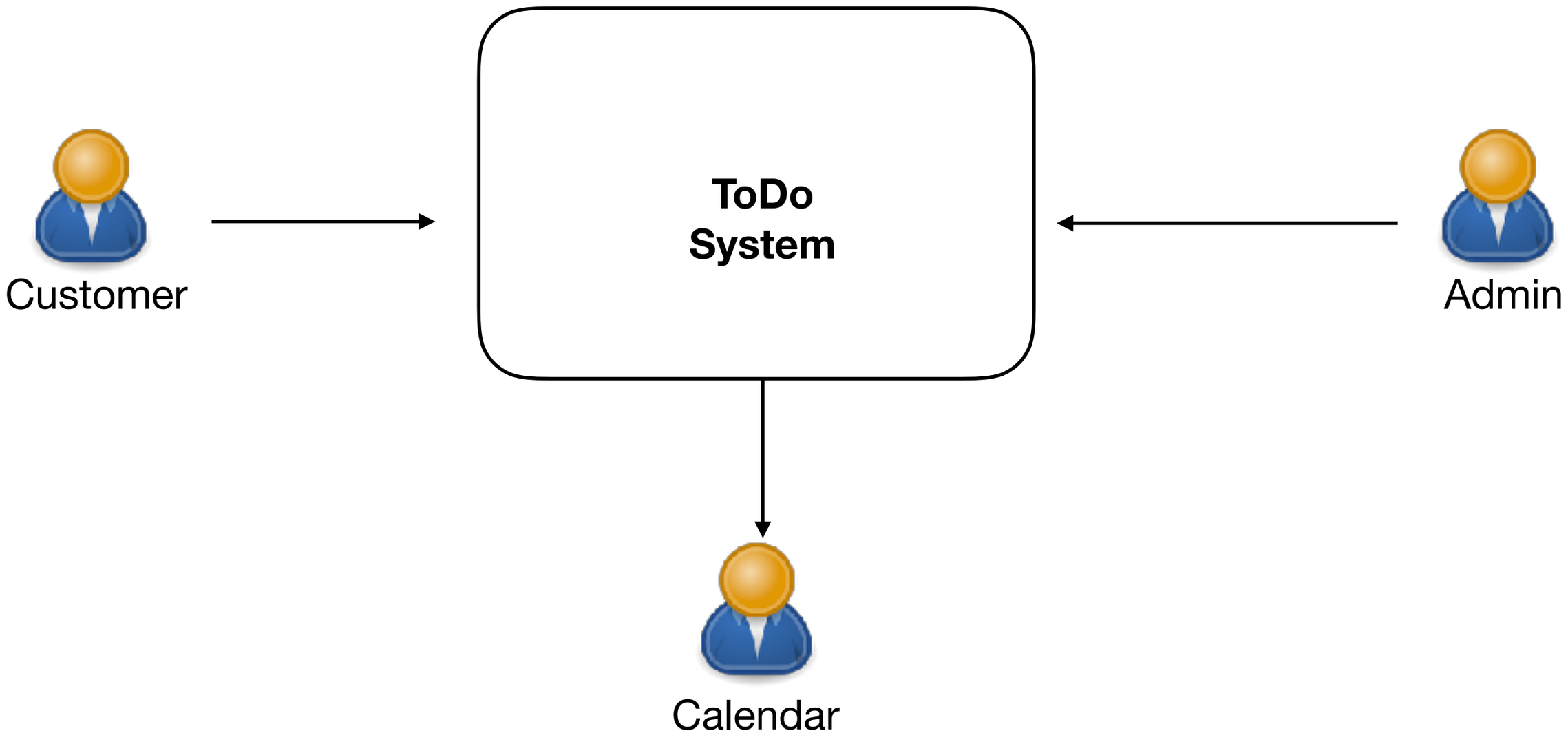
Software Architecture Views

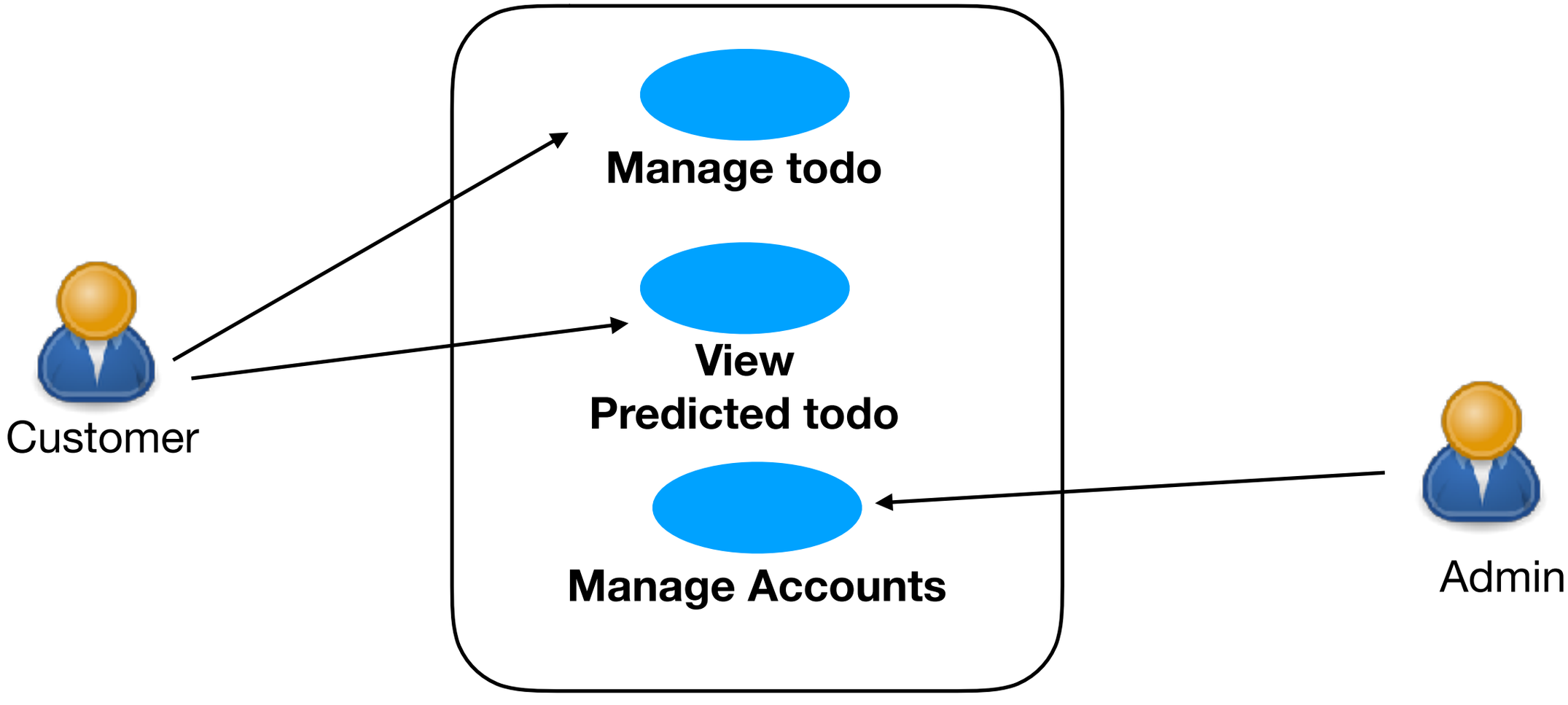
Part 1

Arch Req

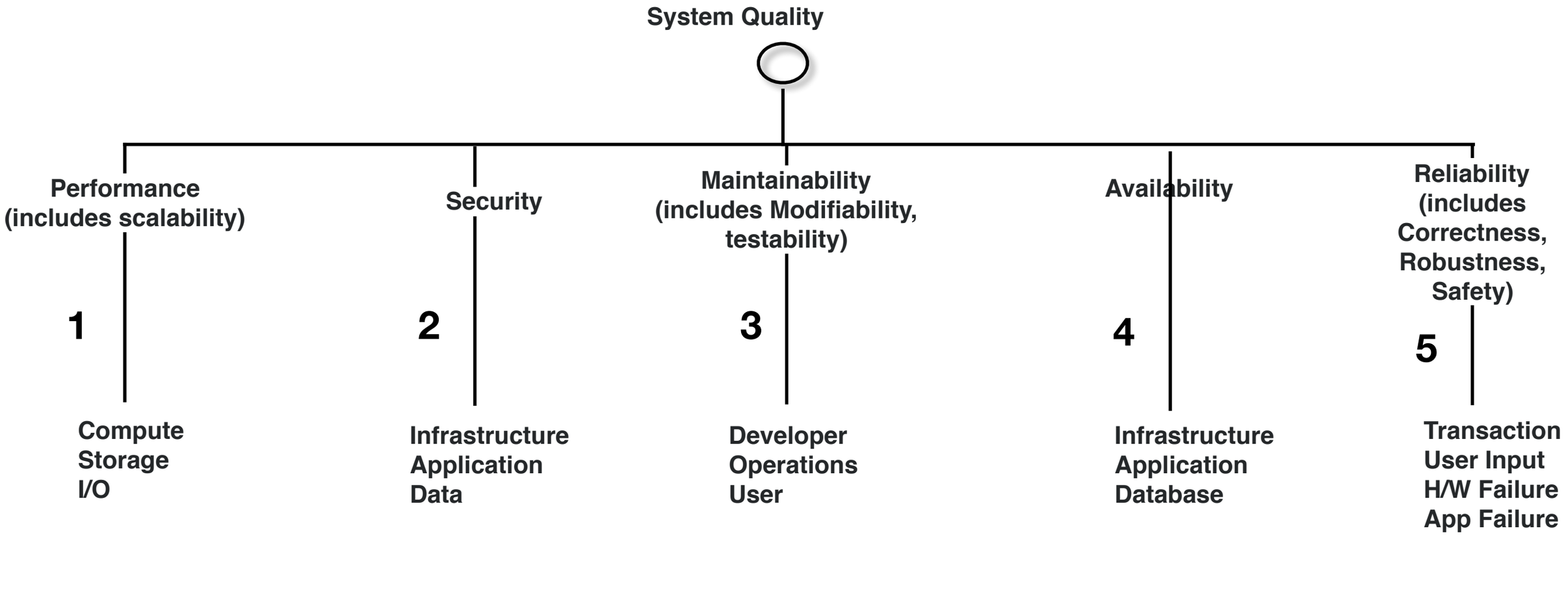
1. Context view



2. Functional view



3. Quality View



Usability
Portability
Interoperability

Quality Attribute Scenarios

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a user	I want to add a todo	In the portal	During peak load.	The todo is added	In < 3 secs
Reliability	As a user	I want to add a todo	From the portal	Duplicate request	Duplicate entry is detected	In 100% of cases
Maintainability	Developer	Change the Configuration engine	In the Frame work	During maintenance	The configuration mechanism is replaced	In < 1 week
Security	unknown identity	requests to add a new todo	In the portal	During Normal load.	block access to the data and record the access attempts	100% probability of detecting the attack, 100% probability of denying access
Availability	The processor	Failed	In the db server	During Operational Hours	Secondary is made Primary	In < 5 minutes

4. Constraints view

It should work on IE 11, Chrome and Firefox

Should work on windows, unix, Mac platforms

Should deploy on Azure Cloud

---

5. Assumptions View

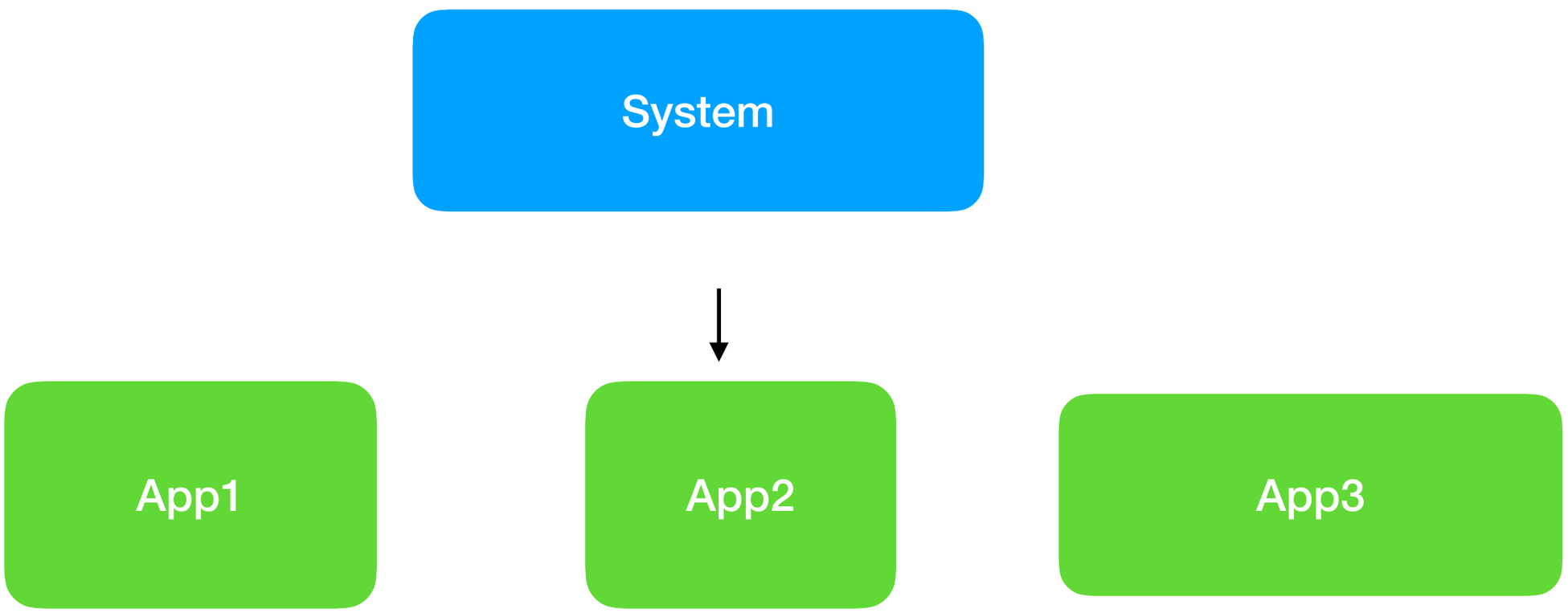
1	Will use open source Technologies only
2	
3	

Part 2

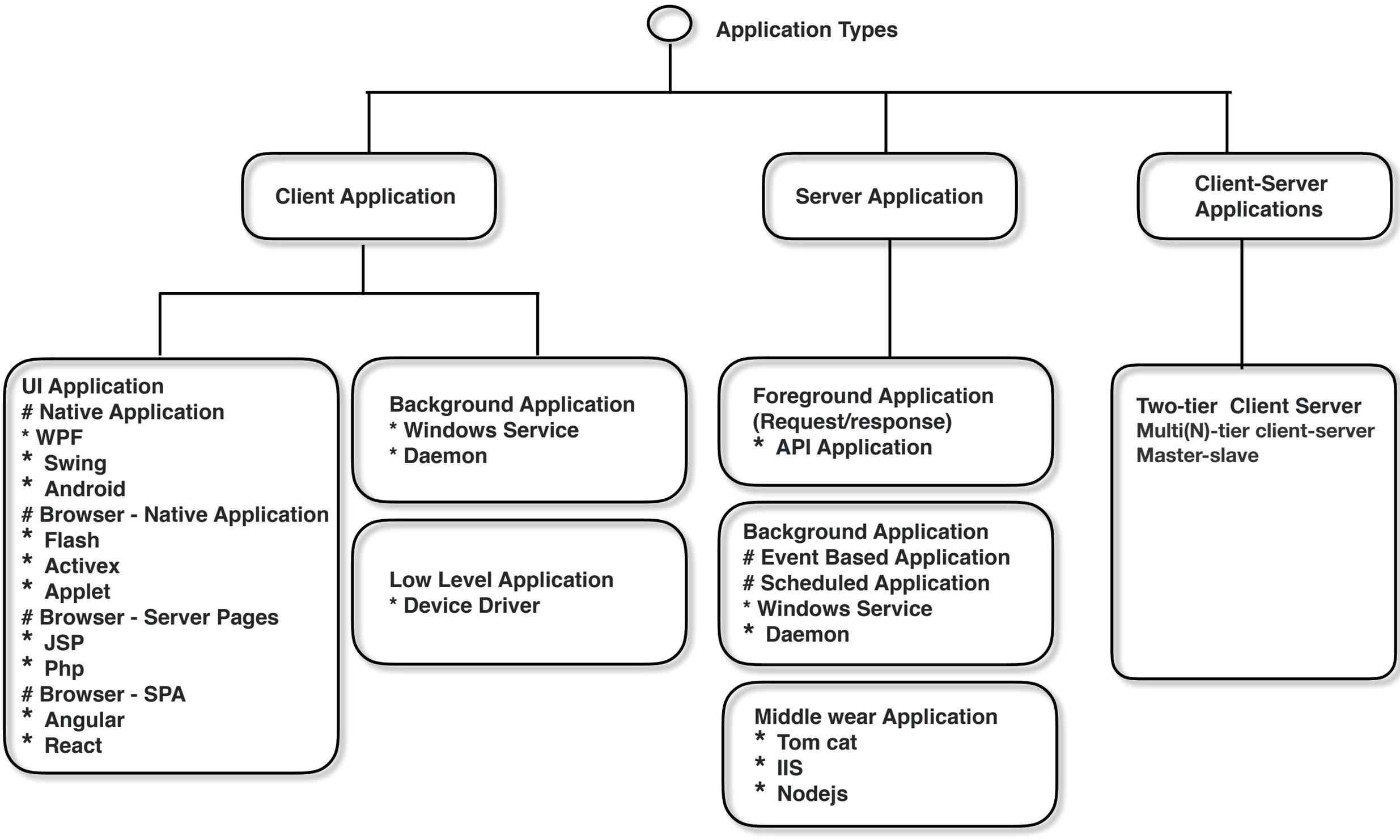
Arch Design

Decisions

6. Application View



Step 1. Decompose the system into Applications

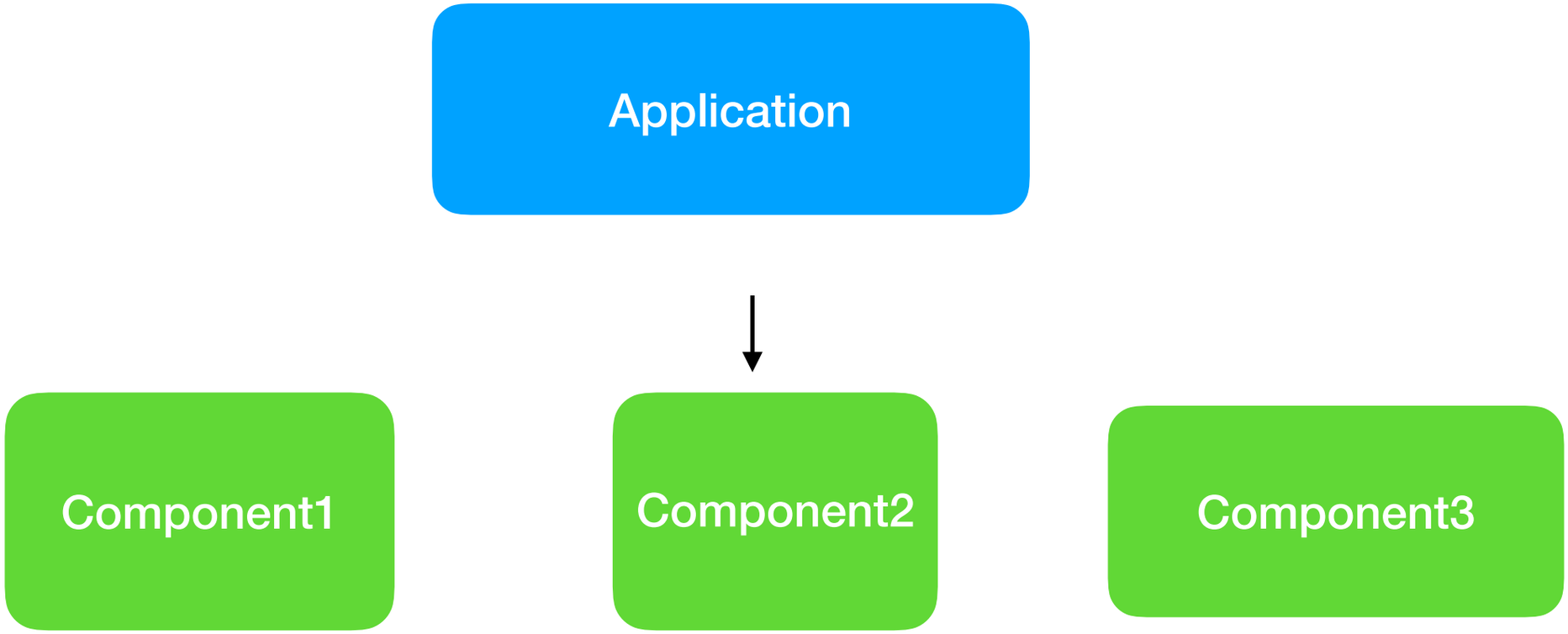


ToDo Portal
(Single Page Application)

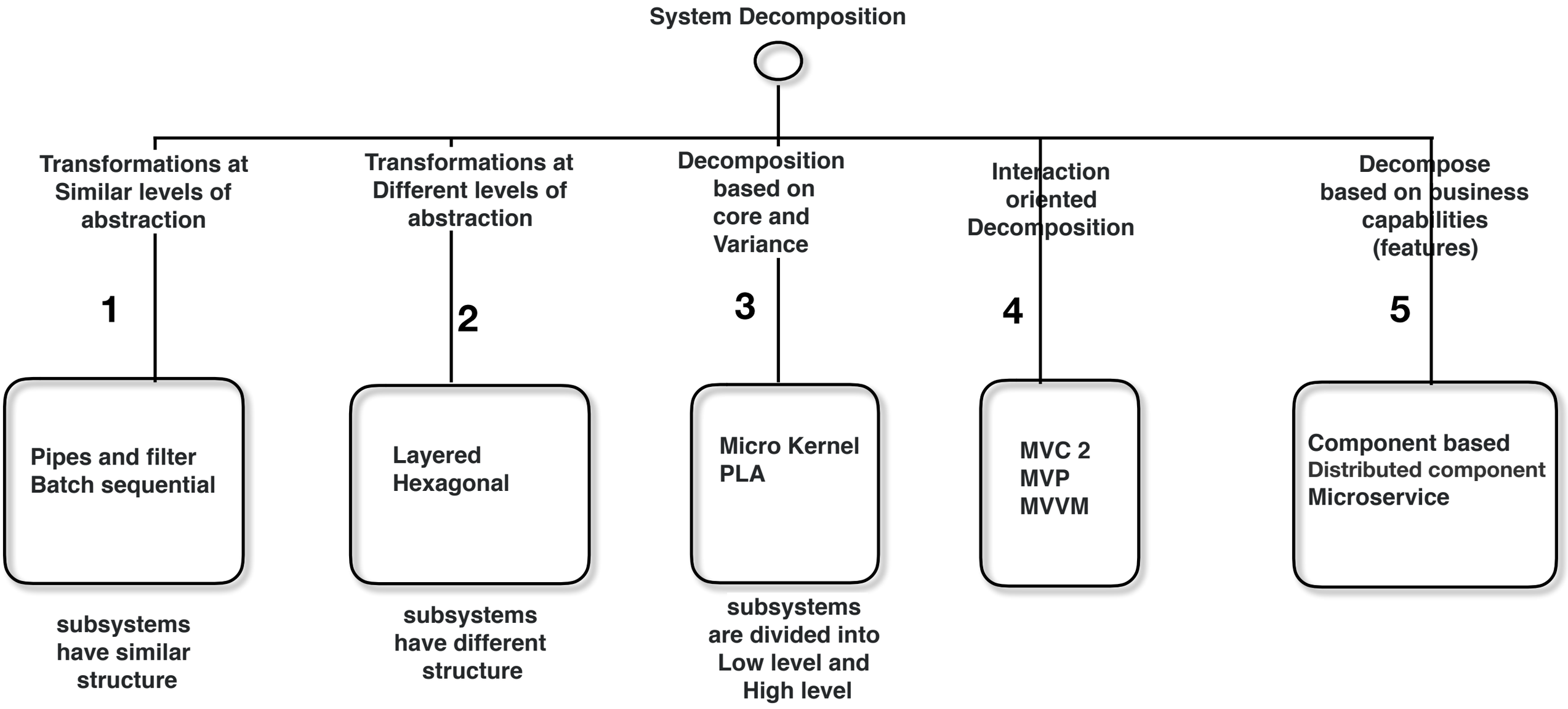
ToDo API Service
(Api Application)

ToDo Job Service
(Background Application)

7. Logical View



Step 1. Decompose application



ToDo Portal
(Single Page Application)

ToDo API Service
(Api Application)

ToDo Calendar Service
(Background Application)

ToDo Prediction Service
(Background Application)

validation,
formatting,
Routing,
State mgmt

View Layer

Controller Layer

Model

Technology

Logic

REST

Technology

Todo API
Service

Api Layer

Domain Layer

Message Adapter

Repository Layer

dto

Technology

Logic

Technology

Msg protocol (?)

