ip: node port (97)
Node ip: node port (97)
Cluster
node port (98)
Node
Service A
Pod
Microservice
A
Cluster ip:port
node port (98)
Node
Pod
Microservice
A
node port (97)
Cluster ip:port
node port (97)
node port (98)
Service B
Pod
Microservice
B


Ip addr: port
Load Balancer
node port (117)
node port (117)
node port (117)
Cluster
Gateway Service
Route pattern
Cluster ip:port
Cluster ip:port
Service A
Service B
Pod
Container
Microservice A
Pod
Microservice A
Pod
Microservice B
Pod
Microservice B

My app

OS

Machine

My app

Middlewear

OS

Machine

My app

Middlewear

Guest OS

VM

Machine

My app

Middlewear

Container

Guest OS

VM

Machine

Kubernetes

Cluster

My app

Middlewear

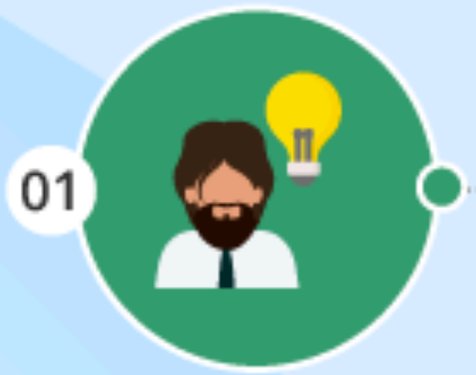
Container

Guest OS

VM

Machine

DEVOPS IMPLEMENTATION ROADMAP



01

Organizing a DevOps initiative

02

Building the DevOps strategy

03

Containerizing

04

Integrating infrastructure automation with CI/CD tools

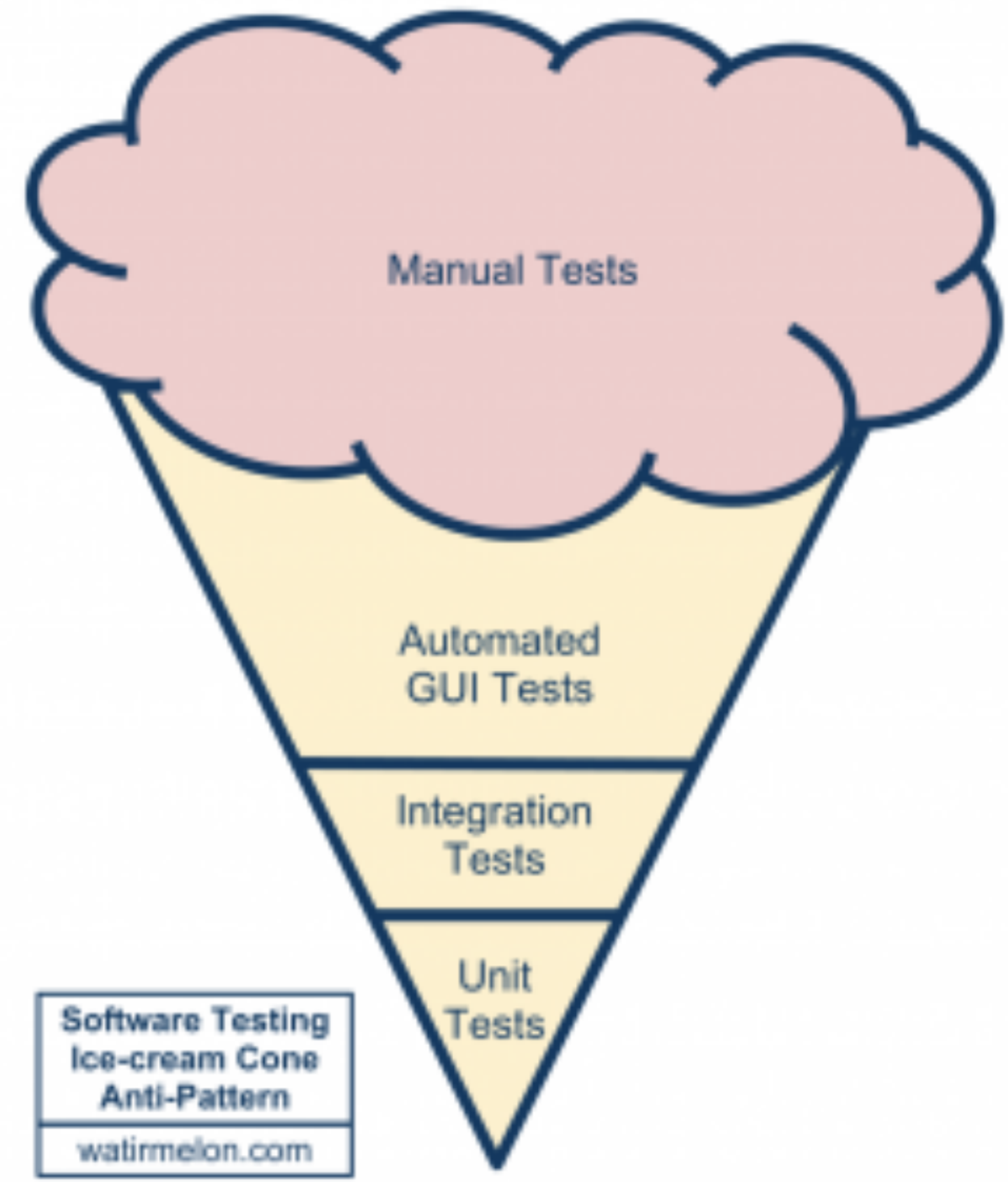
05

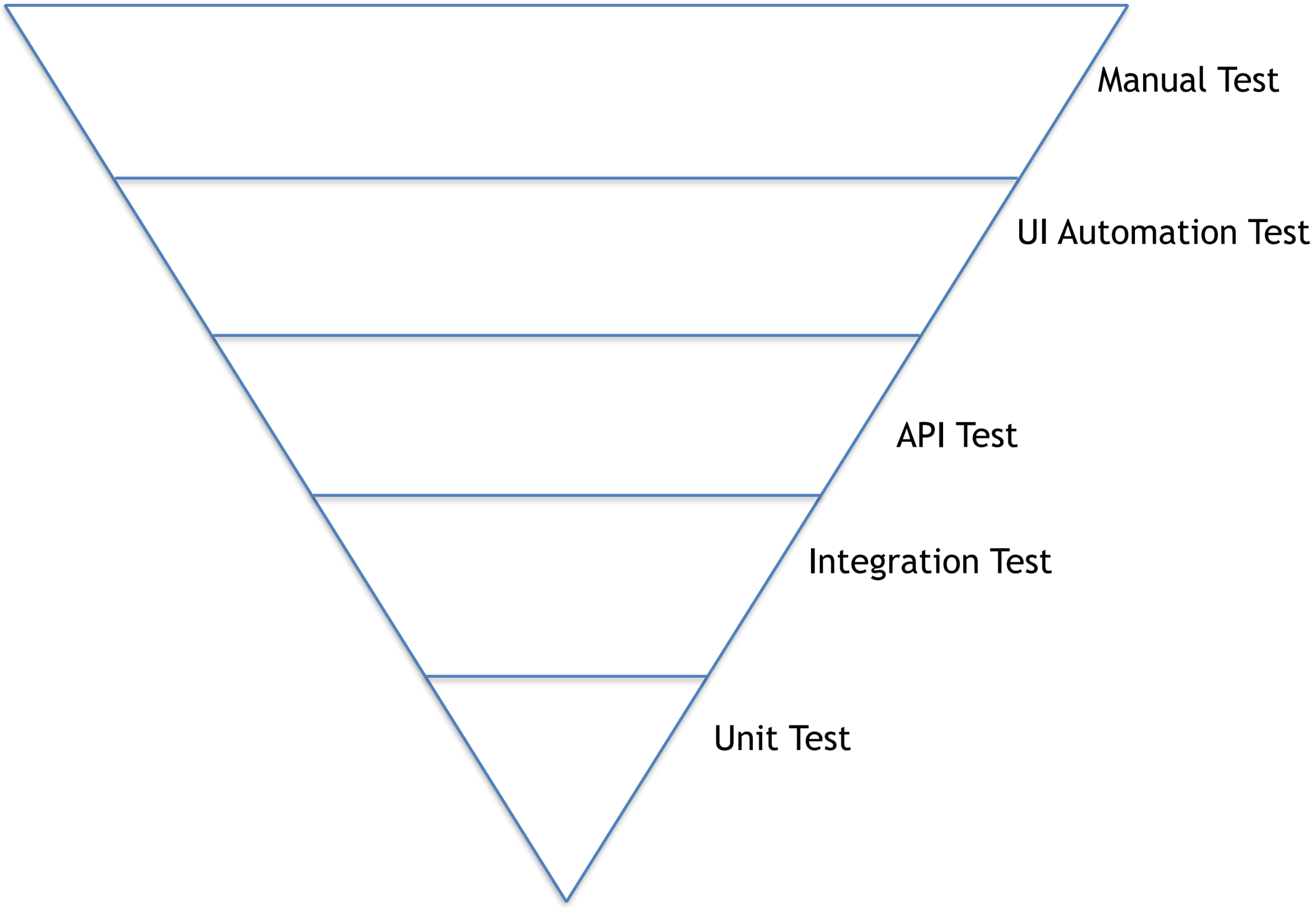
Increasing the amount of test automation and aligning QA with development

06

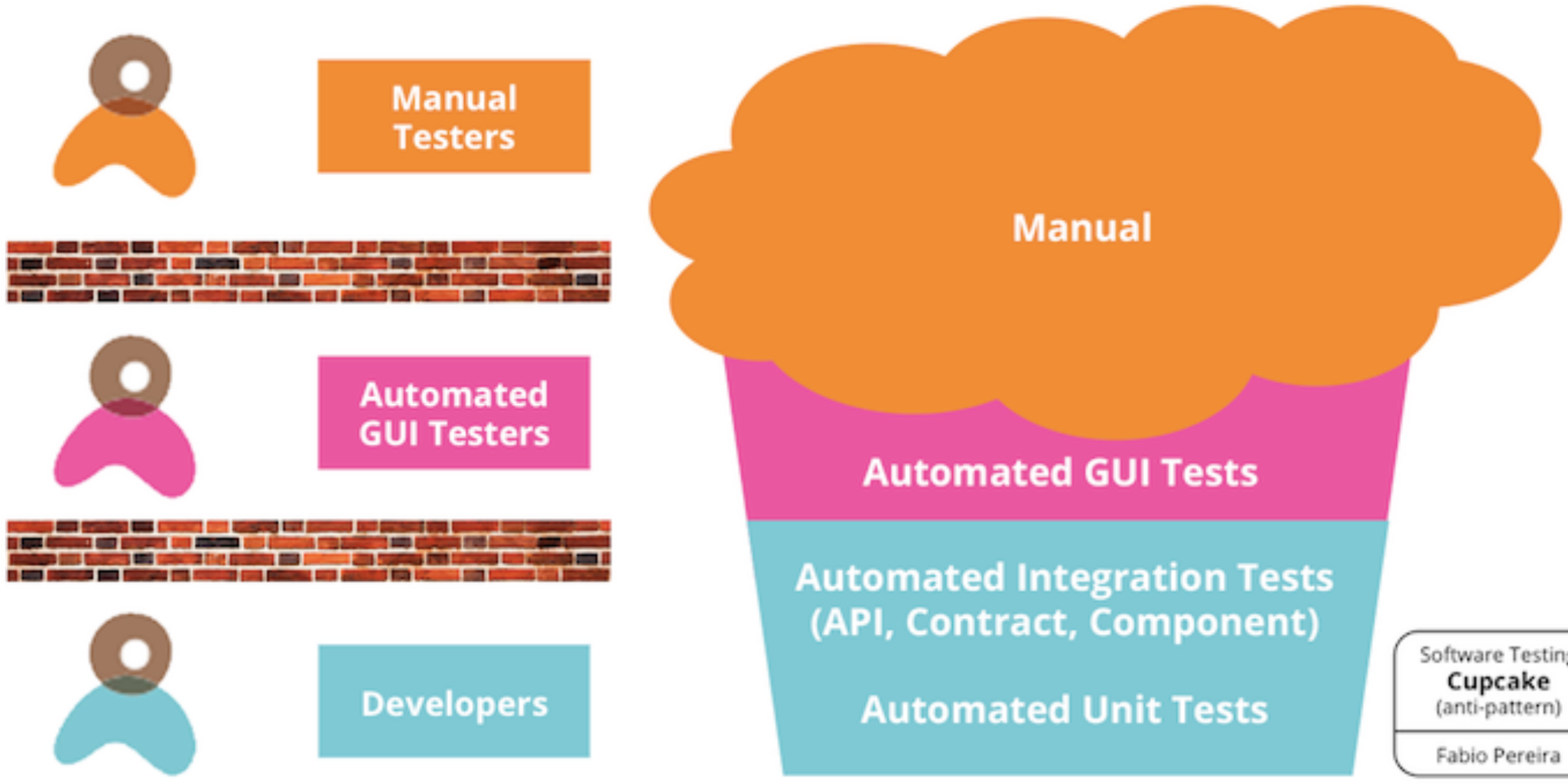
Ensuring total application performance monitoring

Ice Cream Cone Anti Pattern

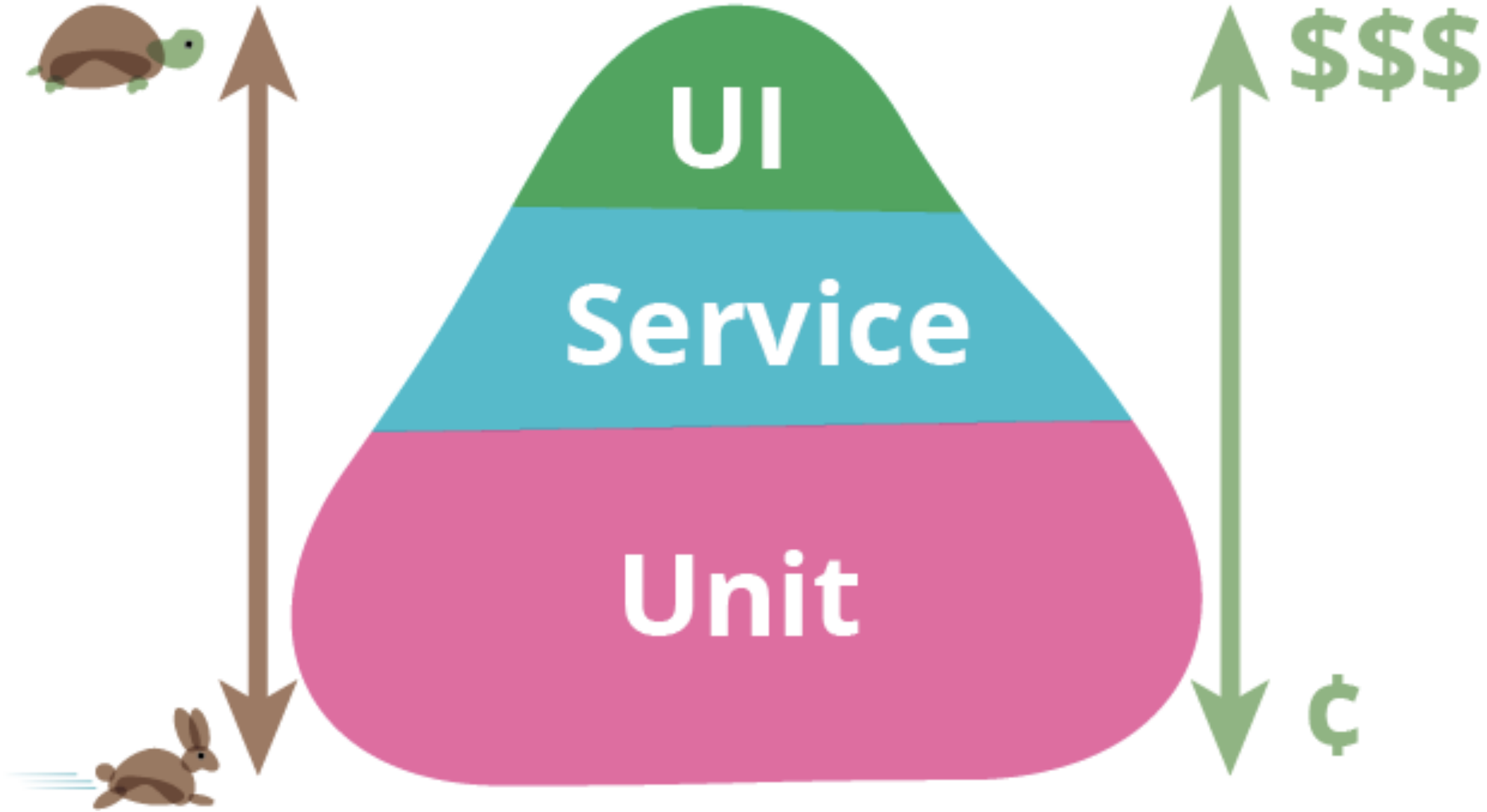


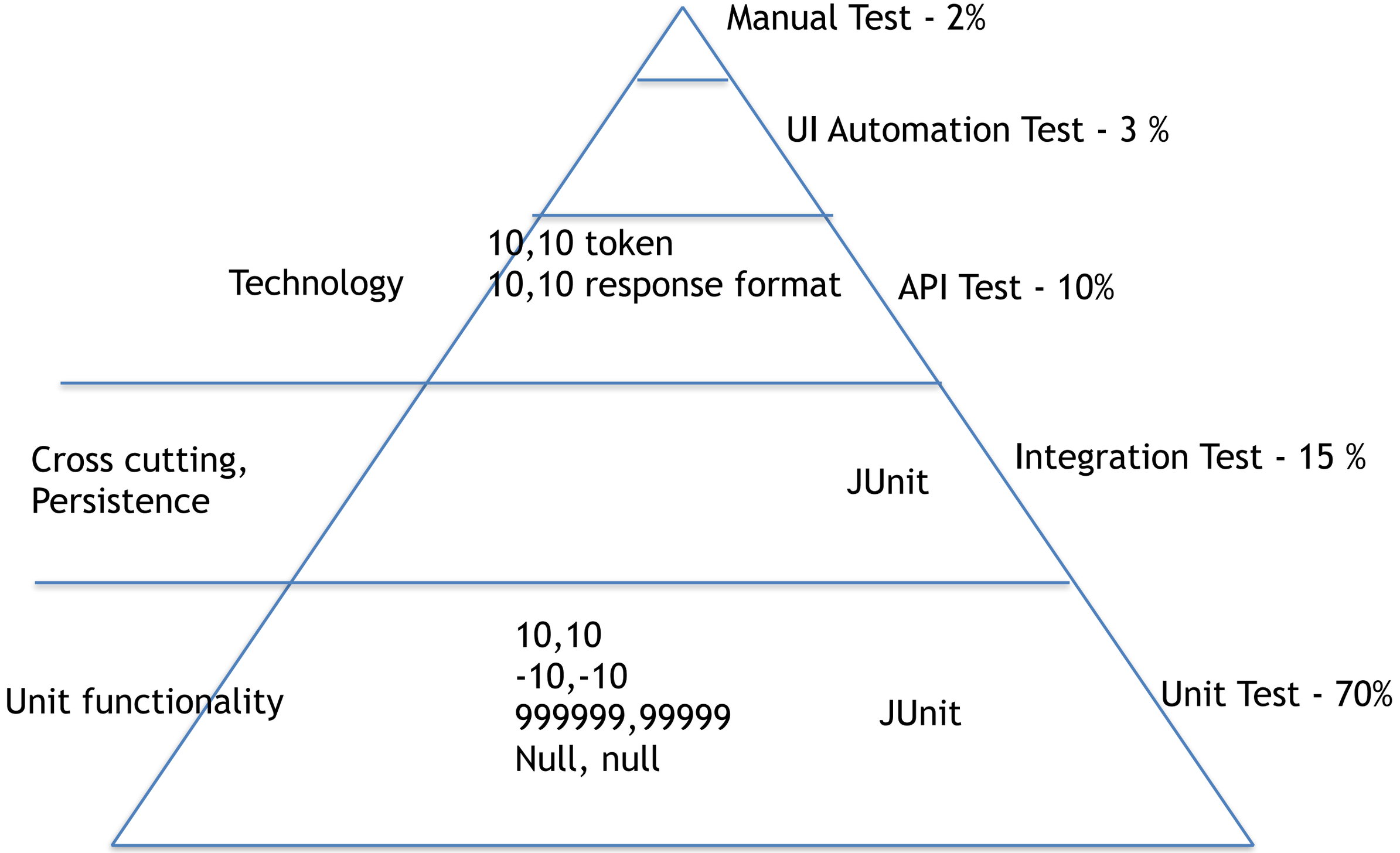

Manual Test
UI Automation Test
API Test
Integration Test
Unit Test

Cup Cake Anti Pattern




Manual Test
UI Automation Test
API Test
Integration Test
Unit Test


UI
Service
Unit
$$$
¢


Manual Test - 2%
UI Automation Test - 3 %
10,10 token
10,10 response format
Technology
API Test - 10%
Cross cutting,
Persistence
JUnit
Integration Test - 15 %
10,10
-10,-10
999999,99999
Null, null
Unit functionality
JUnit
Unit Test - 70%

Manual Test

UI view (get/set/event handling) — UI Automation Test

UI controller (logic heavy) — Unit test

web **Service Layer (no logic, delegates) - 3~5 lines per method** — API Test

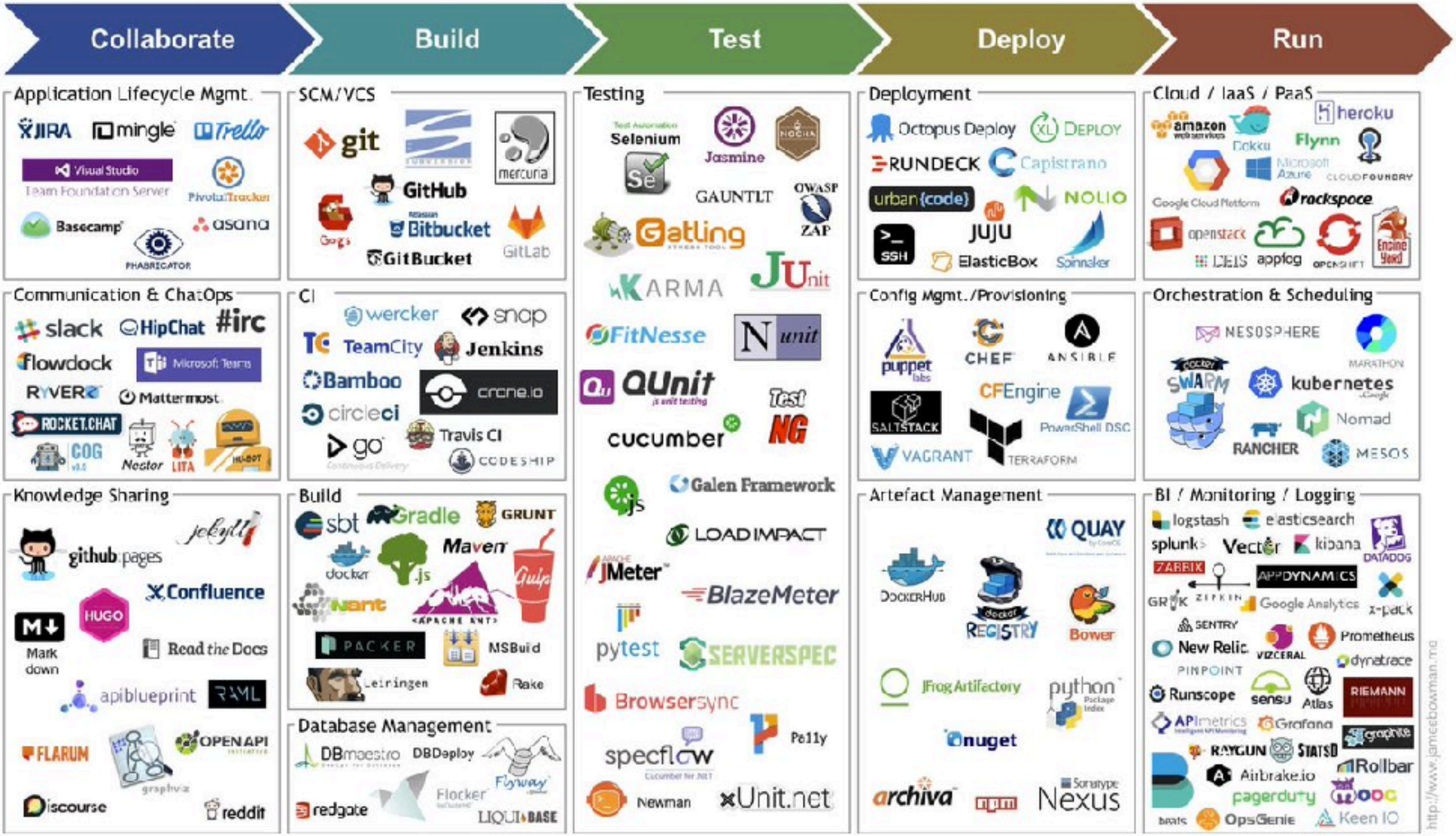
facade Layer (flow) — Integrate test

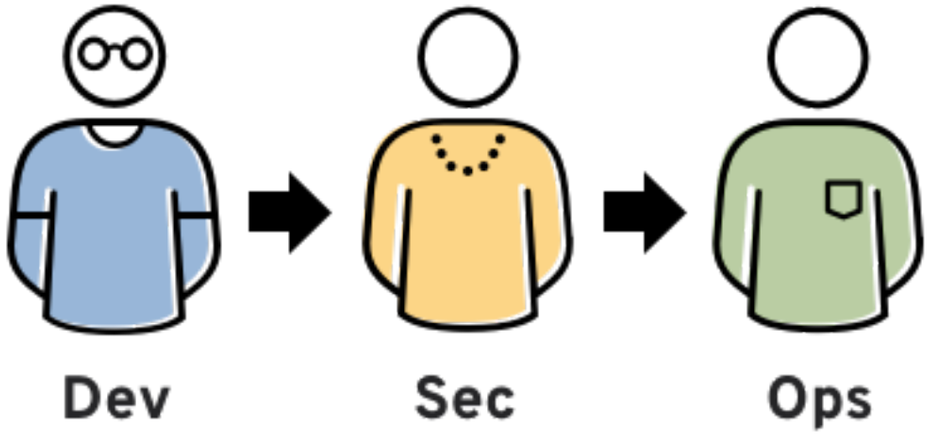
domain Layer (steps) (logic heavy) — Unit test

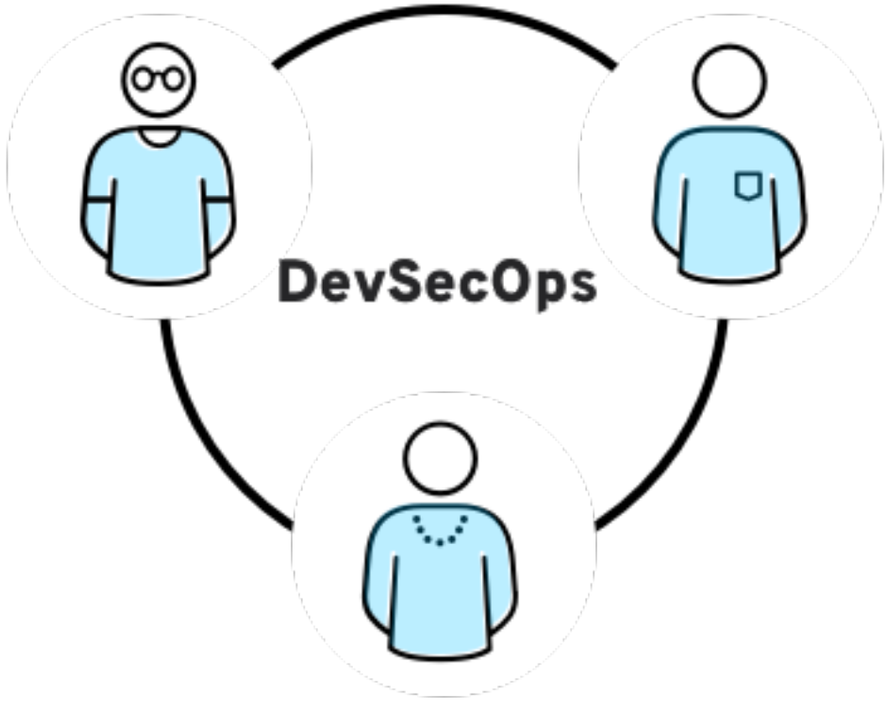
persistence **Layer (CRUD) - Cookie cutter** — Integration Test

Stored procs – logic on large volume of data? — Integration Test

Dto(get/set)

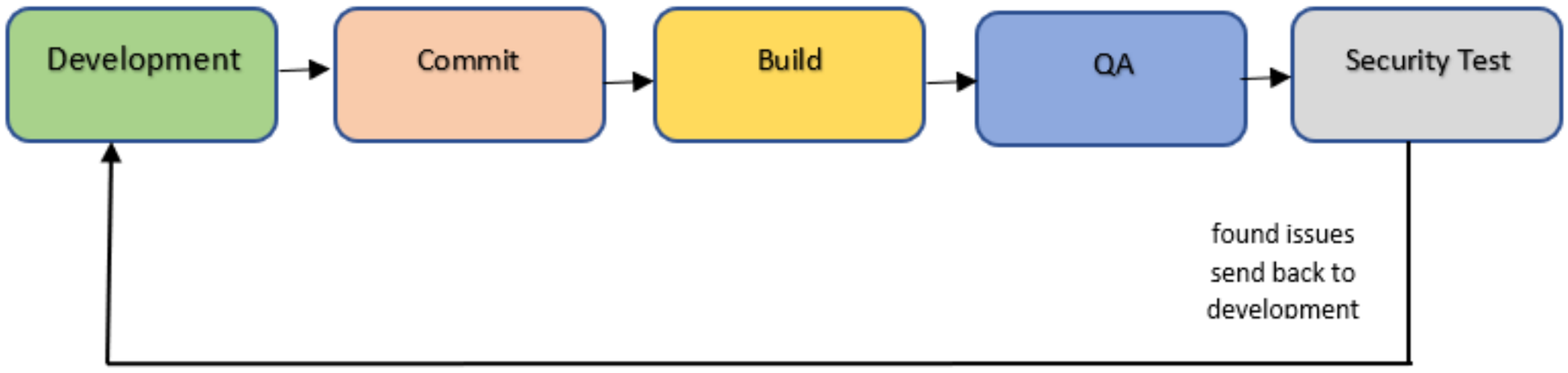

Collaborate
Build
Test
Deploy
Run
Application Lifecycle Mgmt.
JIRA
mingle
Trello
Visual Studio
Team Foundation Server
PivotalTracker
Basecamp
asana
PHABRICATOR
SCM/VCS
git
subversion
mercurial
GitHub
Gogs
Bitbucket
GitBucket
GitLab
Testing
Test Automation
Selenium
Jasmine
NOCHA
GAUNTLT
OWASP
ZAP
Gatling
JUnit
KARMA
FitNesse
N unit
QUnit
js unit testing
cucumber
Test
NG
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
Pa11y
specflow
Cucumber for .NET
Newman
xUnit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
SSH
JUJU
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
Deis
heroku
Flynn
Microsoft Azure
CLOUDFOUNDRY
Google Cloud Platform
rackspace
openstack
DEIS
appfog
OPENSHIFT
Engine Yard
Communication & ChatOps
slack
HipChat
#irc
flowdock
Microsoft Teams
RYVER
Mattermost
ROCKET.CHAT
COG
Nectar
LITA
HUBOT
CI
wercker
snap
TeamCity
Jenkins
Bamboo
circleci
cruise.io
go
Travis CI
CODESHIP
Config Mgmt./Provisioning
puppet labs
CHEF
ANSIBLE
CFEngine
SALTSTACK
PowerShell DSC
VAGRANT
TERRAFORM
Orchestration & Scheduling
MESOSPHERE
MARATHON
SWARM
kubernetes
Nomad
RANCHER
MESOS
Knowledge Sharing
github pages
jekyll
Confluence
HUGO
Mark down
Read the Docs
apiblueprint
RAML
FLARUM
OPEN API
graphviz
Discourse
reddit
Build
sbt
Gradle
GRUNT
docker
Maven
.js
Gulp
nant
<APACHE ANT>
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE
Galen Framework
LOAD IMPACT
BlazeMeter
SERVERSPEC
Artefact Management
QUAY
DockerHub
docker REGISTRY
Bower
JFrog Artifactory
python Package Index
nuget
archiva
npm
Sonatype Nexus
BI / Monitoring / Logging
logstash
elasticsearch
splunk
Vector
kibana
DATADOG
ZABBIX
APPDYNAMICS
GROK
ZIPKIN
Google Analytics
x-pack
SENTRY
New Relic
VIZCERAL
Prometheus
dynatrace
PINPOINT
Runscope
sensu
Atlas
RIEMANN
APImetrics
Grafana
graphite
RAYGUN
STATSD
Rollbar
Airbrake.io
pagerduty
OOOC
beats
OpsGenie
Keen IO
http://www.jamescbowman.net



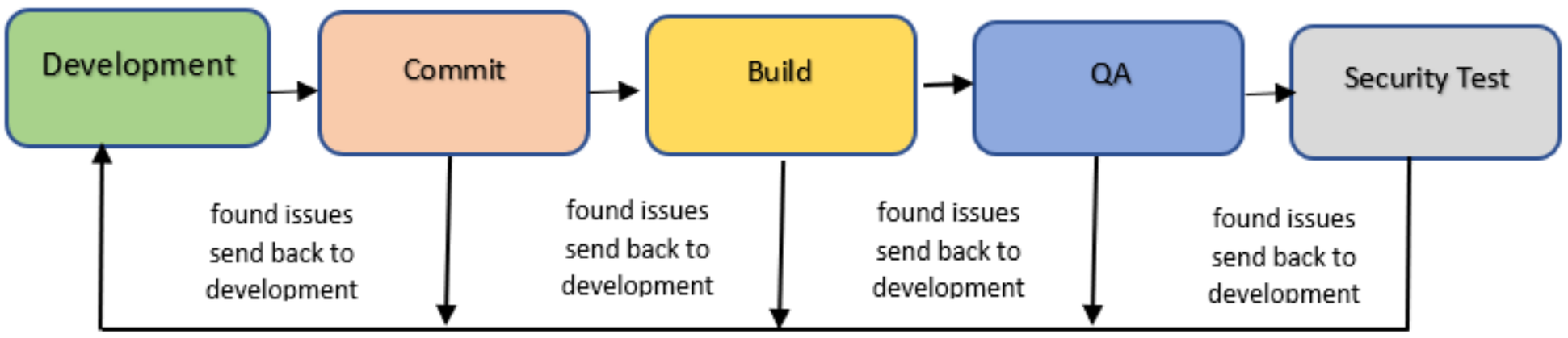


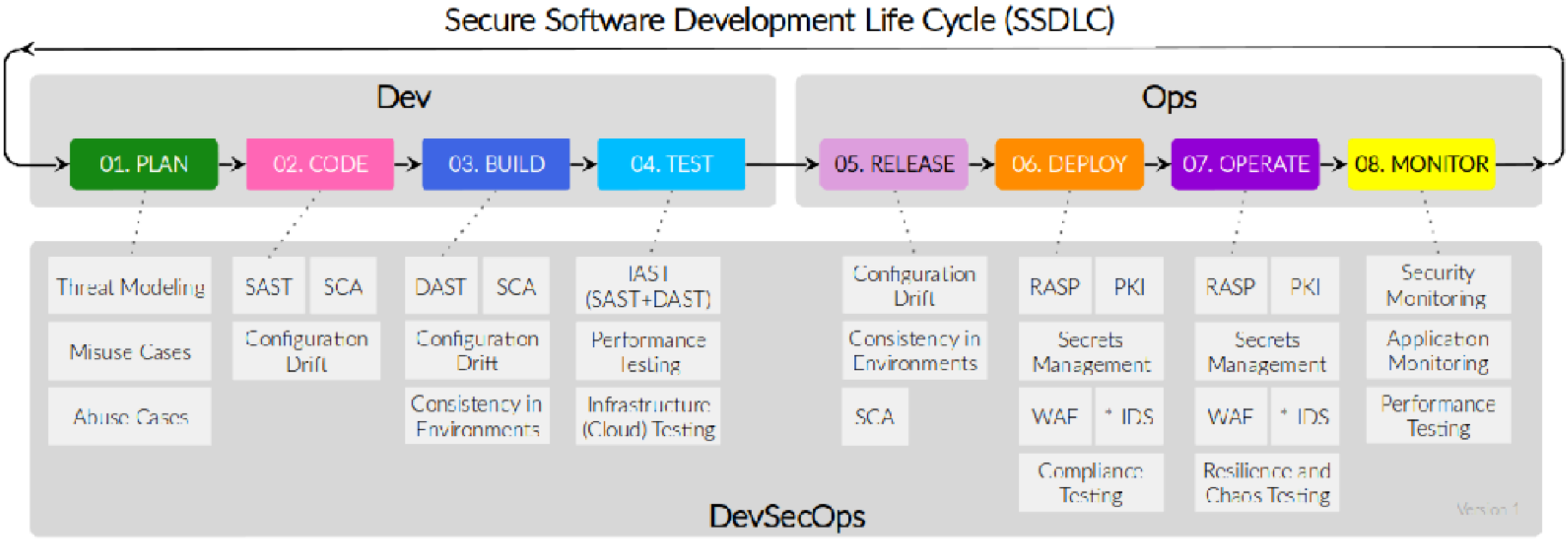
DevSecOps is about injecting security into the DevOps lifecycle.

"Security in traditional way"



"Security in DevSecOps way"




Secure Software Development Life Cycle (SSDLC)
Dev
Ops
01. PLAN
02. CODE
03. BUILD
04. TEST
05. RELEASE
06. DEPLOY
07. OPERATE
08. MONITOR
Threat Modeling
Misuse Cases
Abuse Cases
SAST
SCA
Configuration Drift
DAST
SCA
Configuration Drift
Consistency in Environments
IAST (SAST+DAST)
Performance Testing
Infrastructure (Cloud) Testing
Configuration Drift
Consistency in Environments
SCA
RASP
PKI
Secrets Management
WAF
* IDS
Compliance Testing
RASP
PKI
Secrets Management
WAF
* IDS
Resilience and Chaos Testing
Security Monitoring
Application Monitoring
Performance Testing
DevSecOps

DAST vs. SAST

Vulnerability Coverage

SAST

+ Null Pointer Dereference
+ Threading Issues
+ Code Quality Issues
+ Insecure Crypto Issues
+ Issues in Non Web application Code

- Higher number of FP
- Run time Code generation
- Dynamic Languages (Ruby + Python)

+ SQL Injection
+ Cross Site Scripting (XSS)
+ OS Commanding
+ HTTP Response Splitting
+ LDAP Injection
+ XPATH Injection
+ Path Traversal
+ Buffer Overflows
+ Format String Issues

DAST

+ Runtime Privilege Issues
+ Authentication Issues
+ Session Management Issues
+ Insecure 3rd Party Libraries
+ Business Logic Vulnerabilities
+ Protocol Parser Issues

- Web2.0, JSON, Flash, HTML 5.0,
- Integrity and Availability violations
- Long Execution Times

SAST vs. DAST

Static application security testing (SAST) and dynamic application security testing (DAST) are both methods of testing for security vulnerabilities, but they're used very differently.

Here are some key differences between the two:

White box security testing

- The tester has access to the underlying framework, design, and implementation.
- The application is tested from the inside out.
- This type of testing represents the developer approach.

Black box security testing

- The tester has no knowledge of the technologies or frameworks that the application is built on.
- The application is tested from the outside in.
- This type of testing represents the hacker approach.

Requires source code

- SAST doesn't require a deployed application.
- It analyzes the source code or binary without executing the application.

Requires a running application

- DAST doesn't require source code or binaries.
- It analyzes by executing the application.

Finds vulnerabilities earlier in the SDLC

- The scan can be executed as soon as code is deemed feature-complete.

Finds vulnerabilities toward the end of the SDLC

- Vulnerabilities can be discovered after the development cycle is complete.

Less expensive to fix vulnerabilities

- Since vulnerabilities are found earlier in the SDLC, it's easier and faster to remediate them.
- Findings can often be fixed before the code enters the QA cycle.

More expensive to fix vulnerabilities

- Since vulnerabilities are found toward the end of the SDLC, remediation often gets pushed into the next cycle.
- Critical vulnerabilities may be fixed as an emergency release.

Can't discover run-time and environment-related issues

- Since the tool scans static code, it can't discover run-time vulnerabilities.

Can discover run-time and environment-related issues

- Since the tool uses dynamic analysis on an application, it is able to find run-time vulnerabilities.

Typically supports all kinds of software

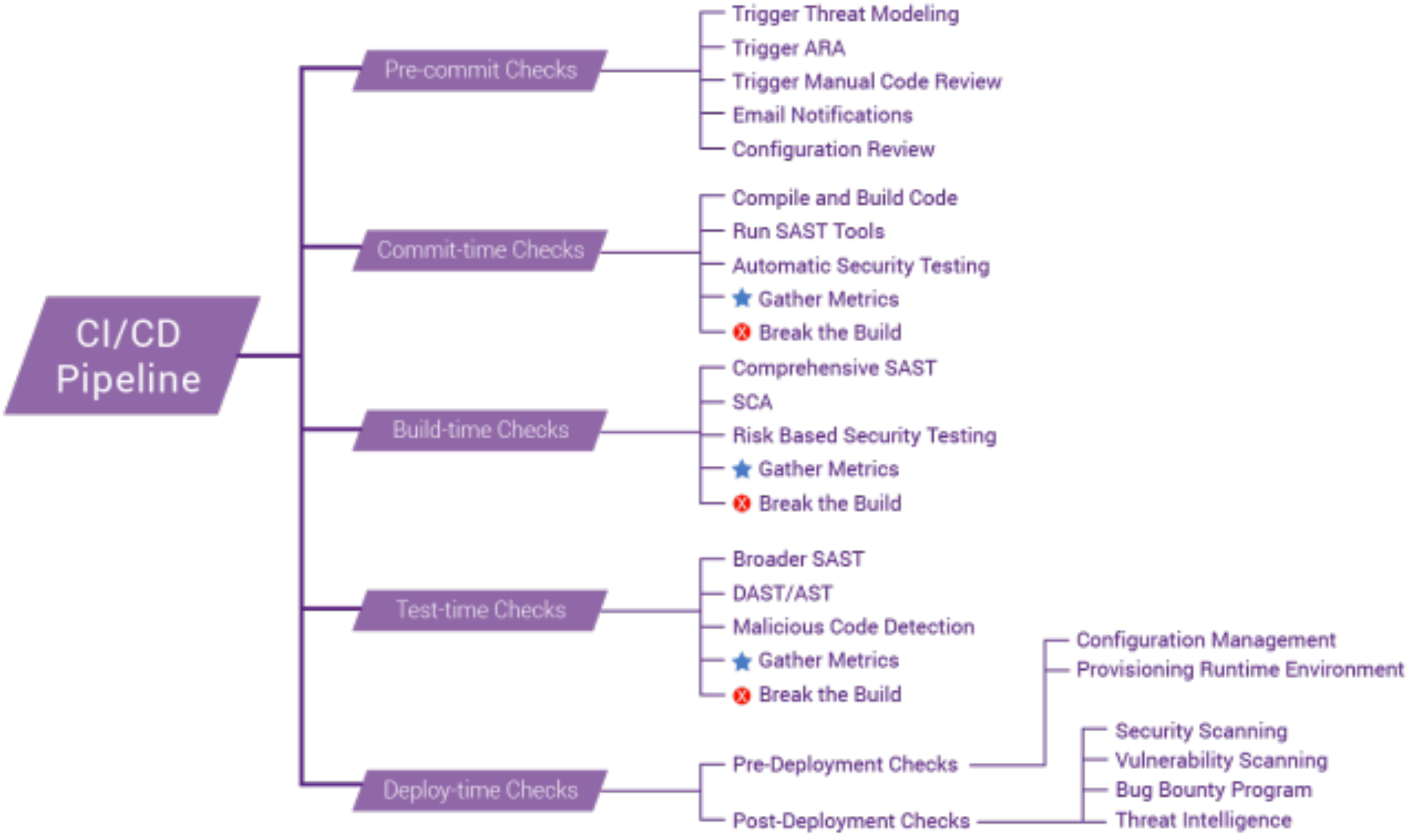
- Examples include web applications, web services, and thick clients.

Typically scans only apps like web applications and web services

- DAST is not useful for other types of software.

SAST and DAST techniques complement each other.

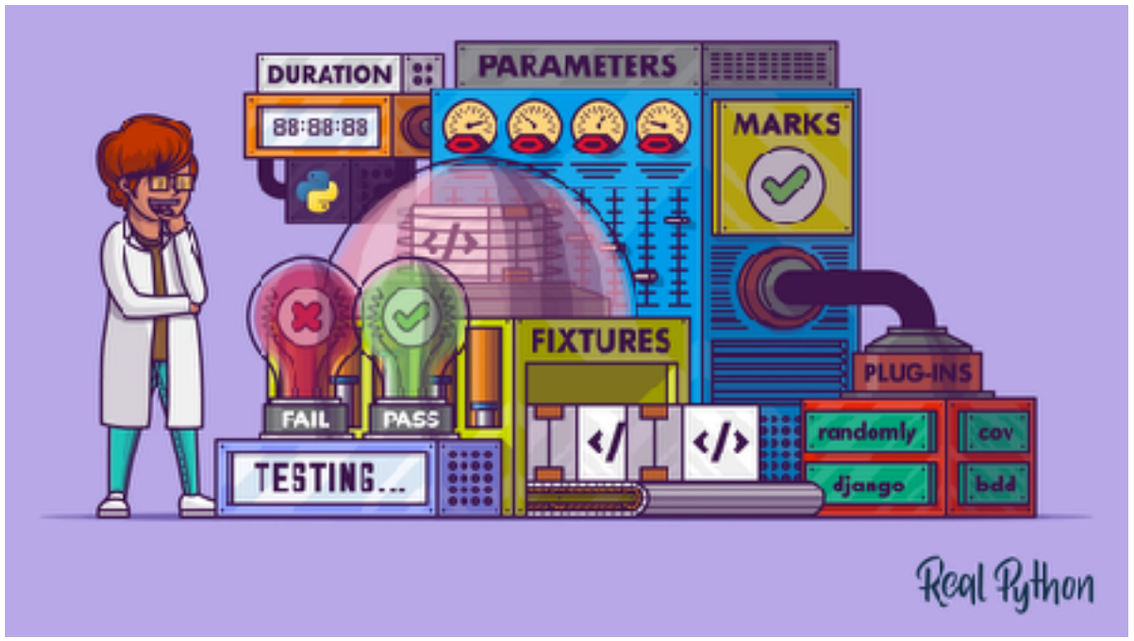
Both need to be carried out for comprehensive testing.


CI/CD Pipeline
Pre-commit Checks
Trigger Threat Modeling
Trigger ARA
Trigger Manual Code Review
Email Notifications
Configuration Review
Commit-time Checks
Compile and Build Code
Run SAST Tools
Automatic Security Testing
Gather Metrics
Break the Build
Build-time Checks
Comprehensive SAST
SCA
Risk Based Security Testing
Gather Metrics
Break the Build
Test-time Checks
Broader SAST
DAST/AST
Malicious Code Detection
Gather Metrics
Break the Build
Deploy-time Checks
Pre-Deployment Checks
Configuration Management
Provisioning Runtime Environment
Post-Deployment Checks
Security Scanning
Vulnerability Scanning
Bug Bounty Program
Threat Intelligence

IAST and RASP is designed to address the shortcomings of SAST and DAST by combining elements of both approaches. **RASP** leverages the same technique as **IAST** by installing an agent within the application

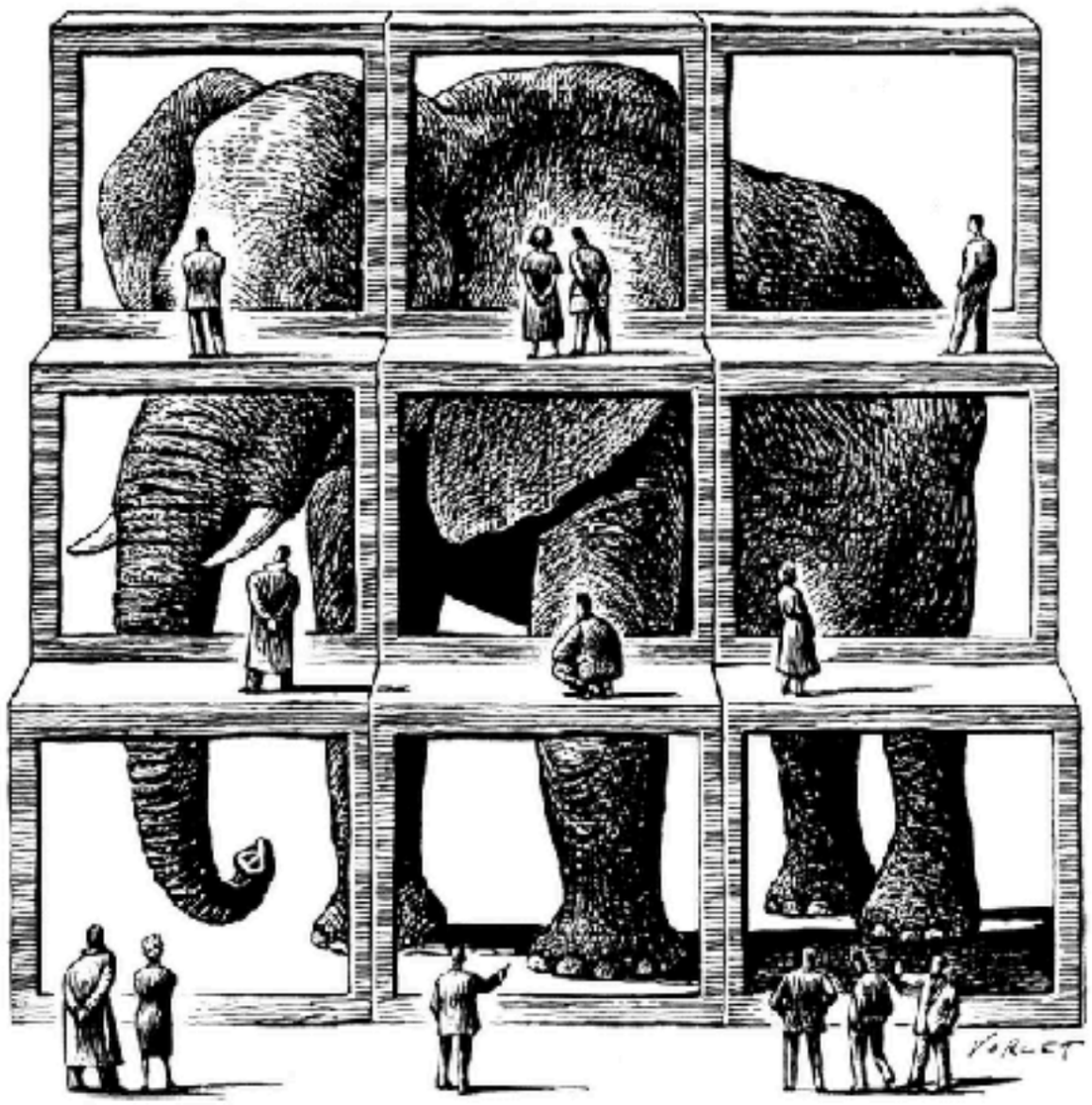
RASP and IAST works by inserting sensors in the existing application code to monitor and control certain critical execution points, in real-time. By means of those techniques, RASP and IAST become part of the system so your applications remain protected wherever they go.

IAST: Interactive application security testing. Monitoring an application for security vulnerabilities while it is running — at testing time.

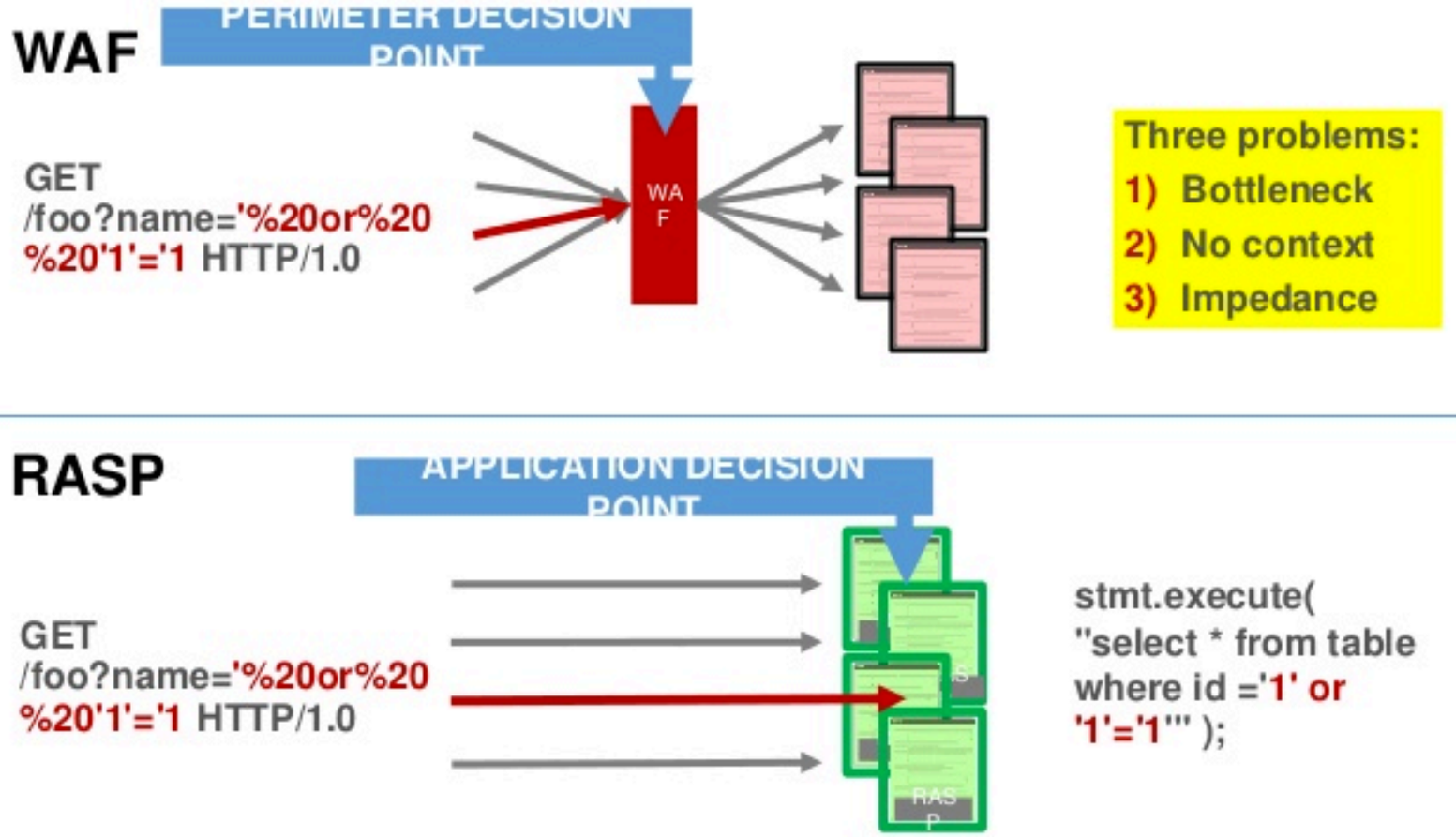


RASP: Runtime application self protection. Monitoring an application to detect attacks while it is running — at production time.

IAST tools look for security vulnerabilities, whereas **RASP** monitors the application for attacks, **and** protects the application against them when it senses an attack happening.



Because the IAST and RASP agent is working inside the app, it can apply its analysis to the entire app all its code; its runtime control and data flow information; its configuration information; HTTP requests and responses; libraries, frameworks and other components; and backend connection information. Access to all that information allows the IAST and RASP engine to cover more code, produce more accurate results and verify a broader range of security rules than either SAST or DAST



A **WAF** is not aware of the true weaknesses of the application, so it must validate all input before it reaches the application itself. Similarly, a WAF is unable to see the consequences of a payload. For instance, a very dangerous consequence of an SQLi payload would be to have two SQL statements, as opposed to one. To circumvent this **lack of context**, some WAFs implement machine learning systems to detect anomalies in the traffic that might indicate attacks. They require a training process so that legitimate traffic can be identified. All this introduces delays and increases the chances of accidentally blocking legitimate traffic, which damages the user experience.

By means of those techniques, RASP gets **full visibility of the internal architecture** details of the applications and **full visibility of the execution flow during runtime**. This means that a RASP can make very smart decisions about what is an attack and what is not.

DevSecOps Pipeline Techstack

Legend: DevOps DevSecOps

INOVO | VENTURE PARTNERS

CI/CD Pipeline

Commit code to repository

Build

Deploy

IDE Plugins

JFrog, snyk, Skim

Pre-Commit Hooks

Talisman, git-secrets, HOUND

Secrets Management

HashiCorp Vault, Secret Manager, AWS Secrets Manager

SAST

Checkmarx, DerScanner, ECG

SCA

GitHub, BLACKDUCK, snyk

DAST

RAPID7, acunetix, CONTRAST SECURITY

IAST

CONTRAST SECURITY, SYNOPSYS, VERACODE

Secure Infrastructure as code

clair, anchore, DOCKSCAN

Compliance as code

KitchenCI, SERVERSPEC, CHEF INSPEC

RASP

Signal Sciences, TREND MICRO, imperva

Web Application Firewall

imperva, CLOUDFLARE, wallarm

Monitoring Tools

DATADOG, Nagios, PerfOps

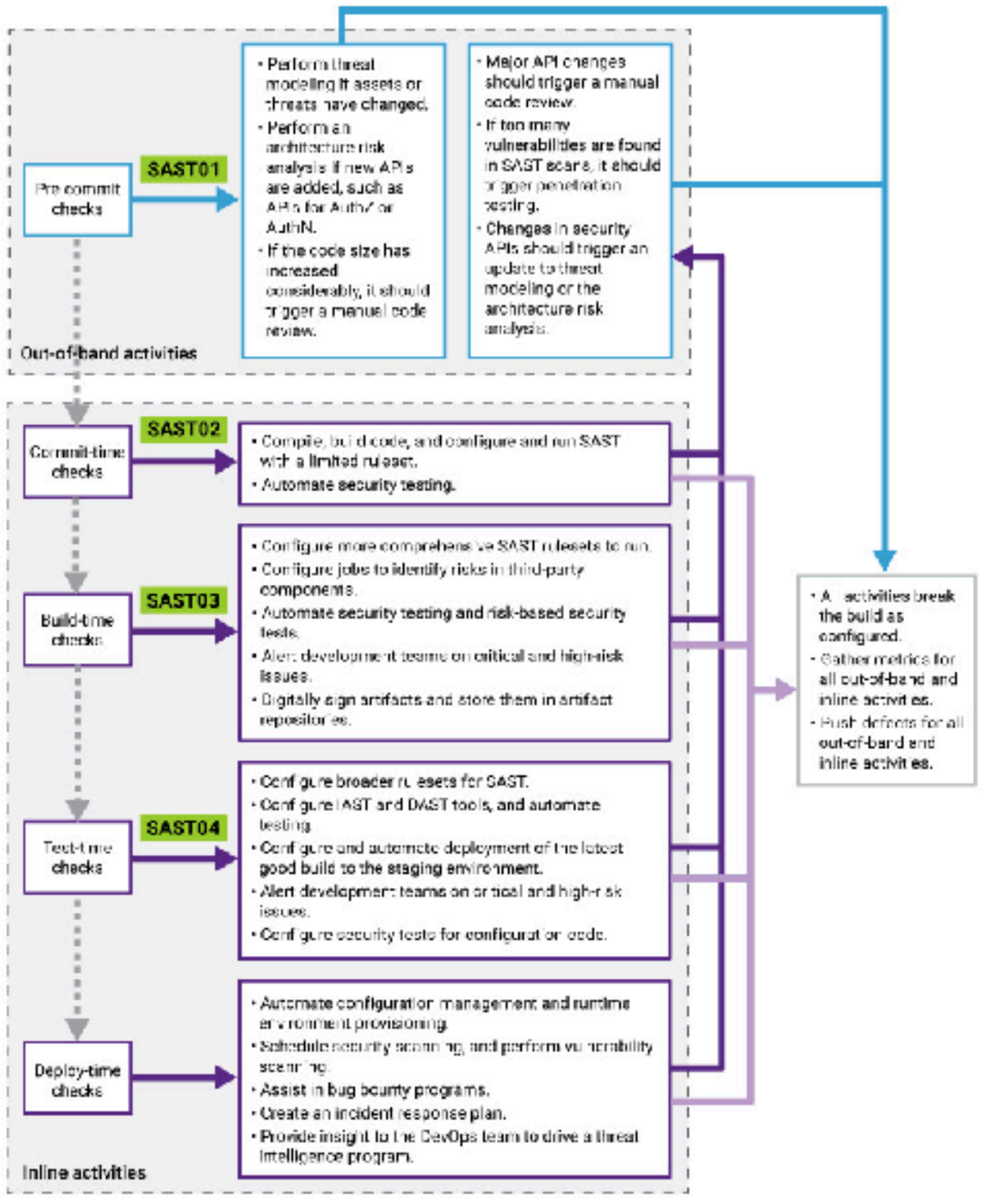
Chaos Engineering

Gremlin, ChaosToolkit

Vulnerability management

Hacker, ARCHERY

SAST tool integration in DevSecOps pipeline



SAST01

The SAST tool runs in the IDE as developers write code. The tool is configured to detect vulnerabilities that have zero false positives, including issues such as SQL injection and XSS. The scan should take seconds.

SAST02

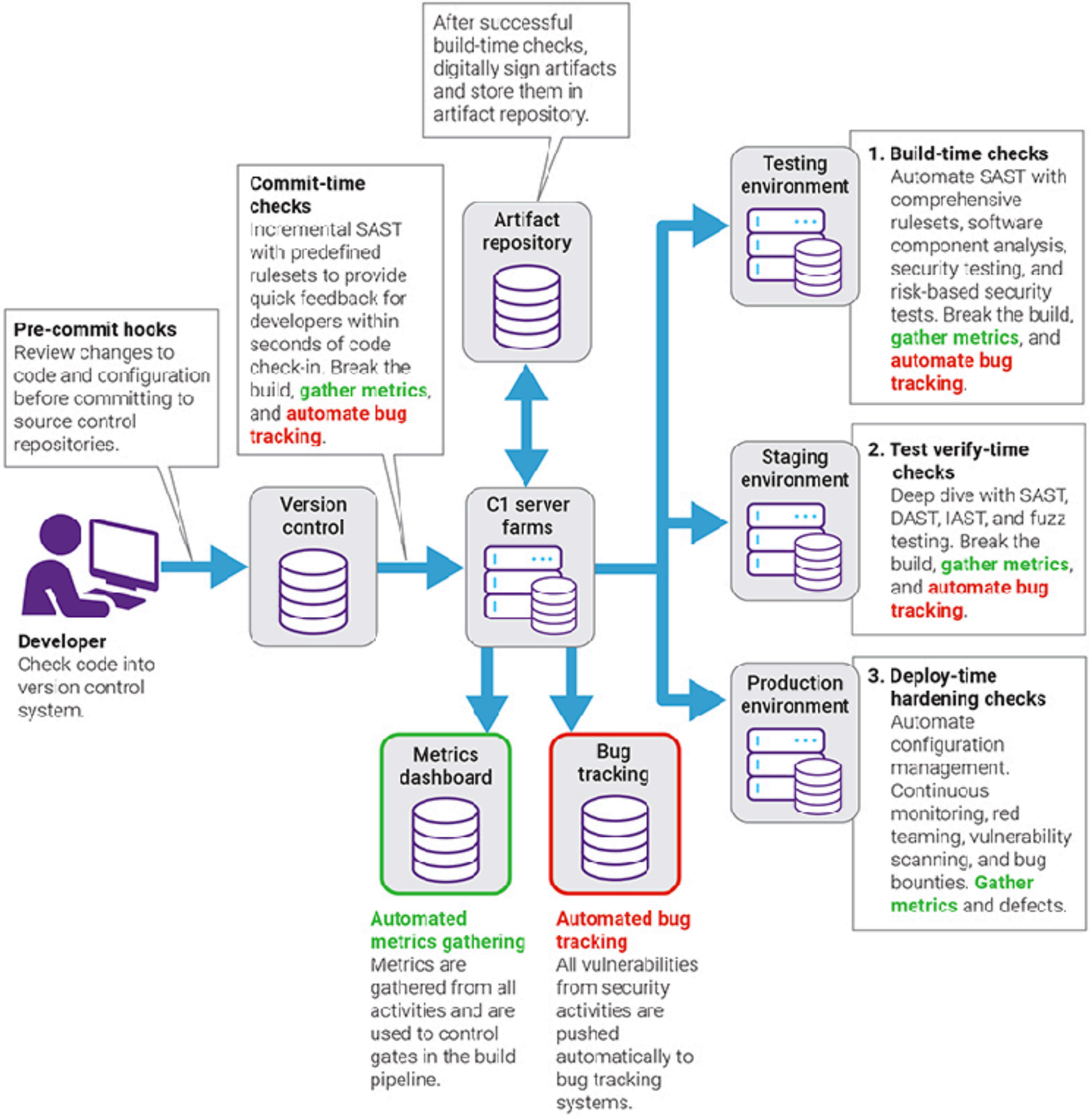
The SAST tool is automated on the CI server. The tool is configured for the client's top 10 issues, such as command injection and hard-coded keys. The tool also uses rules from SAST01. The scan should take 4–5 minutes so developers get feedback fast.