Db protocol(?)

Message Queue

Calendar Sync
Service

Message Layer

Domain Layer

Calendar Adapter

dto

Data Access Layer

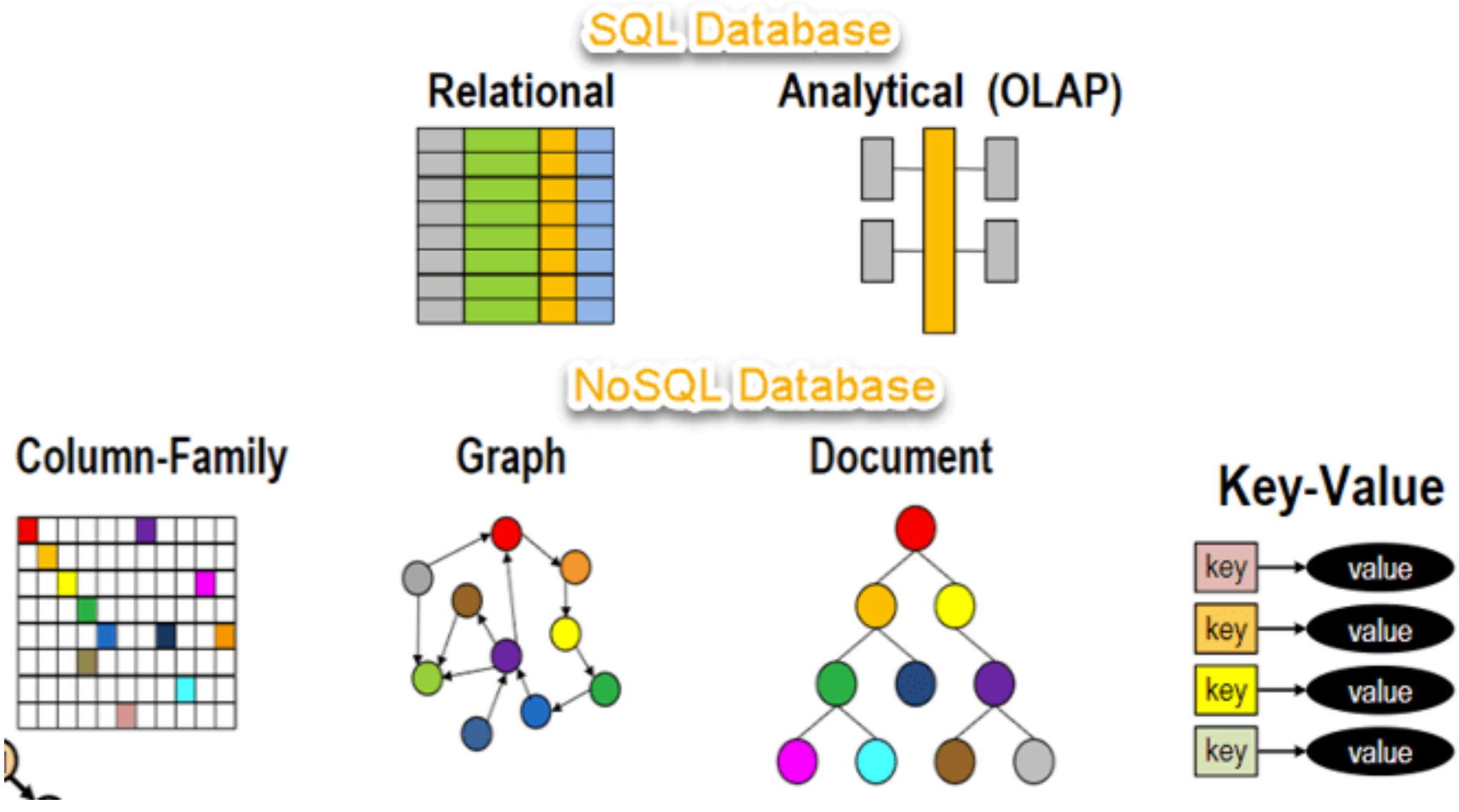
Domain Layer

Message Layer

dto

Todo Prediction
Service

8. Data View


SQL Database
Relational
Analytical (OLAP)
NoSQL Database
Column-Family
Graph
Document
Key-Value
key
value
key
value
key
value
key
value

Step 1. Choose App Persistence Mechanism

Lob Application Storage

< 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

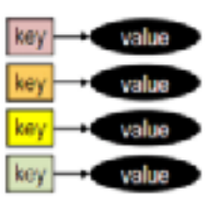
Simple
structure

Persistent

Key Value Store

Transient

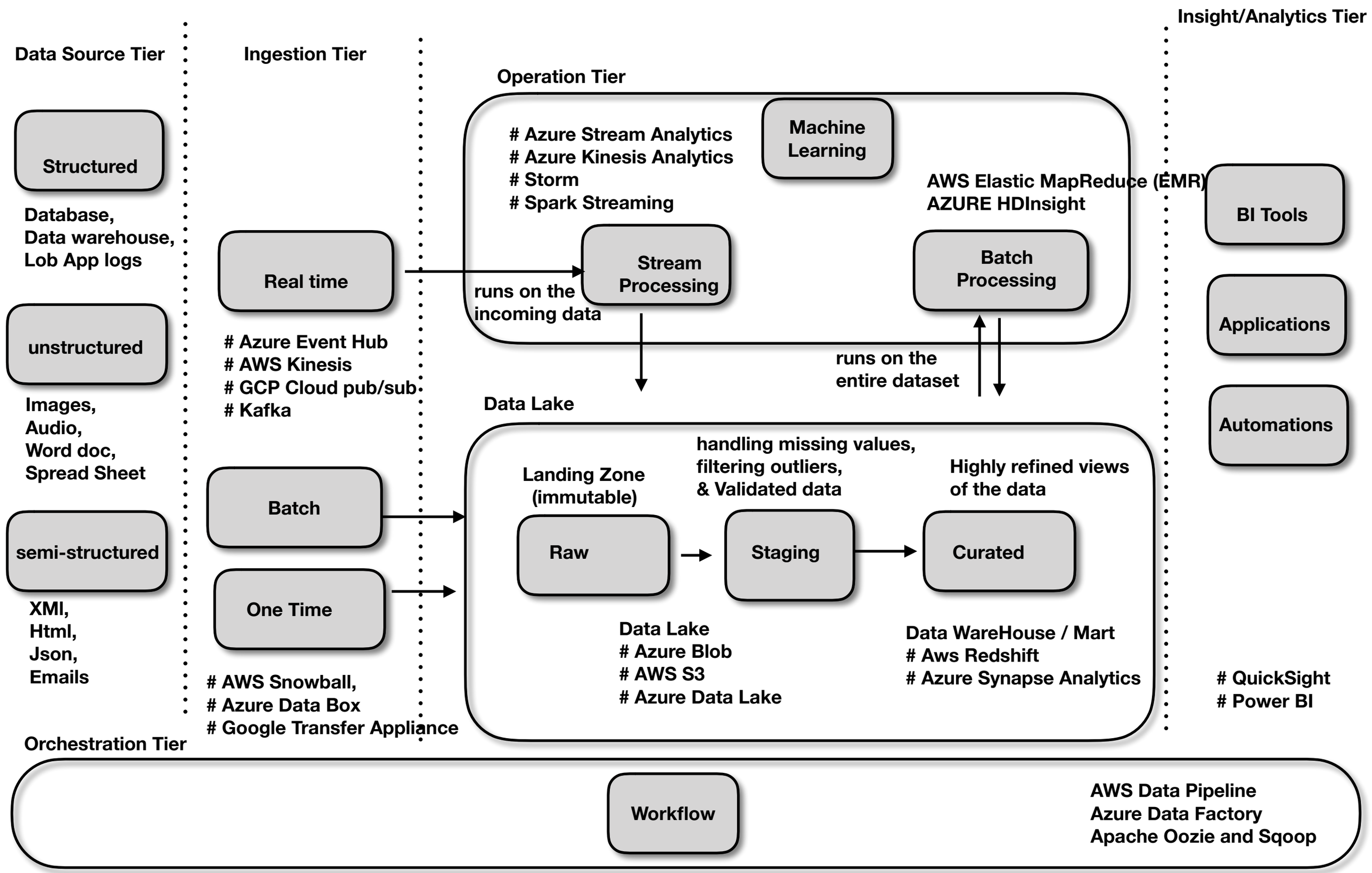
Key Value Cache



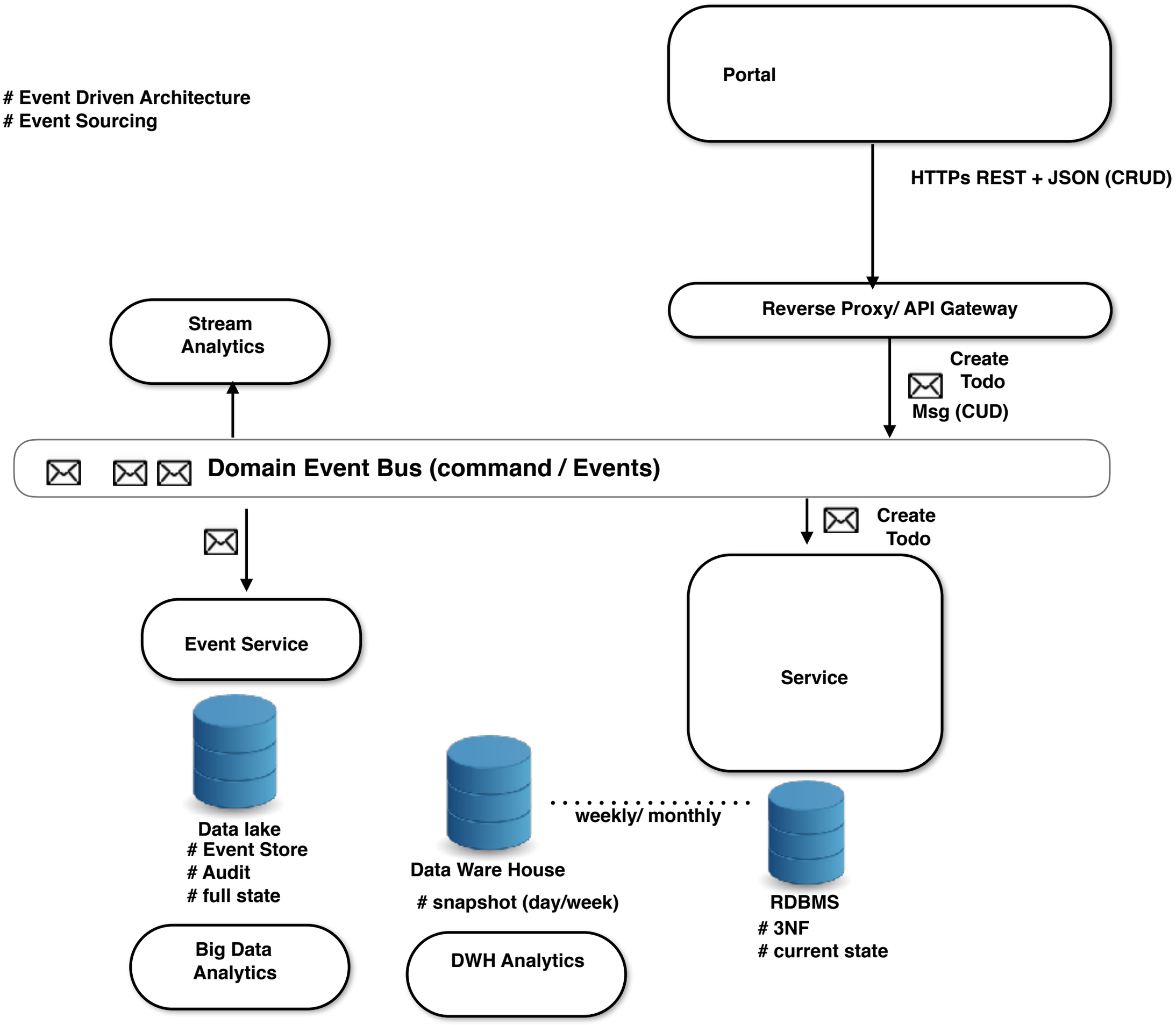
ToDo Api
(Api Application)

Document Db

Step 2. Choose Analytical Persistence Mechanism



Event Driven Architecture
Event Sourcing



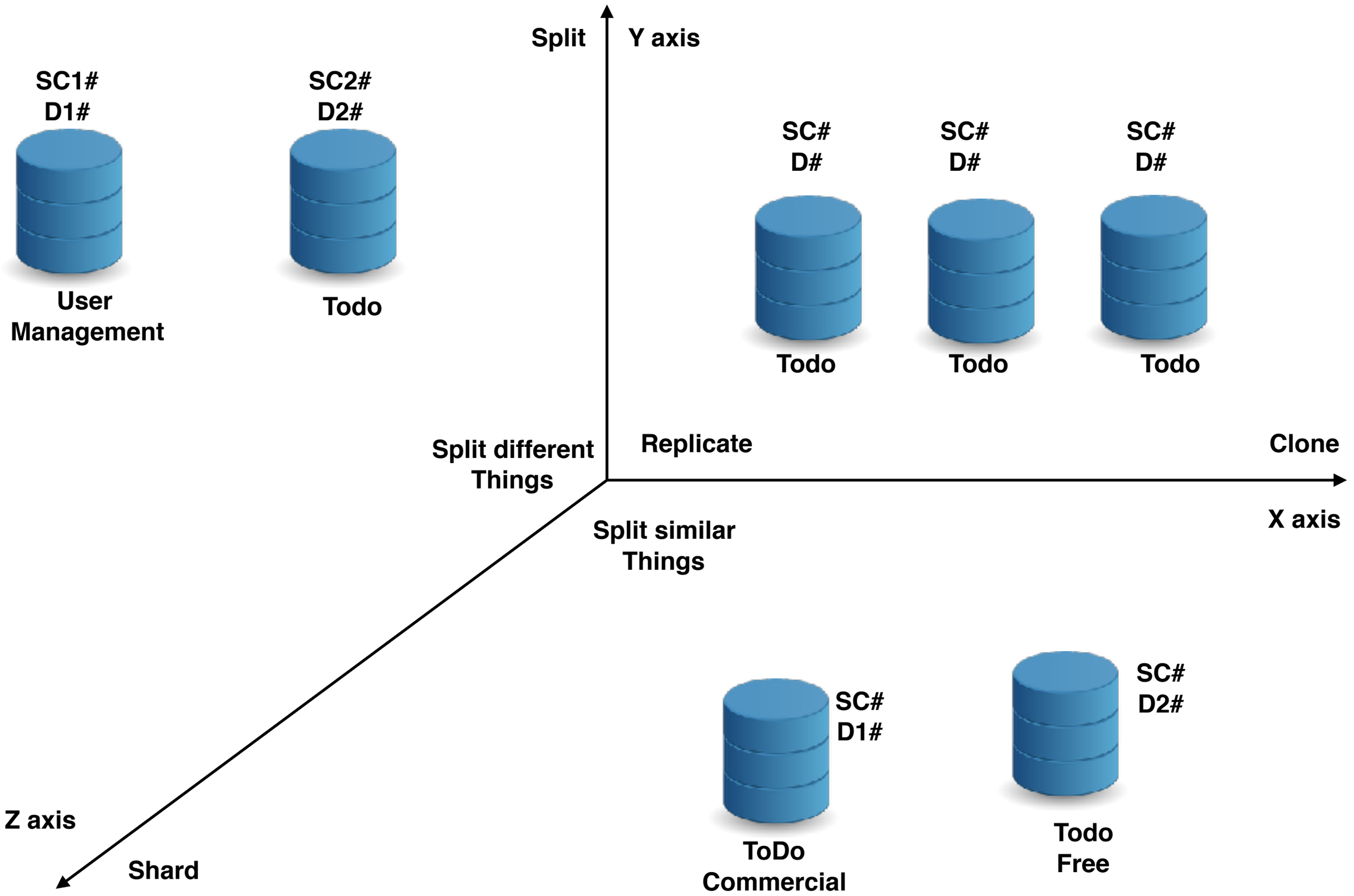
Data Lake

ToDo Prediction Service

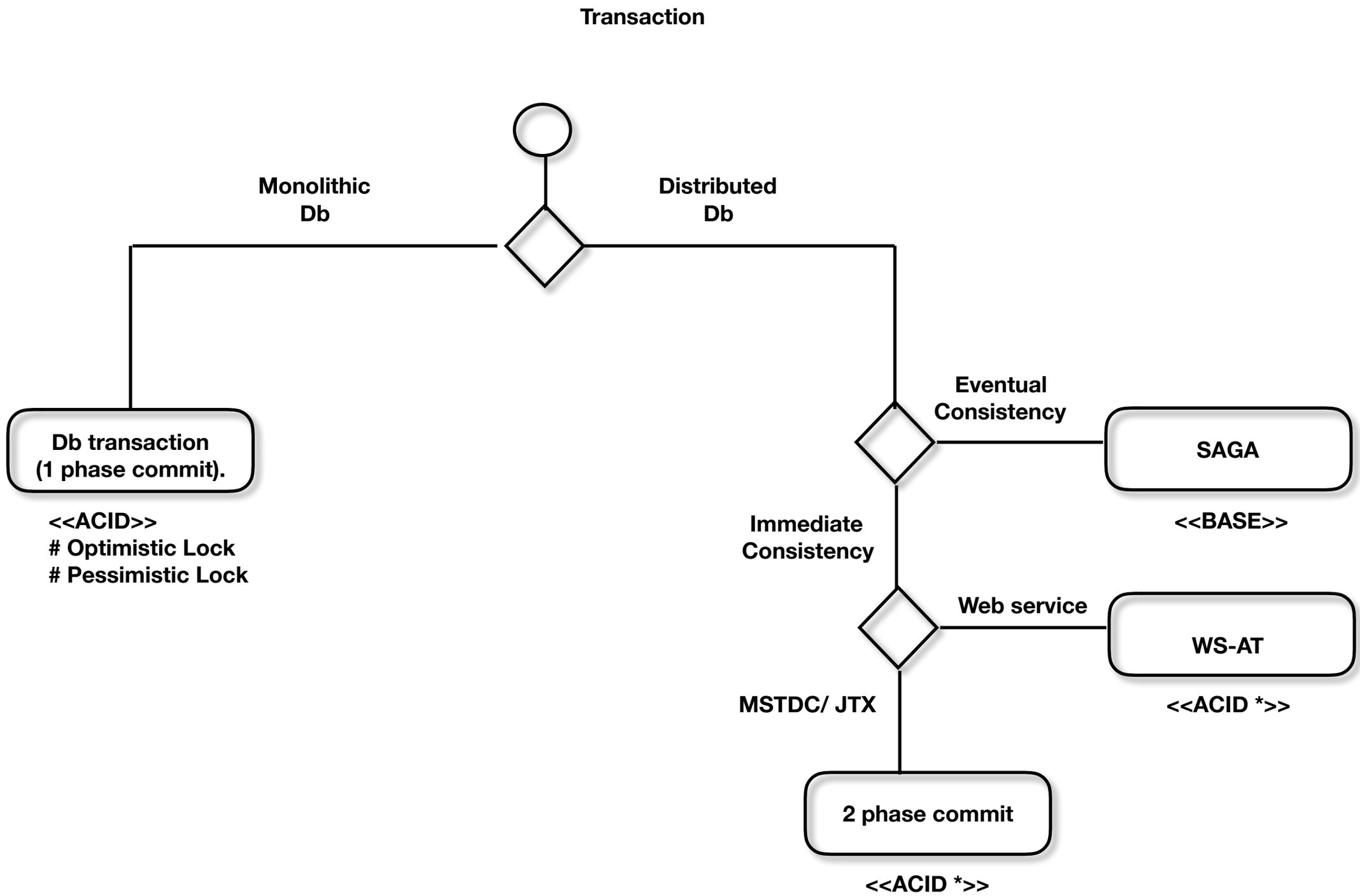
Step 3. Size the storage

Use Case	**Maximum Response Time**	Start of Window	End Of Window	Transactions/ Hour	List of **Database Activity (CRUD)**	Size of Storage Per transaction

Scalability Cube - 50 rules for high Scalability

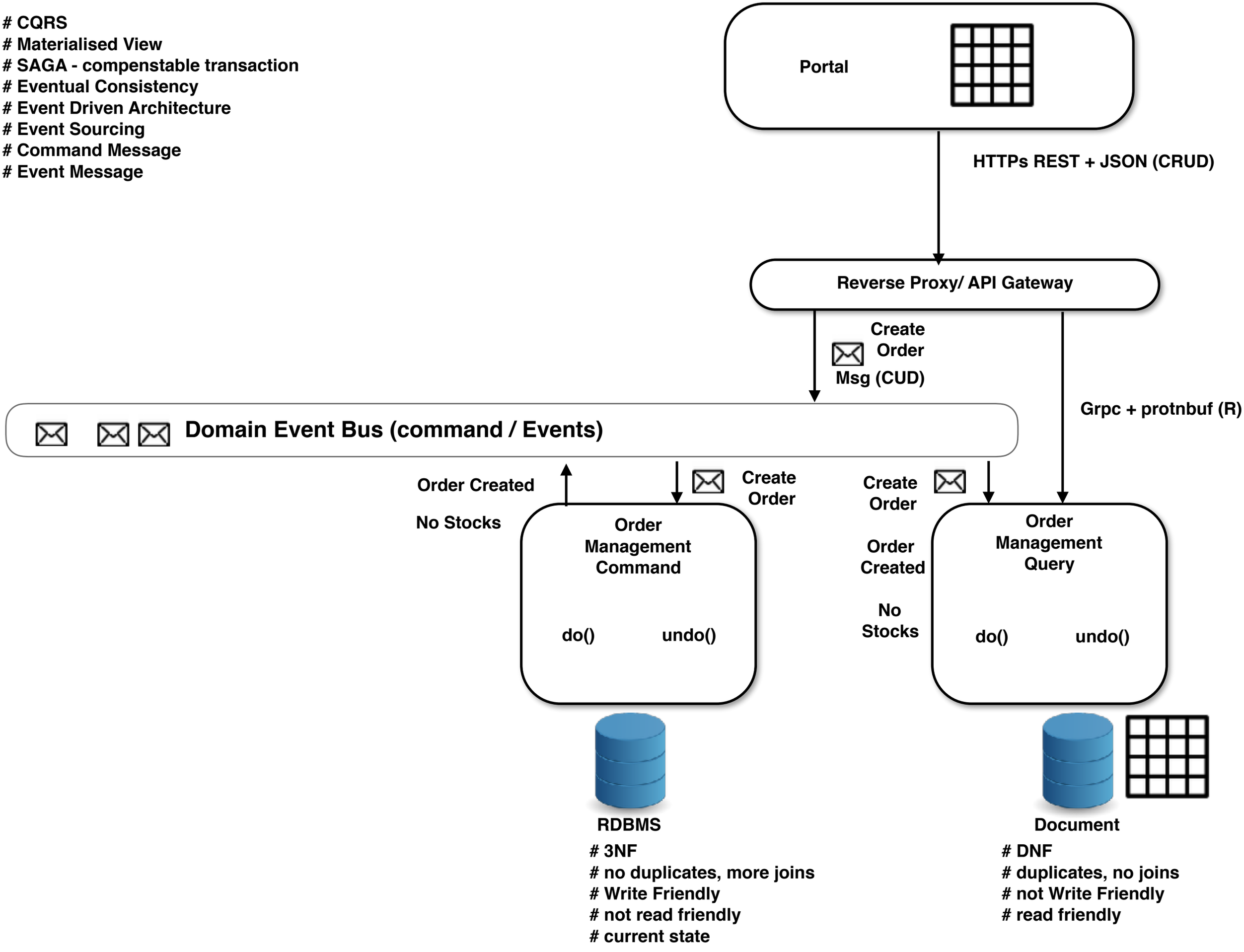


Step 4. Transaction and Concurrency



* 2PC **is not resilient to all possible failure configurations**.