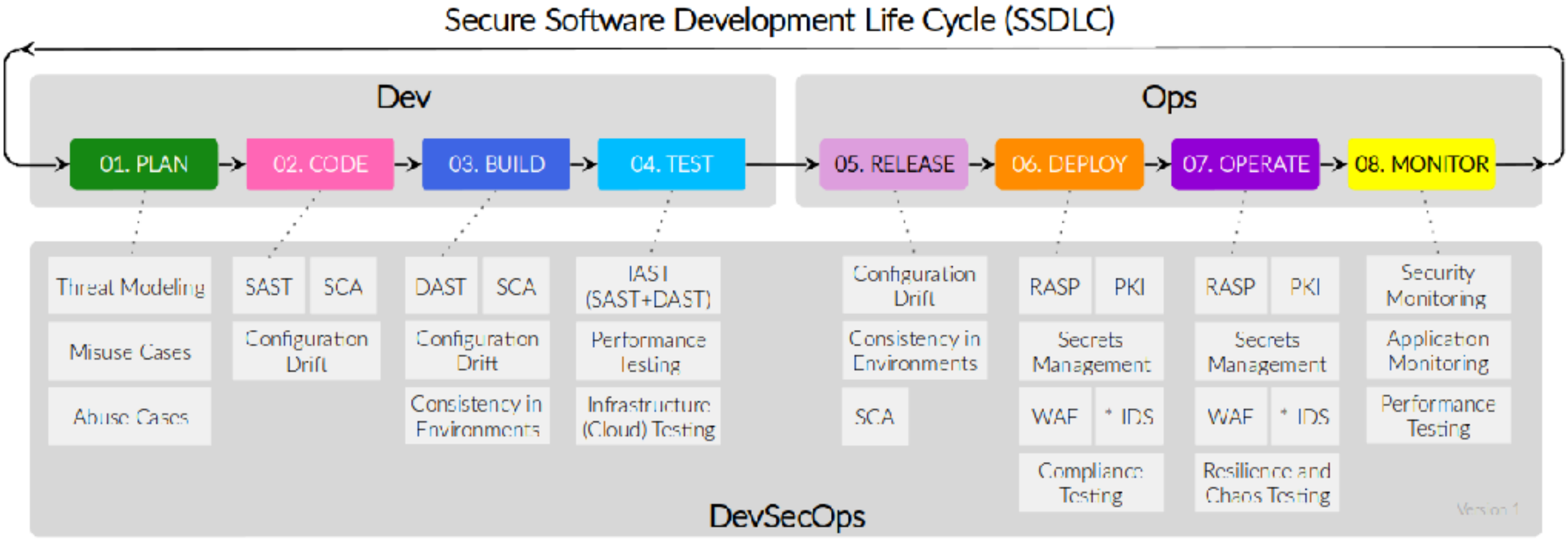
SAST03

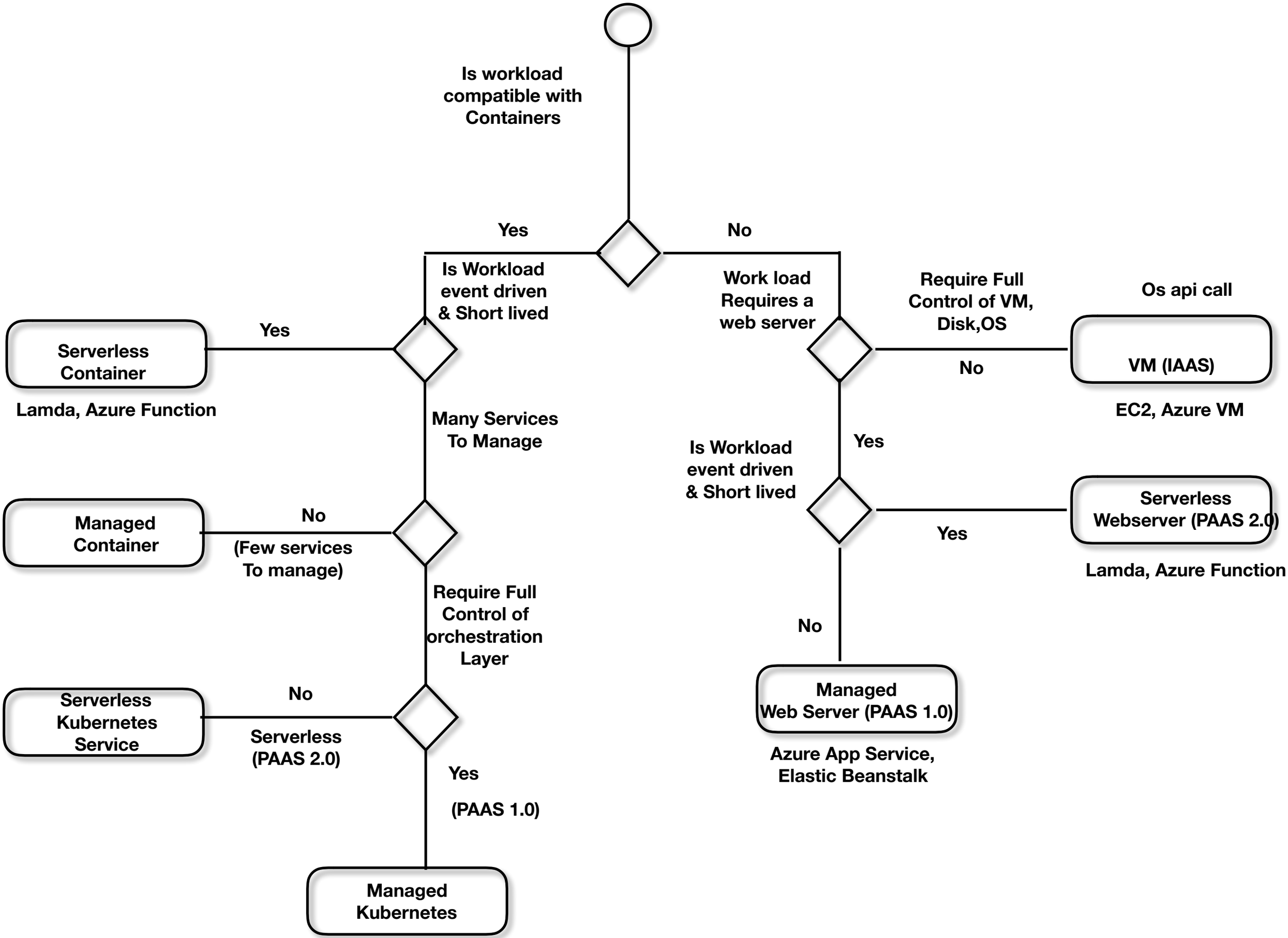
The SAST tool uses rules for the OWASP Top 10 and any customized rulesets written for client-specific APIs. The scan can be run in parallel with other activities and should take 10–15 minutes.

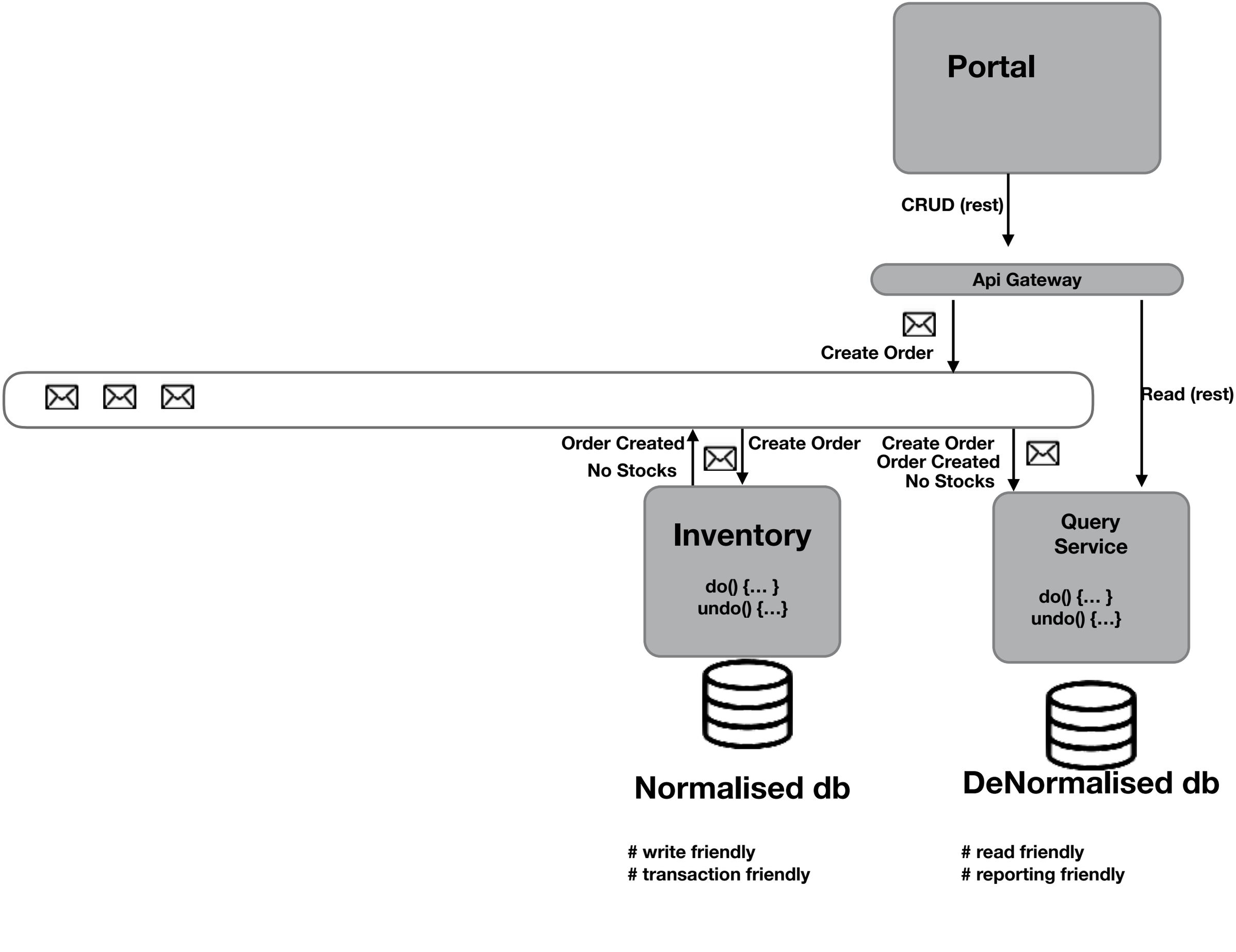
SAST04

The SAST tool uses comprehensive rulesets. All previous rulesets are excluded. The goal is to find issues before the code goes to production. The scan should take anywhere from an hour to 3–4 hours, depending on production velocity.

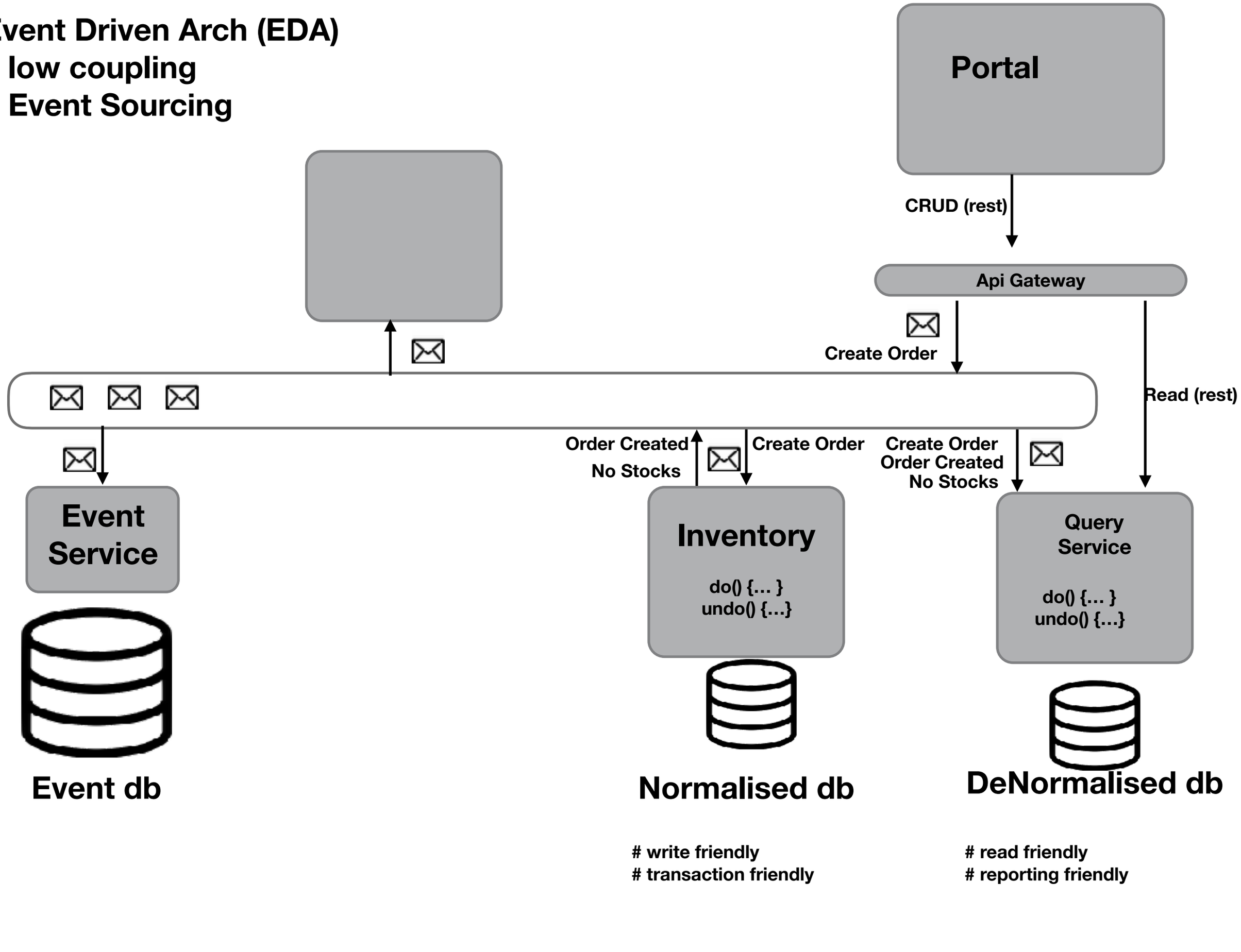

After successful build-time checks, digitally sign artifacts and store them in artifact repository.
Commit-time checks
Incremental SAST with predefined rulesets to provide quick feedback for developers within seconds of code check-in. Break the build, gather metrics, and automate bug tracking.
Pre-commit hocks
Review changes to code and configuration before committing to source control repositories.
Artifact repository
Version control
C1 server farms
Testing environment
1. Build-time checks
Automate SAST with comprehensive rulesets, software component analysis, security testing, and risk-based security tests. Break the build, gather metrics, and automate bug tracking.
Staging environment
2. Test verify-time checks
Deep dive with SAST, DAST, IAST, and fuzz testing. Break the build, gather metrics, and automate bug tracking.
Production environment
3. Deploy-time hardening checks
Automate configuration management. Continuous monitoring, red teaming, vulnerability scanning, and bug bounties. Gather metrics and defects.
Developer
Check code into version control system.
Metrics dashboard
Bug tracking
Automated metrics gathering
Metrics are gathered from all activities and are used to control gates in the build pipeline.
Automated bug tracking
All vulnerabilities from security activities are pushed automatically to bug tracking systems.


Secure Software Development Life Cycle (SSDLC)
Dev
Ops
01. PLAN
02. CODE
03. BUILD
04. TEST
05. RELEASE
06. DEPLOY
07. OPERATE
08. MONITOR
Threat Modeling
Misuse Cases
Abuse Cases
SAST
SCA
Configuration Drift
DAST
SCA
Configuration Drift
Consistency in Environments
IAST (SAST+DAST)
Performance Testing
Infrastructure (Cloud) Testing
Configuration Drift
Consistency in Environments
SCA
RASP
PKI
Secrets Management
WAF
* IDS
Compliance Testing
RASP
PKI
Secrets Management
WAF
* IDS
Resilience and Chaos Testing
Security Monitoring
Application Monitoring
Performance Testing
DevSecOps


Is workload compatible with Containers
Yes
No
Is Workload event driven & Short lived
Yes
Serverless Container
Lamda, Azure Function
Many Services To Manage
No
Managed Container
(Few services To manage)
Require Full Control of orchestration Layer
No
Serverless Kubernetes Service
Serverless (PAAS 2.0)
Yes
(PAAS 1.0)
Managed Kubernetes
Work load Requires a web server
Require Full Control of VM, Disk,OS
Os api call
No
VM (IAAS)
EC2, Azure VM
Yes
Is Workload event driven & Short lived
Yes
Serverless Webserver (PAAS 2.0)
Lamda, Azure Function
No
Managed Web Server (PAAS 1.0)
Azure App Service, Elastic Beanstalk


Portal
CRUD (rest)
Api Gateway
Create Order
Read (rest)
Order Created
No Stocks
Create Order
Create Order
Order Created
No Stocks
Inventory
do() {... }
undo() {...}
Query
Service
do() {... }
undo() {...}
Normalised db
DeNormalised db
write friendly
transaction friendly
read friendly
reporting friendly

Event Driven Arch (EDA)
low coupling
Event Sourcing



write friendly
transaction friendly

read friendly
reporting friendly

Portal

CRUD (rest)

Stream Analytics

Api Gateway

Create Order

Read (rest)

Order Created
No Stocks

Create Order

Create Order
Order Created
No Stocks

Event Service

Inventory

do() {... }
undo() {...}

Query Service

do() {... }
undo() {...}

Event db

full state

Big data Analytics

Archive

Data ware House

Snap Shots
(daily/ weekly / monthly)

WareHouse Analytics

Normalised db

write friendly
transaction friendly
current state

DeNormalised db

read friendly
reporting friendly

Data Source Tier

Ingestion Tier

Operation Tier

Insight/Analytics Tier

Structured

unstructured

semi-structured

Real time

Azure Event Hub
AWS Kinesis
GCP Cloud pub/sub
Kafka

Batch

One Time

AWS Snowball,
Azure Data Box
Google Transfer Appliance

Azure Stream Analytics
Azure Kinesis Analytics
Storm
Spark Streaming

Machine Learning

Stream Processing

runs on the incoming data

AWS Elastic MapReduce (EMR)
AZURE HDInsight

Batch Processing

runs on the entire dataset

Storage Tier

Landing Zone

Raw

handling missing values,
filtering outliers,
& Validated data

Staging

Data Lake
AWS S3
Azure Data Lake

Highly refined views of the data

Curated

Analytical data store
Aws Redshift
Azure Synapse Analytics

BI Tools

Applications

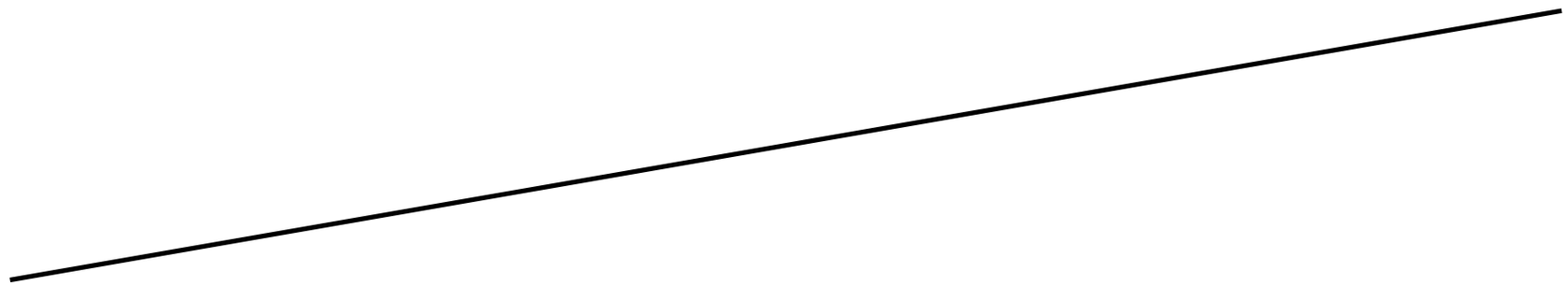
Automations

QuickSight
Power BI

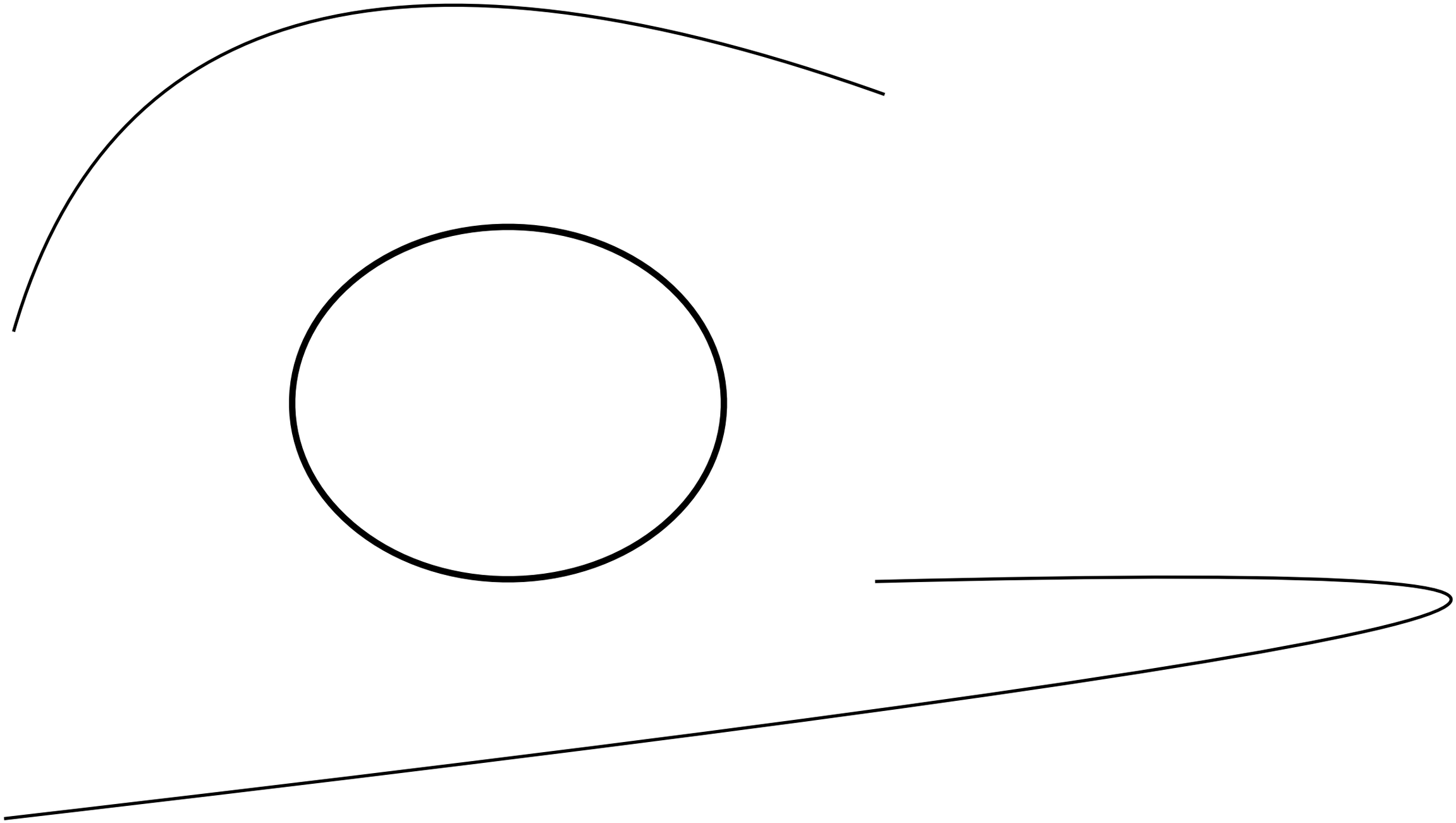
Orchestration Tier

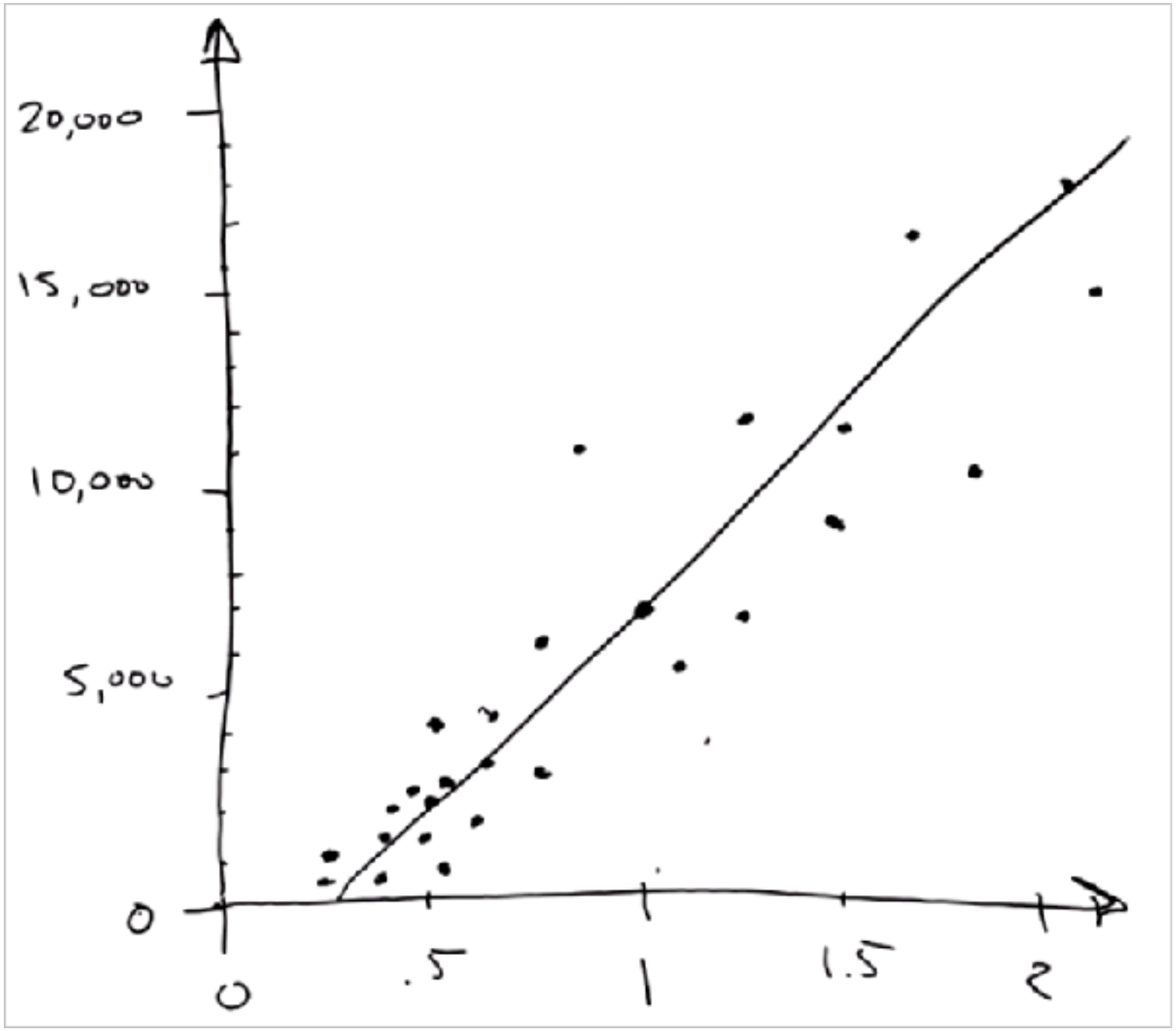
Workflow

AWS Data Pipeline
Azure Data Factory
Apache Oozie and Sqoop



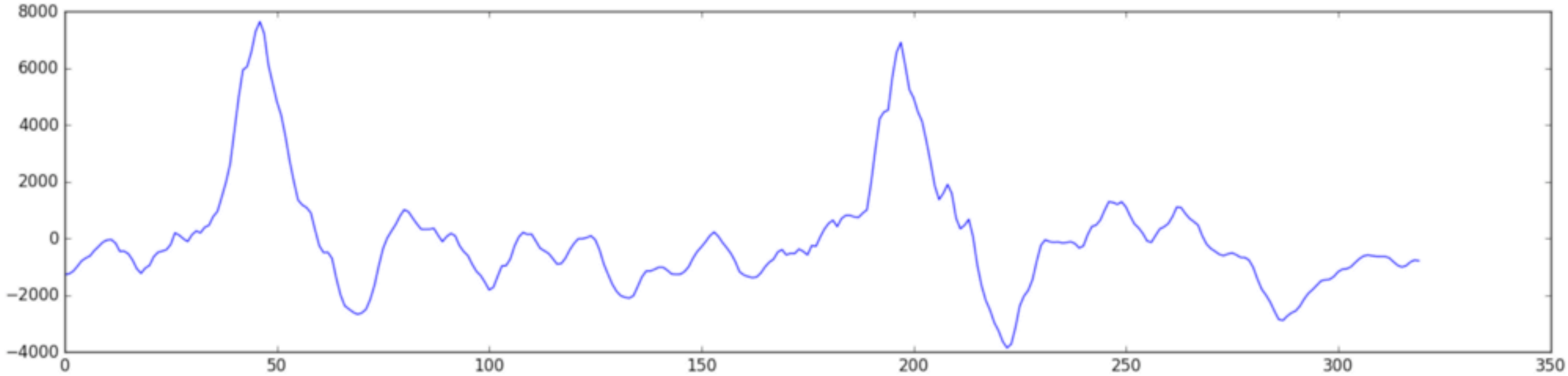
The vast majority of machine learning are concerned with just drawing lines.



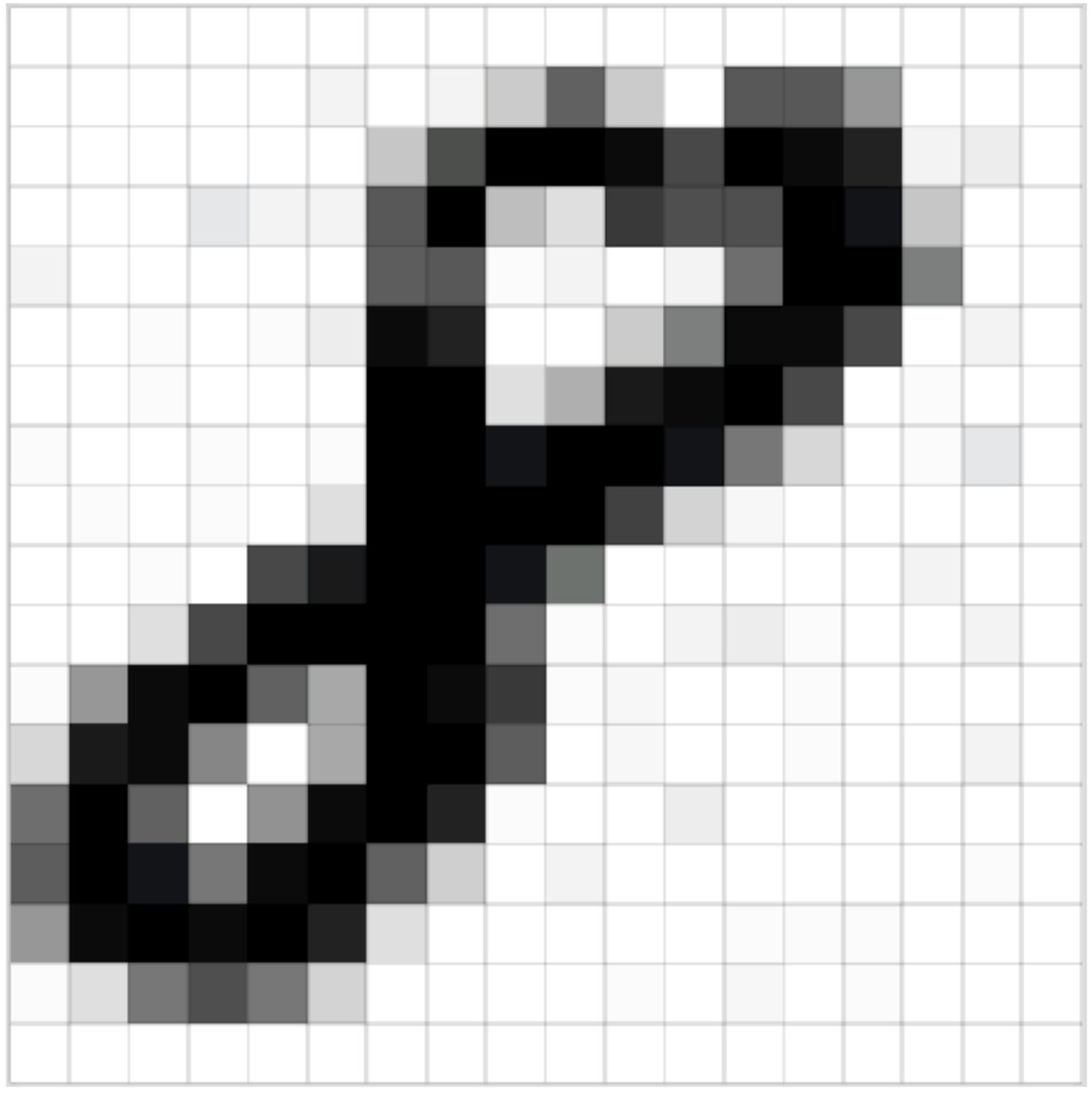


In particular, machine learning is all about drawing lines through data.

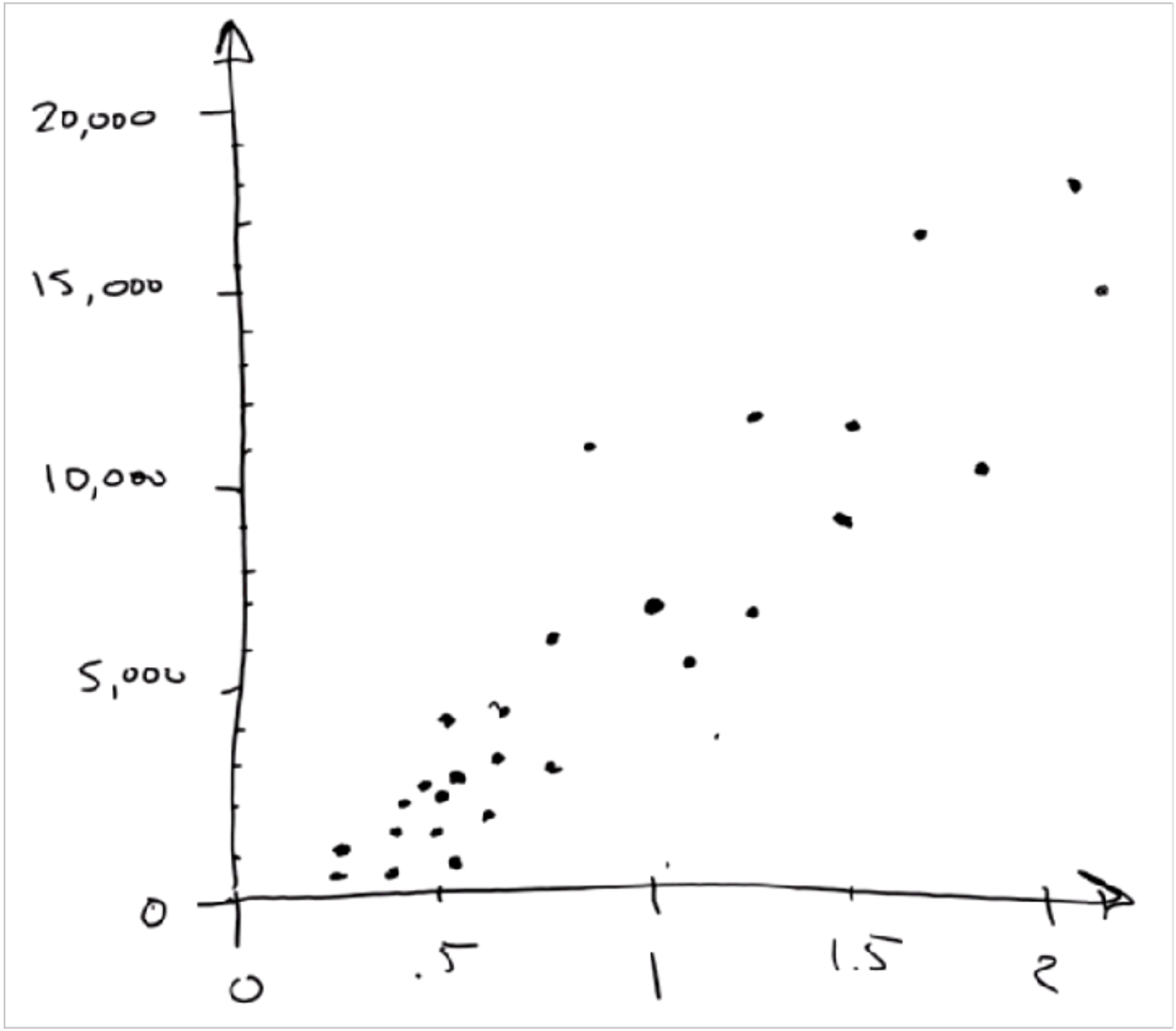
Carats	Price
1.01	7,366
.49	985
.31	544
1.51	9,140
.37	493
.73	3,011
1.53	11,413
.56	1,814
.41	876
.74	2690
.63	1,190
.6	4,172
2.06	11,764
1.1	4,682
1.31	6,171

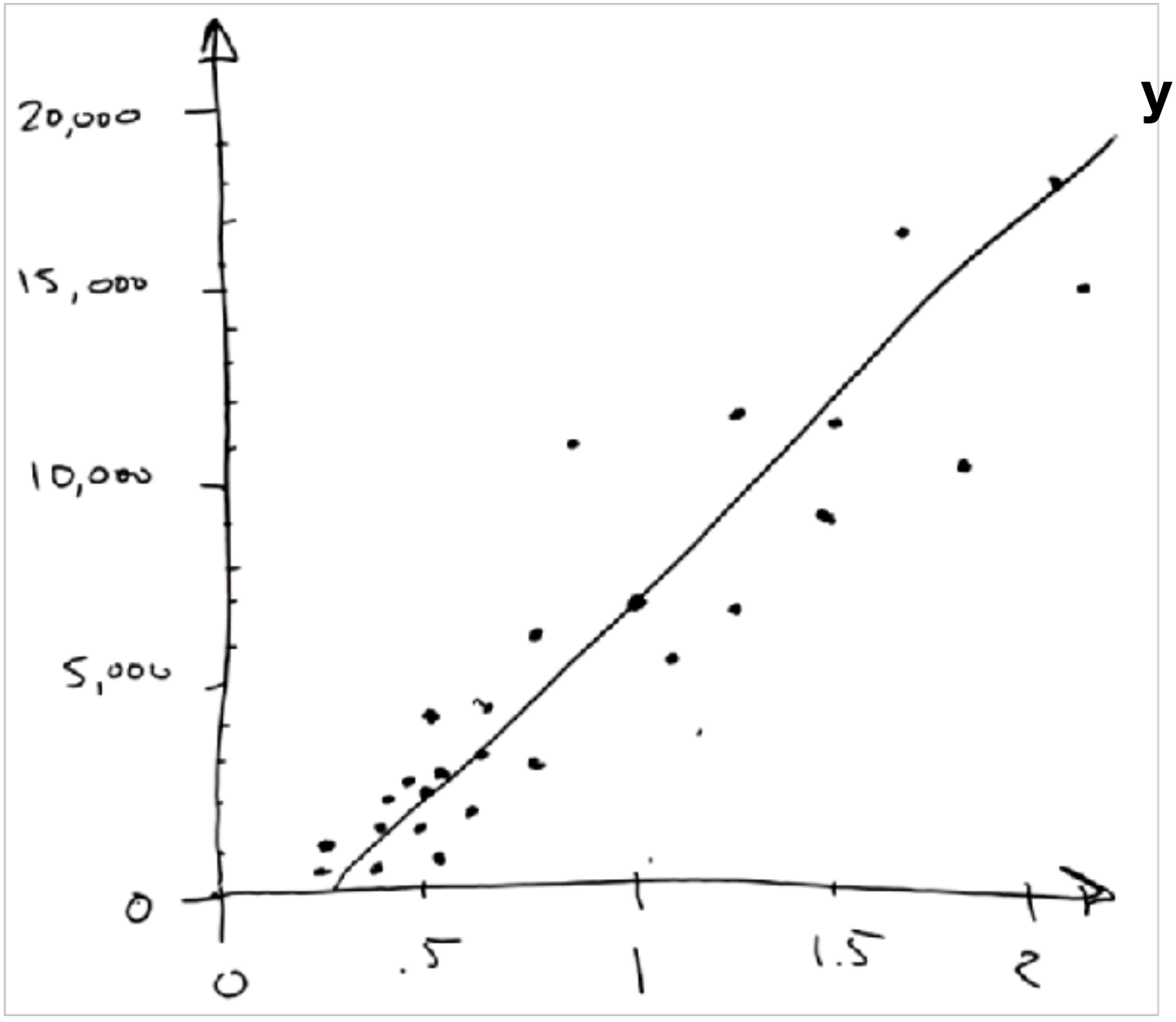



```
[-1274, -1252, -1160, -986, -792, -692, -614, -429, -286, -134, -57, -41, -169, -456, -450, -541, -761, -1067, -1231, -1047, -952, -645, -489, -448
, -397, -212, 193, 114, -17, -110, 128, 261, 198, 390, 461, 772, 948, 1451, 1974, 2624, 3793, 4968, 5939, 6057, 6581, 7302, 7640, 7223, 6119, 5461,
 4820, 4353, 3611, 2740, 2004, 1349, 1178, 1085, 901, 301, -262, -499, -488, -707, -1406, -1997, -2377, -2494, -2605, -2675, -2627, -2500, -2148, -
1648, -970, -364, 13, 260, 494, 788, 1011, 938, 717, 507, 323, 324, 325, 350, 103, -113, 64, 176, 93, -249, -461, -606, -909, -1159, -1307, -1544]
```



No matter what the information is, for a machine learning algorithm to correctly process it, it should always be transformed into a number.


20,000
15,000
10,000
5,000
0
0
.5
1
1.5
2

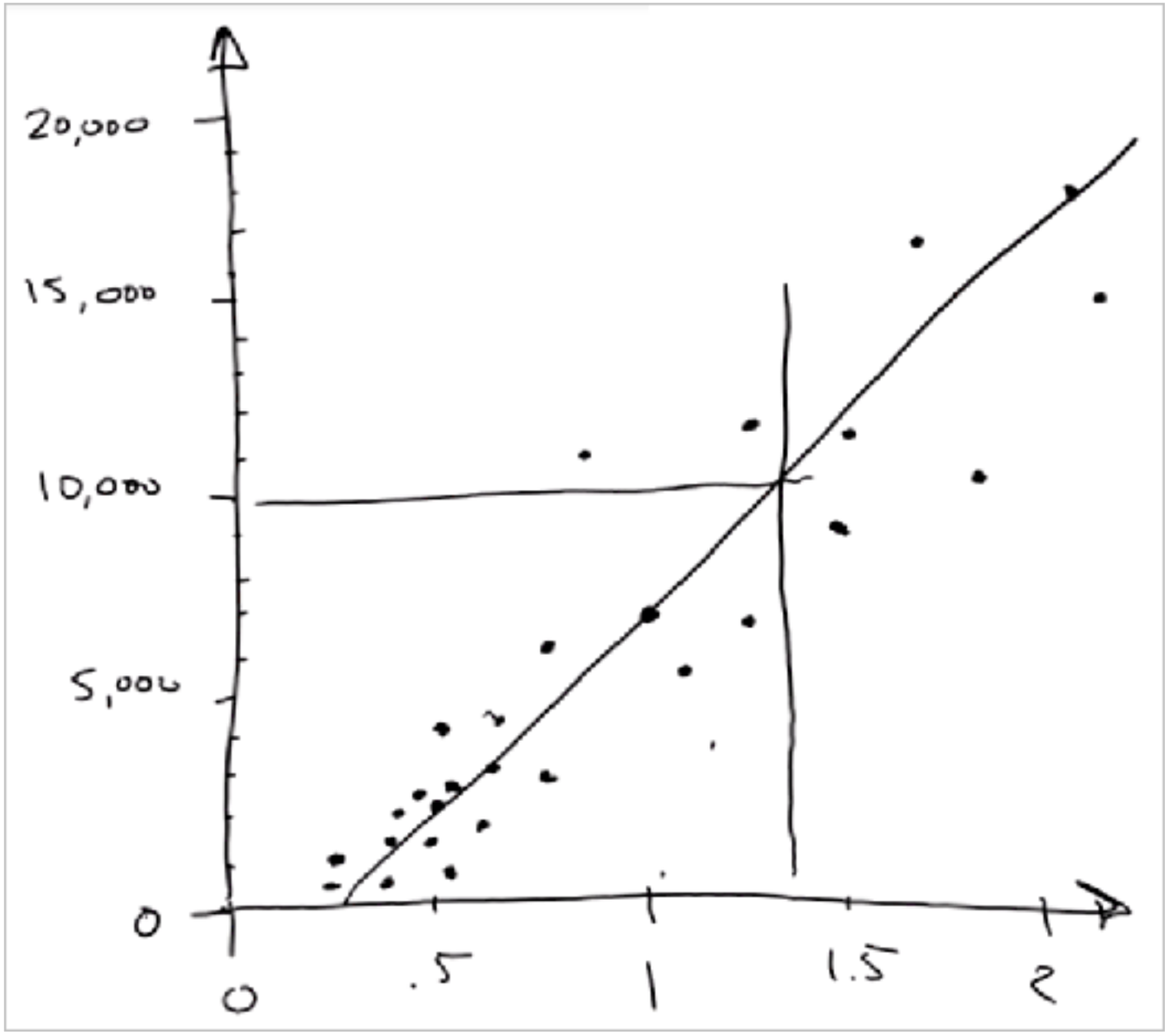


$y = mx + c$

$m = 9258$

$c = -2500$

$$y = mx + c$$



m= 9258

c= -2500

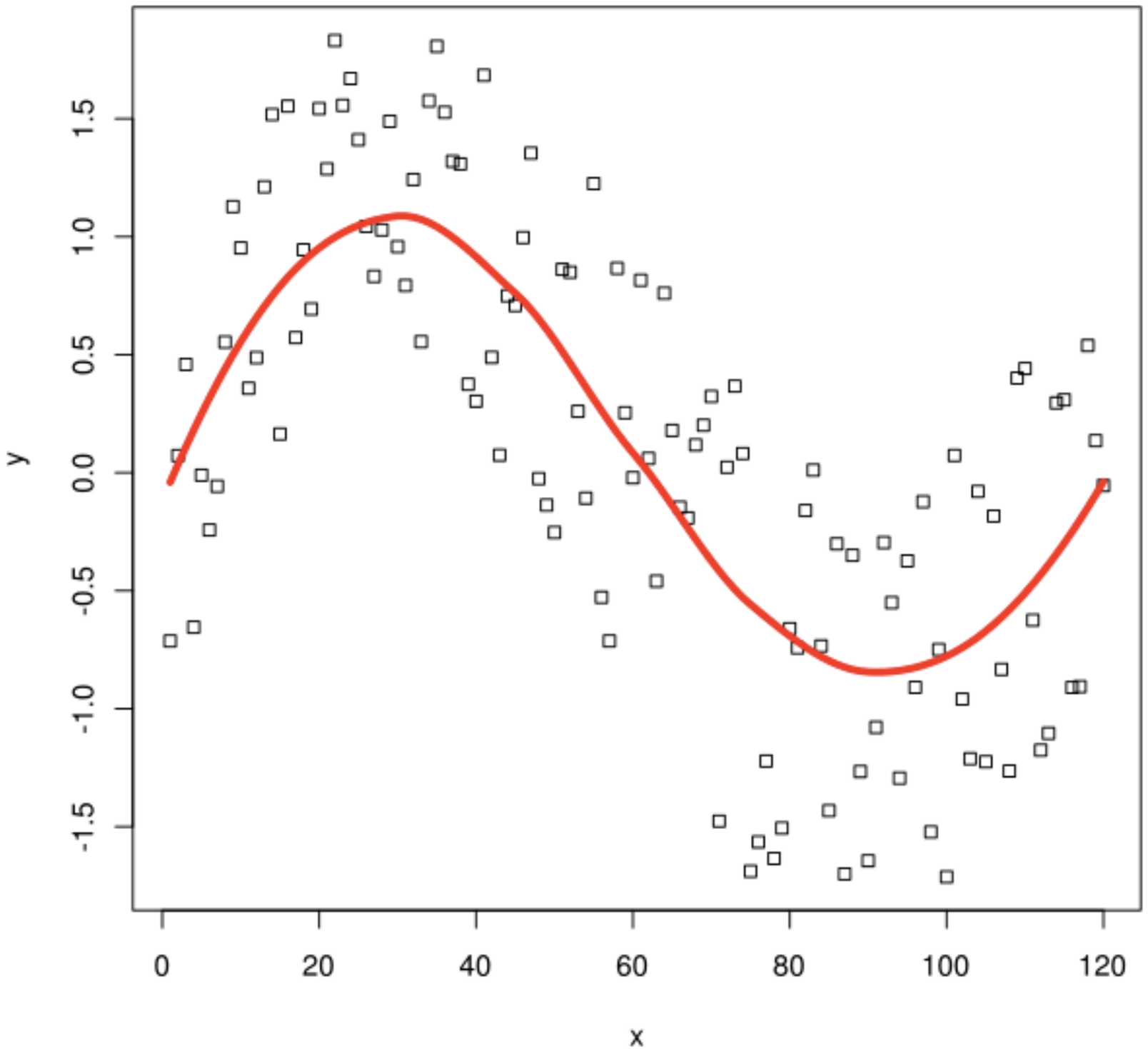
- How much will a 1.35 carat diamond cost?

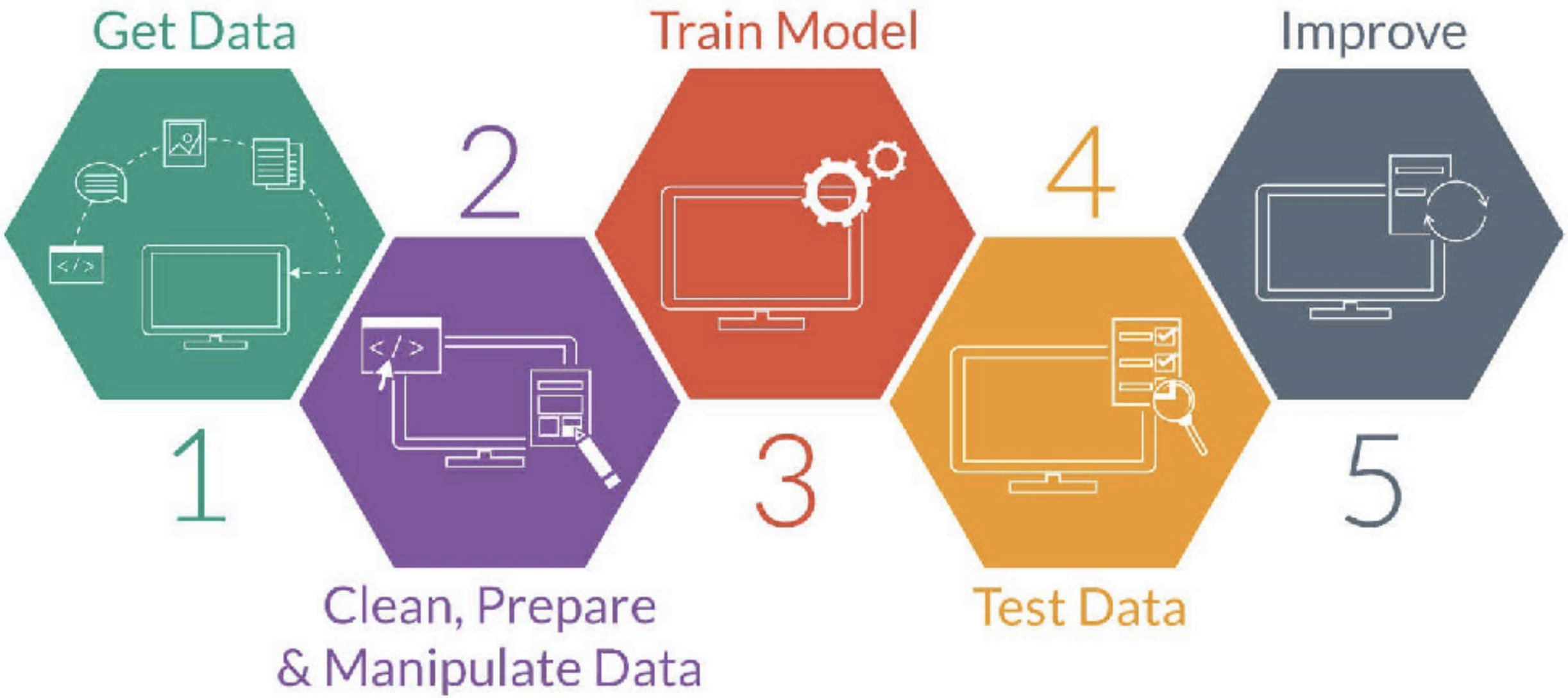
Input	Output
d 1	P1
d 2	P2
d 3	P3

Prediction = model(data)

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	$250,000
2	800	Hipsterton	$300,000
2	850	Normaltown	$150,000
1	550	Normaltown	$78,000
4	2000	Skid Row	$150,000

Price = model(bedrooms, sq feet, neighborhood)


y
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
0
20
40
60
80
100
120
x


Get Data
1
Clean, Prepare
& Manipulate Data
2
Train Model
3
Test Data
4
Improve
5

ML

Supervised

Un Supervised

Semi Supervised

Reinforcement

Supervised

x1	x2	x3	y
d	d	d	P1
d	d	d	P2
d	d	d	P3

prediction = model(x1,x2,x3)

UnSupervised

x1	x2	x3
d	d	d
d	d	d
d	d	d

group = model(x1,x2,x3)

Semi Supervised

x1	x2	x3	y
d	d	d	
d	d	d	P2
d	d	d	
d	d	d	P4

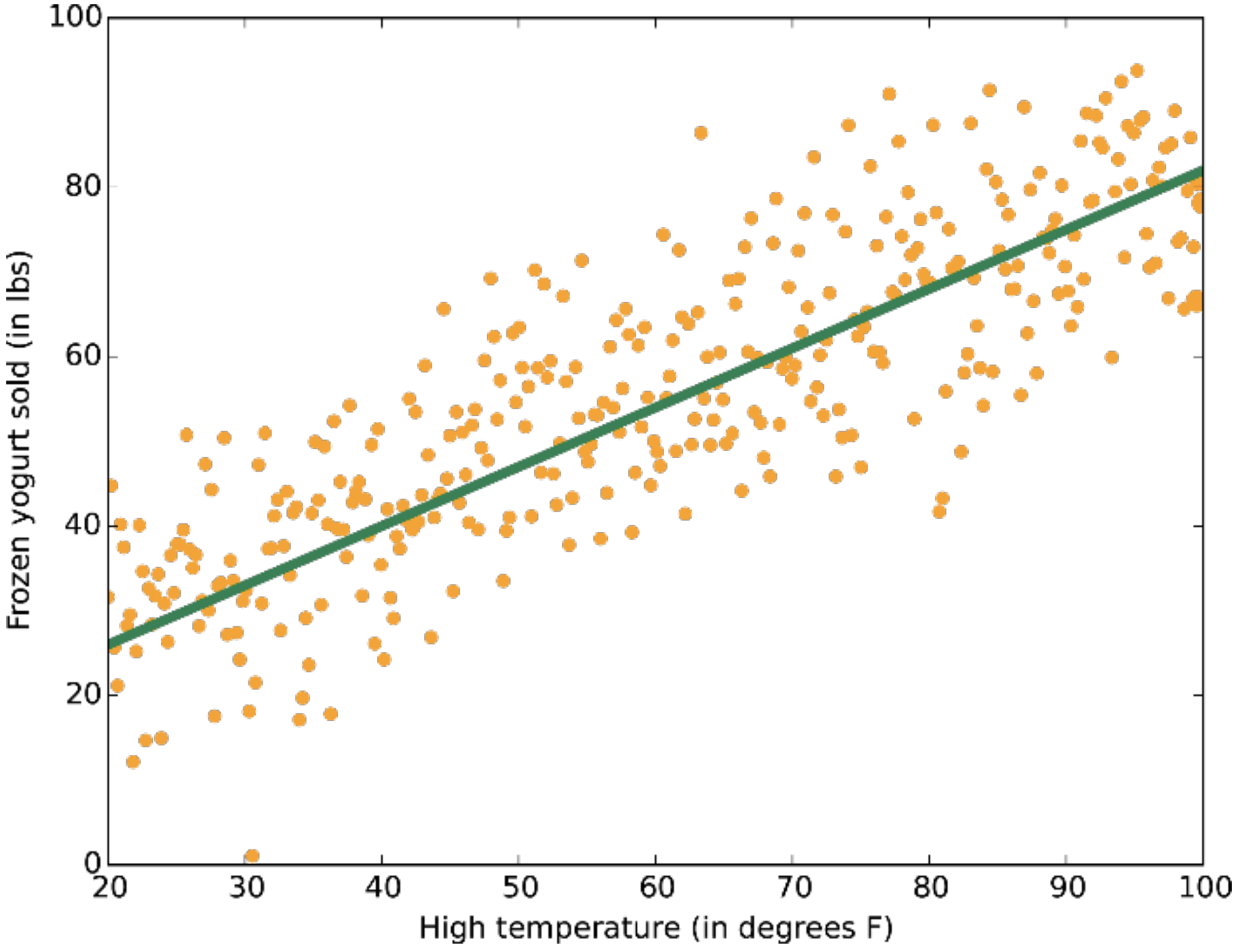
prediction = model(x1,x2,x3)