CQRS
Materialised View
SAGA - compenstable transaction
Eventual Consistency
Event Driven Architecture
Event Sourcing
Command Message
Event Message



All transactions are encapsulated within a call from the BusinessLayer to a stored procedure. Any SQL errors within a stored procedure force a rollback and the error is raised to the BusinessLayer. Account data integrity is maintained through optimistic concurrency. A unique version timestamp is returned with each *head* Account object. During an update, the saveAccount stored procedure verifies the timestamp of the modified account is the same as the one in the database. If not, another user had previously updated the Account, so the update is rejected. The user is prompted to refresh a current version of the account before making edits.

```
    if @version <> (select version from Account where account_id =
@AccountPK)
    begin
      rollback transaction
      raiserror (60001, 11, 1, 'Account')
    end
```

Step 5. Add Metadata

Data Language
Data Timezone
Data Version
Tenant
Created By
Created Date
Last Modified Date
…

To do
Portal

HTTPs REST + JSON (CRUD)

Calendar Sync
Service

Reverse Proxy/ API Gateway

Create
Todo
Msg (CUD)

Grpc + protnbuf (R)

Domain Event Bus (command / Events)

Create
Todo

ToDo
Prediction Service

Event Service

Todo
Service

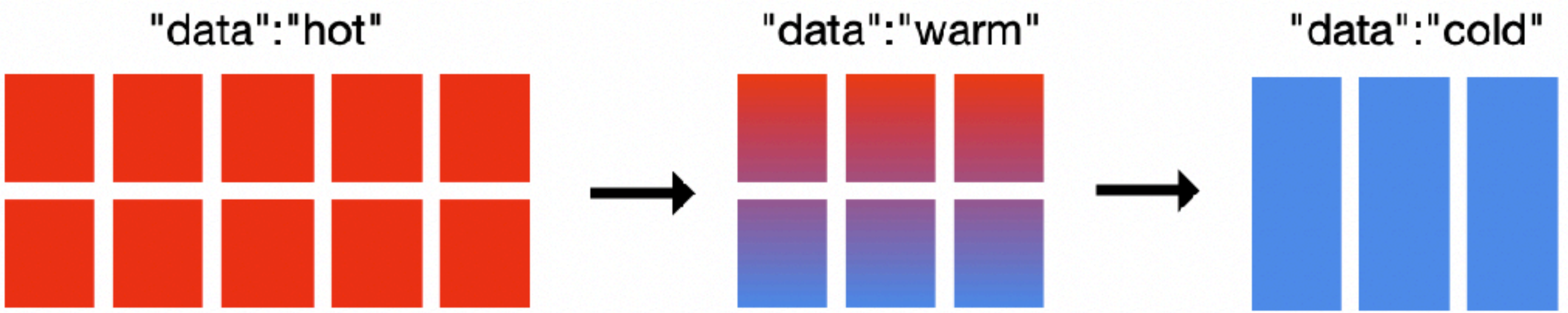
Data Lake

Event Store

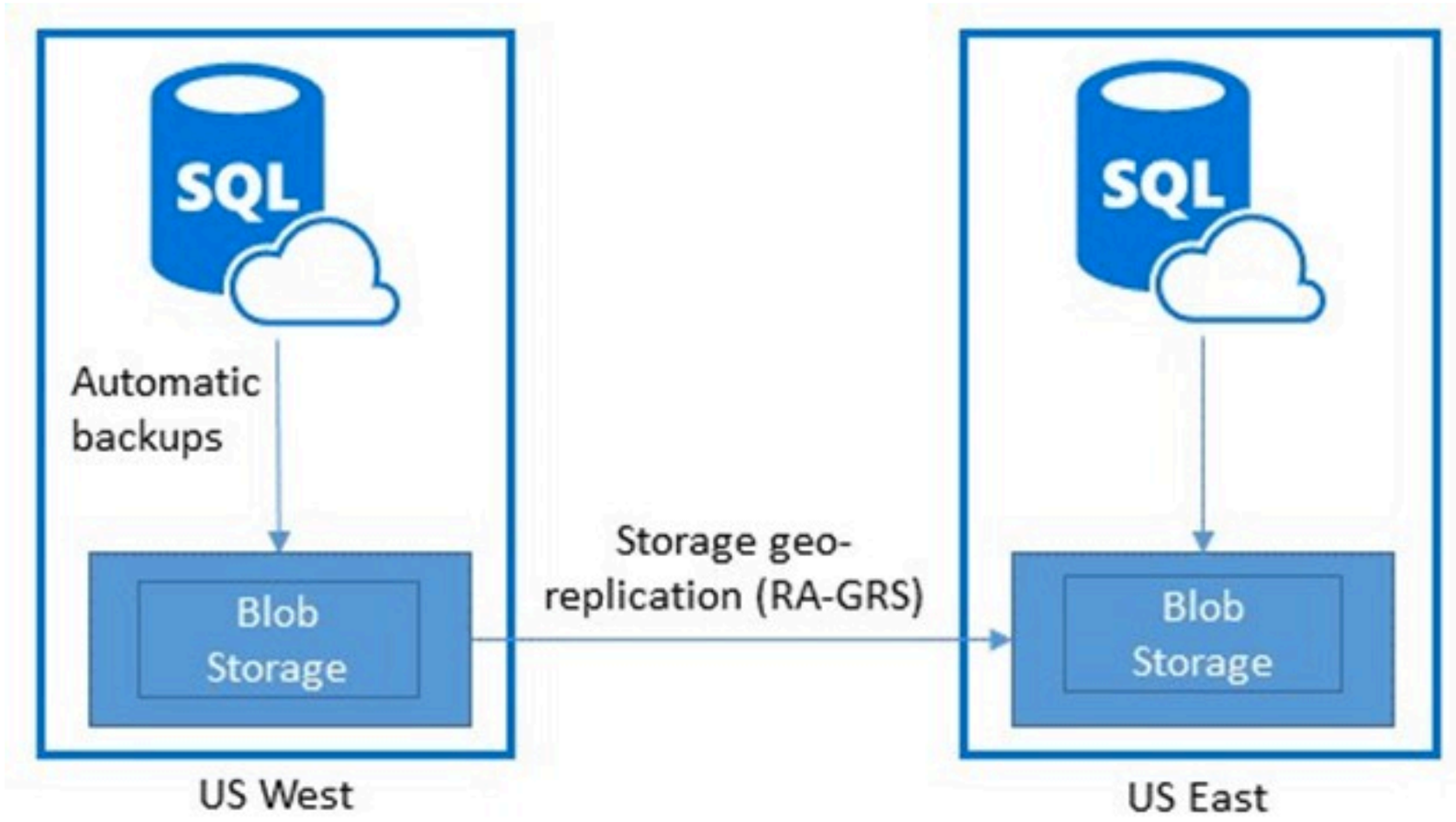
RDBMS

Lob Data

Step 6. Archive



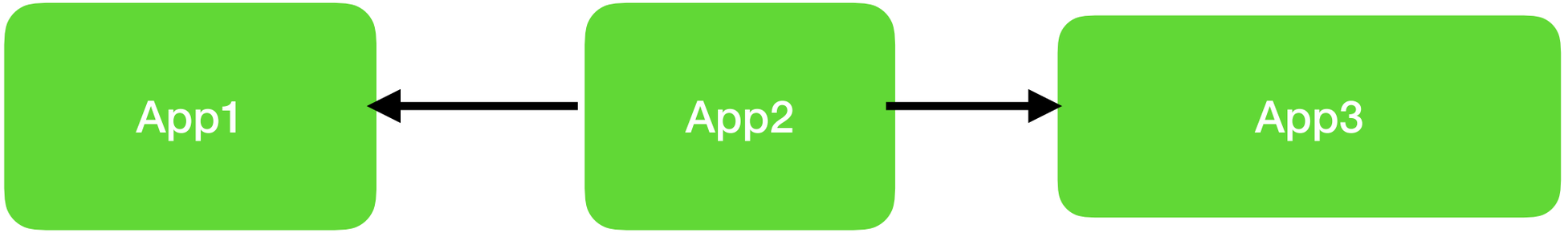
Step 7. Backup and Recovery



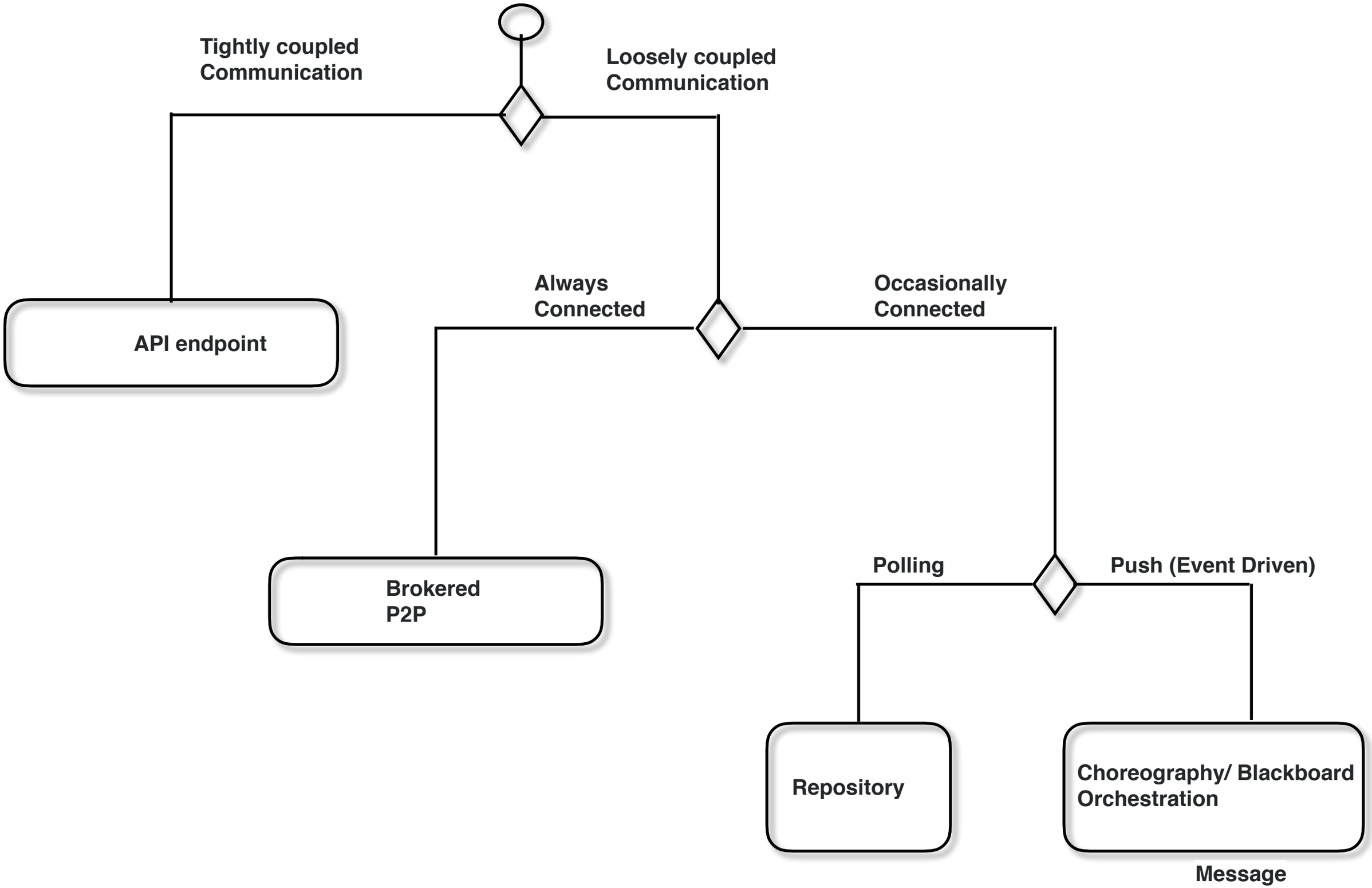
* Full backups
* differential backups
* transaction log backups

* Standard Geo-Replication
* Active Geo-Replication

9. Integration View



Step 2. Choose Communication Mechanism



ToDo Portal
(Single Page Application)

Https(443)

ToDo API Service
(Api Application)

Document Db

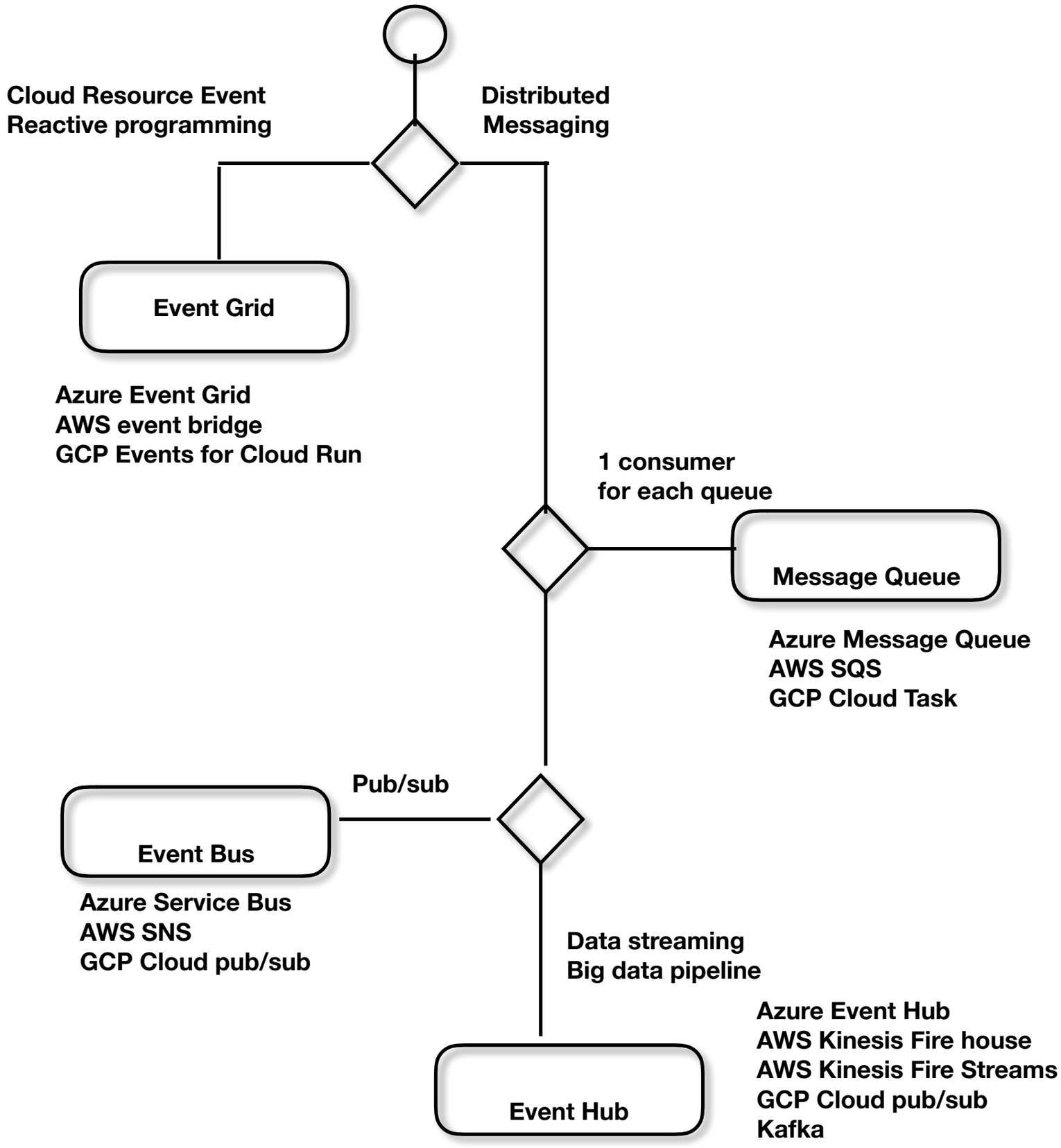
ToDo Calendar Service
(Background Application)

Message

Data Lake

ToDo Prediction Service
(Background Application)

Step 3. Choose Message Engine



ToDo Portal
(Single Page Application)

Https(443)

ToDo API Service
(Api Application)

ToDo Calendar Service
(Background Application)

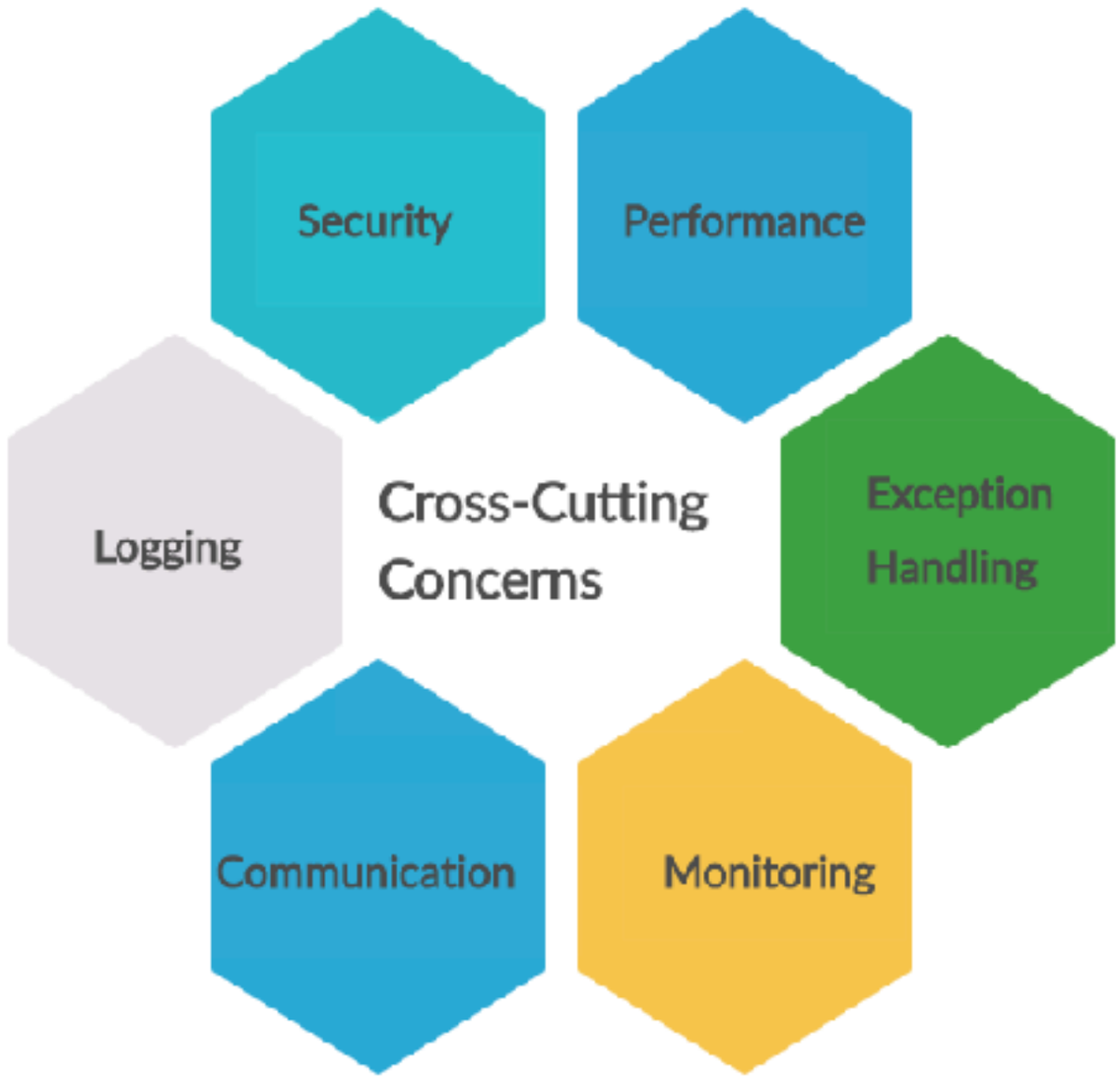
Document Db

Message queue

Data Lake

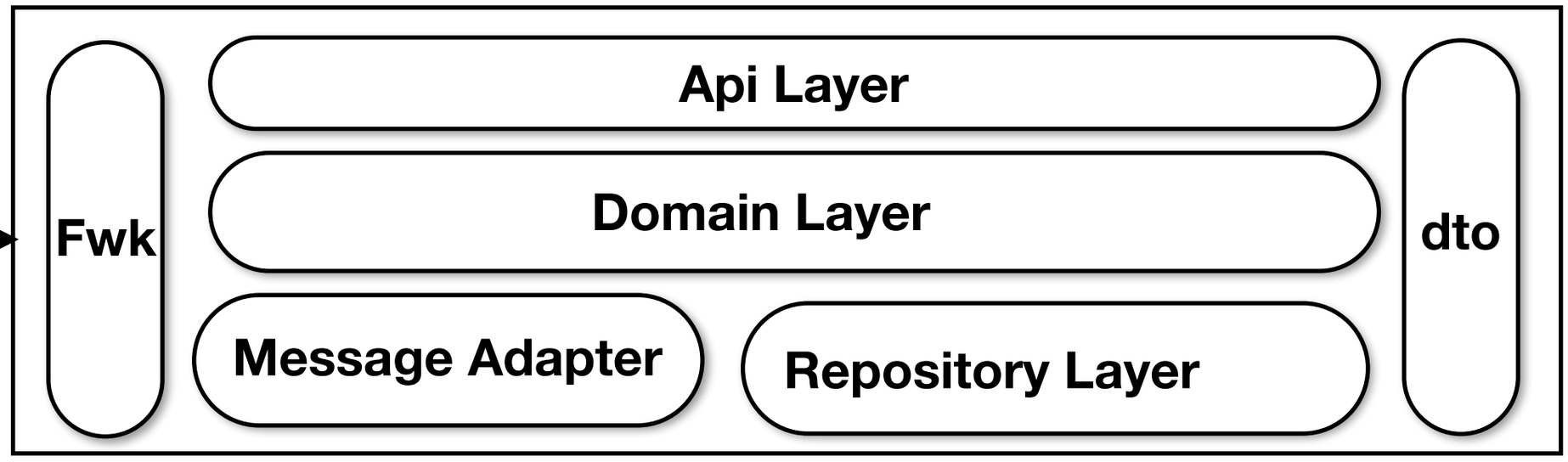
ToDo Prediction Service
(Background Application)