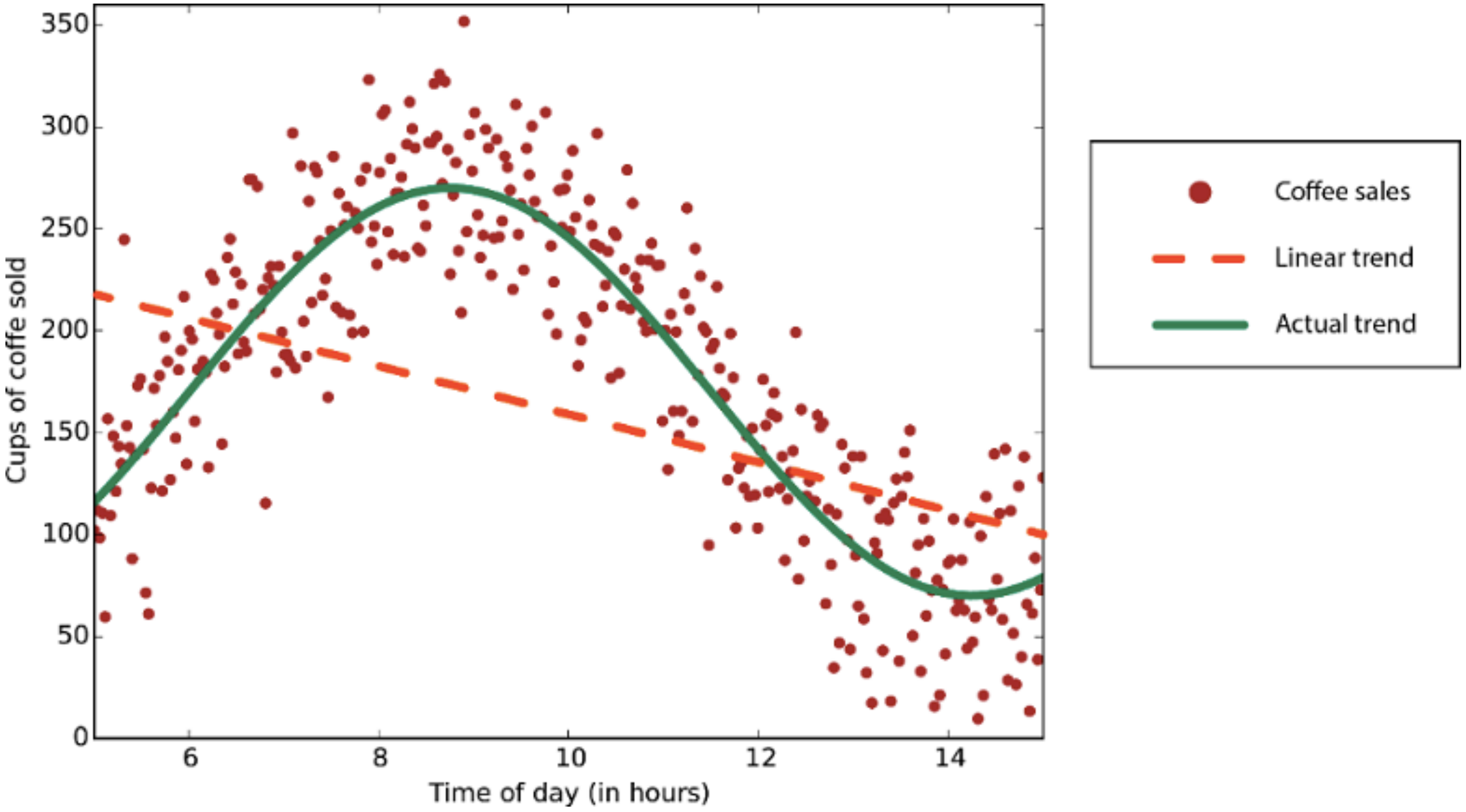
There are only **5** questions

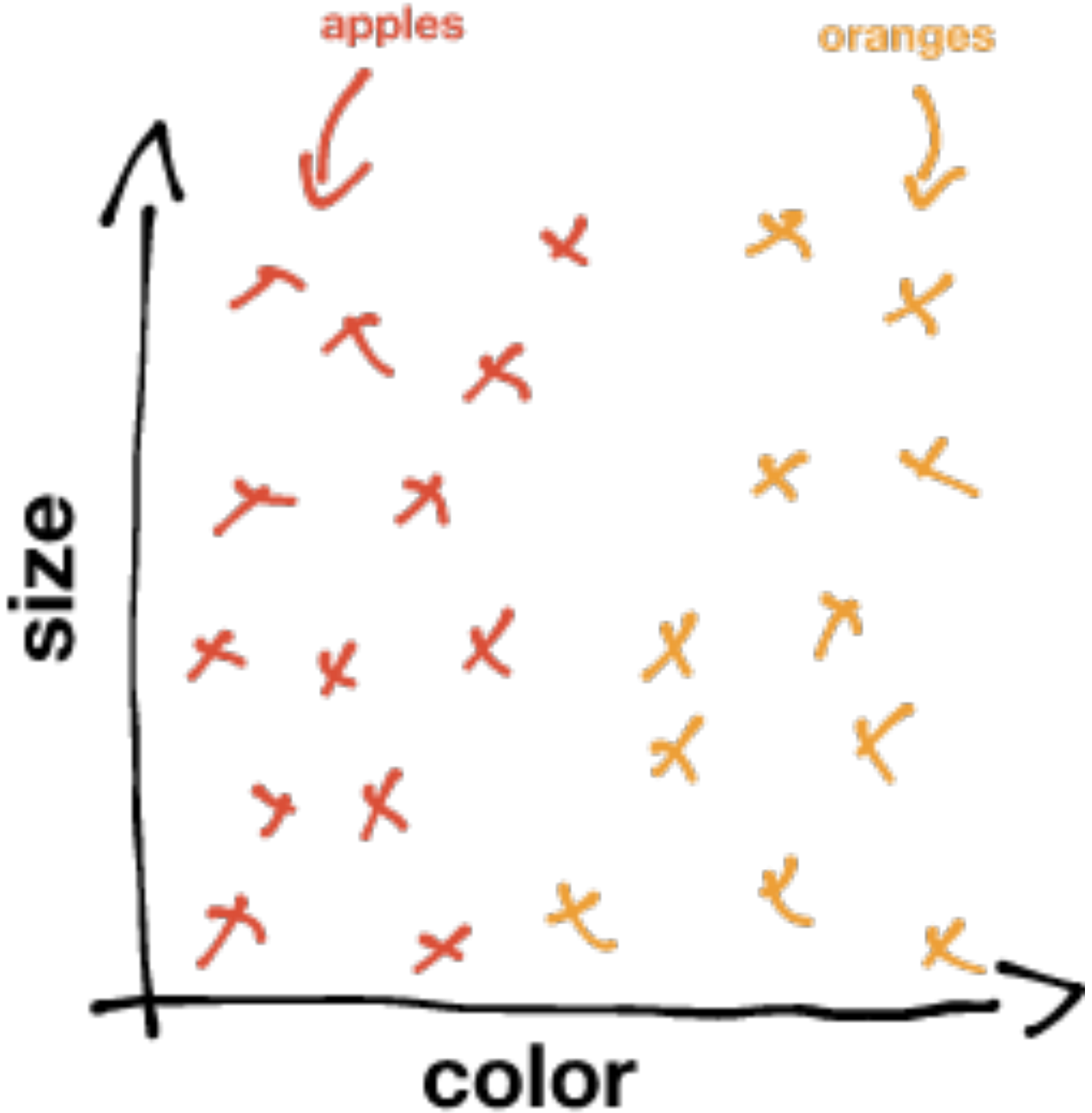
that

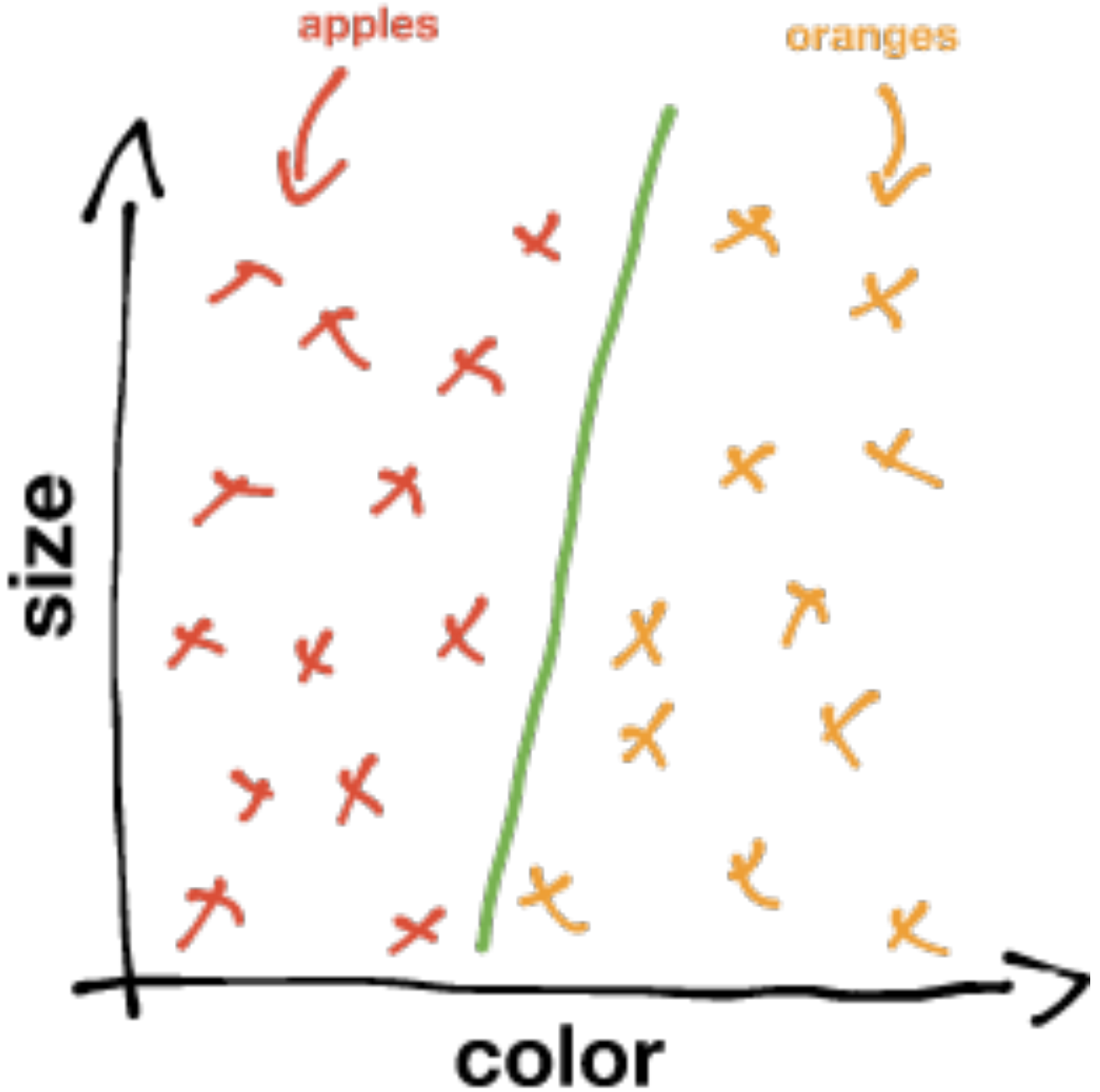
ML can answer

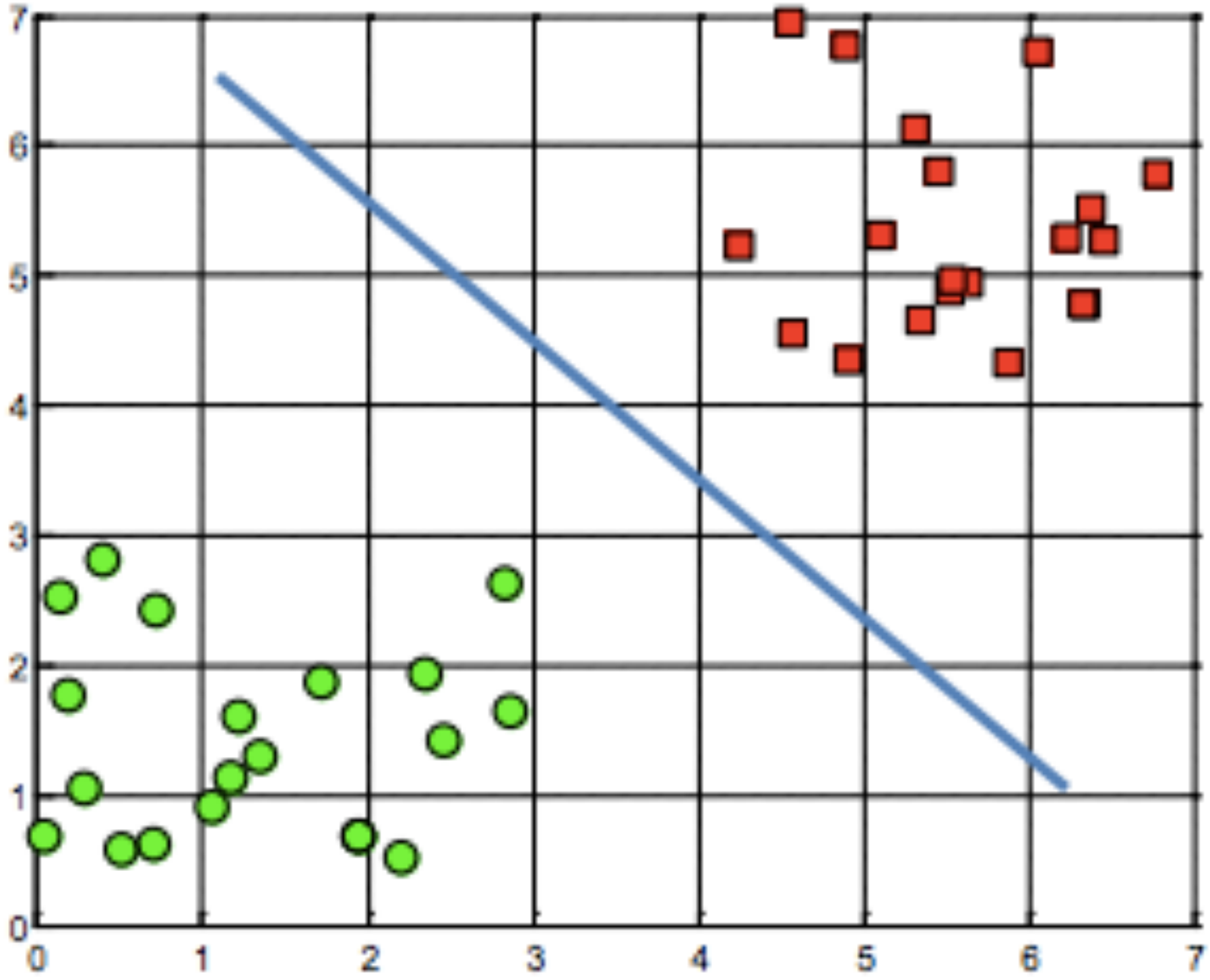
1. Is this A or B? - Classification
2. How much – or – How many? - Regression
3. How is this organized? - Clustering
4. Is this weird? - Anomaly detection
5. What should I do next? - Reinforcement

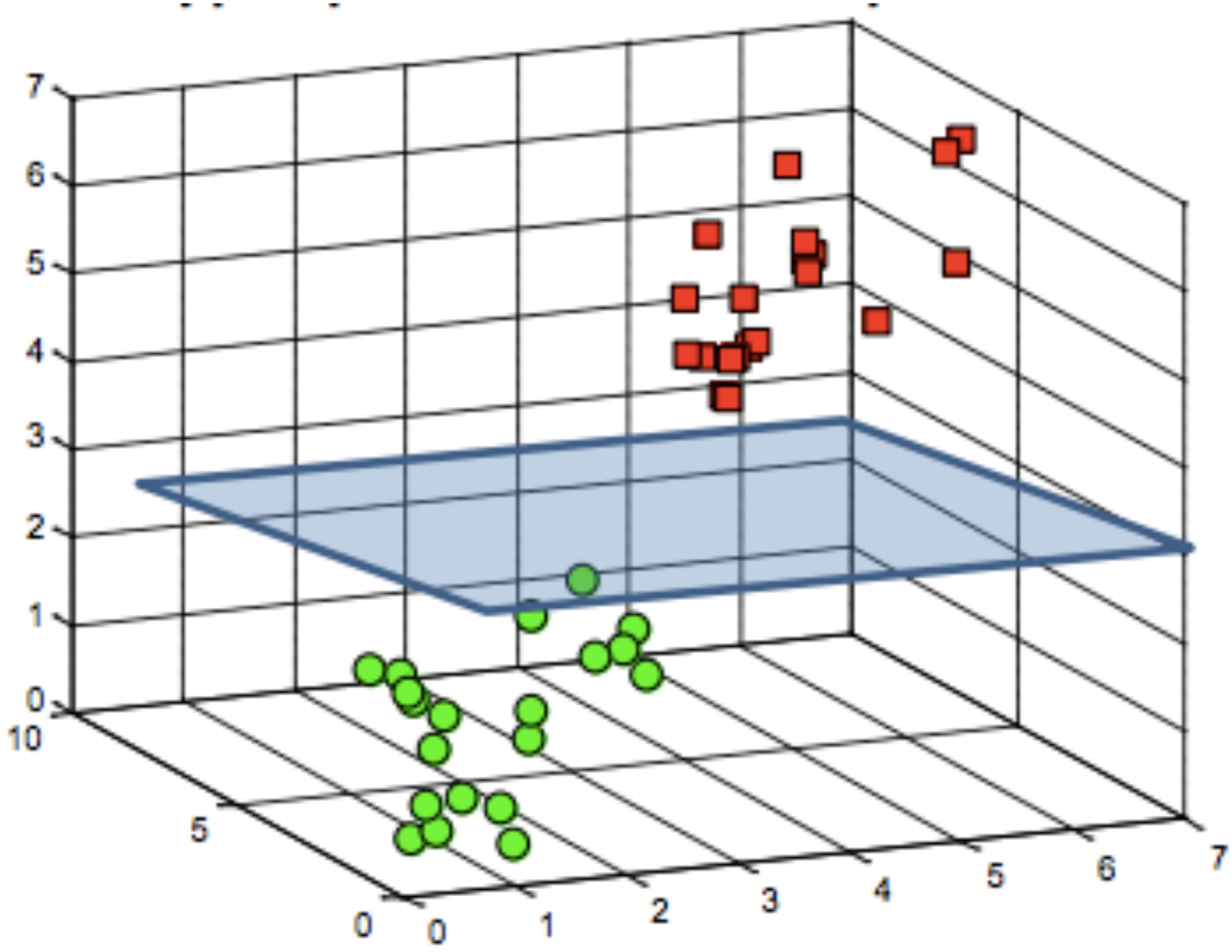

100
80
60
40
20
0
Frozen yogurt sold (in lbs)
20
30
40
50
60
70
80
90
100
High temperature (in degrees F)

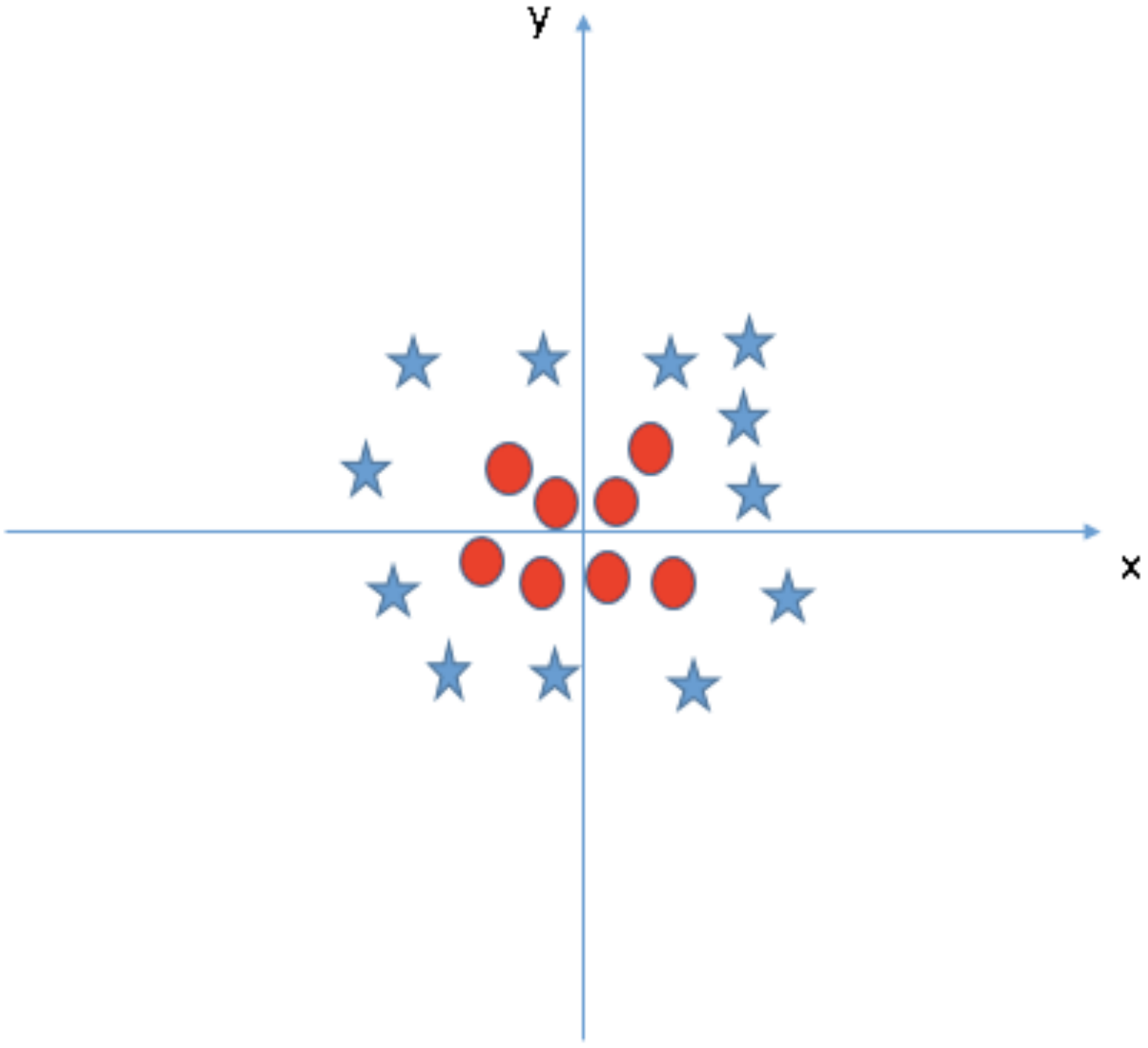

350
300
250
200
150
100
50
0
Cups of coffe sold
6
8
10
12
14
Time of day (in hours)
Coffee sales
Linear trend
Actual trend