10. Cross cutting concerns

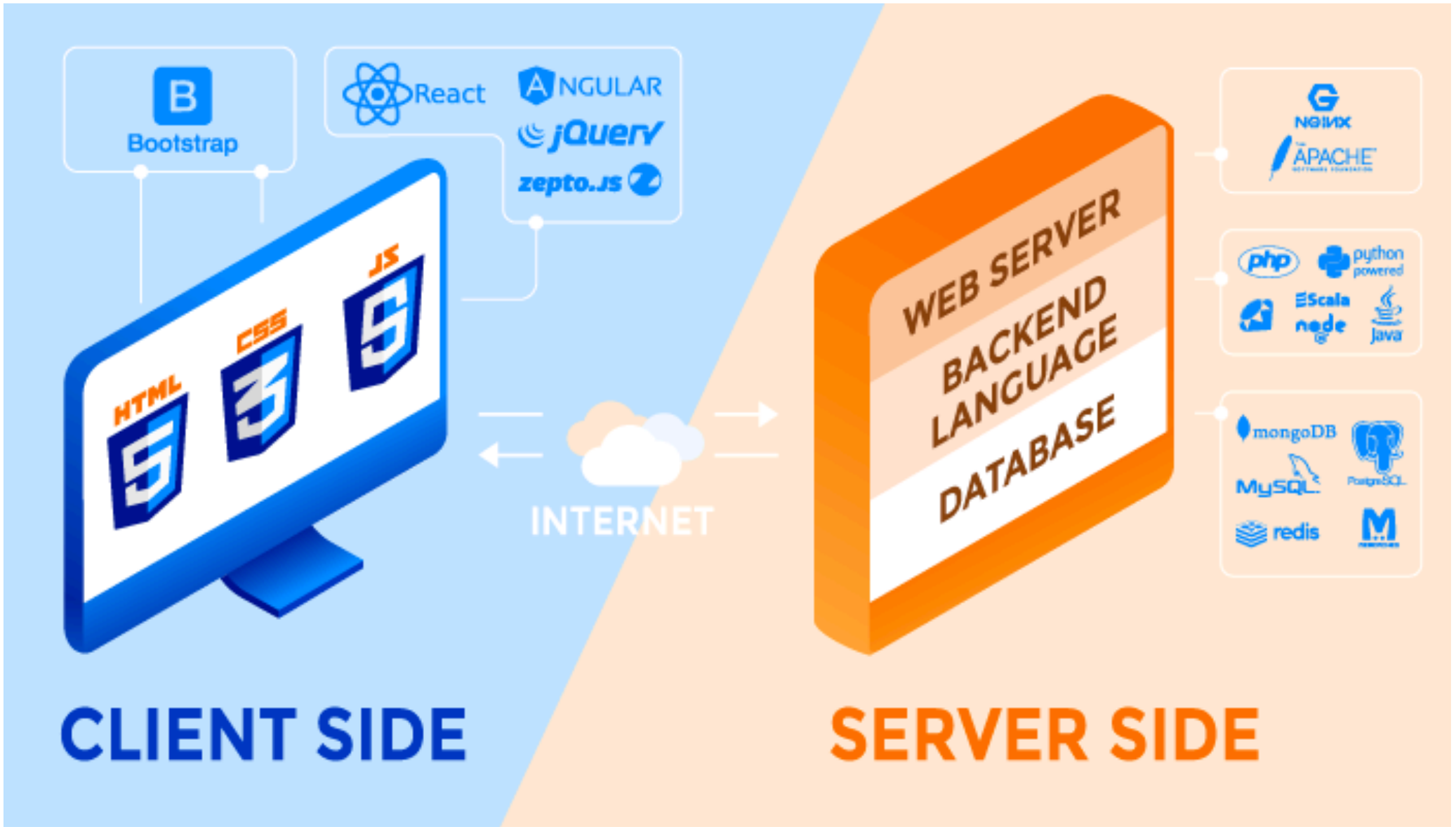


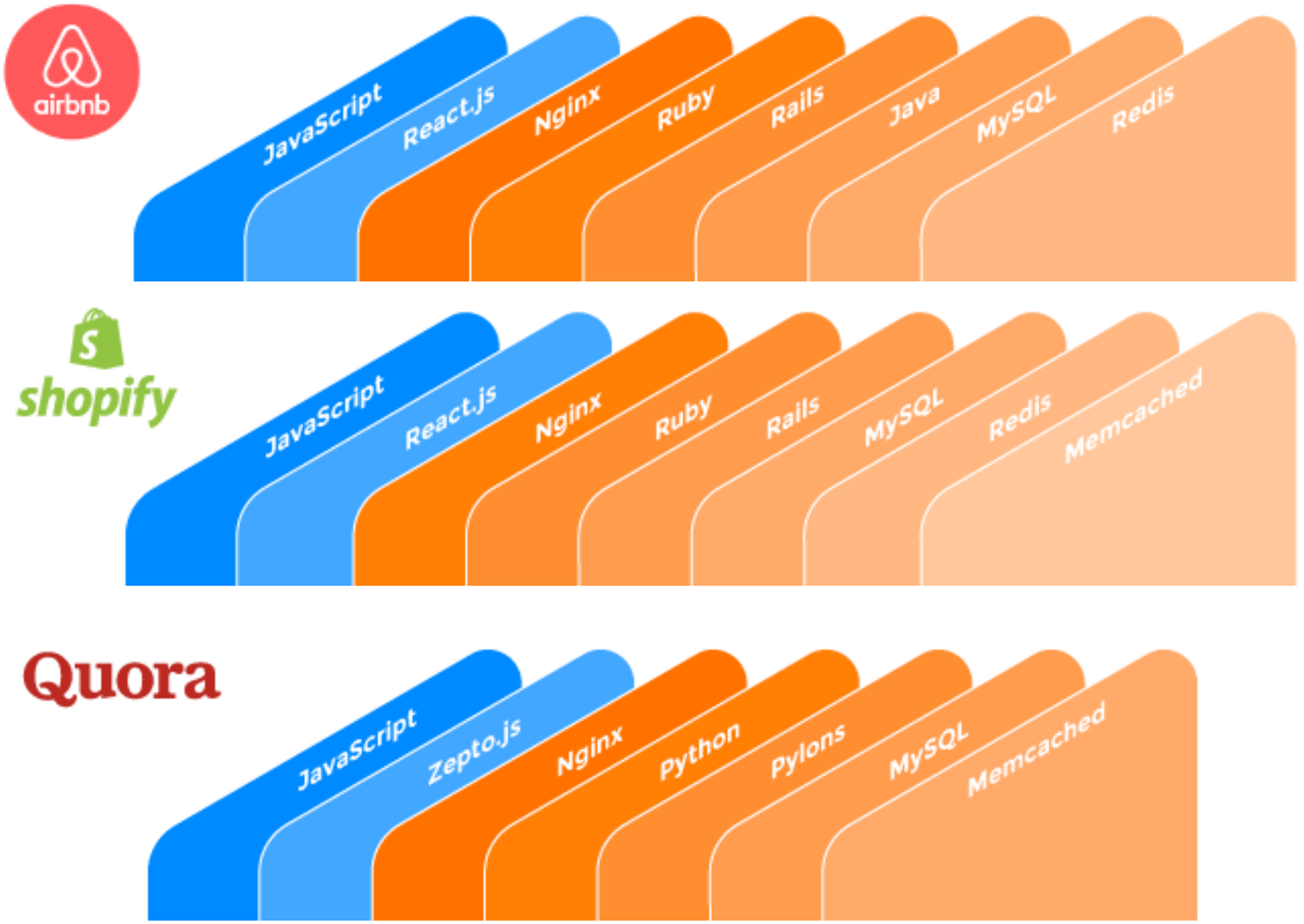
The **crosscutting concern** is a concern which is needed in almost every module of an application.

Logging,
Exception Handling,
Input Validation,
Syncronization,
Transaction,
Audit,
Authentication,
Authorization,
Encryption,



11. Technology View



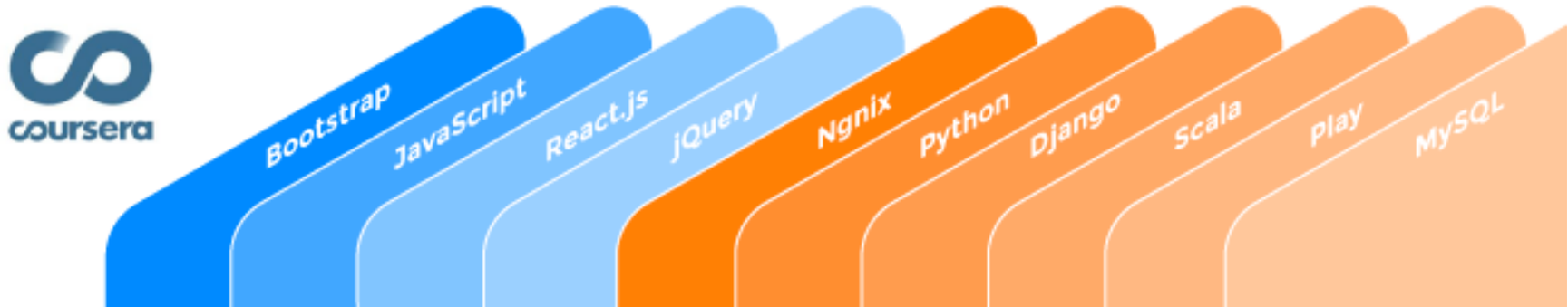

airbnb
JavaScript
React.js
Nginx
Ruby
Rails
Java
MySQL
Redis

shopify
JavaScript
React.js
Nginx
Ruby
Rails
MySQL
Redis
Memcached

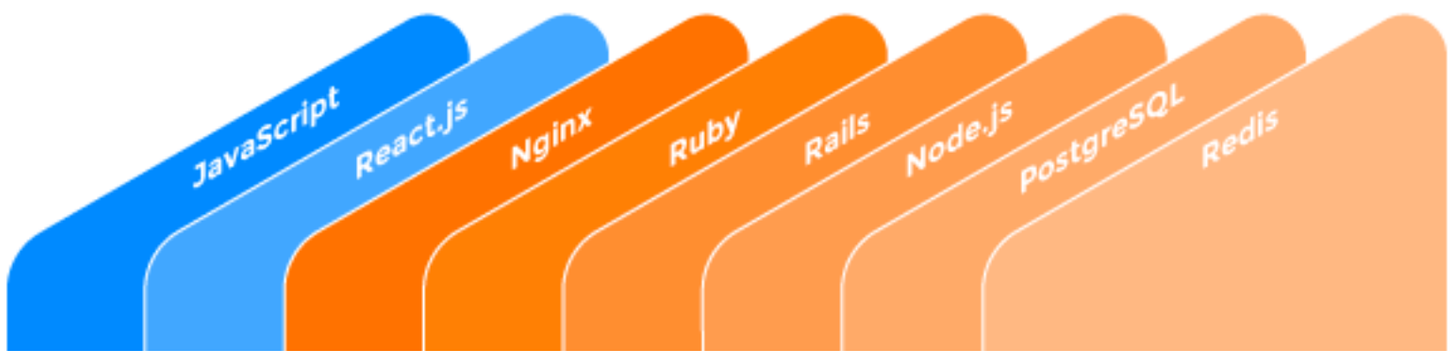
Quora
JavaScript
Zepto.js
Nginx
Python
Pylons
MySQL
Memcached

JavaScript
React.js
Nginx
Python
Django
PostgreSQL
Redis

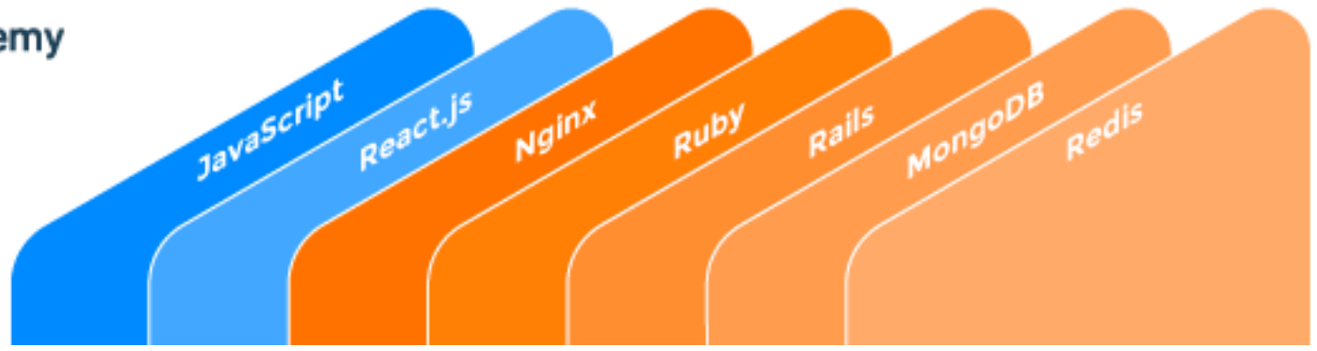
[Instagram]


coursera
Bootstrap
JavaScript
React.js
jQuery
Ngnix
Python
Django
Scala
Play
MySQL

Product Hunt

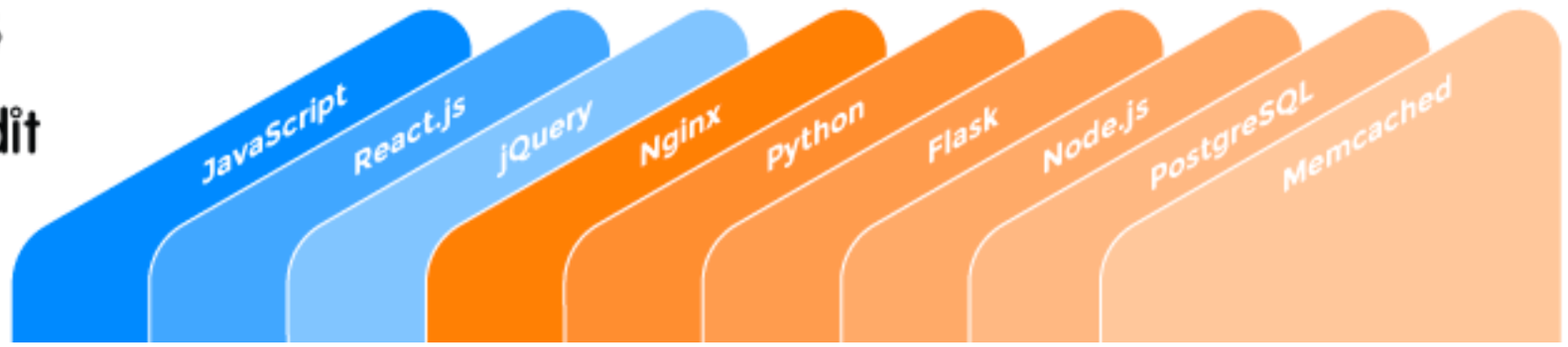


codecademy



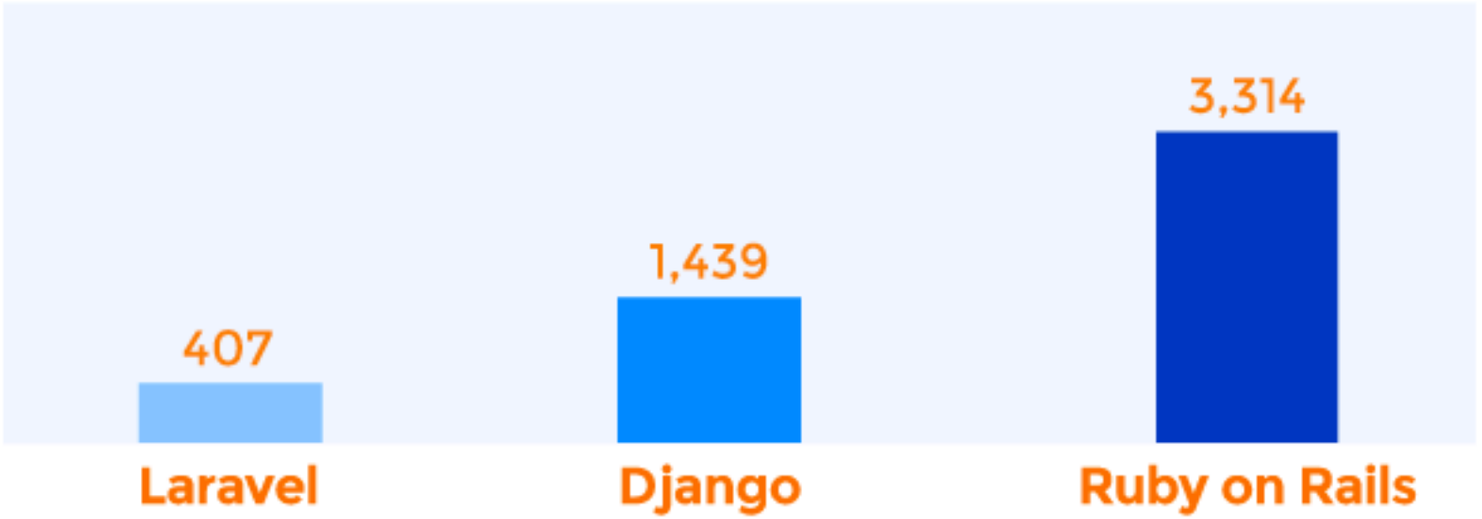
Pinterest

reddit



NUMBER OF CONTRIBUTORS ON GITHUB

JUNE 2017



AVERAGE DEVELOPER SALARIES IN THE US

BY TECHNOLOGY

Ruby	Python	PHP	JAVA
$55.78/hour	$55.19/hour	$42.68/hour	$54.13/hour

Source: Indeed.com

ToDo Portal
(Single Page Application)

NGINX

ANGULAR

Https(443)

ToDo API Service
(Api Application)

Apache Web Server

python

Flask

ToDo Calendar Service
(Background Application)

python

Document Db

mongoDB

Message queue

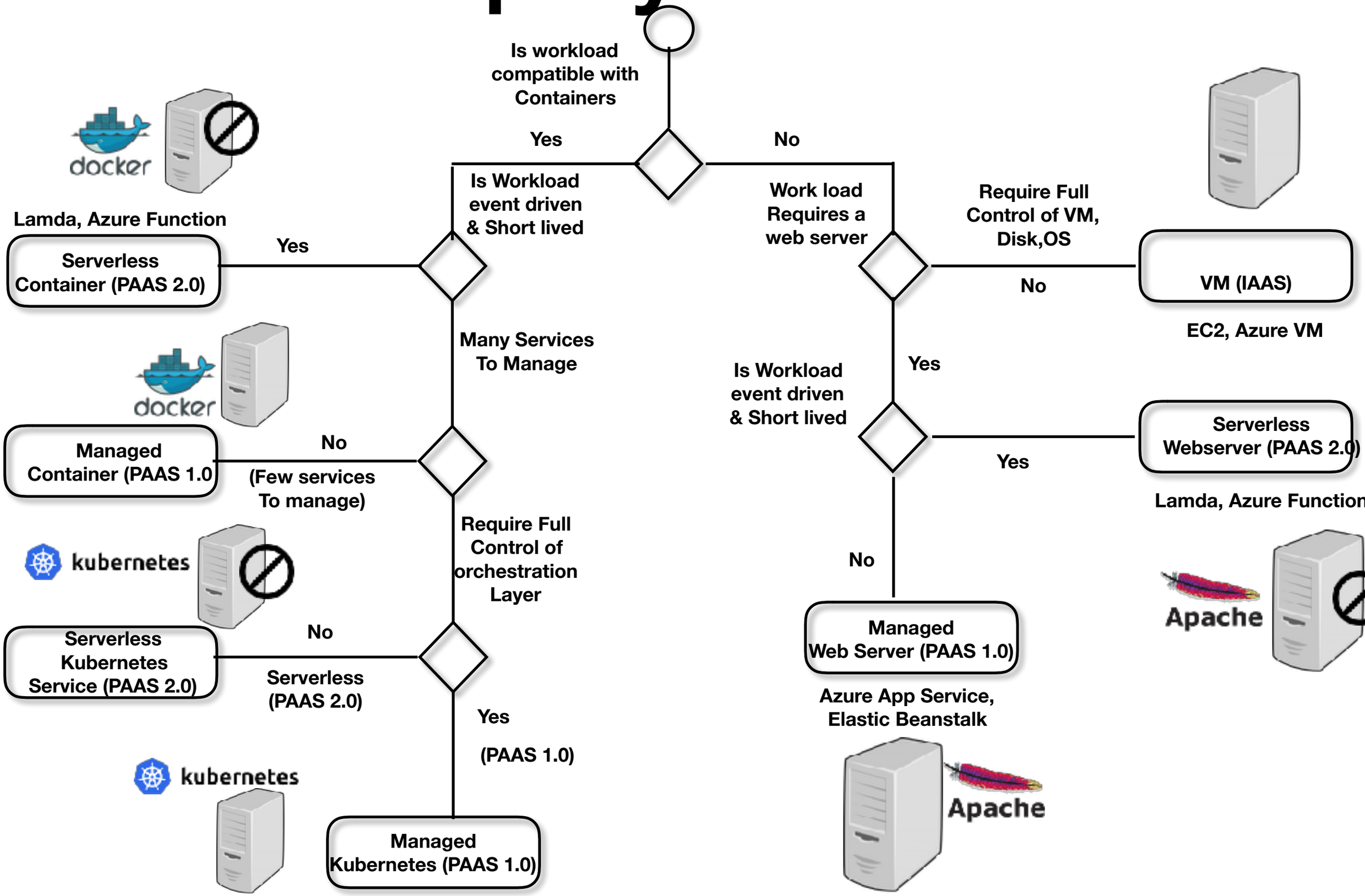
Azure Queue

Azure Data Lake

ToDo Prediction Service
(Background Application)

python

12. Deployment View



ToDo Portal
(Single Page Application)

NGINX

ANGULAR

kubernetes

Https(443)

ToDo API Service
(Api Application)