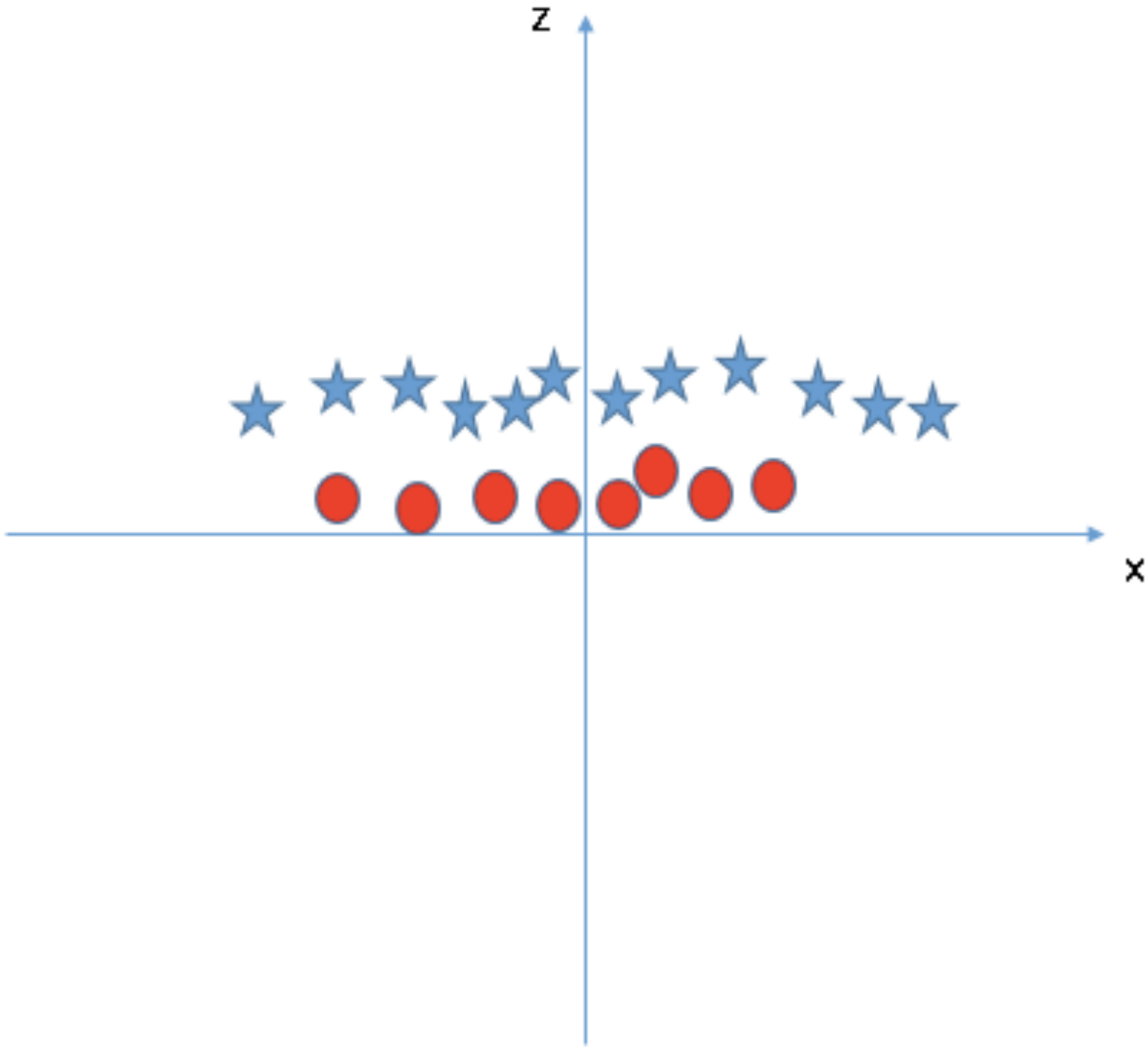

apples
oranges
size
color

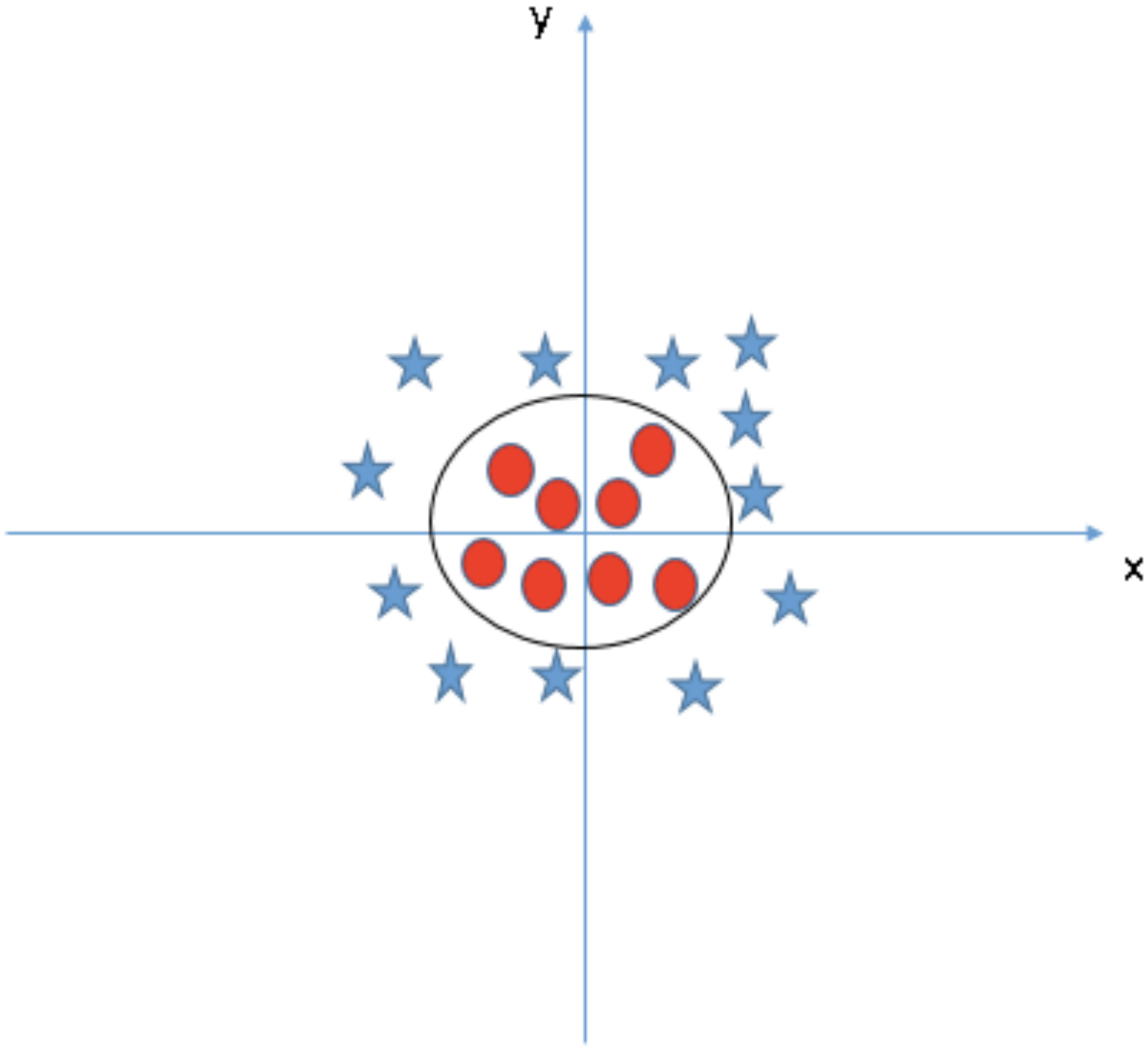

apples
oranges
size
color

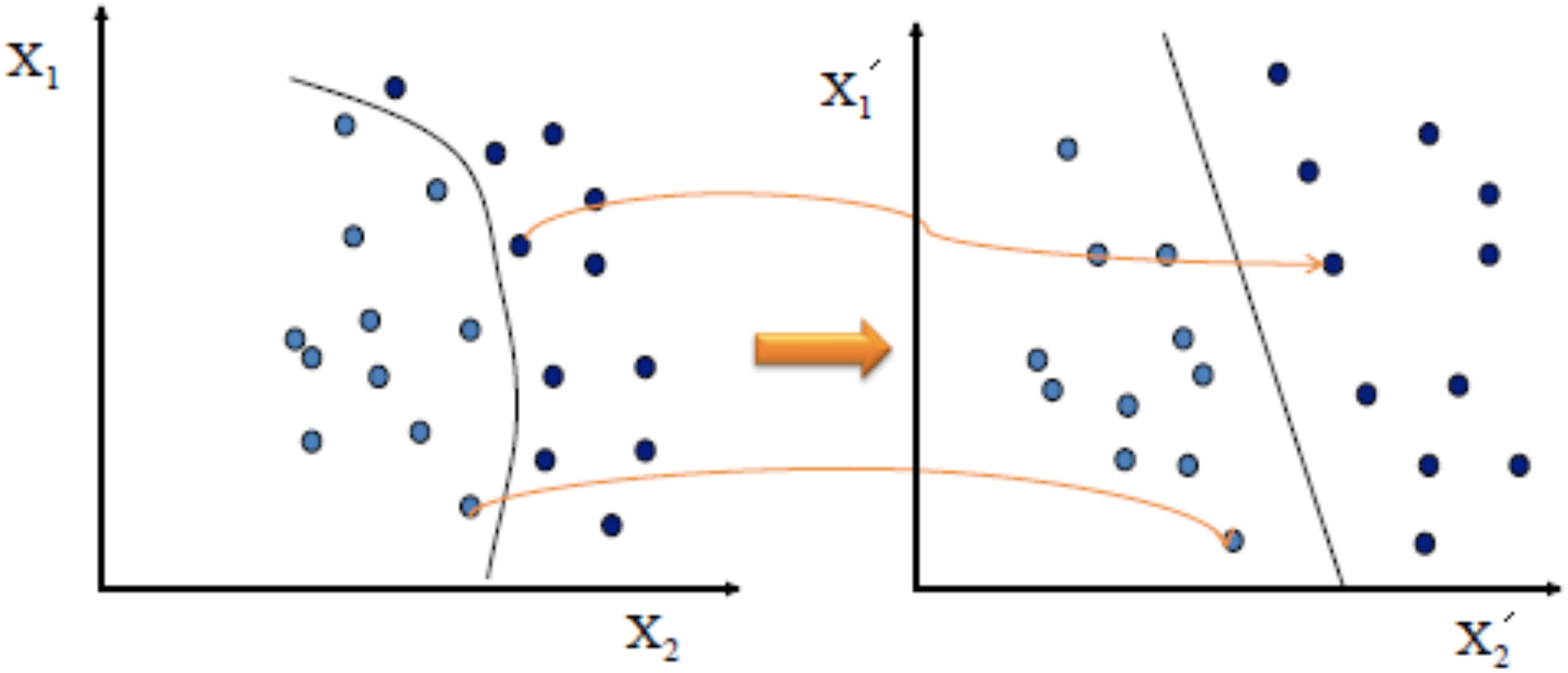

7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7

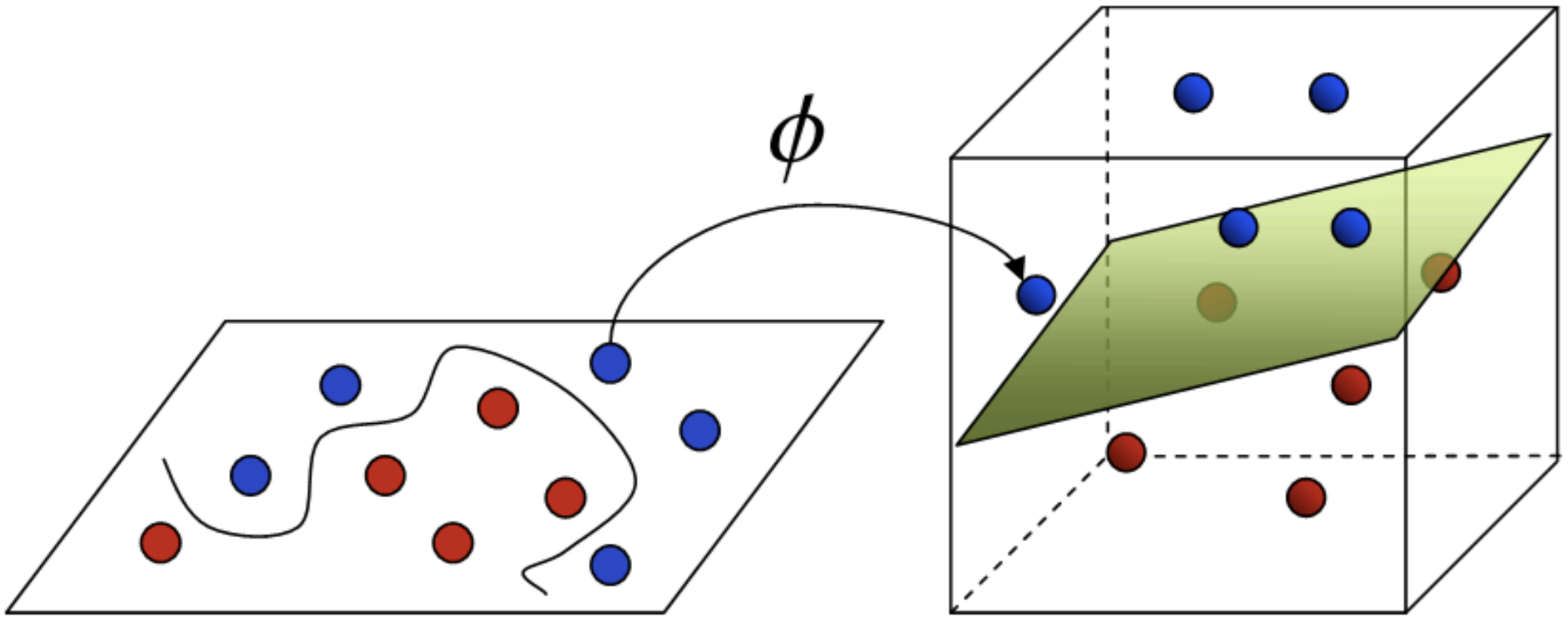

7
6
5
4
3
2
1
0
10
5
0
0
1
2
3
4
5
6
7


y
x

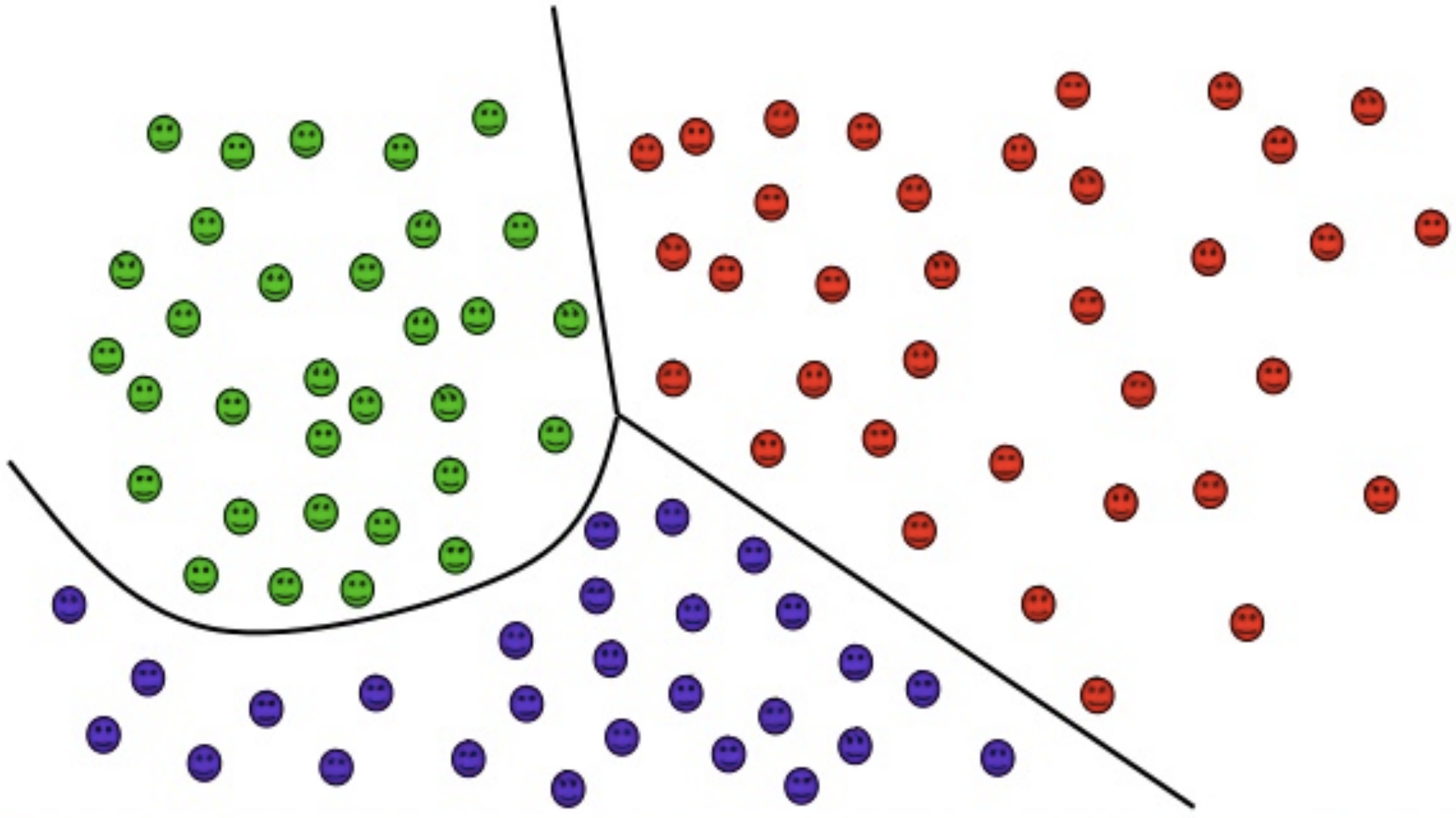

z
x

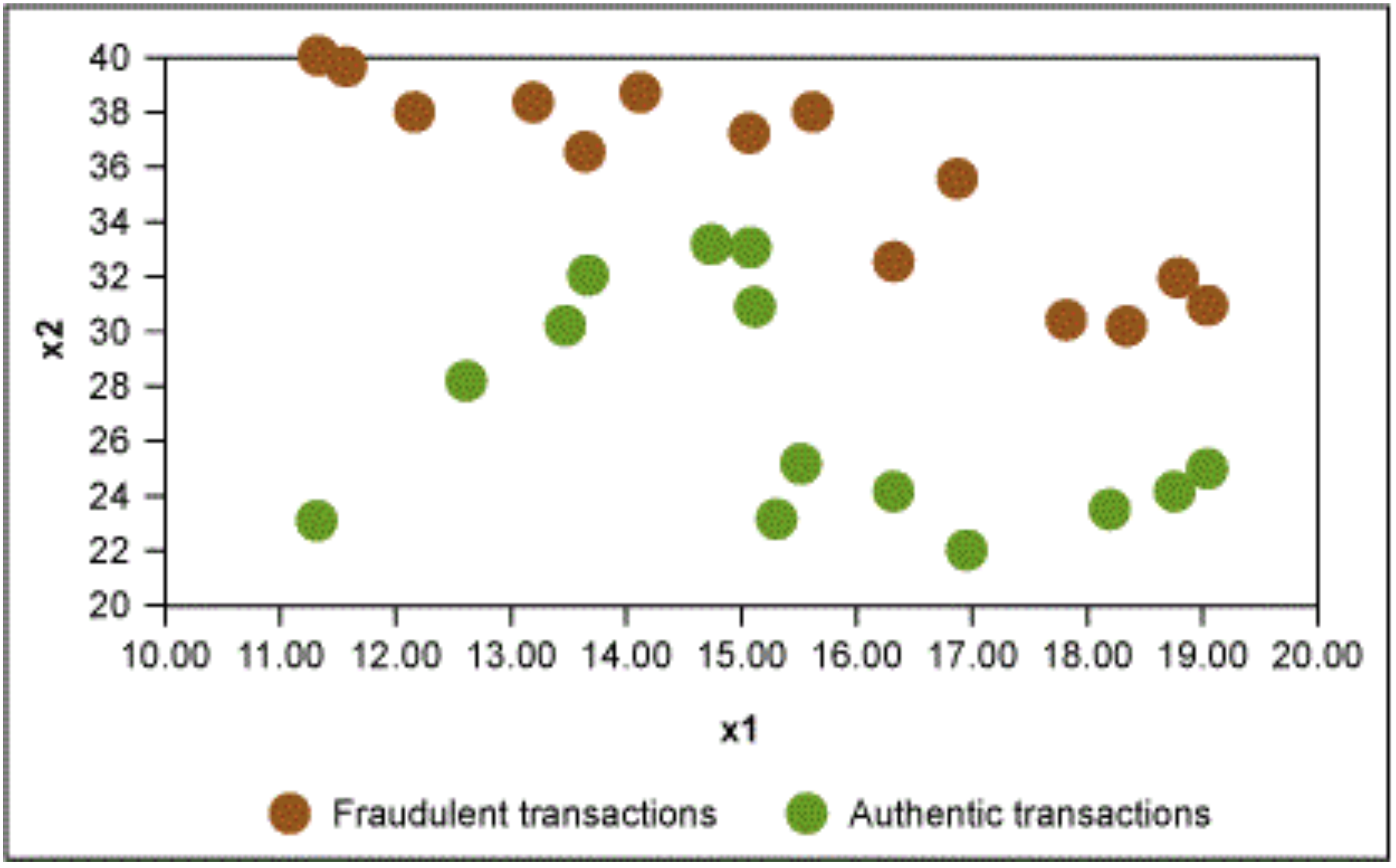

y
x

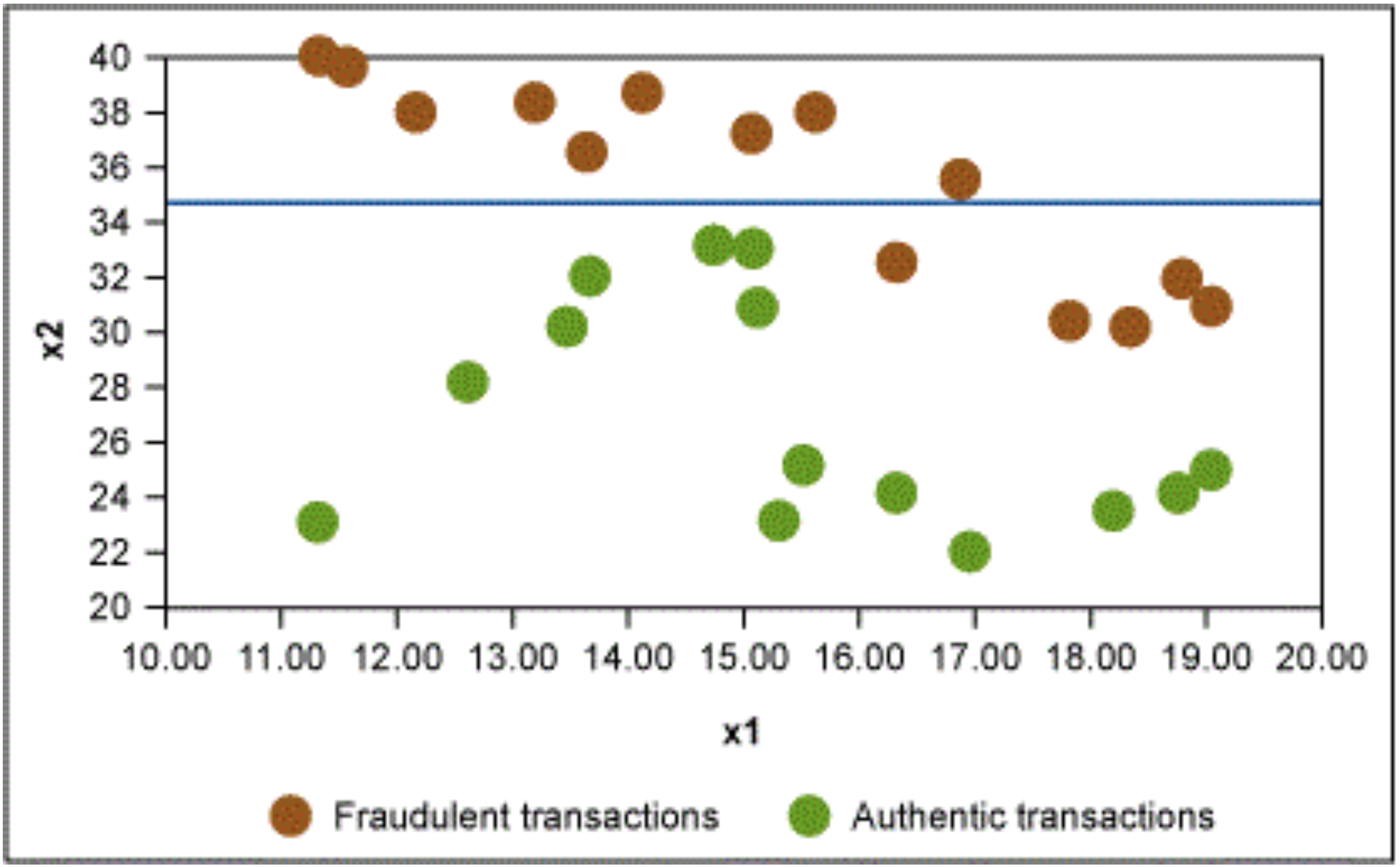

X1
X2
X1´
X2´

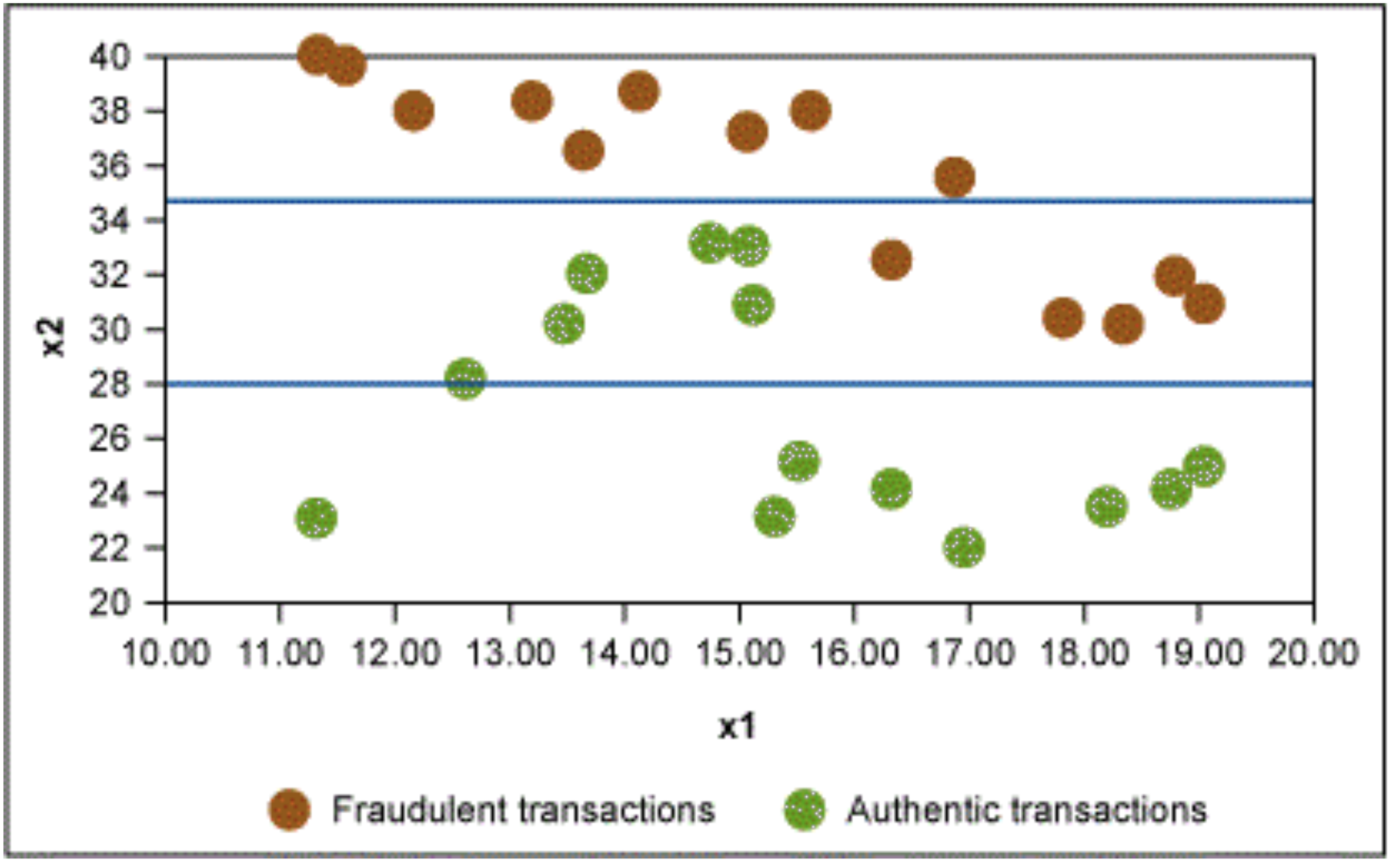

ϕ
Input Space
Feature Space

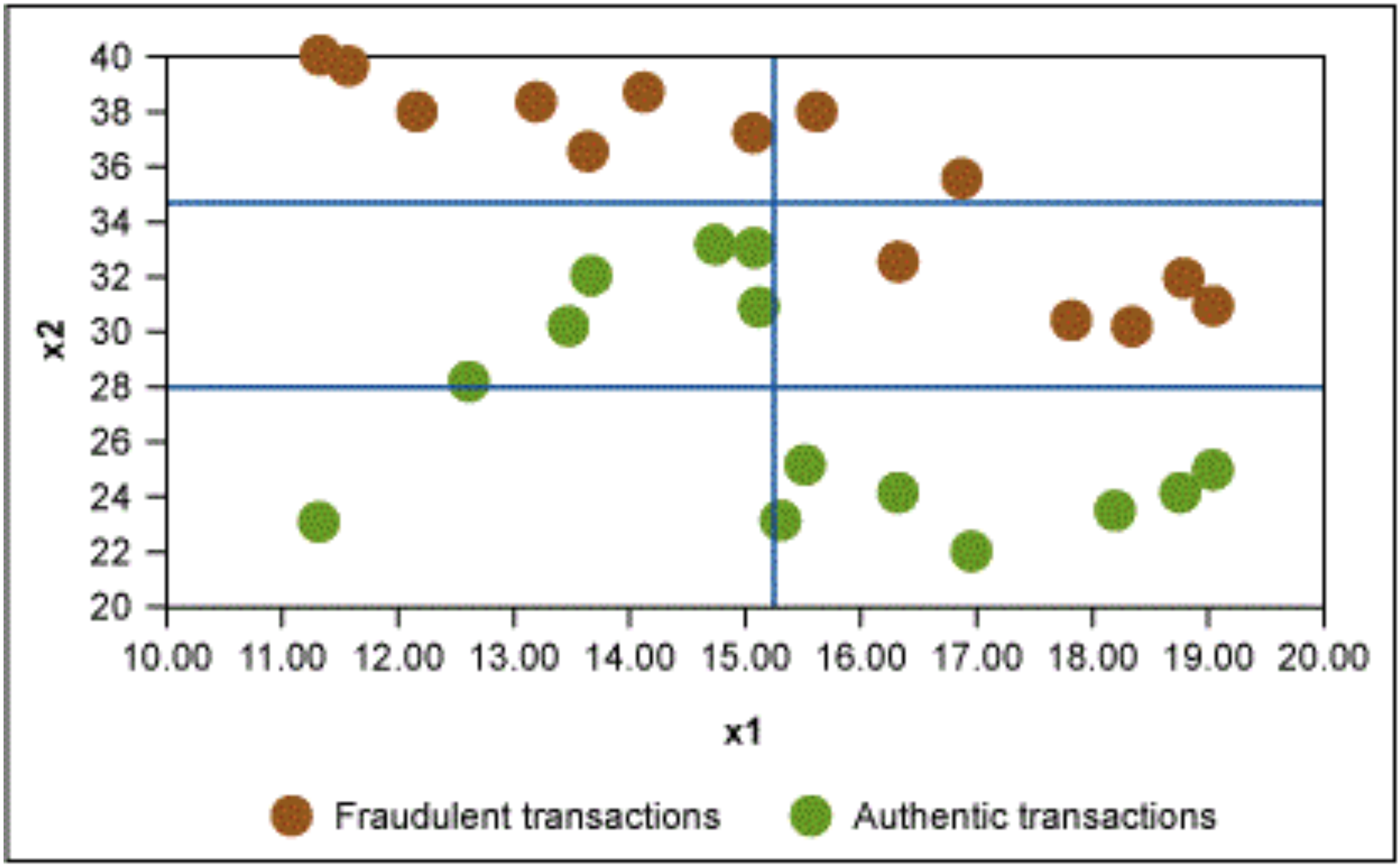
Multi-class classification

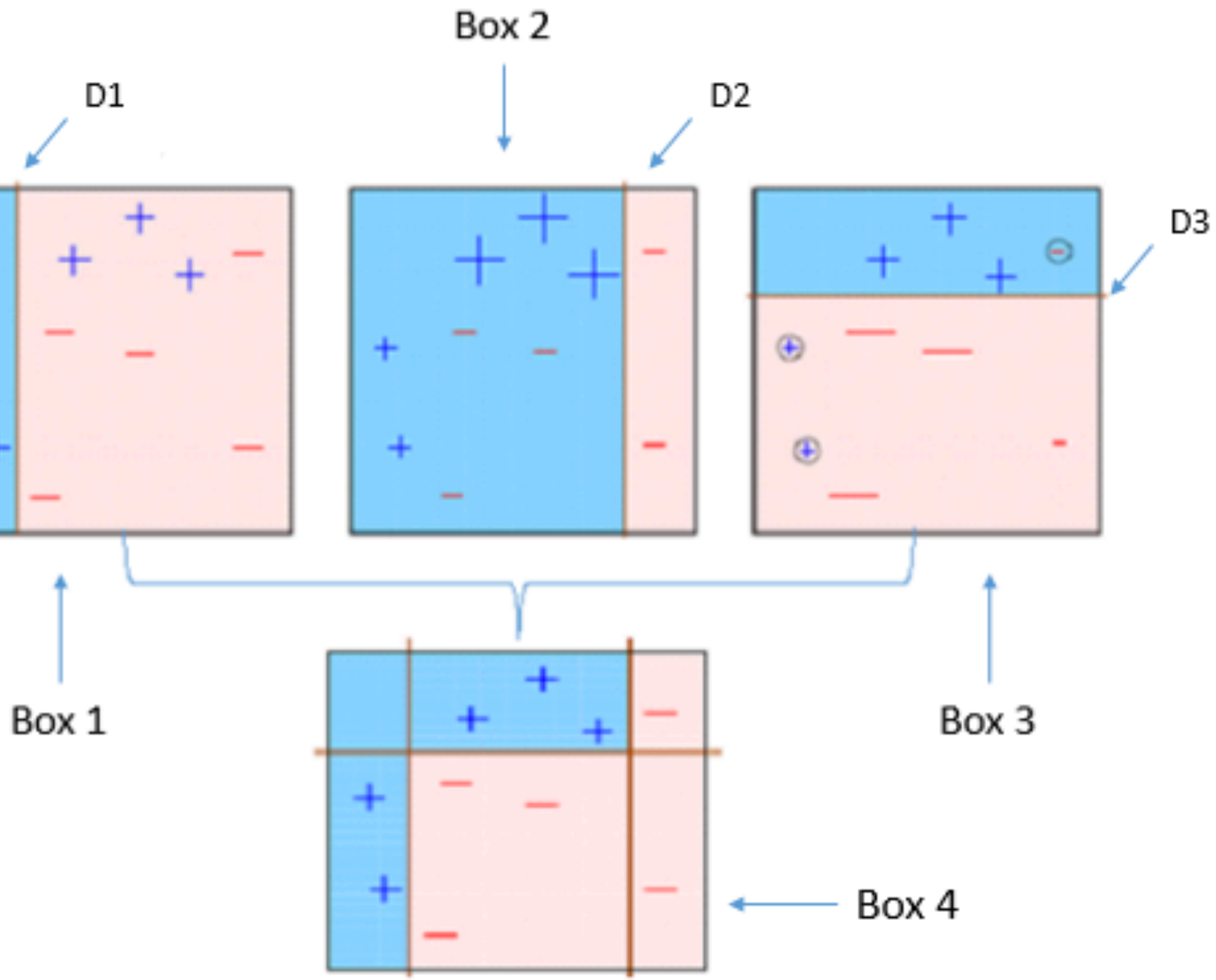



40
38
36
34
32
30
28
26
24
22
20
x2
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00
19.00
20.00
x1
Fraudulent transactions
Authentic transactions


40
38
36
34
32
30
28
26
24
22
20
x2
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00
19.00
20.00
x1
Fraudulent transactions
Authentic transactions

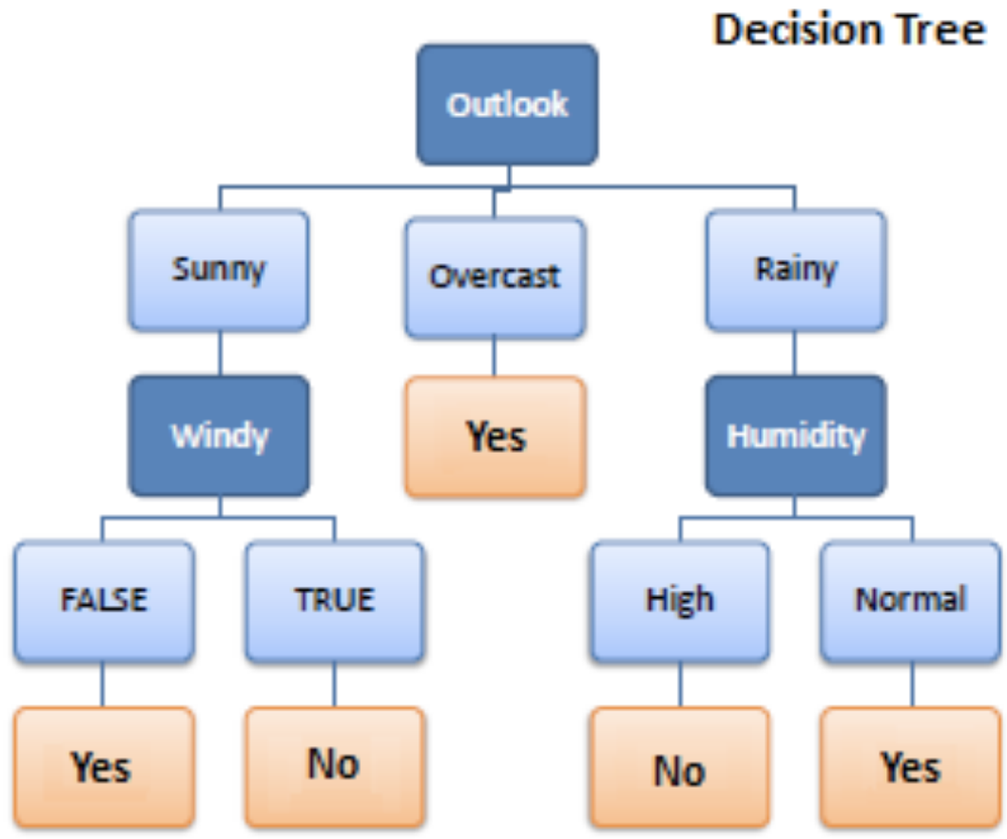

40
38
36
34
32
30
28
26
24
22
20
x2
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00
19.00
20.00
x1
Fraudulent transactions
Authentic transactions

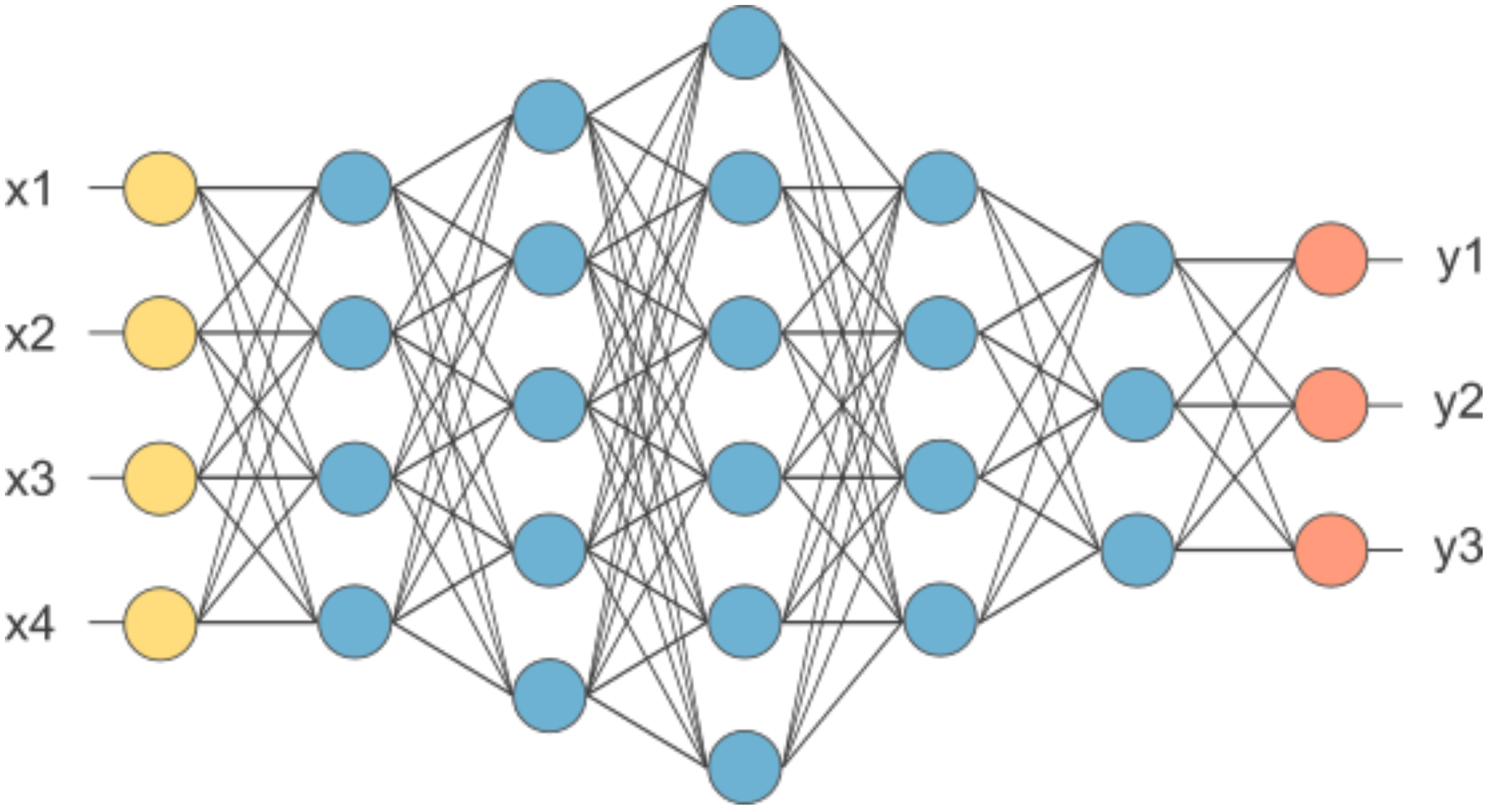

40
38
36
34
32
30
28
26
24
22
20
x2
10.00
11.00
12.00
13.00
14.00
15.00
16.00
17.00
18.00
19.00
20.00
x1
Fraudulent transactions
Authentic transactions


Box 2
D1
D2
D3
Box 1
Box 3
Box 4

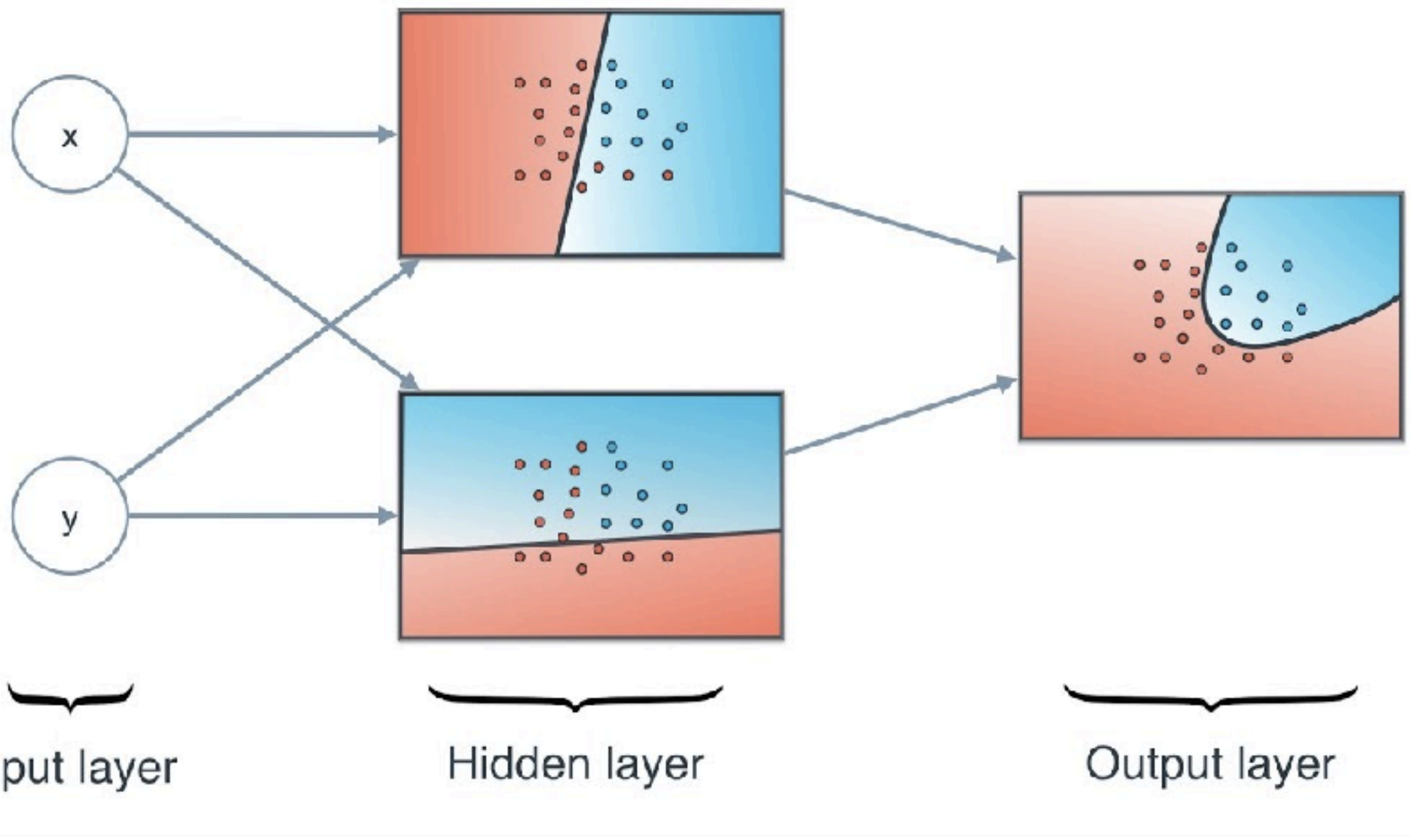
Predictors | Target

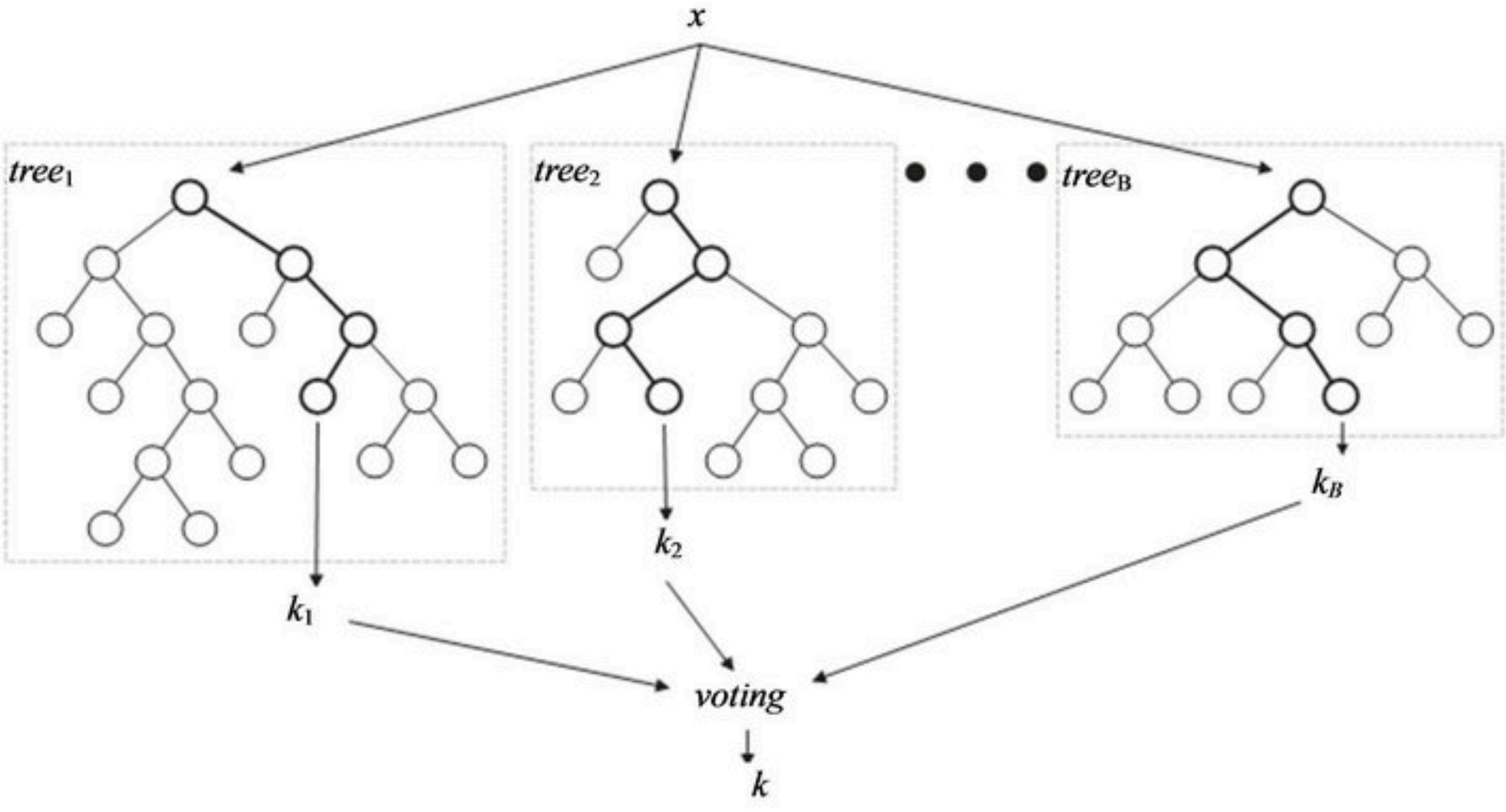
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No




x1
x2
x3
x4
y1
y2
y3

Neural Network




x
tree1
tree2
treeB
k1
k2
kB
voting
k


y
85
80
75
70
65
60
55
50
45
40
Number of Friends
155
158
f
163
m
170
f
158
m
155
m
178
m
183
f
180
m
191
f
0
150
155
160
165
170
175
180
185
190
195
200
x
Number of Selfies

	POP	PER CAPITA	REGION
Singapore	4m	51.431$	Asia
Brazil	207m	14.023$	South America
USA	324m	57.436$	North America
Hong Kong	7m	42.431$	Asia
Malaysia	30m	9.500$	Asia
Japan	127m	39.000$	Asia
Syria	18m	2.058$	Middle East
Slovenia	2m	33.579$	Europe
Italy	60m	30.050$	Europe

Brazil
Pop: 207m
Per capita: 14.023$

Hong Kong
Pop: 7m
Per capita: 42.431$

Japan
Pop:127m
Per capita: 39.000$

Syria
Pop: 18m
Per capita: 2.058$

Malaysia
Pop: 30m
Per capita: 9.500$

Italy
Pop: 60m
Per capita: 30.050$

USA
Pop: 324m
Per capita: 57.436$

Slovenia
Pop: 2m
Per capita: 33.579$

Singapore
Pop: 4m
Per capita: 51.431$

Classification

	CATEGORY
Hurricane Irma	5
Hurricane Harvey	4
Hurricane El Nino	1

Classification

Wonder Woman ✓

Pirates Of The Caribbean ✗

Classification

	PERFORMANCE RATING
Victor	Above Average
Michelle	Excellent
Joe	Poor

Regression

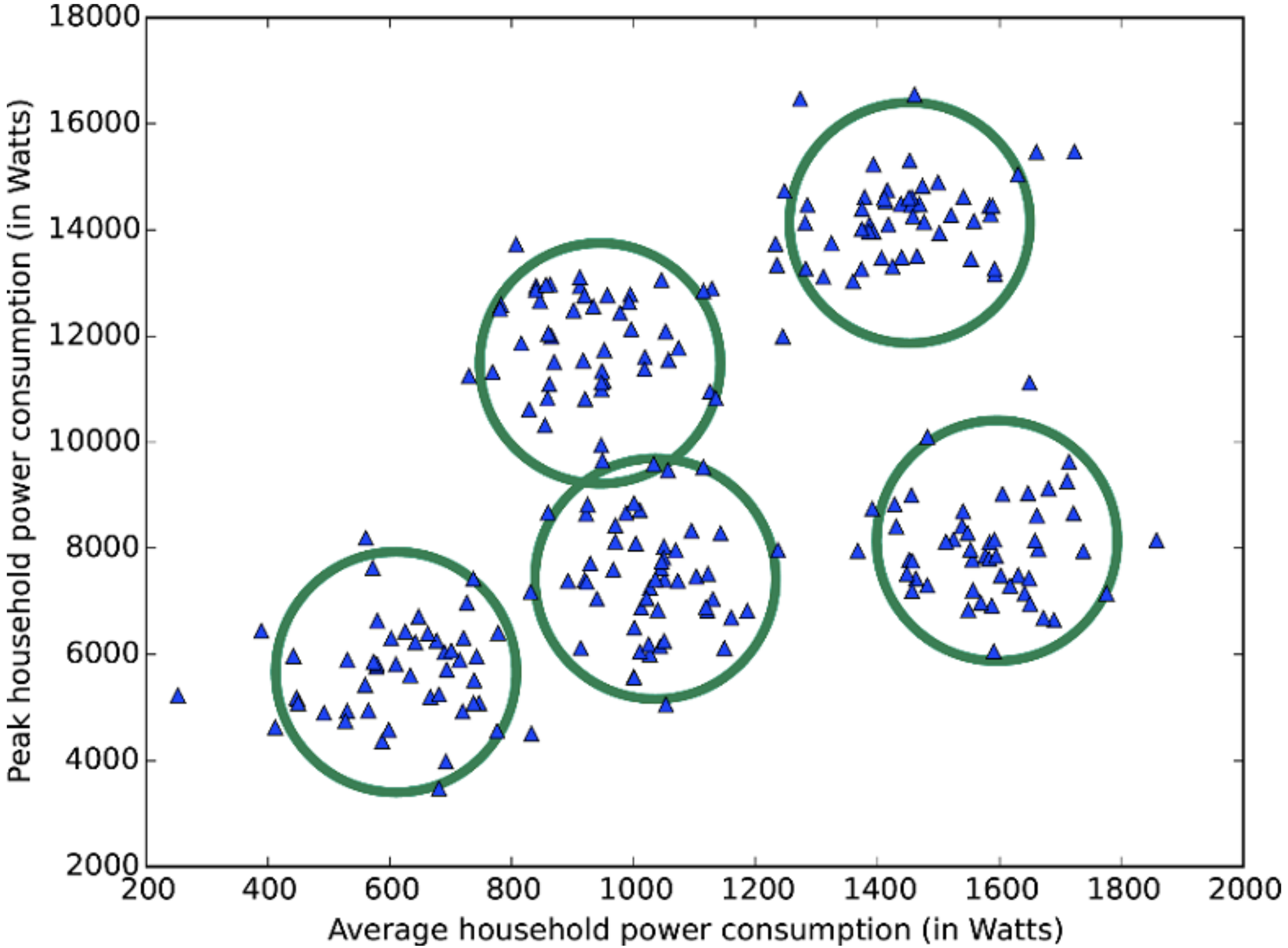
	CAT	WIND SPEED
Hurricane Irma	5	320 km/hr
Hurricane Harvey	4	240 km/hr
Hurricane El Nino	1	190 km/hr

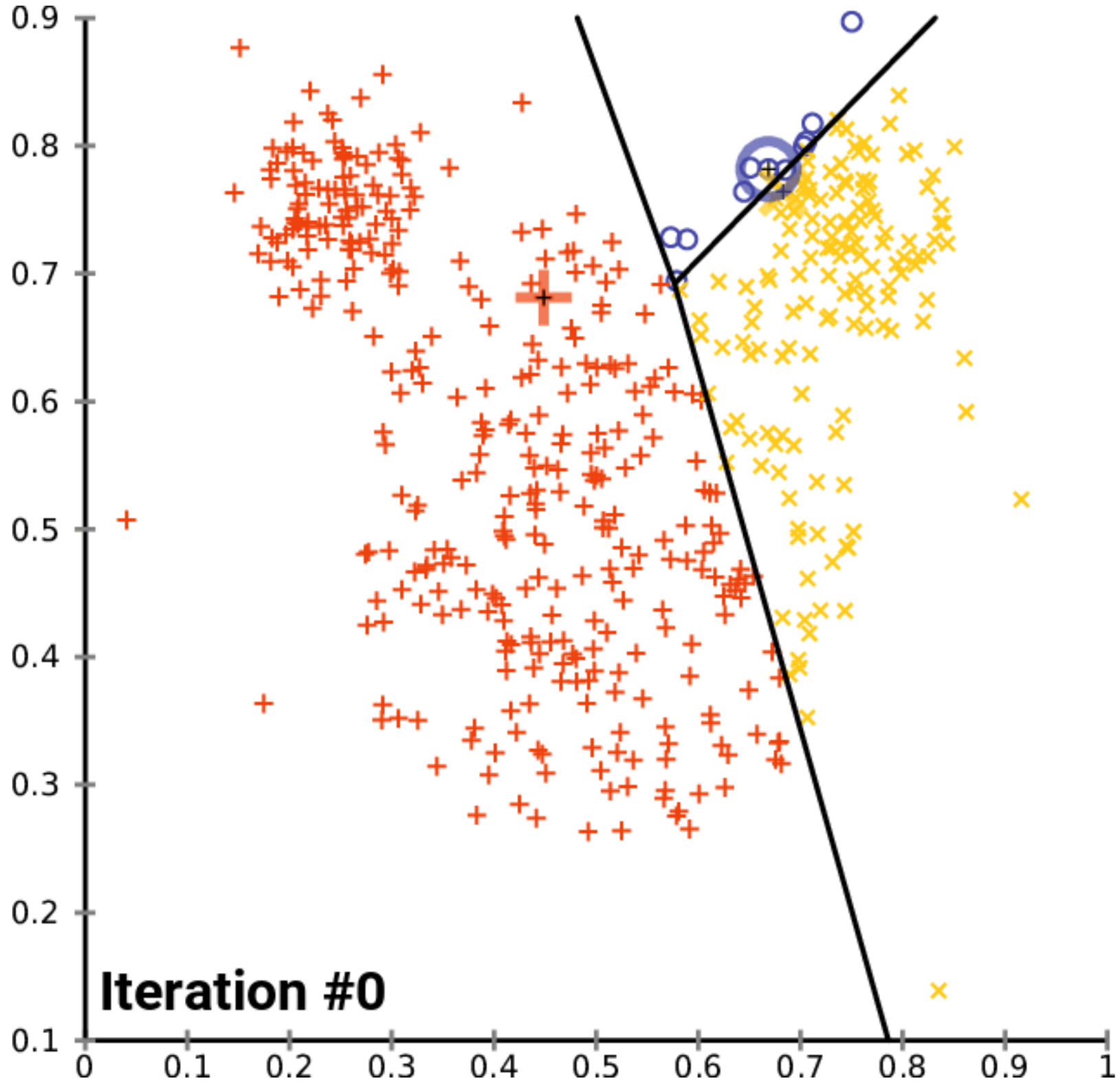
Regression

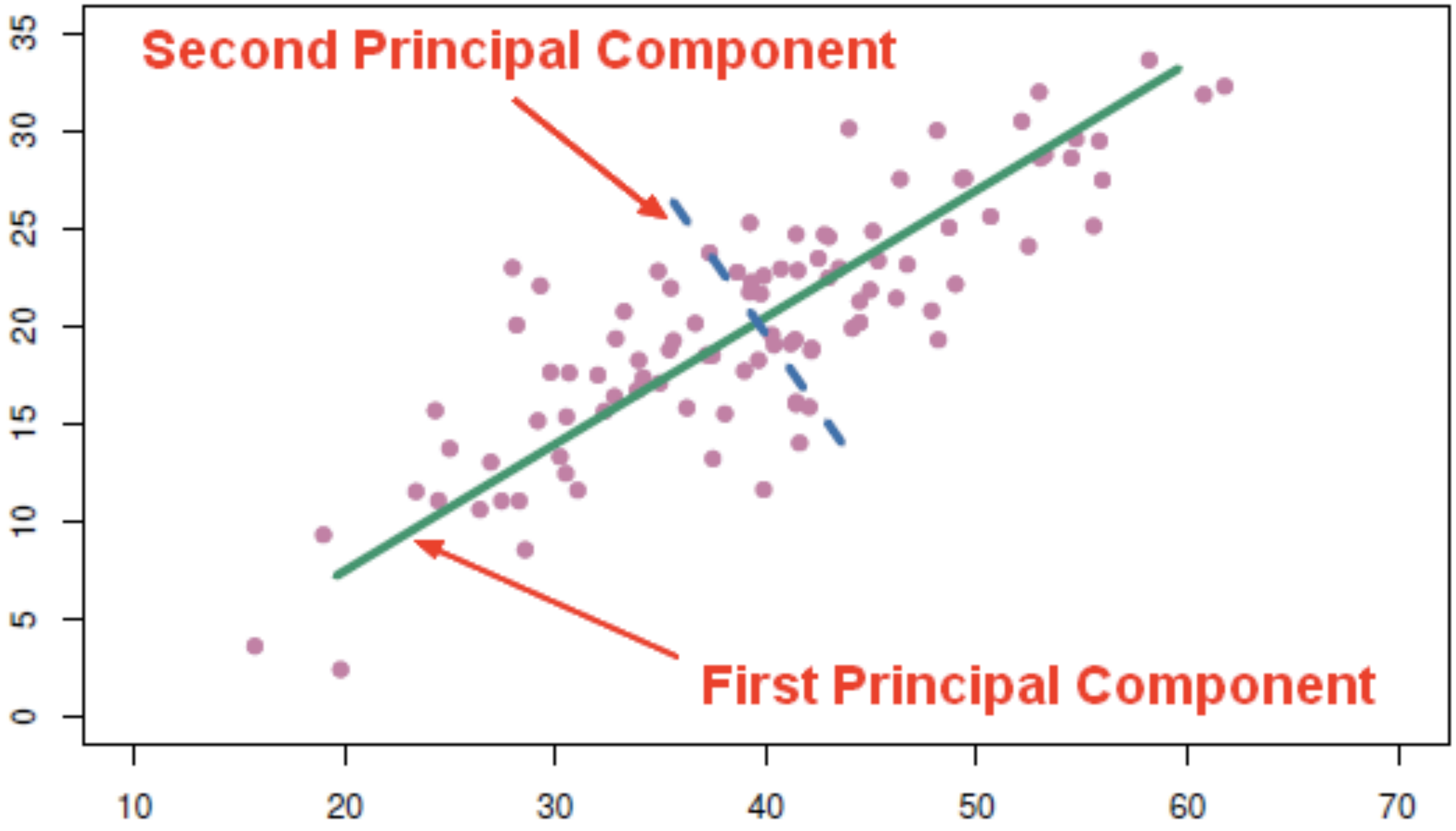
		REVENUE
Wonder Women	✓	250 million USD
Pirates Of The Caribbean	✗	120 million USD

Regression

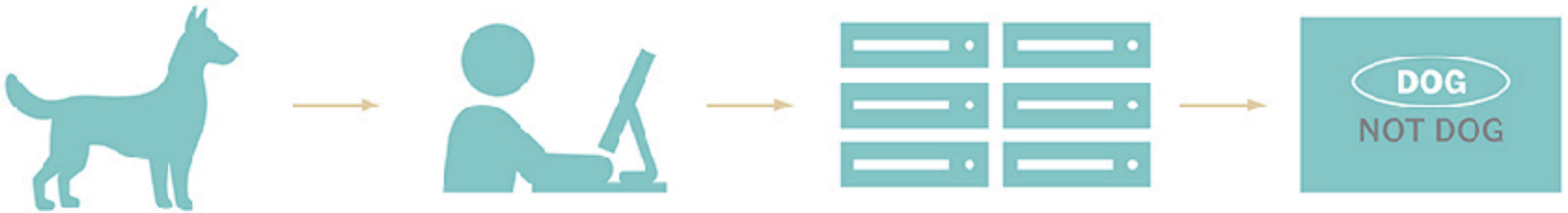
	RATING	SALES
Victor	Above Average	40
Michelle	Excellent	30
Joe	Poor	20


18000
16000
14000
12000
10000
8000
6000
4000
2000
Peak household power consumption (in Watts)
200
400
600
800
1000
1200
1400
1600
1800
2000
Average household power consumption (in Watts)


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Iteration #0


Second Principal Component
First Principal Component
0
5
10
15
20
25
30
35
10
20
30
40
50
60
70

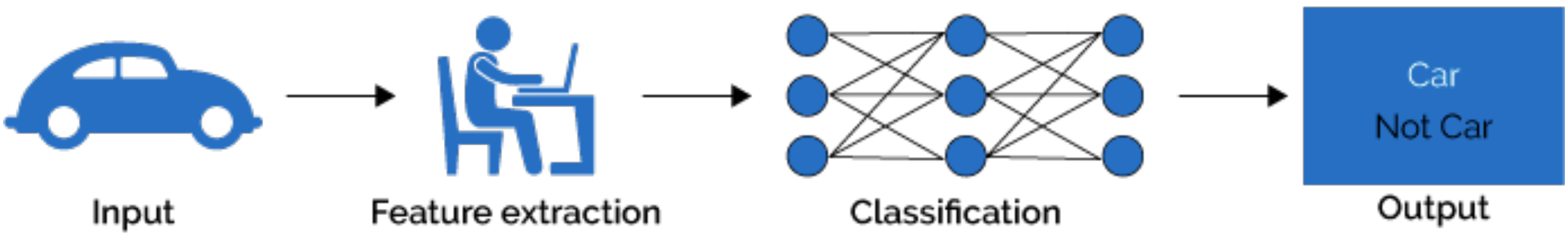
TRADITIONAL MACHINE LEARNING



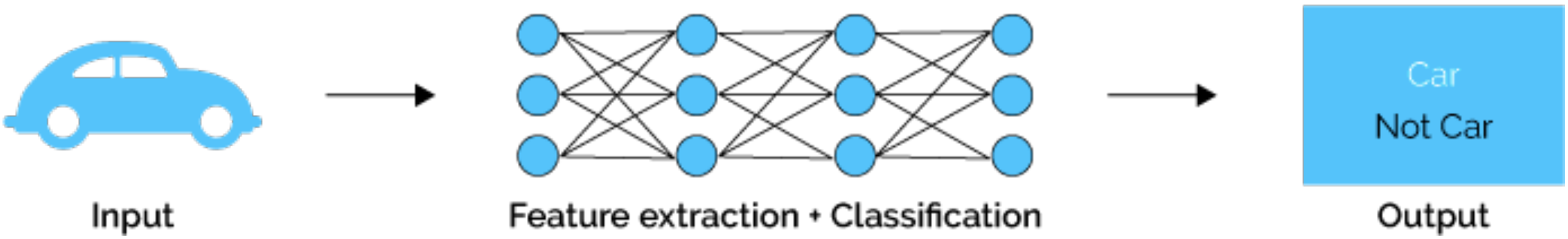
DEEP LEARNING

ILLUSTRATION: BUBAONE/ISTOCK

Machine Learning

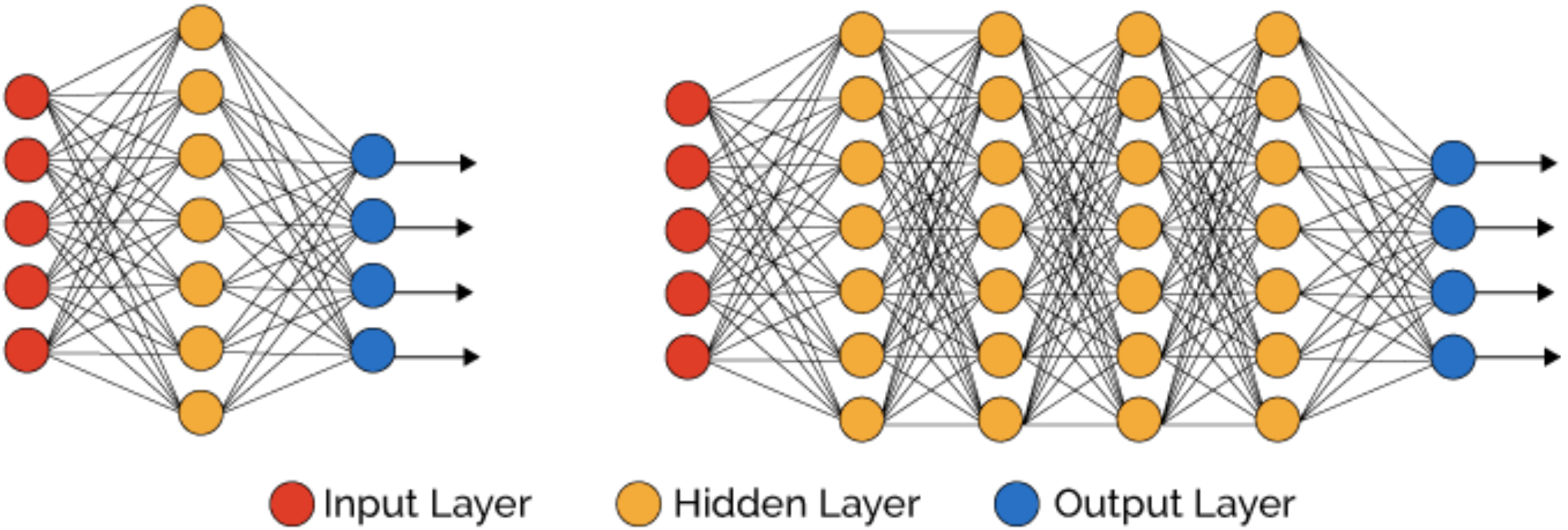


Deep Learning

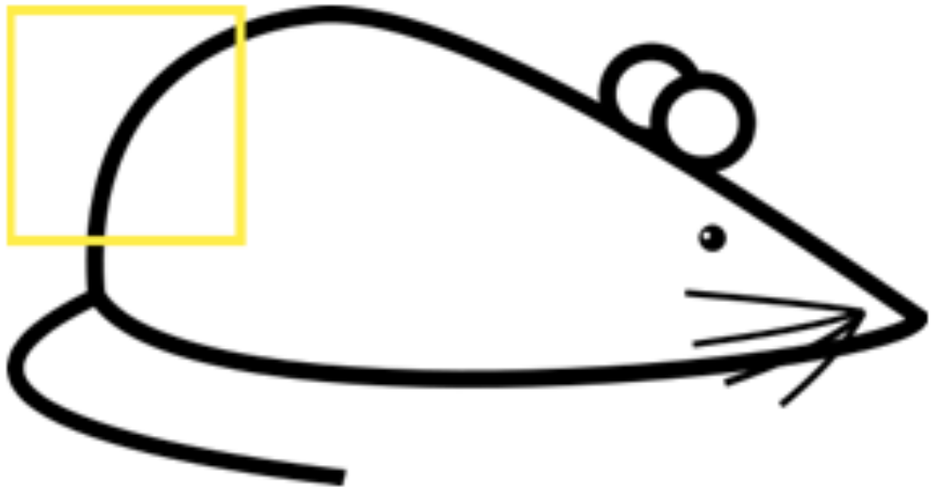


Simple Neural Network

Deep Learning Neural Network



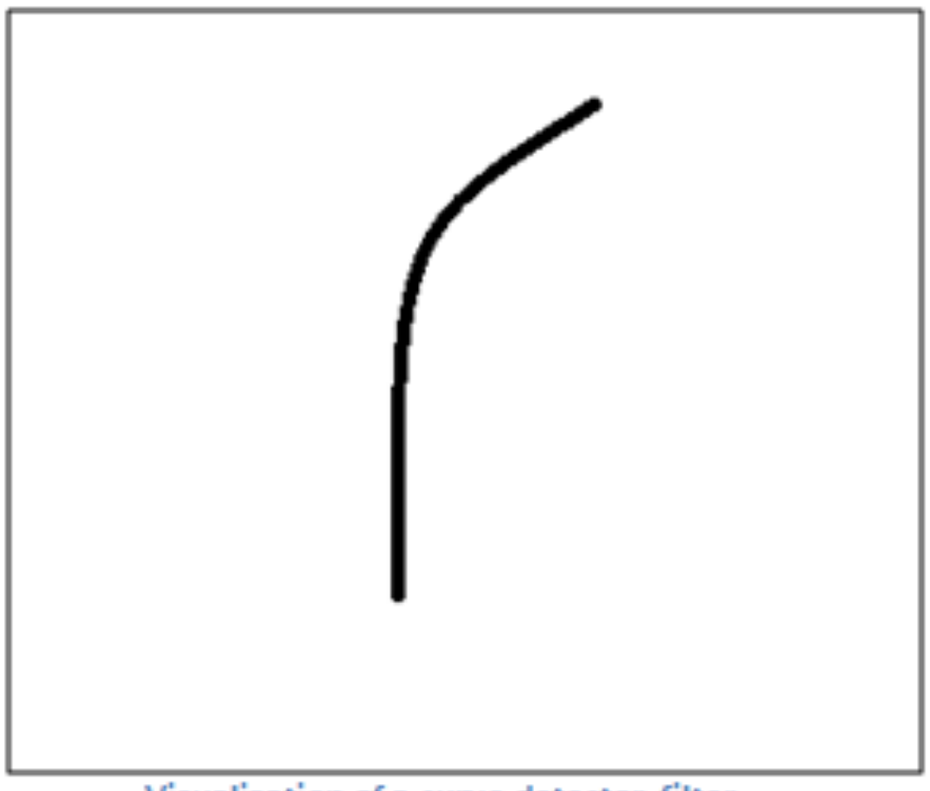
Original image



Visualization of the filter on the image

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

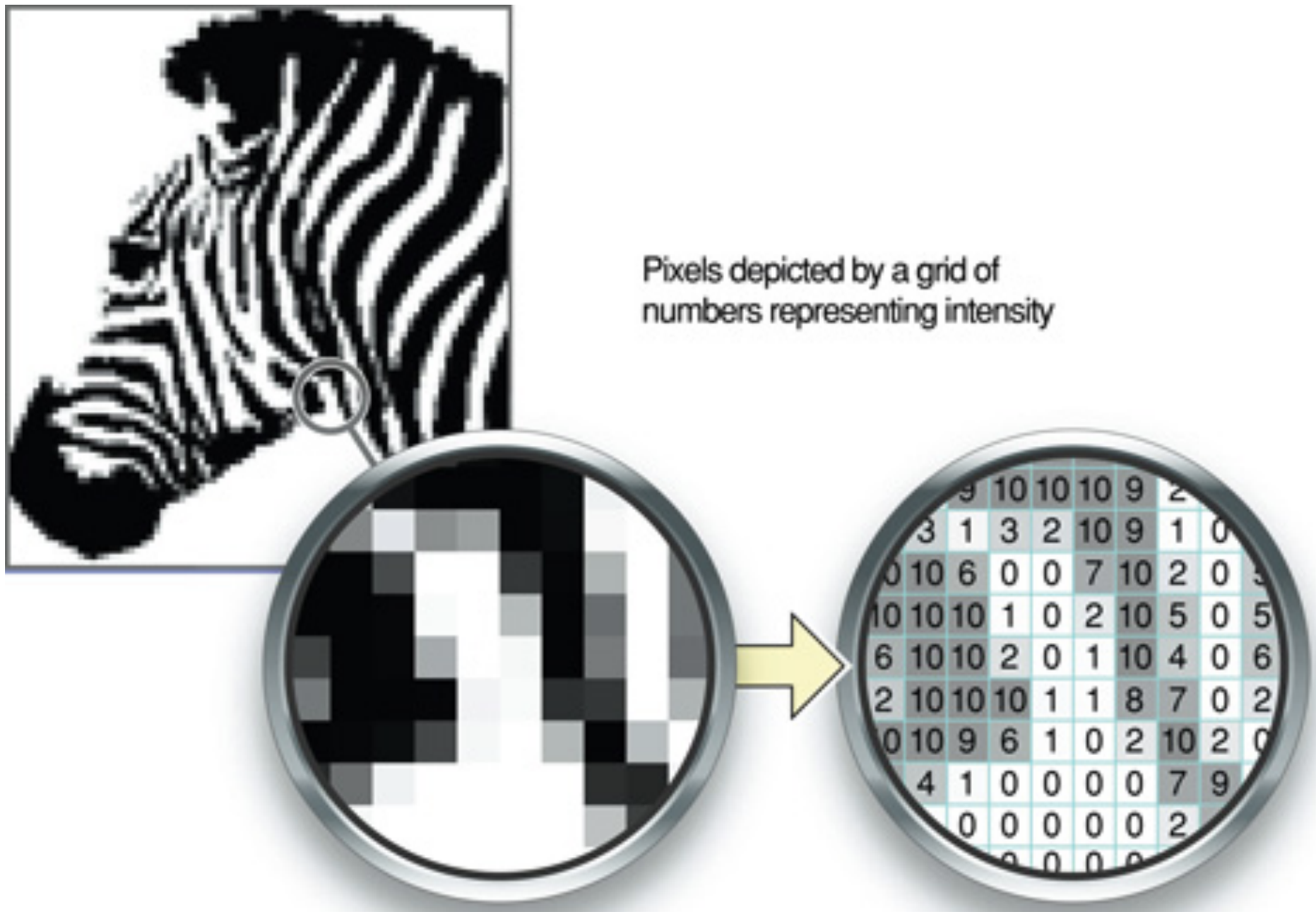
Pixel representation of receptive field

*

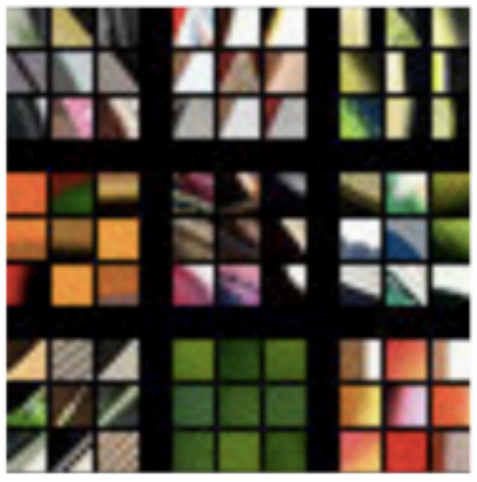
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