Apache Web Server

python

Flask

ToDo Calendar Service
(Background Application)

python

ToDo Prediction Service
(Background Application)

python

Message queue

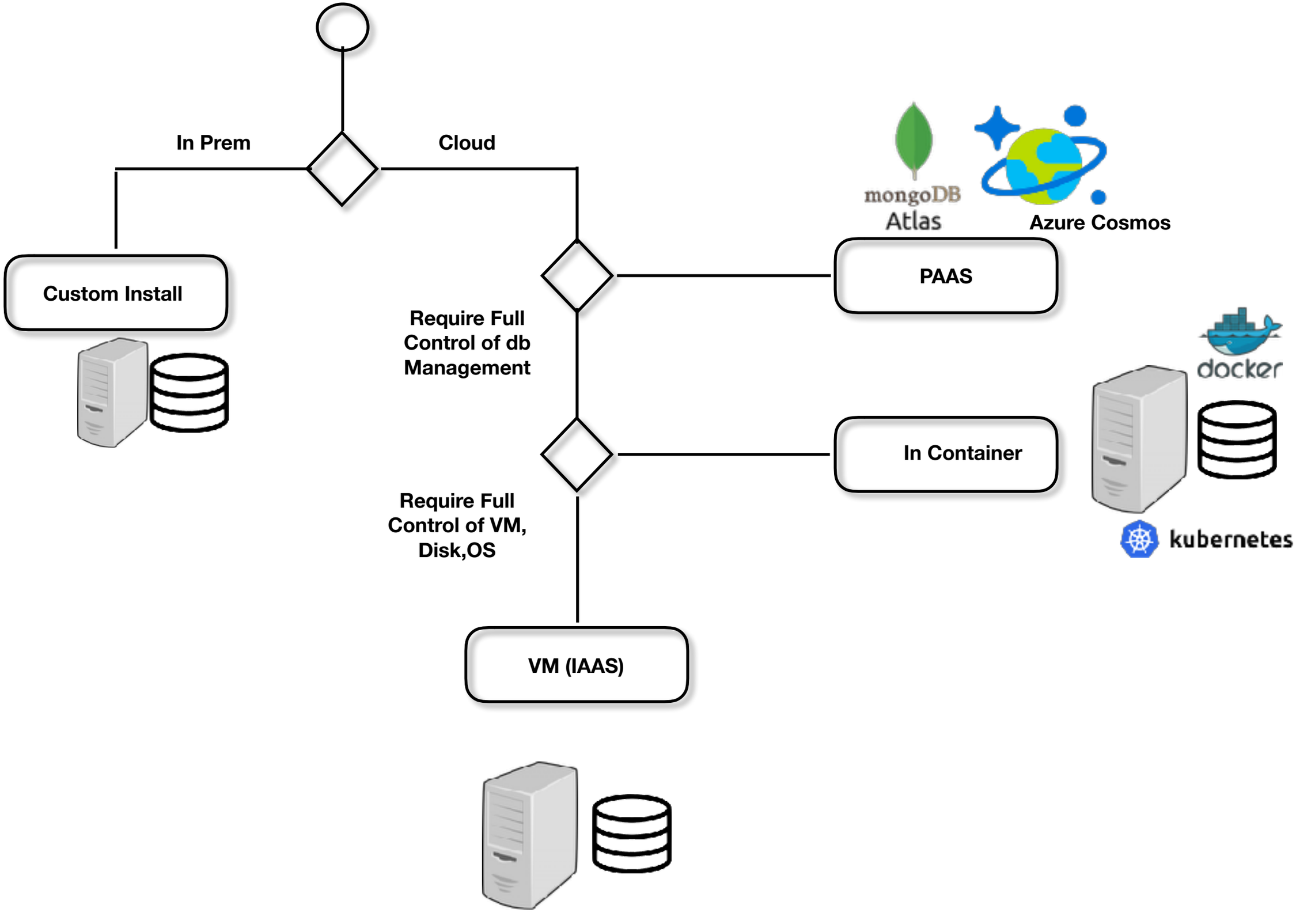
Azure Queue

Document Db

mongoDB

Azure Data Lake

Deploying Data Tier



ToDo Portal
(Single Page Application)

NGINX

ANGULAR

kubernetes

Https(443)

ToDo API Service
(Api Application)

Apache Web Server

python

Flask

ToDo Calendar Service
(Background Application)

python

ToDo Prediction Service
(Background Application)

python

Message queue

Azure Queue

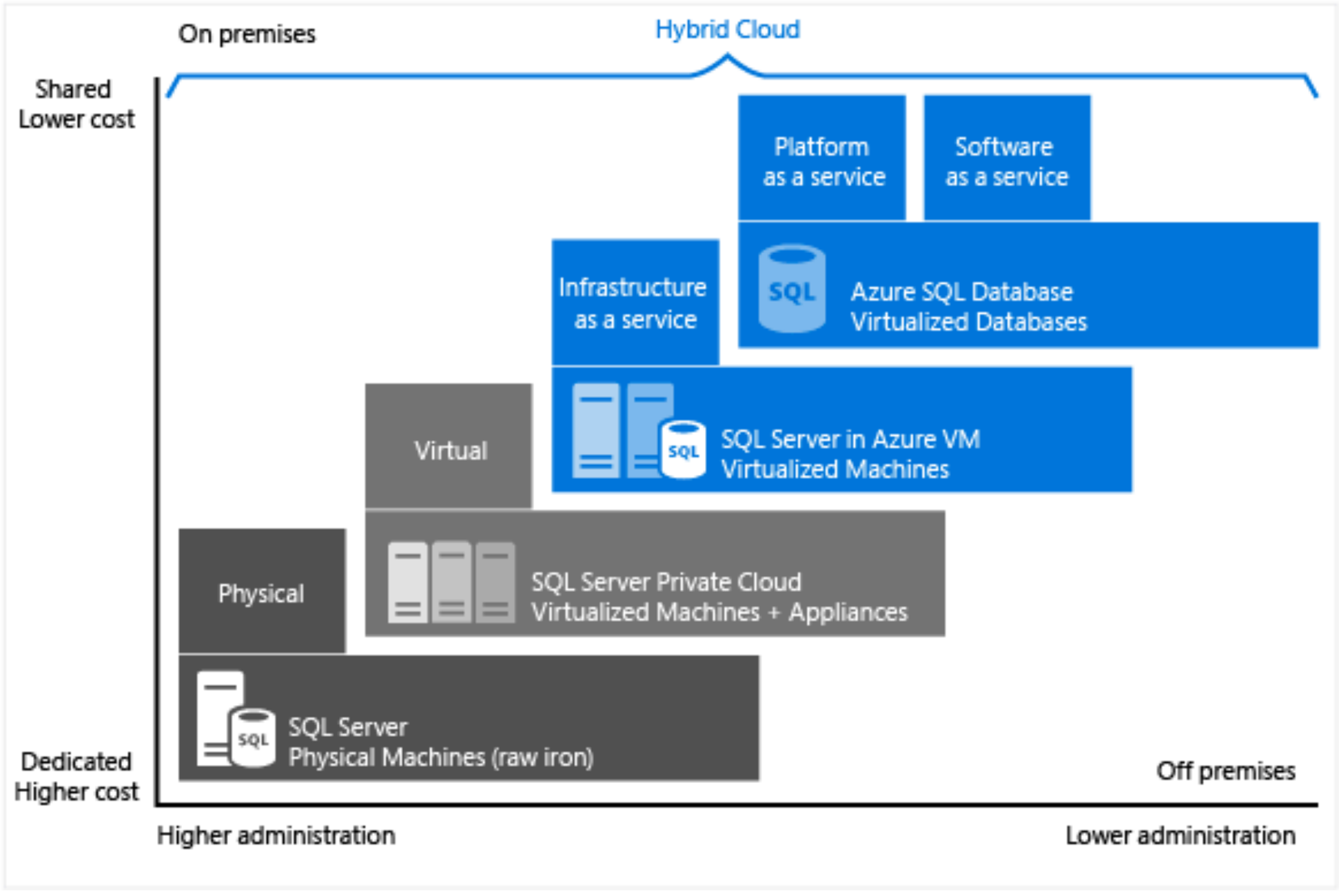
Document Db

mongoDB

mongoDB
Atlas

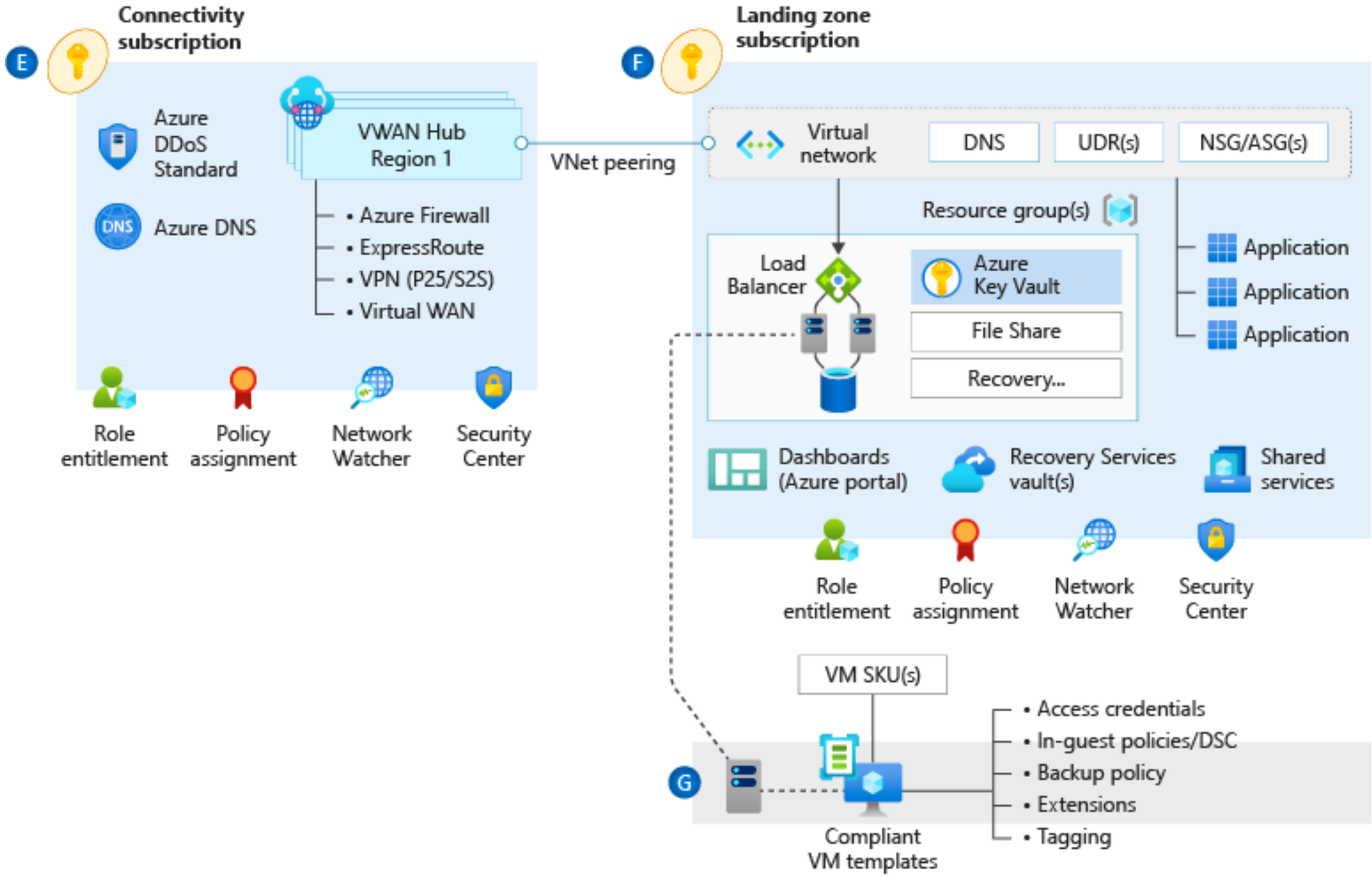
Azure Data Lake

Deploying Data Tier



13. Infrastructure View

Physical View



Public IP address

vnet

asg subnet

Network Load Balancer

asg subnet

NGINX Ingress controller

ToDo Portal
(Single Page Application)
NGINX ANGULAR

kubernetes

Https(443)

ToDo API Service
(Api Application)
Flask
python
Apache Web Server

ToDo Calendar Service
(Background Application)
python

ToDo Prediction Service
(Background Application)
python

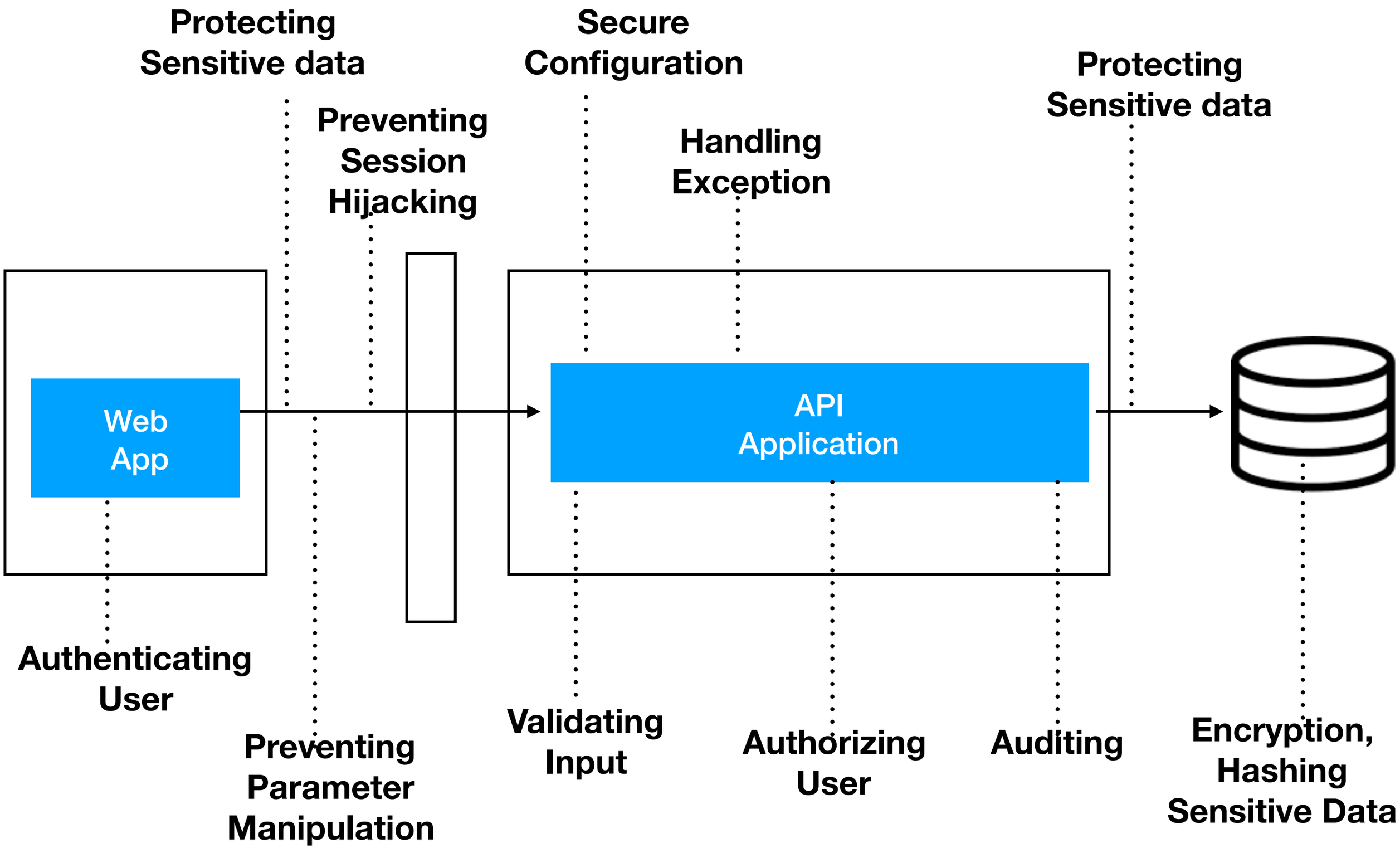
Document Db
mongoDB

mongoDB Atlas

Message queue
Azure Queue

Azure Data Lake

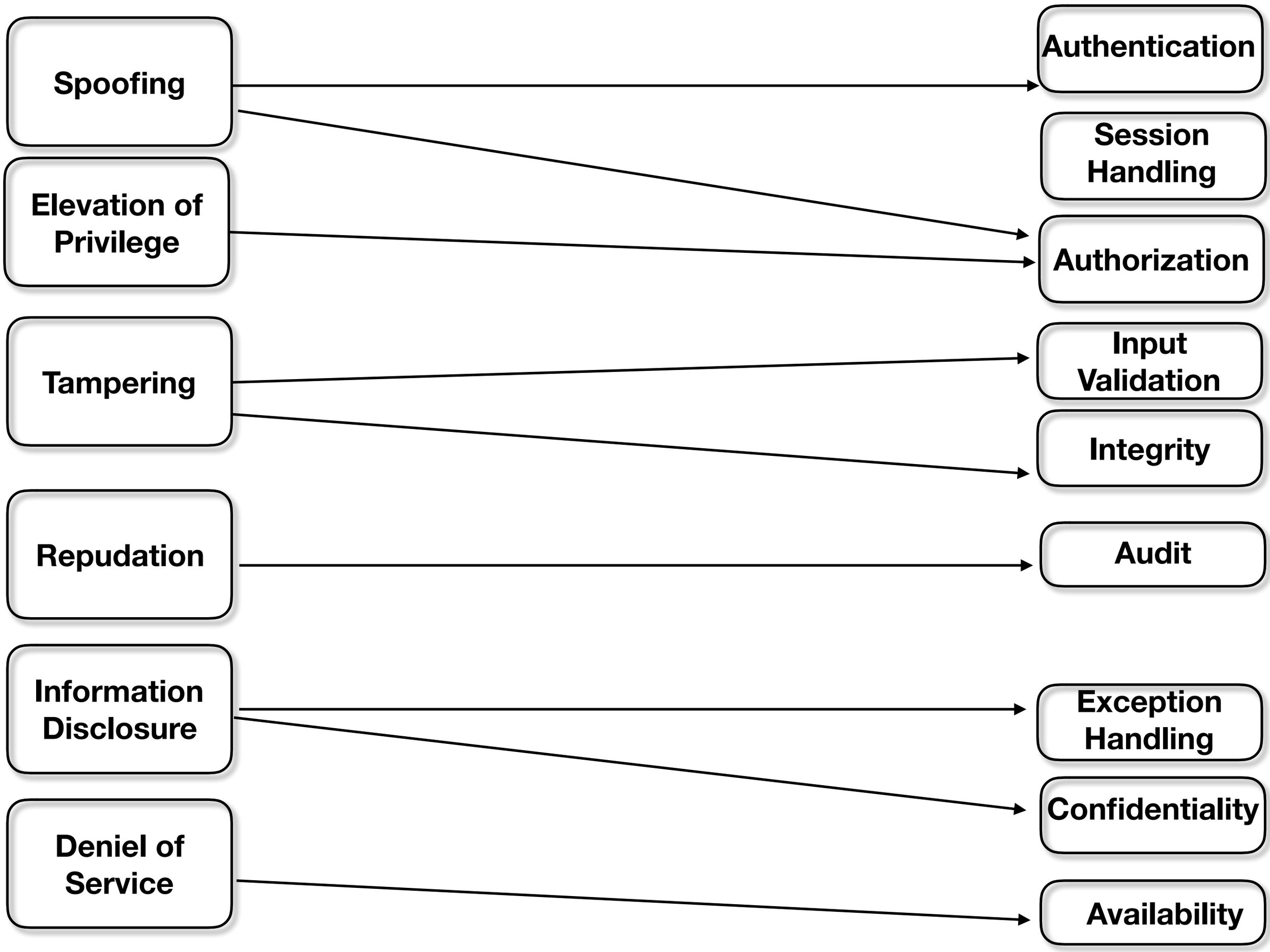
14. Application Security View



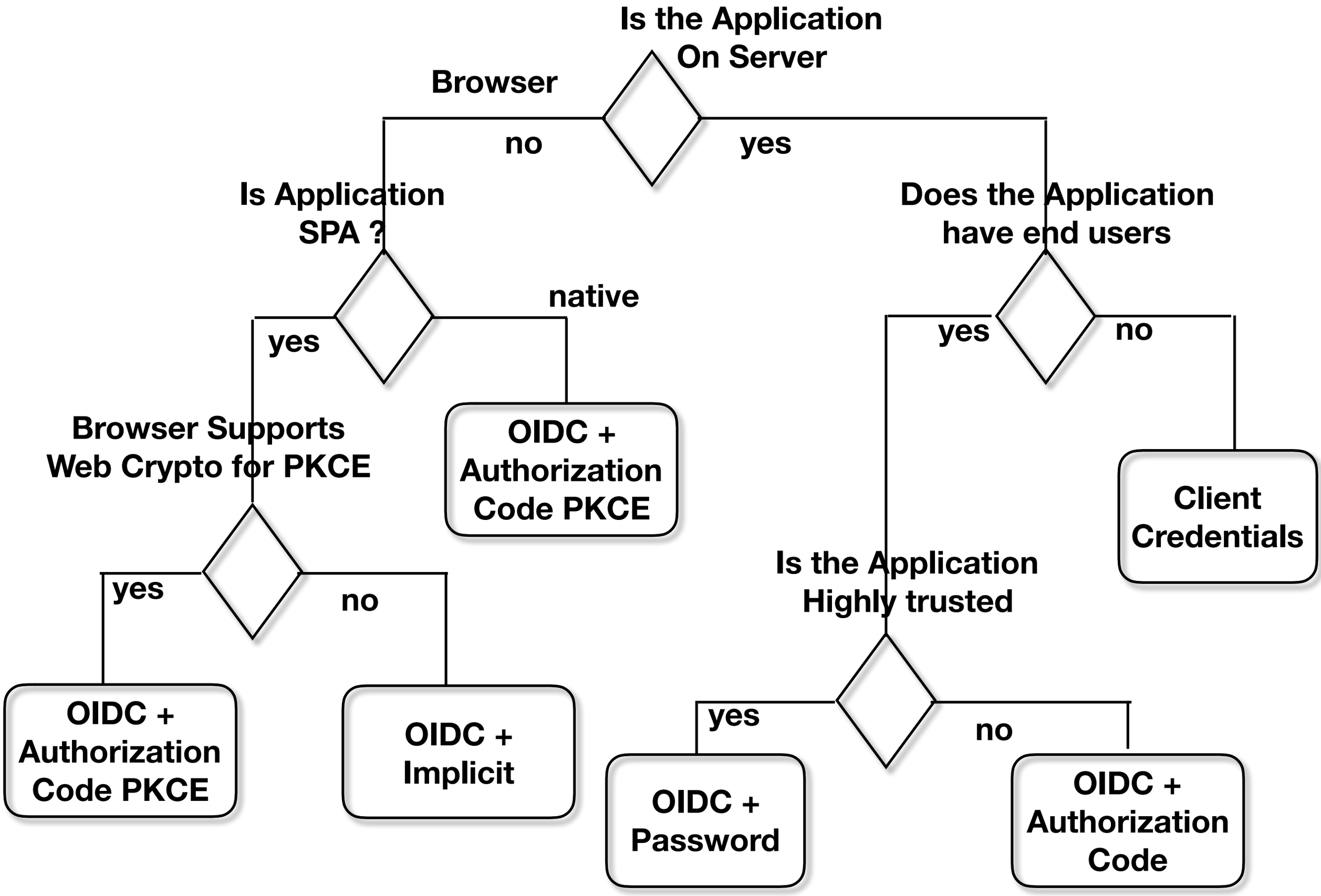
Threat Modeling Method	Features
STRIDE	• Helps identify relevant mitigating techniques • Is the most mature • Is easy to use but is time consuming
PASTA	• Helps identify relevant mitigating techniques • Directly contributes to risk management • Encourages collaboration among stakeholders • Contains built-in prioritization of threat mitigation • Is laborious but has rich documentation
LINDDUN	• Helps identify relevant mitigation techniques • Contains built-in prioritization of threat mitigation • Can be labor intensive and time consuming
CVSS	• Contains built-in prioritization of threat mitigation • Has consistent results when repeated • Has automated components • Has score calculations that are not transparent
Attack Trees	• Helps identify relevant mitigation techniques • Has consistent results when repeated • Is easy to use if you already have a thorough understanding of the system
Persona non Grata	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Has consistent results when repeated • Tends to detect only some subsets of threats
Security Cards	• Encourages collaboration among stakeholders • Targets out-of-the-ordinary threats • Leads to many false positives
hTMM	• Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated
Quantitative TMM	• Contains built-in prioritization of threat mitigation • Has automated components • Has consistent results when repeated
Trike	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has automated components • Has vague, insufficient documentation
VAST Modeling	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Has automated components • Is explicitly designed to be scalable • Has little publicly available documentation
OCTAVE	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Is explicitly designed to be scalable • Is time consuming and has vague documentation

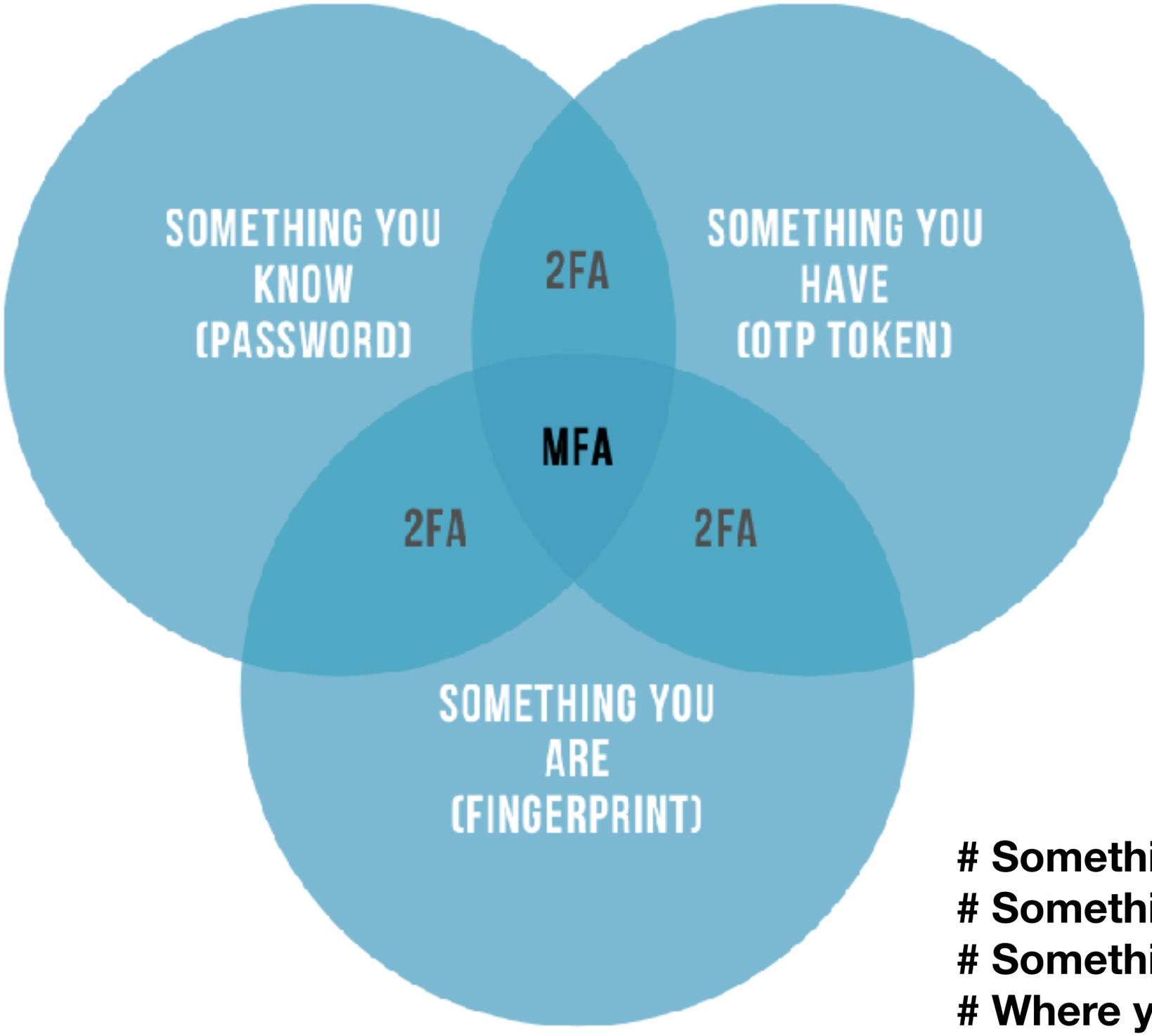
PASTA

STAGE I - Definition of the Objectives (DO)	• DO 1.1 - Document the business requirements • DO 1.2 – Define the security/compliance requirements • DO 1.3 – Define the business impact • DO 1.4 – Determine the risk profile
Stage II - Definition of the Technical Scope (DTS)	• DTS 2.1 – Enumerate Software components • DTS 2.2 – Identify Actors & Data Sinks/Source • DTS 2.3 – Enumerate System-Level services • DTS 2.4 – Enumerate 3rd Party infrastructure. • DTS 2.5 – Assert completeness of secure design.
Stage III - Application Decomposition and Analysis (ADA)	• ADA 3.1 – Enumerate all application use cases • ADA 3.2 – Document Data Flow Diagrams (DFDs) • ADA 3.3 – Security functional analysis & the use of trust boundaries
Stage IV - Threat Analysis (TA)	• TA 4.1 – Analyze the overall threat scenario • TA 4.2 – Gather threat information from internal threat sources • TA 4.3 – Gather threat information from External threat sources • TA 4.4 – Update the threat libraries • TA 4.5 – Threat agents to assets mapping. • TA 4.6 – Assignment of the probabilistic values for identified threats
Stage V - Weakness and Vulnerability Analysis (WVA)	• WVA 5.1 – Review/correlate existing vulnerabilities • WVA 5.2 – Identify weak design patterns in the architecture • WVA 5.3 – Map threats to vulnerabilities • WVA 5.4 – Provide Context risk Analysis based upon Threat-Vulnerability • WVA 5.5 – Conduct targeted vulnerability testing
Stage VI - Attack Modeling & Simulation (AMS)	• AMS 6.1 – Analyze the attack scenarios • AMS 6.2 – Update the attack library/vectors and the control framework • AMS 6.3 – Identify the attack surface and enumerate the attack vectors • AMS 6.4 – Assess the probability and impact of each attack scenario. • AMS 6.5 – Derive a set of cases to test existing countermeasures. • AMS 6.6 – Conduct attack driven security tests and simulations
STAGE VII - Risk Analysis & Management (RAM)	• RAM 7.1 – Calculate the risk of each threat • RAM 7.2 – Identify countermeasures and risk mitigations measures • RAM 7.3 – Calculate the residual risks • RAM 7.4 – Recommend strategies to manage risks


Spoofing
Elevation of Privilege
Tampering
Repudation
Information Disclosure
Deniel of Service
Authentication
Session Handling
Authorization
Input Validation
Integrity
Audit
Exception Handling
Confidentiality
Availability

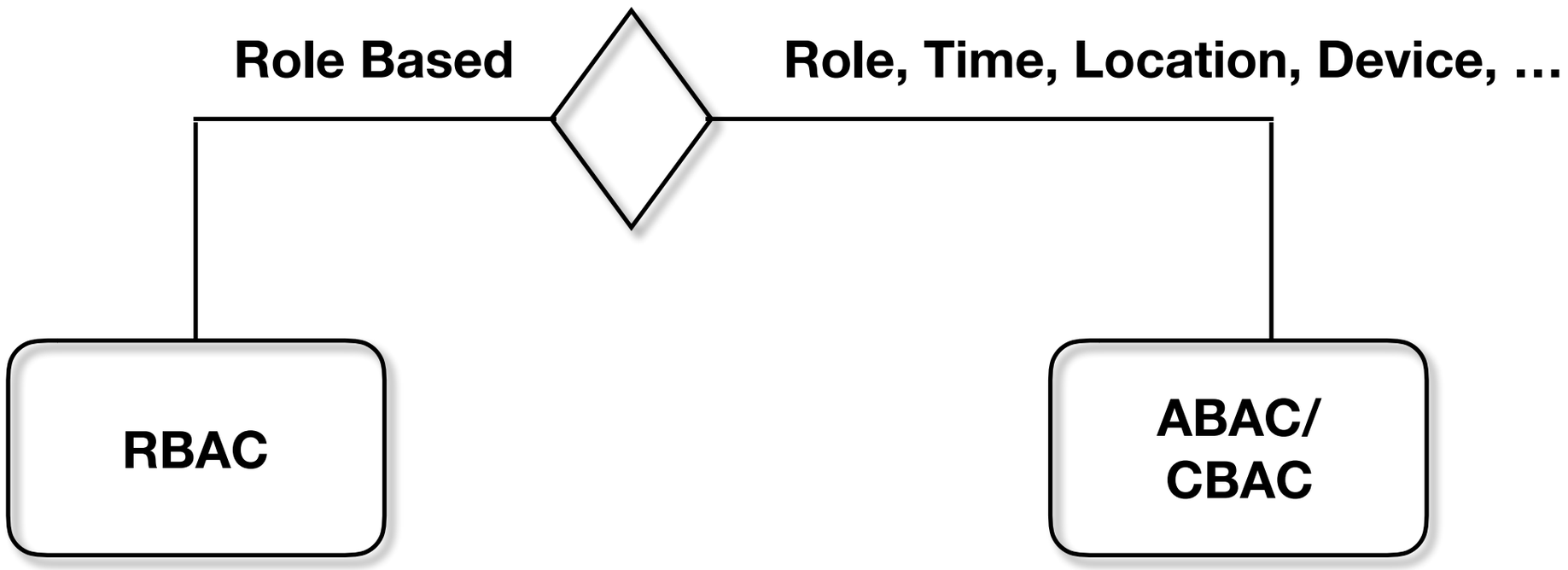
Step 1. Authentication



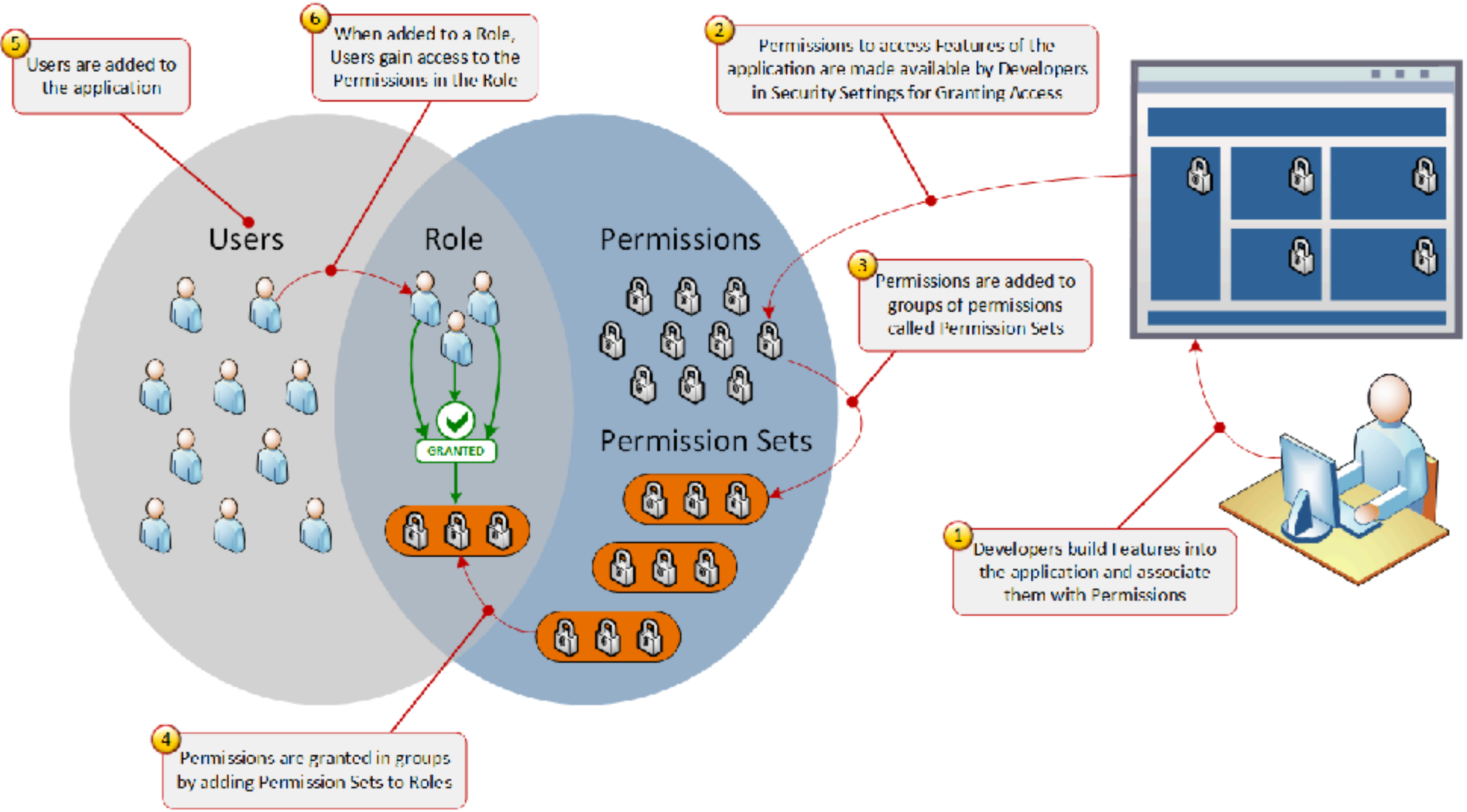


 # Something you know
 # Something you Have
 # Something you are
 # Where you are
 # Something you do

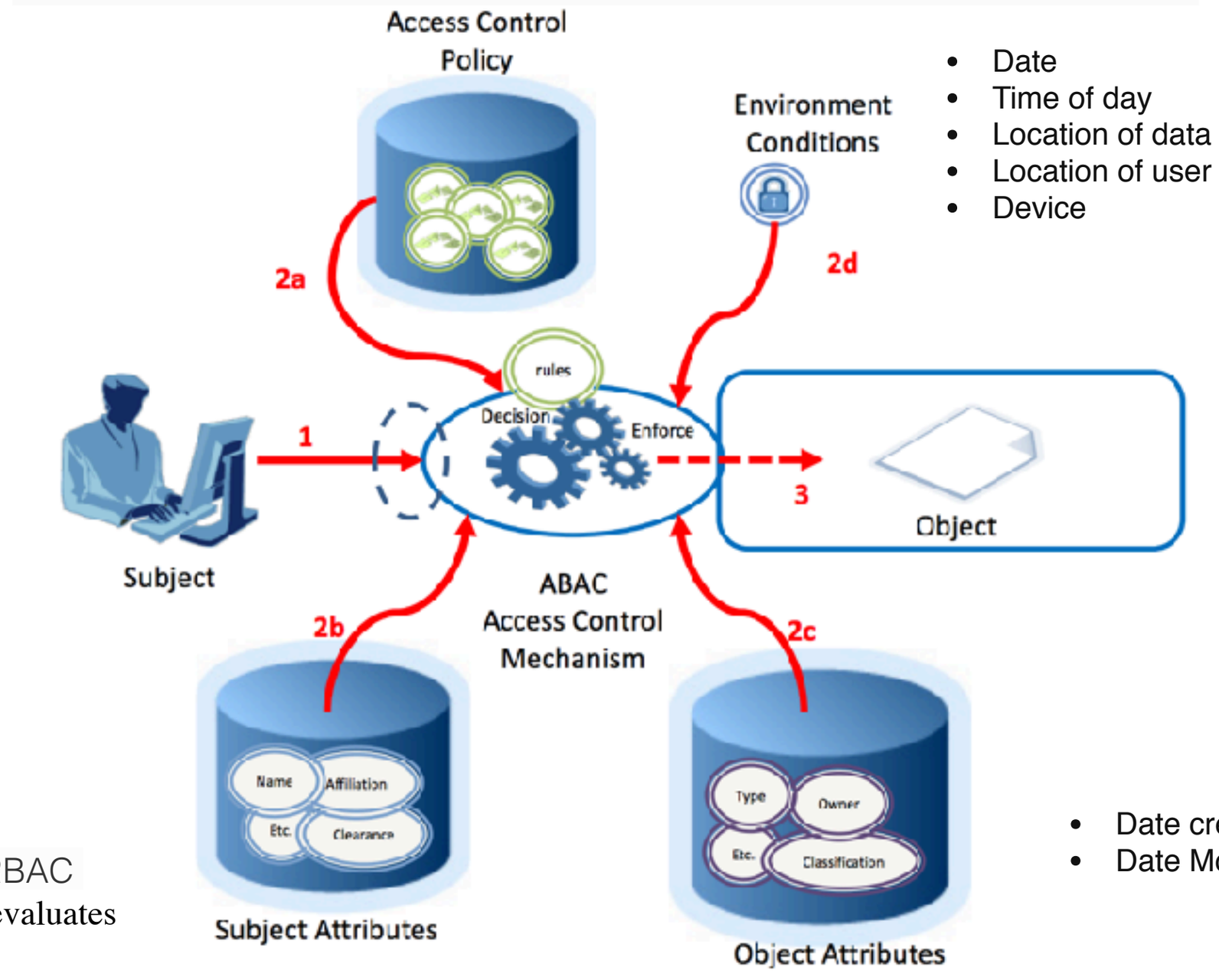
Step 2. Authorization



Role Based Access Control (RBAC)



Attribute Based Access Control (ABAC) / Claim Based Access Control (CBAC)



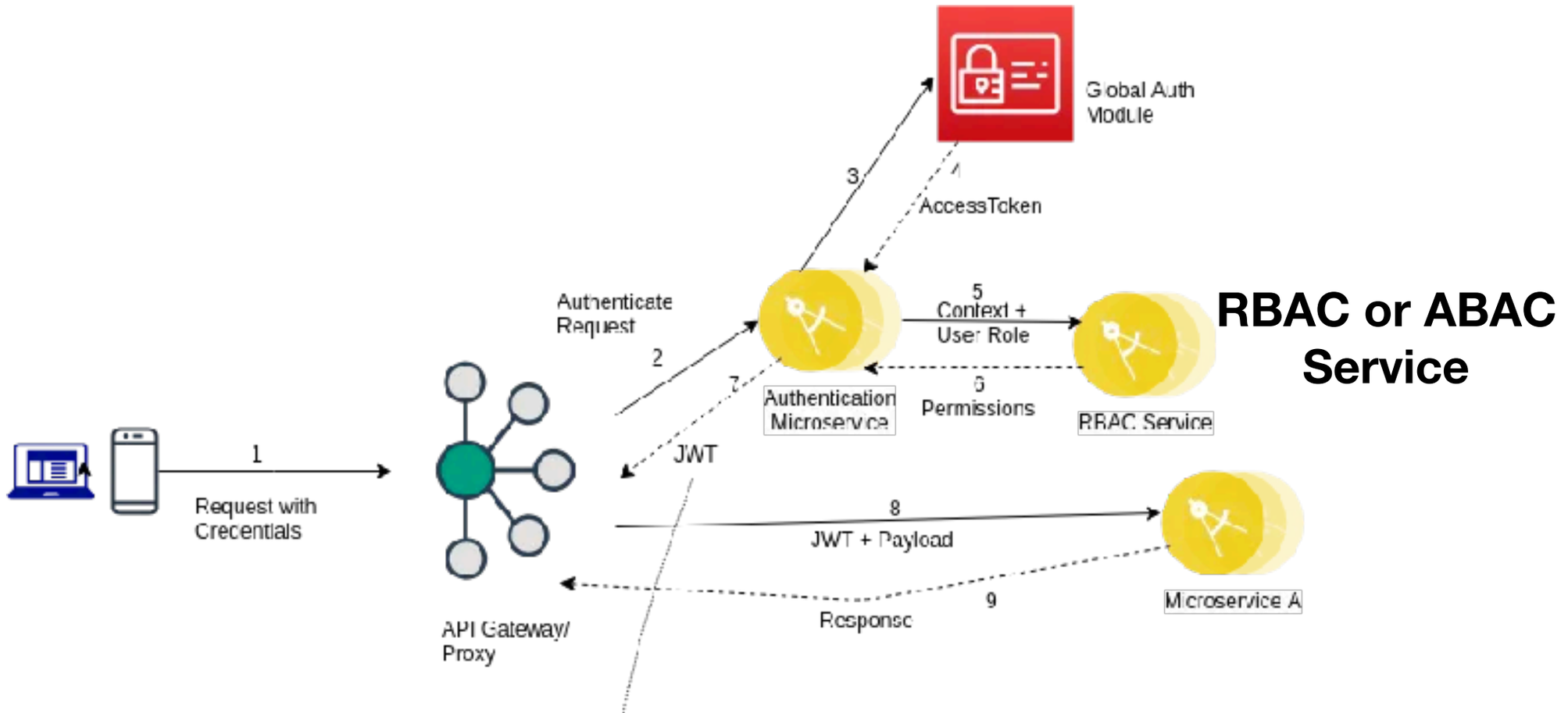
- Date
- Time of day
- Location of data
- Location of user
- Device

- Date created
- Date Modified

ABAC is an extension of RBAC

Access Control Mechanism evaluates

a) Rules (RBAC),
b) Subject Attributes,
c) Object Attributes
d) Environment Conditions
to compute a decision



RBAC or ABAC Service


```
{
"UserData": {
"userId":"#{USER_ID}",
"email":'#{USER_EMAIL}",
"first_name':"#{USER_FIRST_NAME}",
"last_name":'#{USER_LAST_NAME}',
"name":"#{USER_NAME}",
"roles": {
"#{ROLE_NAME}":"#{ROLE_ID}"
},
"permissions": {
"#{PERMISSION_NAME}":'#{PERMISSION_ID}",
}
},
"exp':148761231
}
```