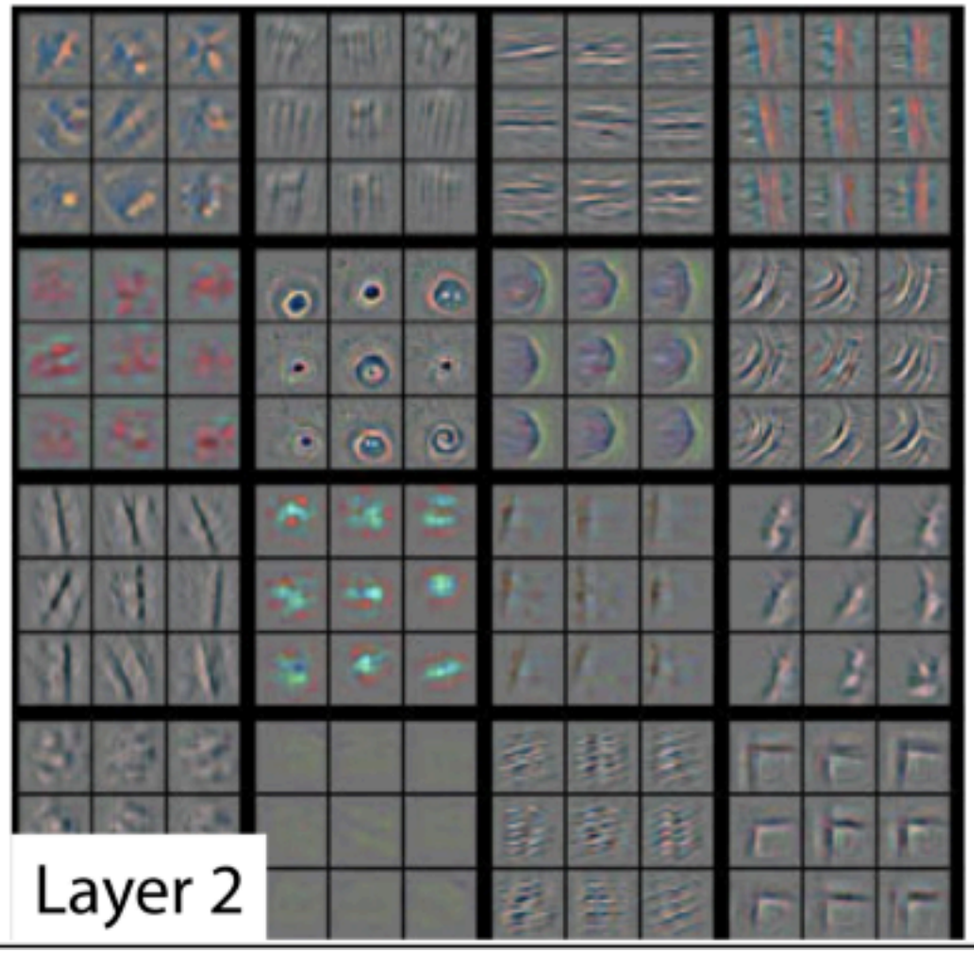
Pixel representation of filter

Multiplication and Summation = 0

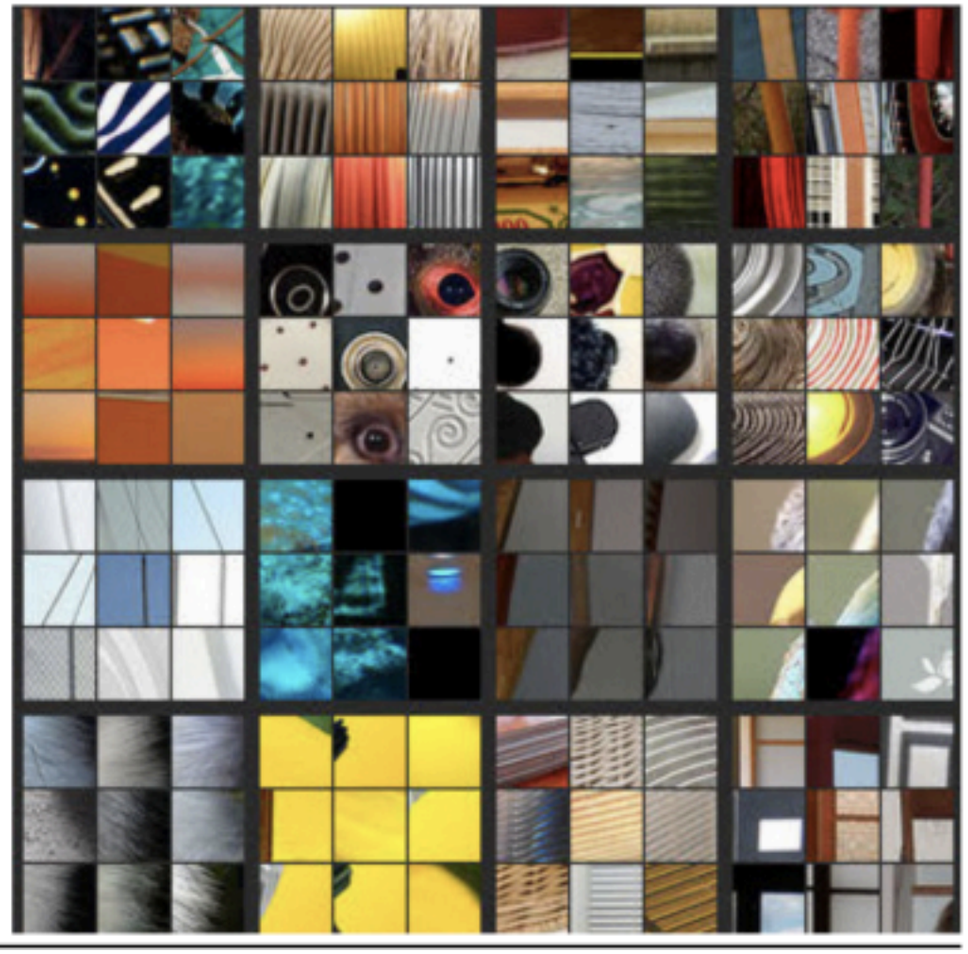

Pixels depicted by a grid of
numbers representing intensity
9 10 10 10 9 2
3 1 3 2 10 9 1 0
0 10 6 0 0 7 10 2 0
10 10 10 1 0 2 10 5 0 5
6 10 10 2 0 1 10 4 0 6
2 10 10 10 1 1 8 7 0 2
0 10 9 6 1 0 2 10 2 0
4 1 0 0 0 0 7 9
0 0 0 0 0 2

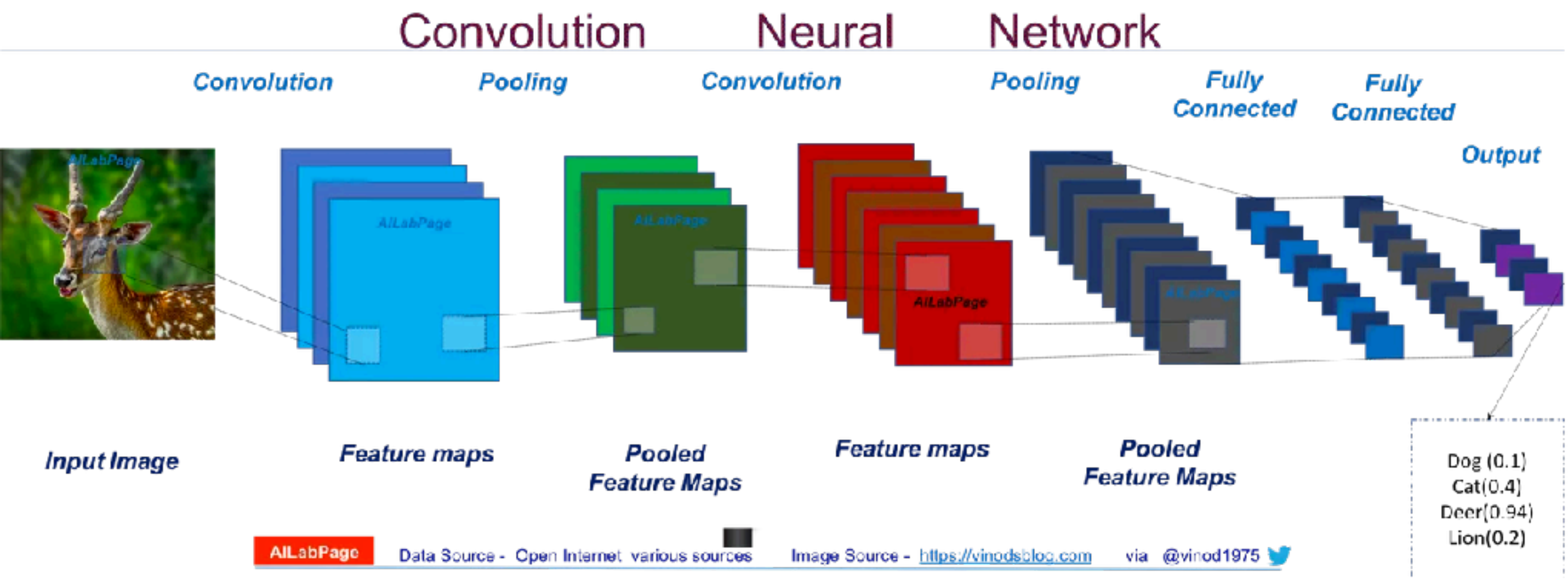
Layer 1

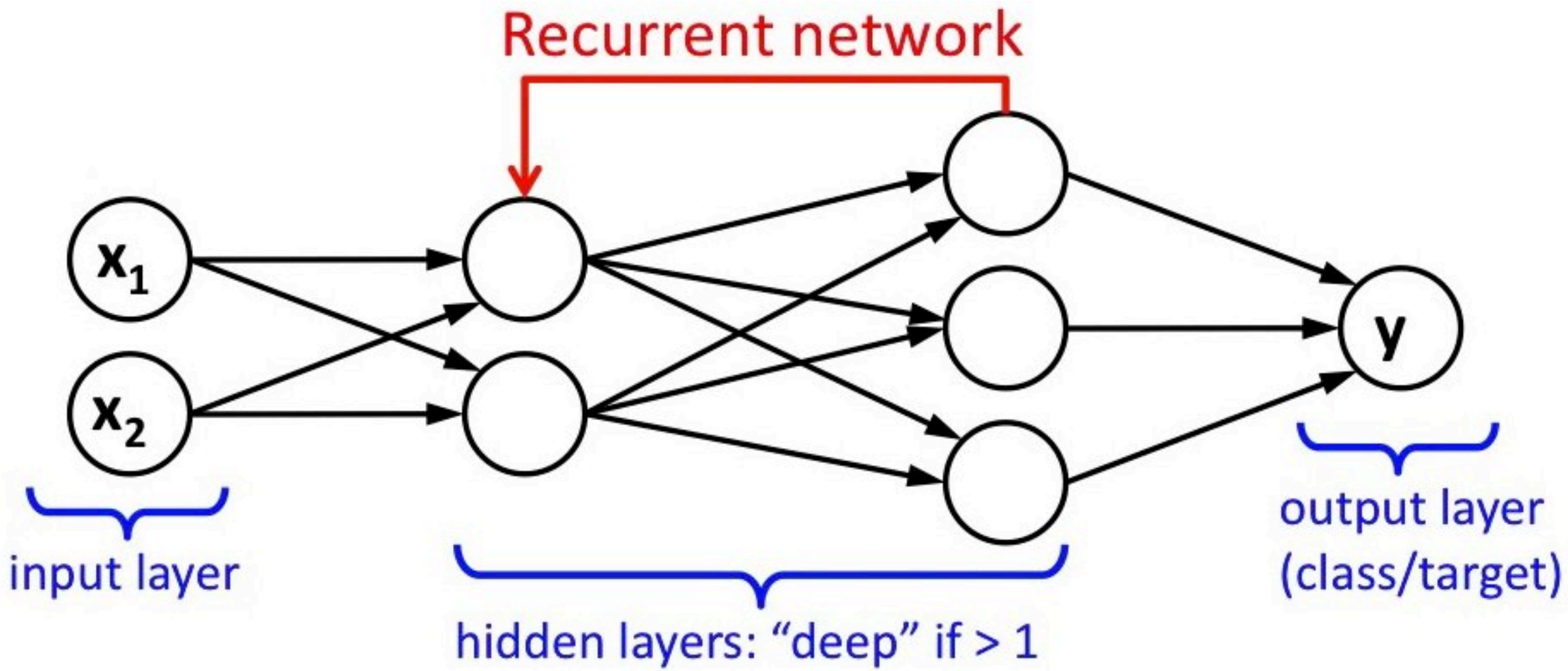




Layer 2




Convolution Neural Network
Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output
AILabPage
Input Image
Feature maps
Pooled Feature Maps
Feature maps
Pooled Feature Maps
Dog (0.1)
Cat(0.4)
Deer(0.94)
Lion(0.2)
AILabPage
Data Source - Open Internet various sources
Image Source - https://vinodsblog.com
via @vinod1975


Recurrent network
x_1
x_2
y
input layer
hidden layers: “deep” if > 1
output layer
(class/target)

Input

Stateful Model

Output

20ms slice of audio

Recurrent Neural Network

The model's current state influences the next calculation.

Likelihood saying 'A'

Likelihood saying 'B'

Likelihood saying 'C'

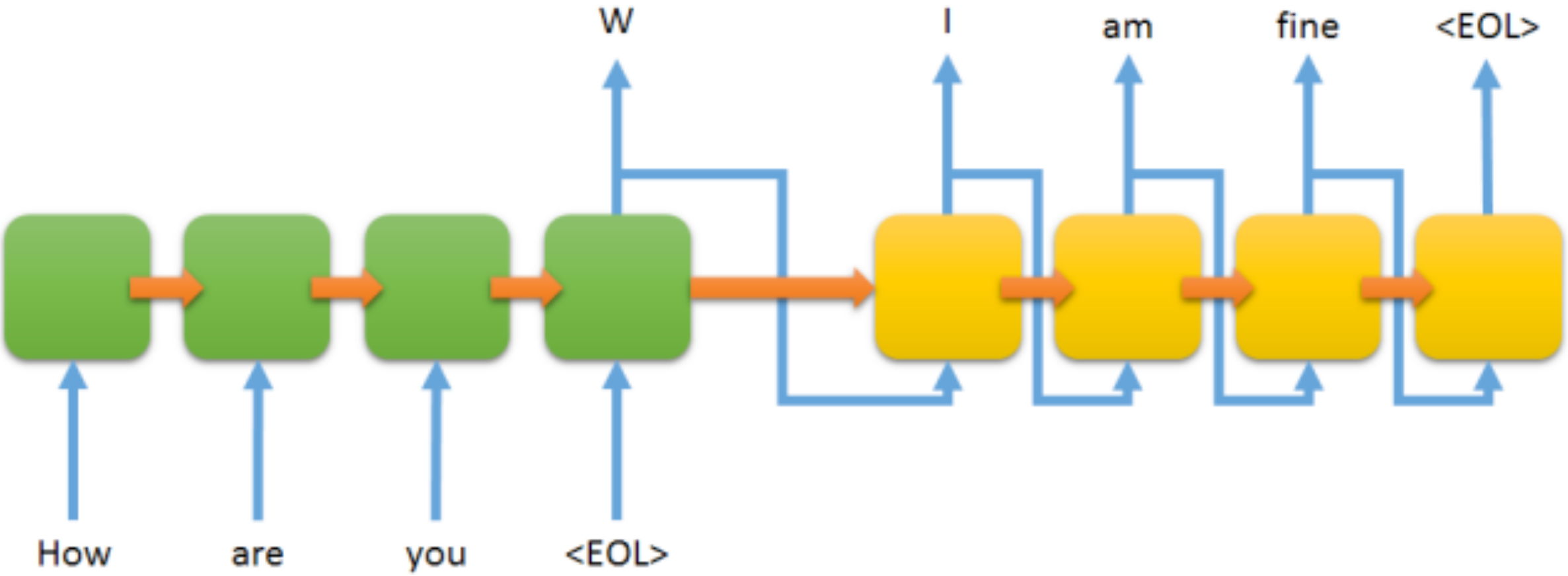
And so on...

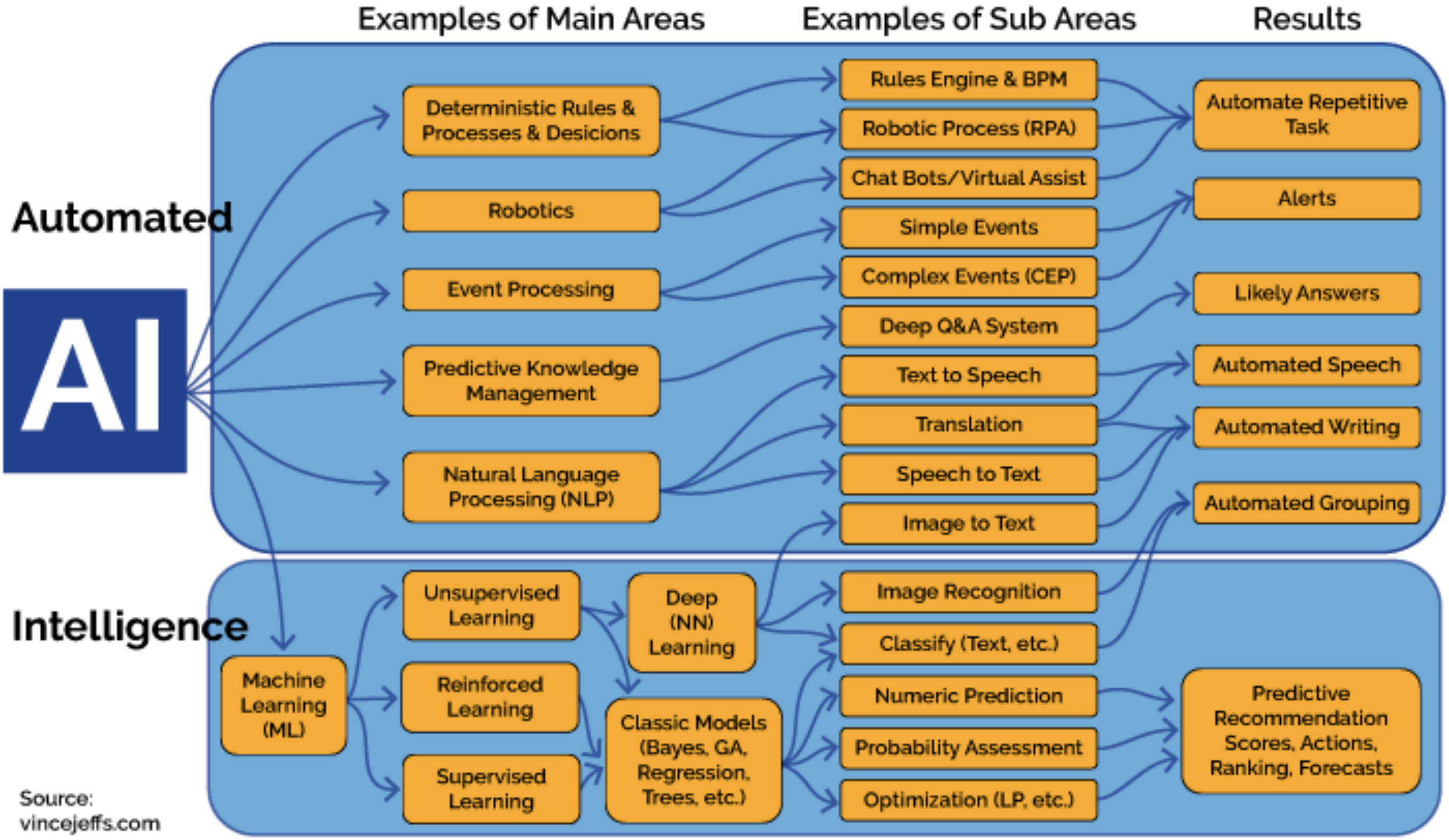
Image

RNN

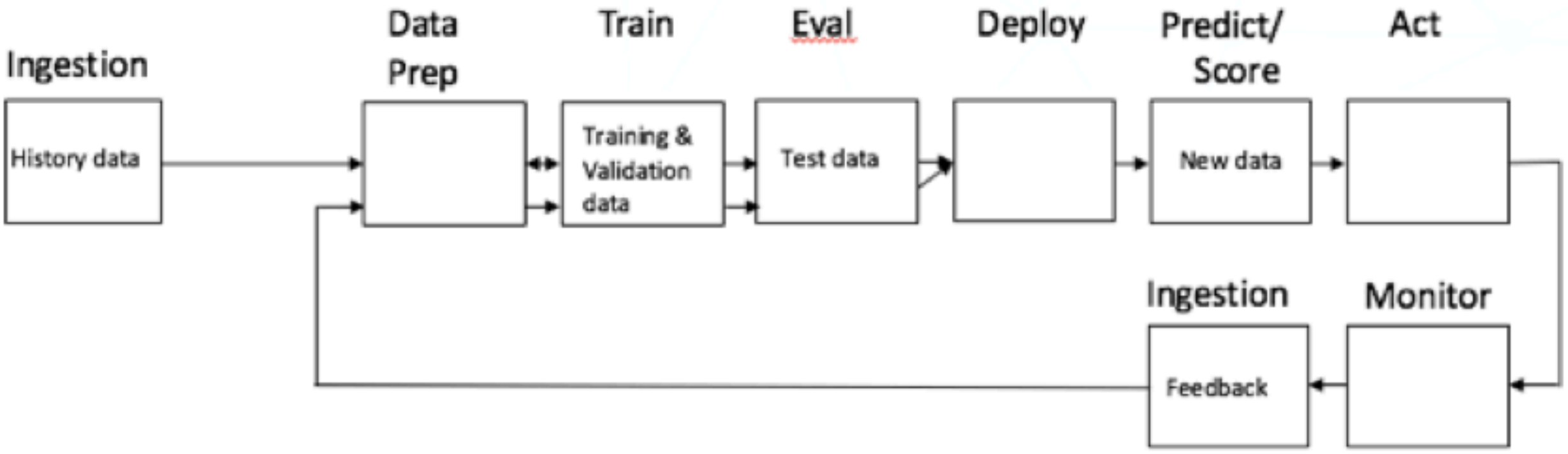
A person is sitting in a beach holding guitar.

Sequence of text

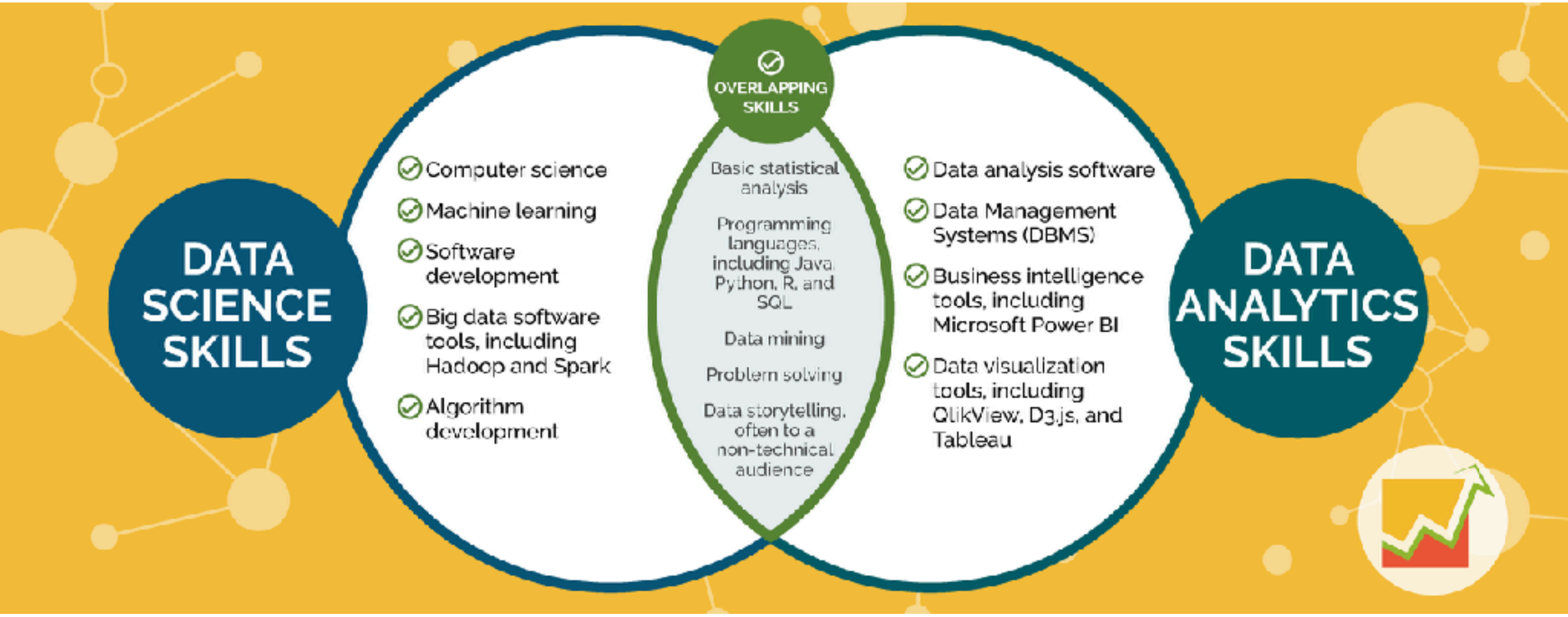

W
I
am
fine
<EOL>
How
are
you
<EOL>
LSTM Encoder
LSTM Decoder



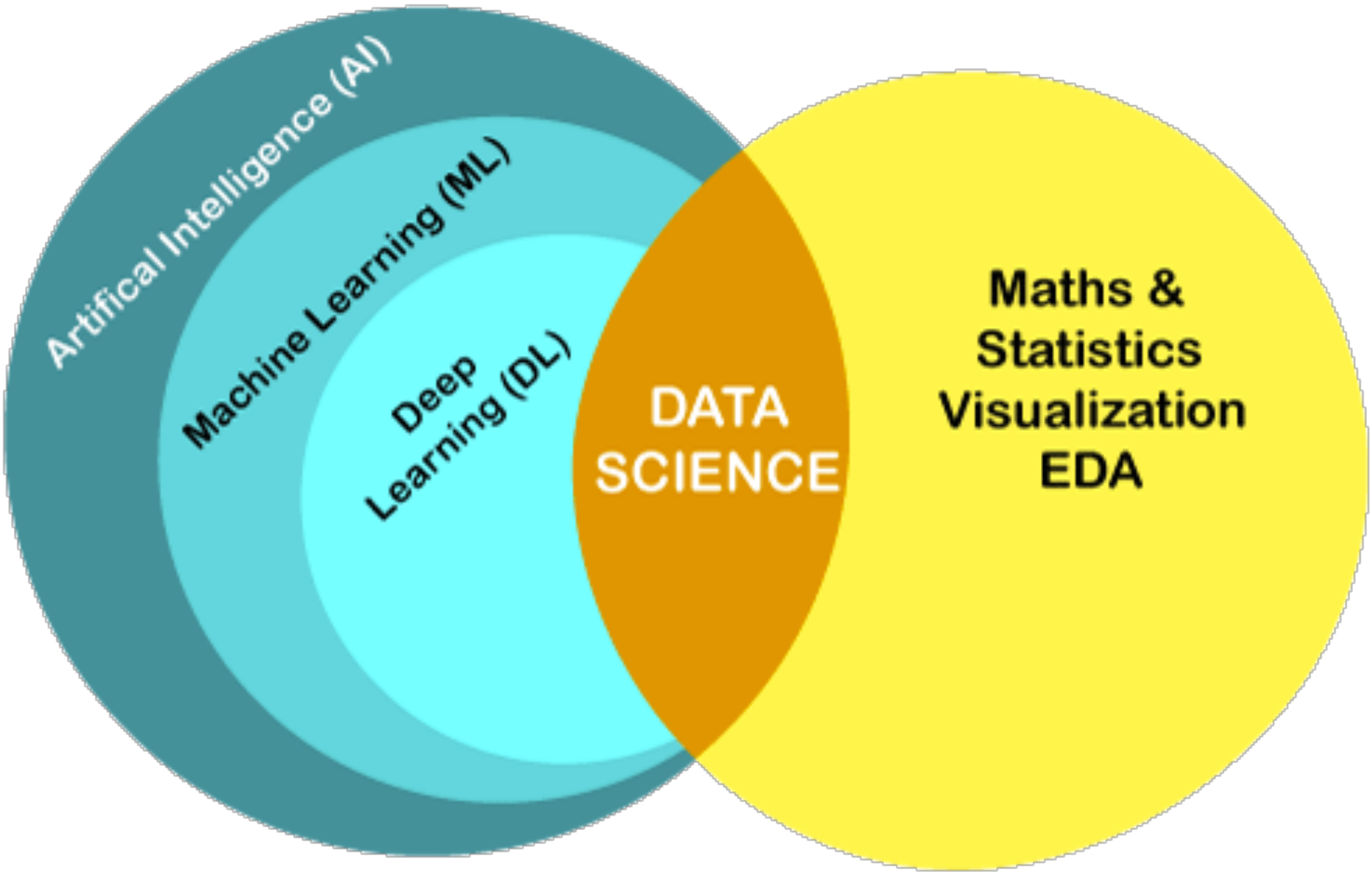
Source:
vincejeffs.com


Ingestion
Data Prep
Train
Eval
Deploy
Predict/ Score
Act
History data
Training & Validation data
Test data
New data
Ingestion
Monitor
Feedback

A Data Analyst is responsible for collecting and interpreting data.



. Data Scientist is in charge of making predictions to help businesses take accurate decisions.


Artifical Intelligence (AI)
Machine Learning (ML)
Deep Learning (DL)
DATA SCIENCE
Maths & Statistics Visualization EDA

Data Source Tier

Ingestion Tier

Operation Tier

Insight/Analytics Tier

Structured

unstructured

semi-structured

Real time

Azure Event Hub
AWS Kinesis
GCP Cloud pub/sub
Kafka

Batch

One Time

AWS Snowball,
Azure Data Box
Google Transfer Appliance

Azure Stream Analytics
Azure Kinesis Analytics
Storm
Spark Streaming

Machine Learning

Stream Processing

runs on the incoming data

AWS Elastic MapReduce (EMR)
AZURE HDInsight

Batch Processing

runs on the entire dataset

Storage Tier

Landing Zone

Raw

handling missing values,
filtering outliers,
& Validated data

Staging

Data Lake
AWS S3
Azure Data Lake

Highly refined views of the data

Curated

Analytical data store
Aws Redshift
Azure Synapse Analytics

BI Tools

Applications

Automations

QuickSight
Power BI

Orchestration Tier

Workflow

AWS Data Pipeline
Azure Data Factory
Apache Oozie and Sqoop

Portal

CRUD (rest)

Stream Analytics

Api Gateway

Create Order

Read (rest)

Order Created
No Stocks

Create Order

Create Order
Order Created
No Stocks

Event Service

Inventory

do() {... }
undo() {...}

Query Service

do() {... }
undo() {...}

Event db

full state

Big data Analytics

Archive

Data ware House

Snap Shots
(daily/ weekly / monthly)

WareHouse Analytics

Normalised db

write friendly
transaction friendly
current state

DeNormalised db

read friendly
reporting friendly