Step 3. Audit

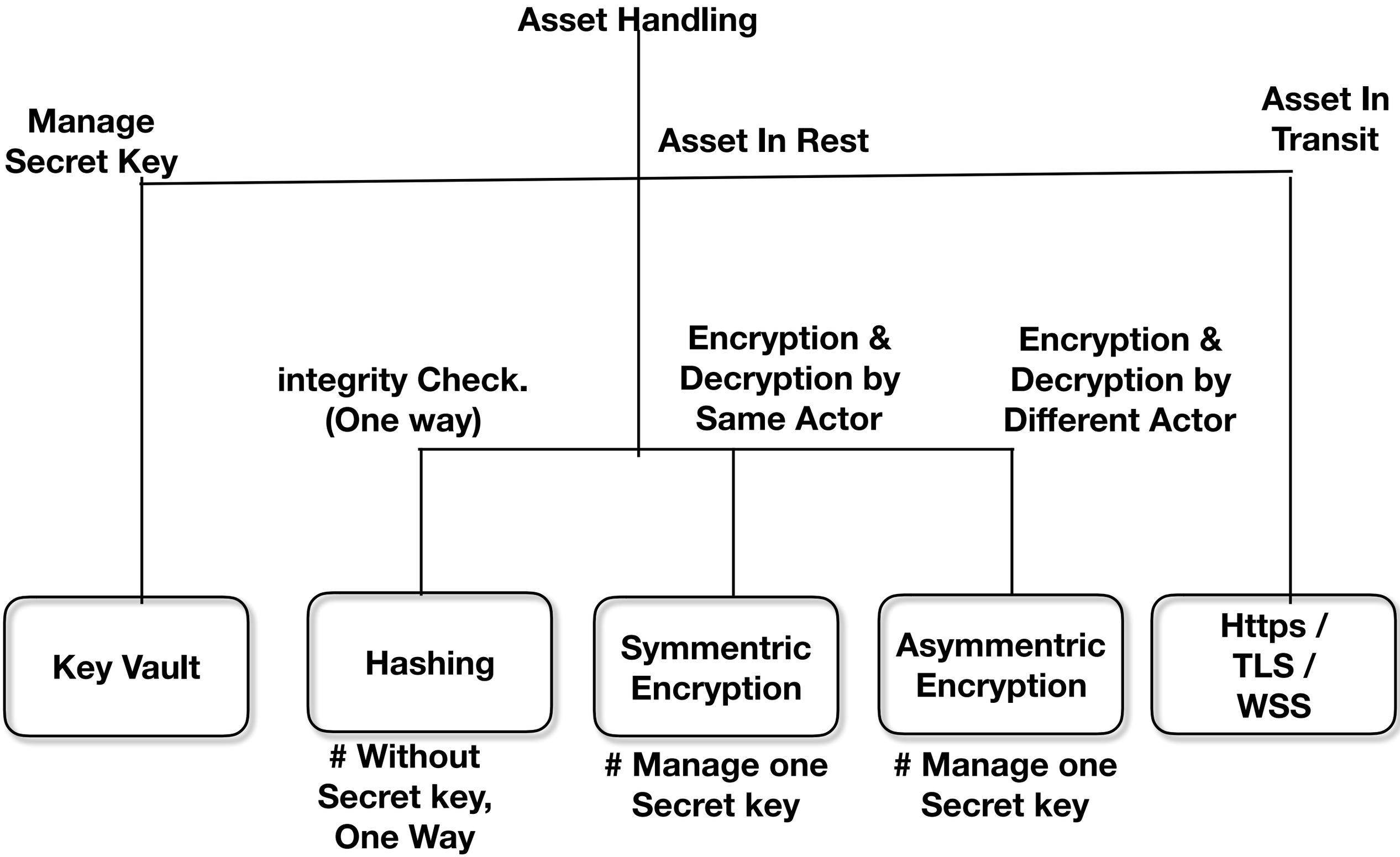
who - Identity

what - Action

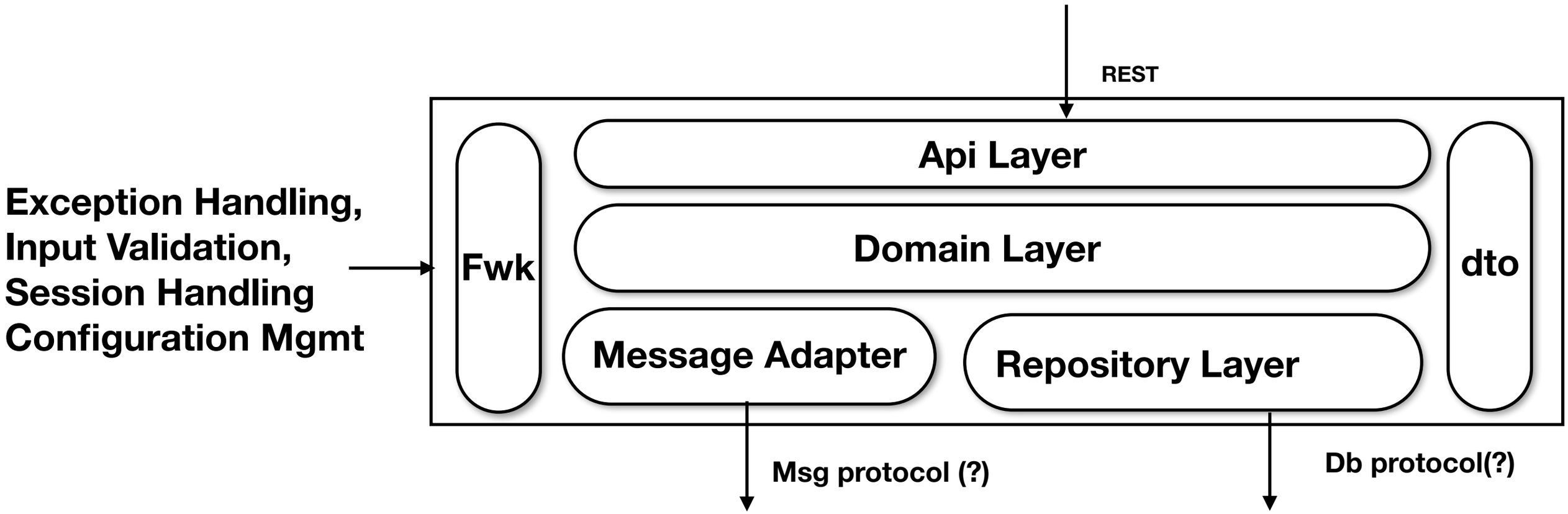
where - Component/Service/Object/Method

when - Timestamp:

Step 4. Privacy and Confidentiality



Step 5. Cross cutting concerns



Public IP address

vnet

asg subnet

WAF

Network Load Balancer

Application Gateway

asg subnet

NGINX Ingress controller

ToDo Portal
(Single Page Application)

NGINX ANGULAR

kubernetes

Https(443)

ToDo API Service
(Api Application)

Flask

python

Apache Web Server

ToDo Calendar Service
(Background Application)

python

ToDo Prediction Service
(Background Application)

python

mongoDB Atlas

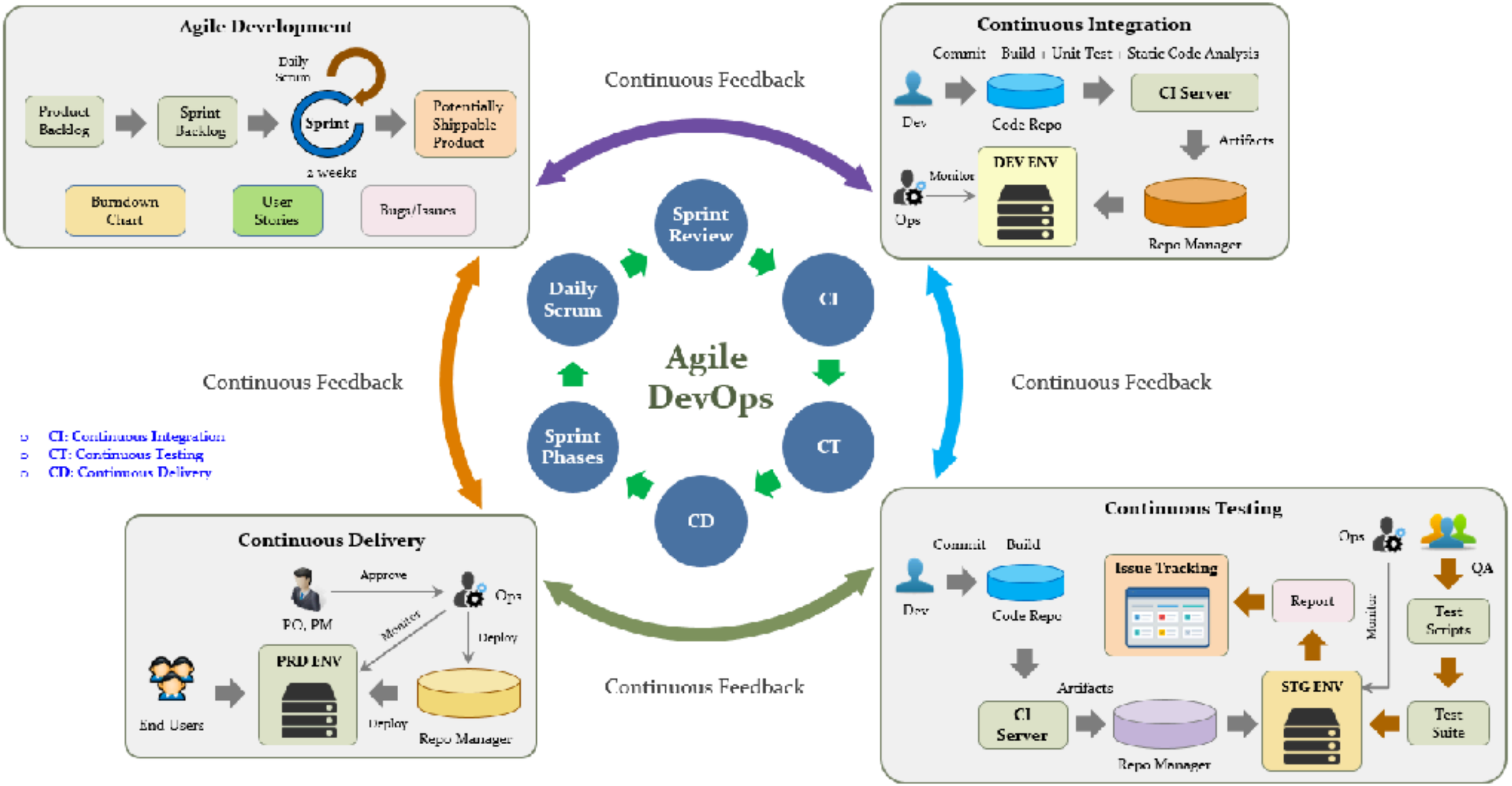
mongoDB

Message queue

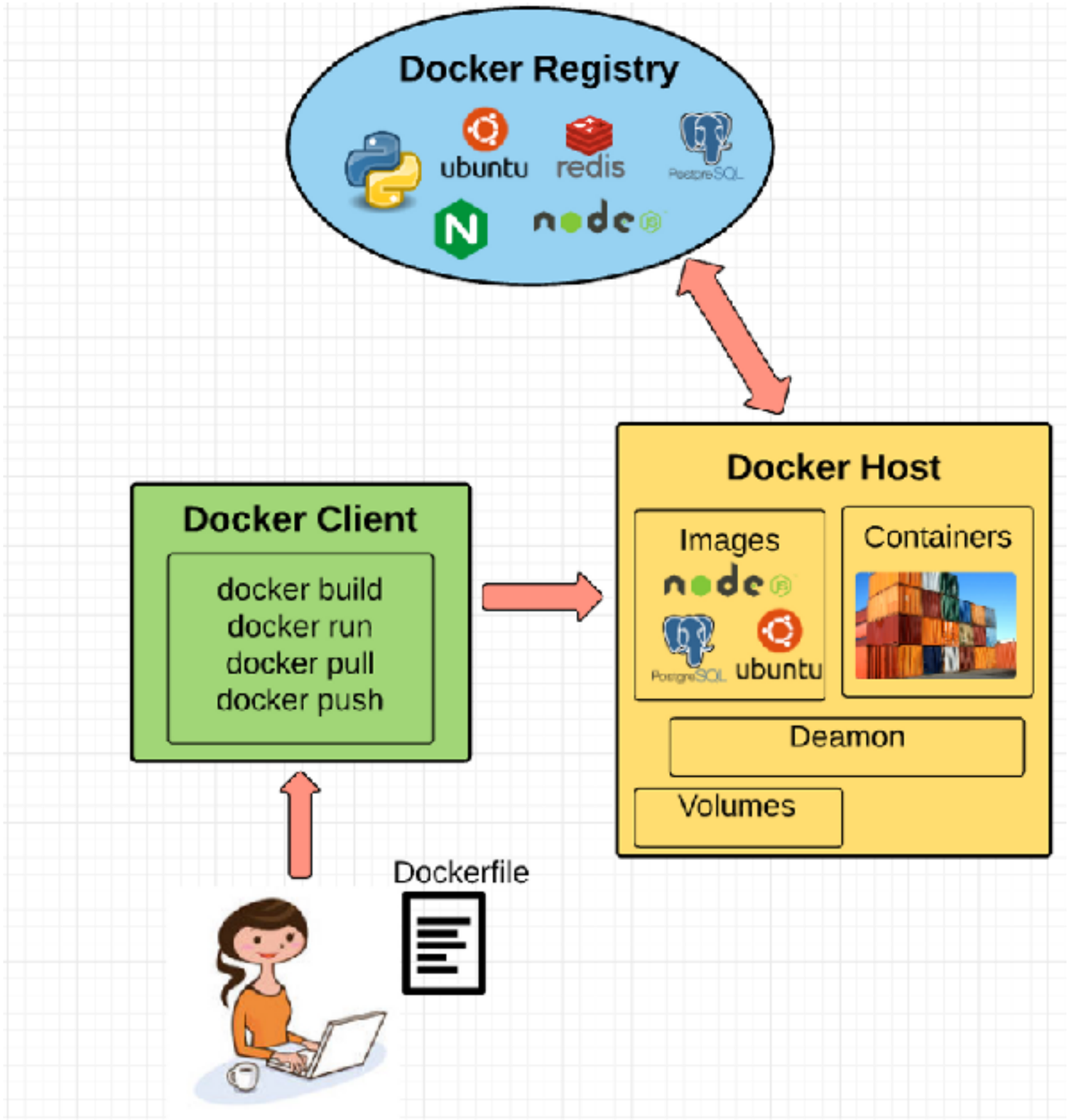
Azure Queue

Azure Data Lake

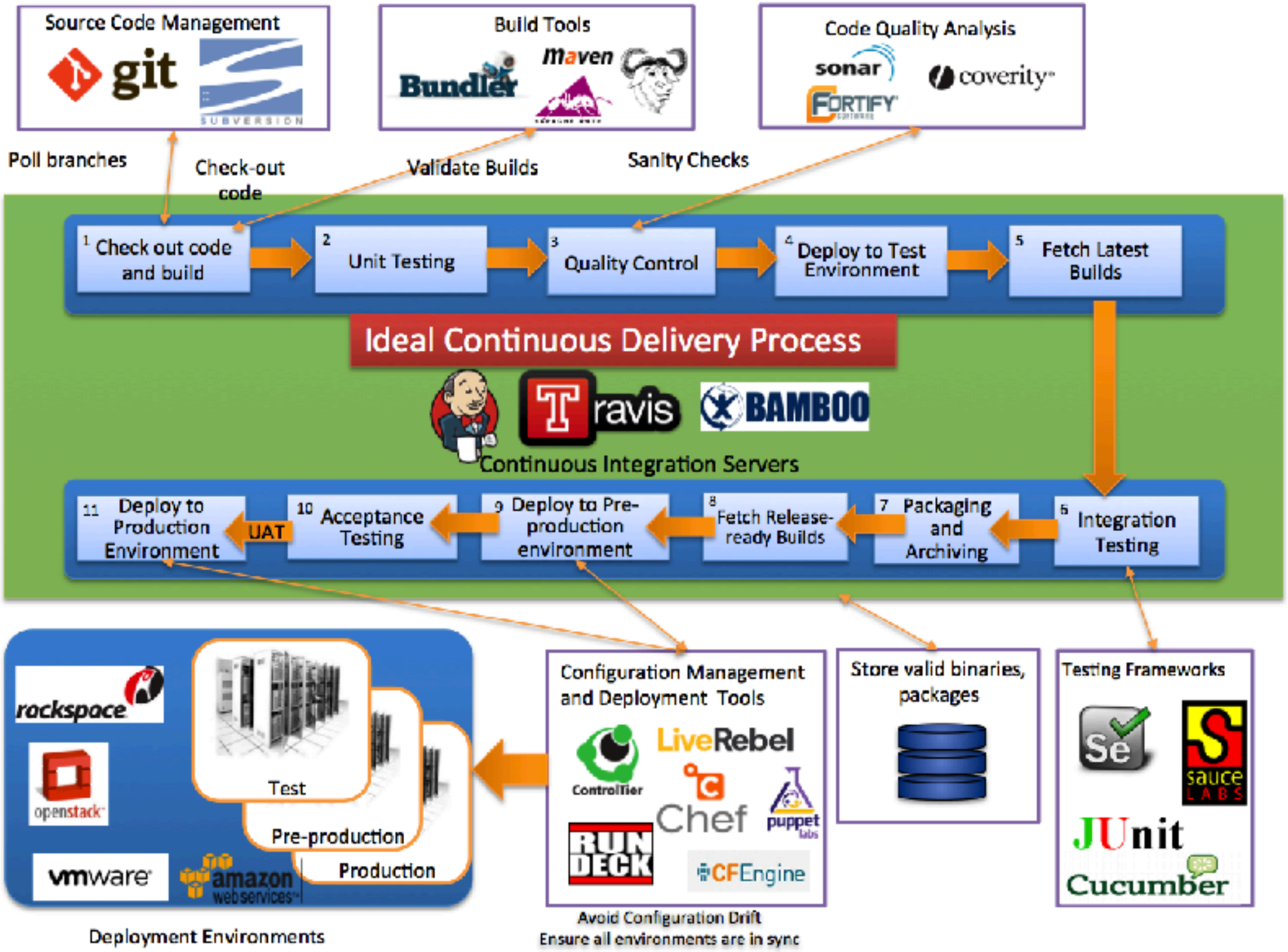
15. DevOps View



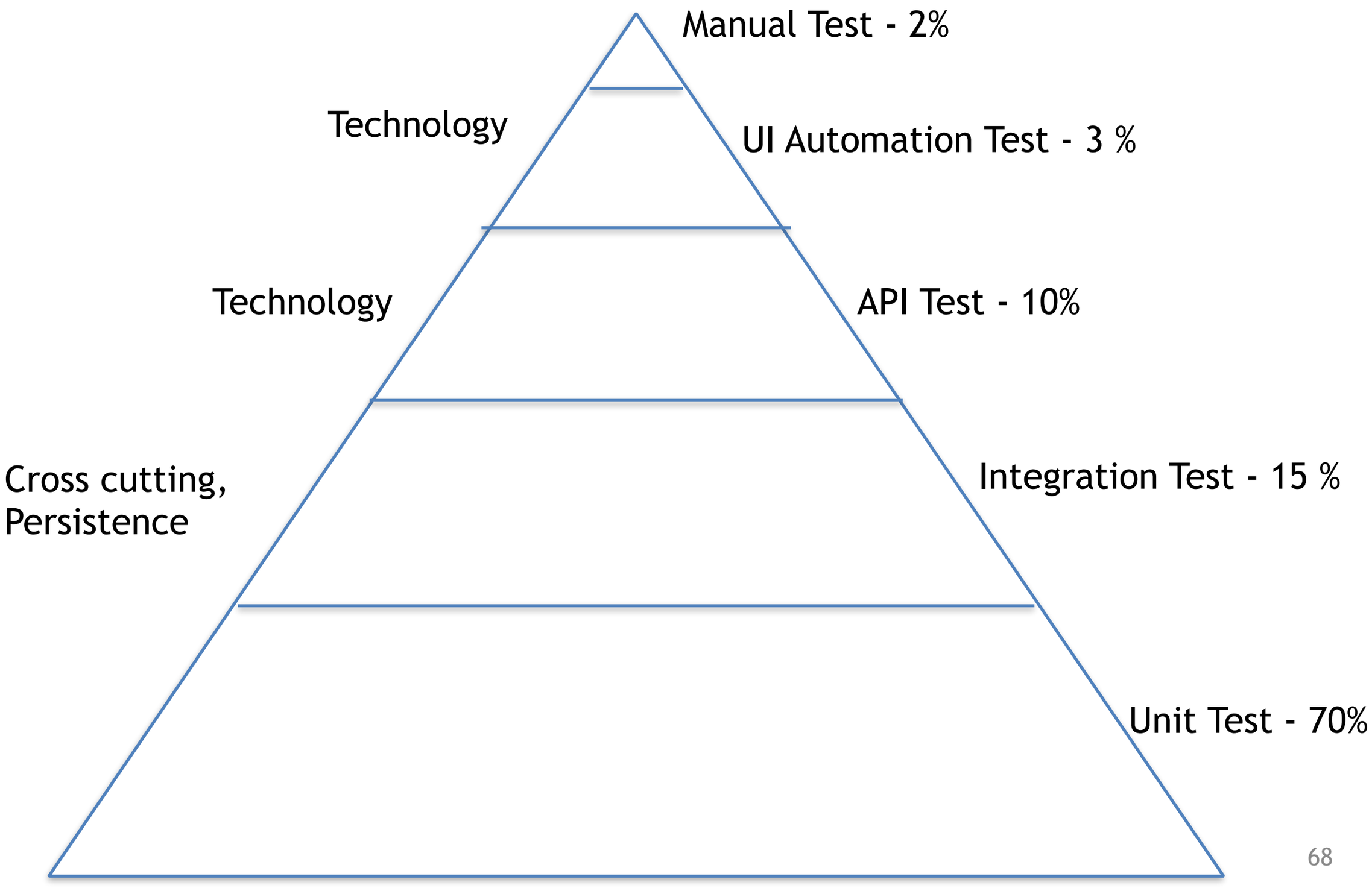
Step 1. Containerizing



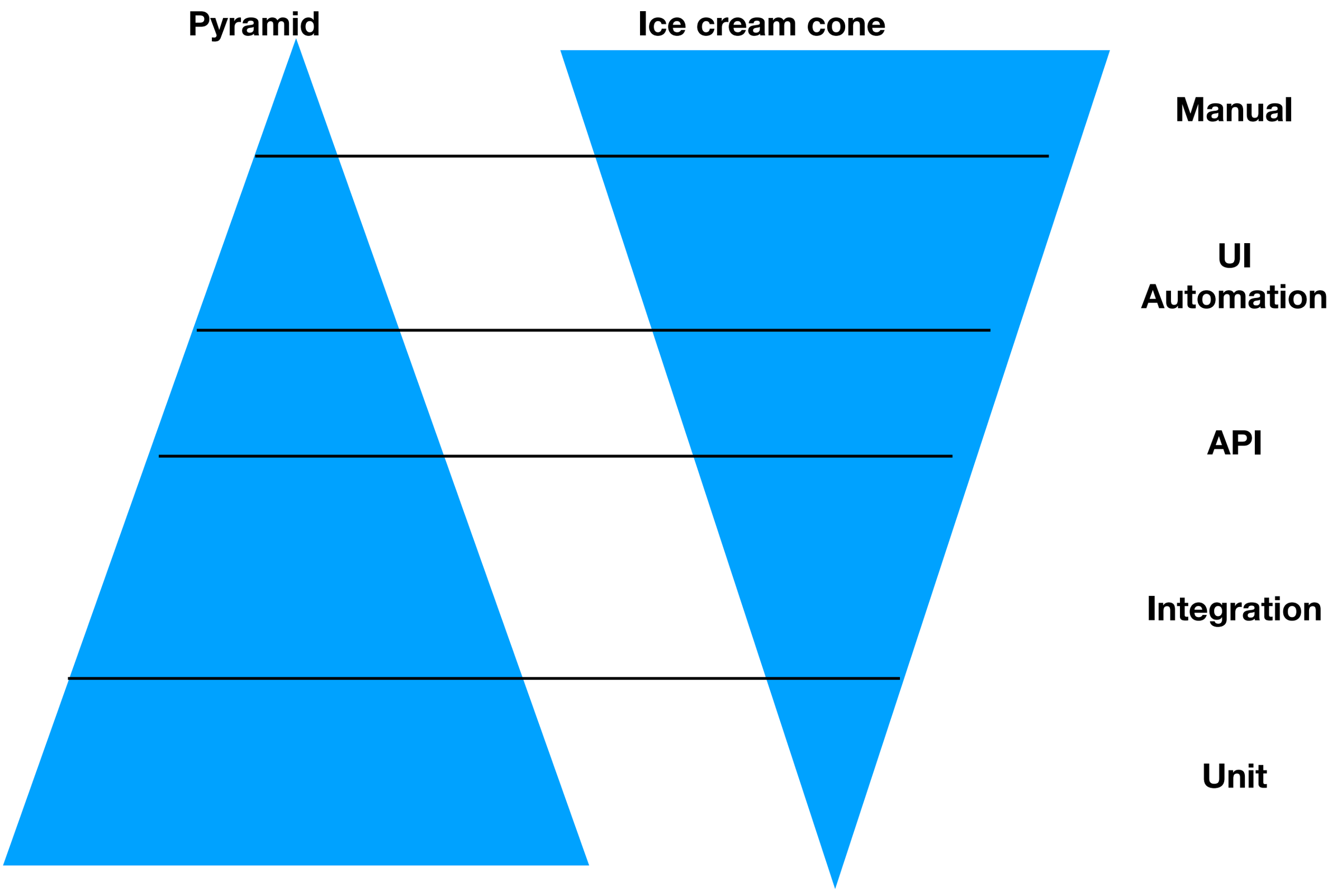
Step 2. Integrating infrastructure automation with CI/CD

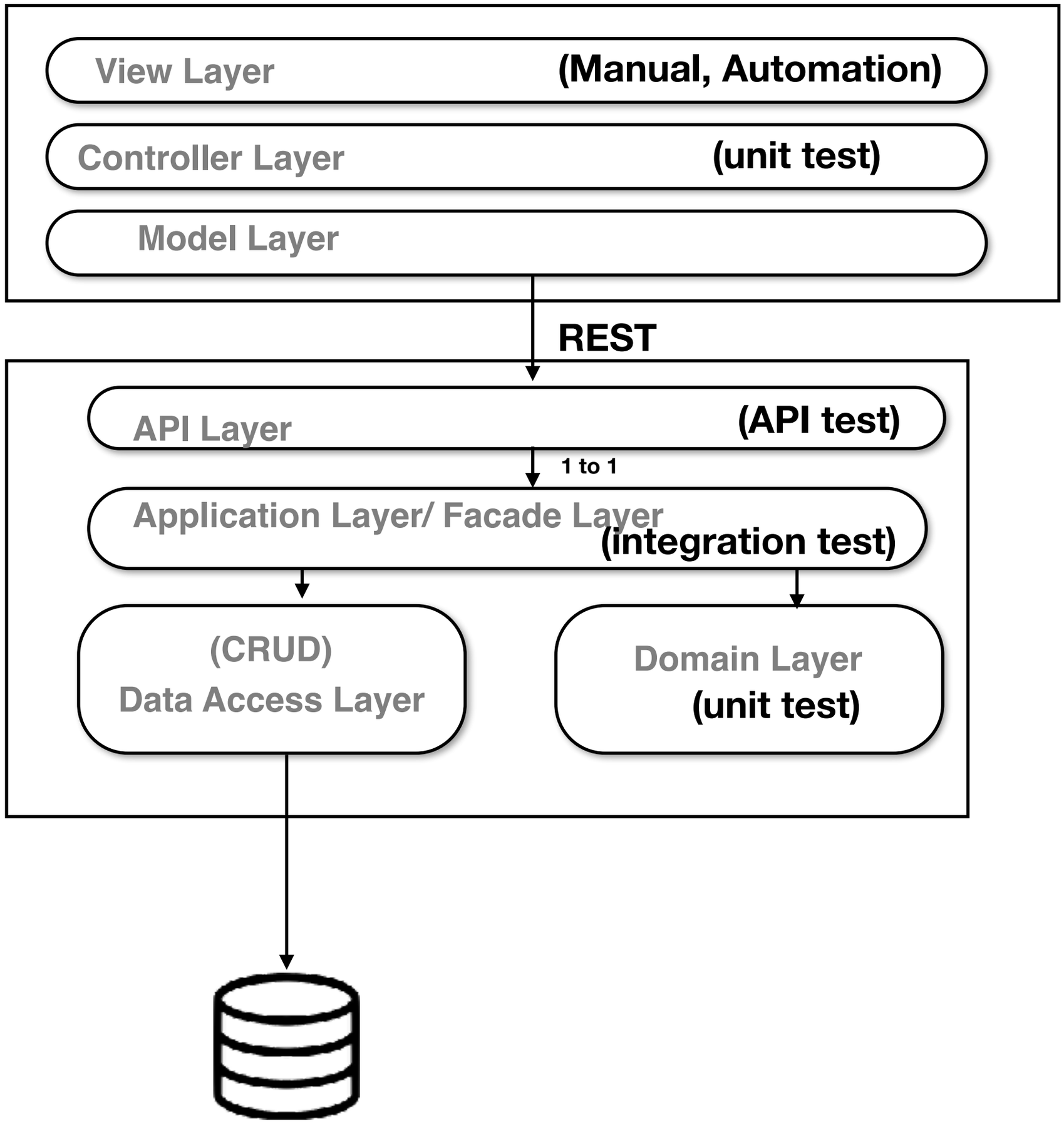


Step 3. test automation and aligning QA with development

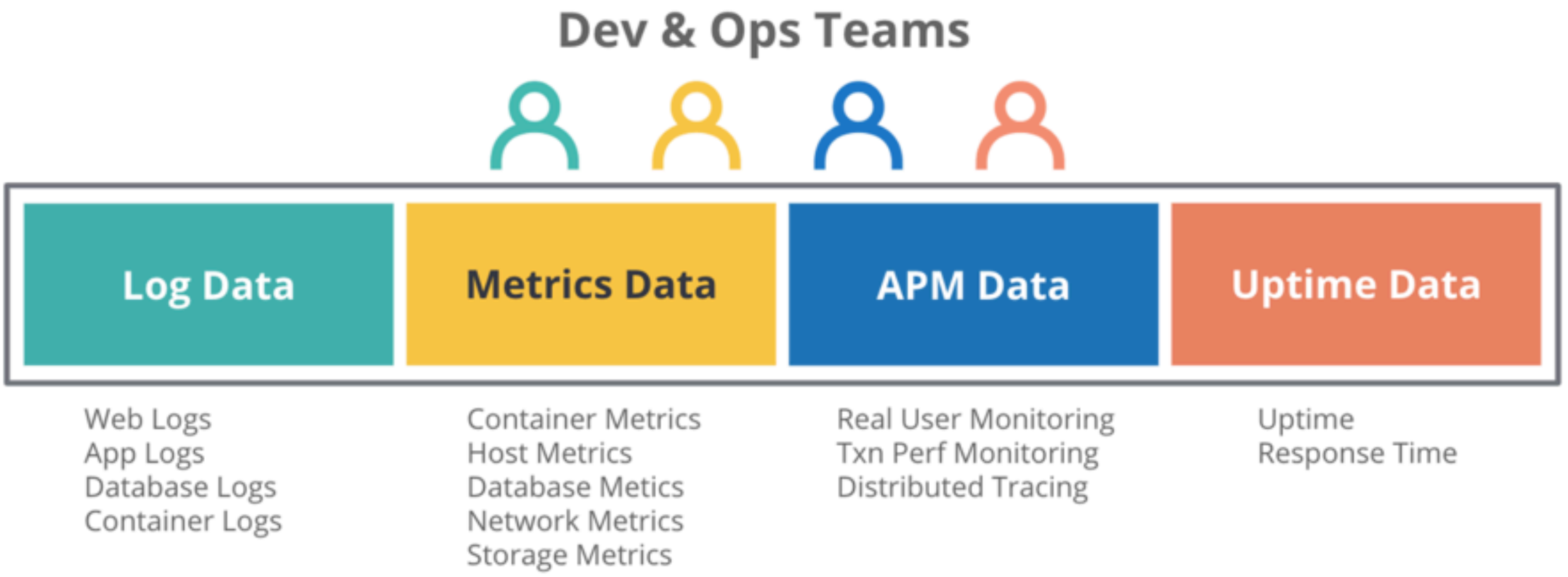


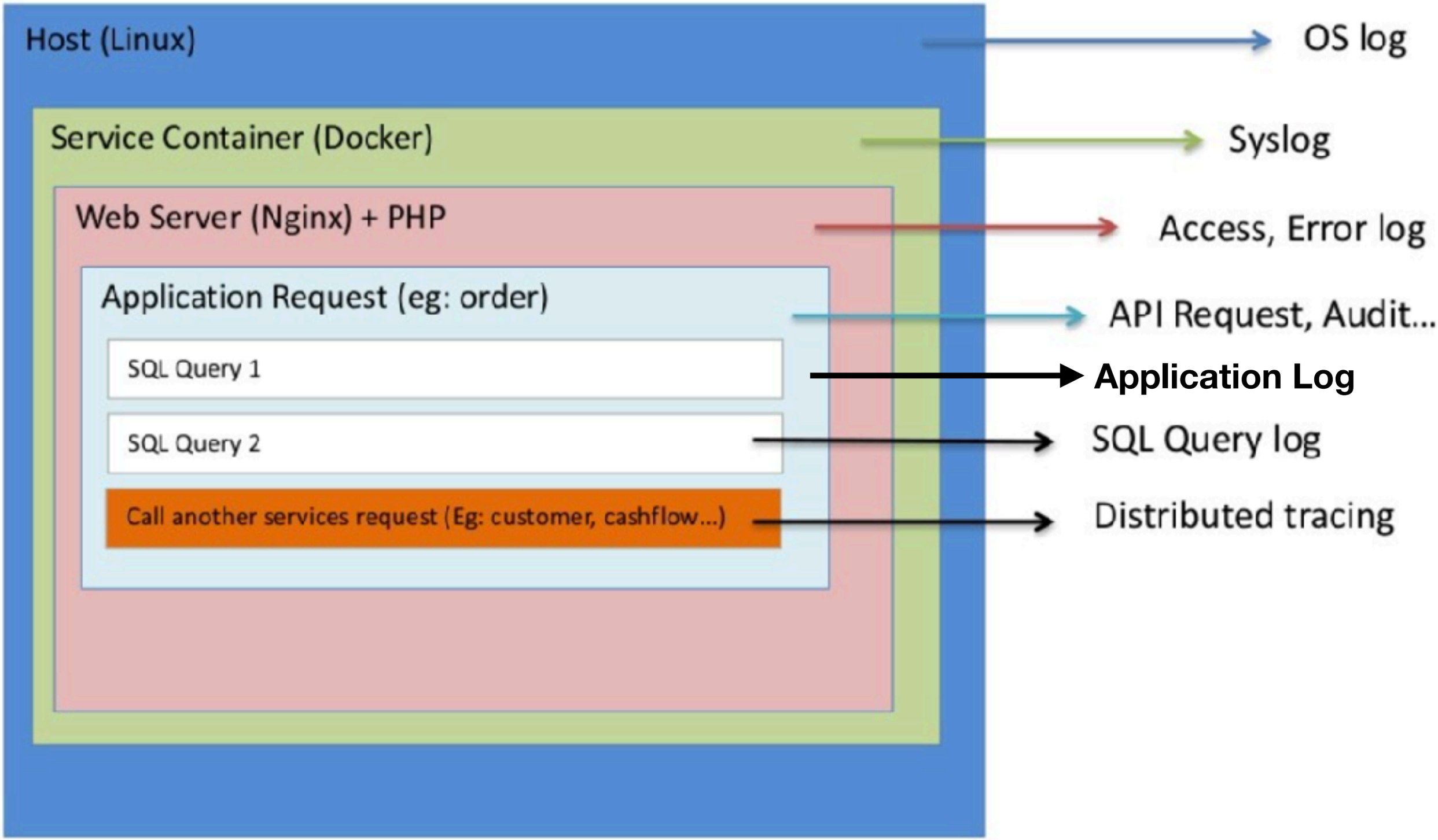
Pyramid Test

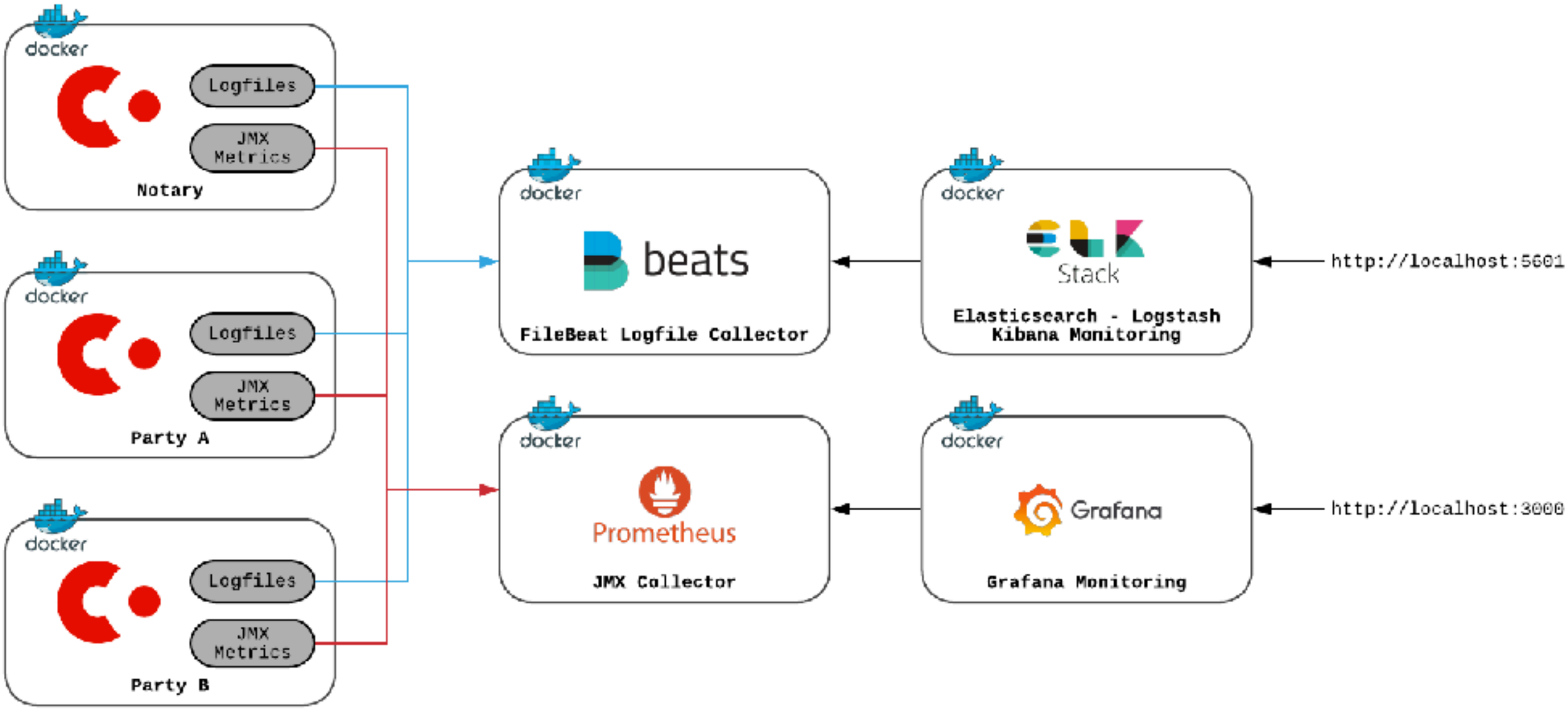



View Layer
(Manual, Automation)
Controller Layer
(unit test)
Model Layer
REST
API Layer
(API test)
1 to 1
Application Layer/ Facade Layer
(integration test)
(CRUD)
Data Access Layer
Domain Layer
(unit test)

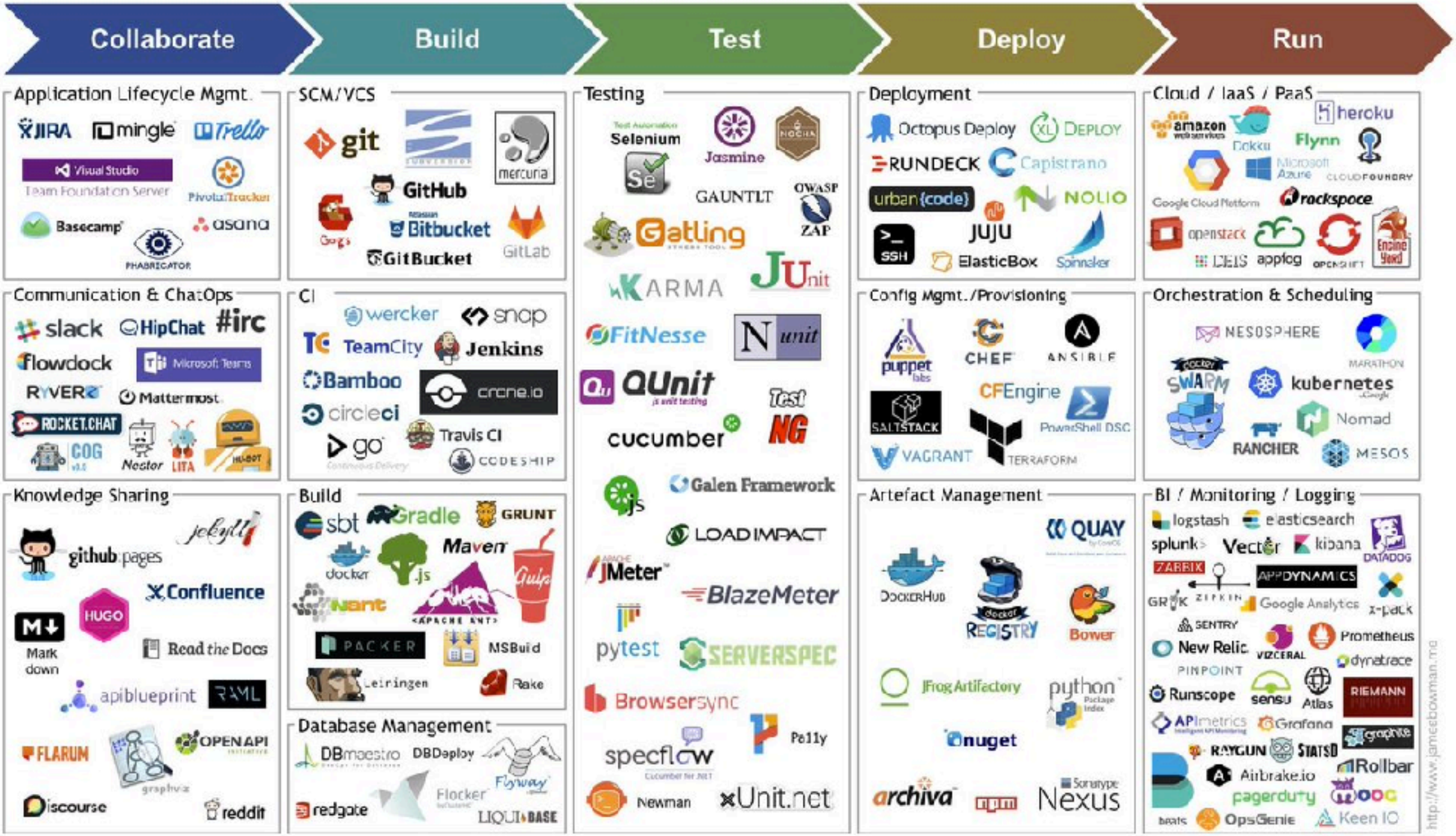
Step 4. Setup Monitoring



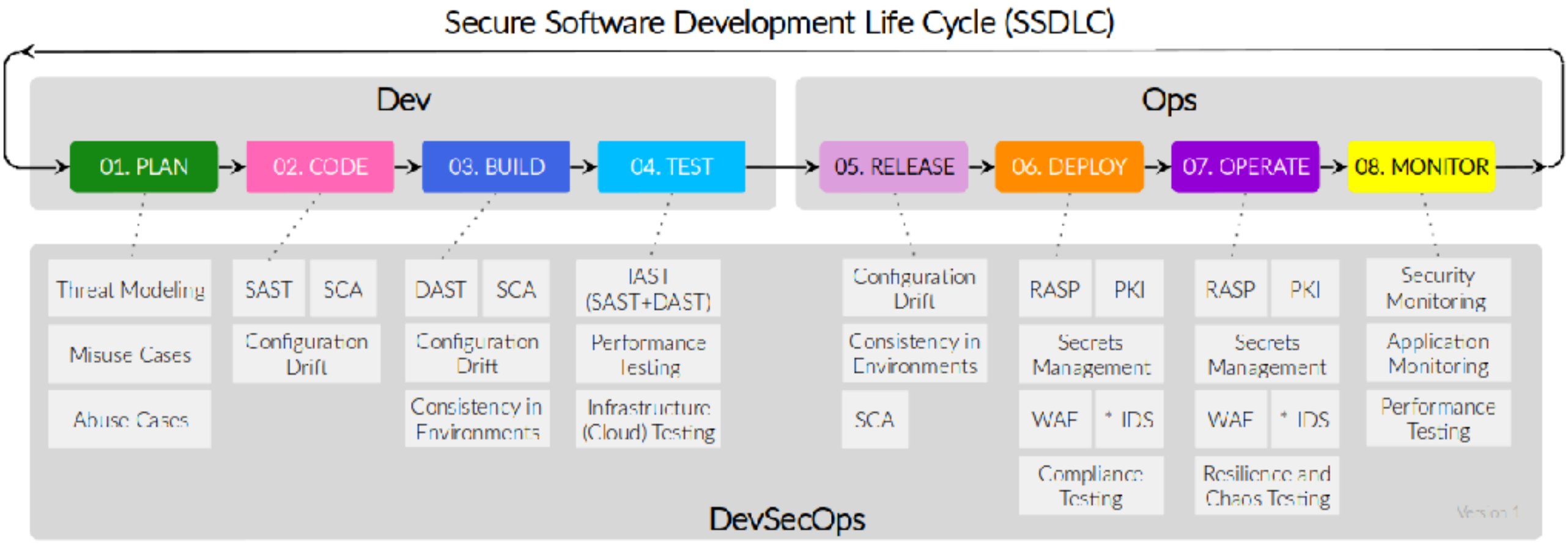

Host (Linux)
Service Container (Docker)
Web Server (Nginx) + PHP
Application Request (eg: order)
SQL Query 1
SQL Query 2
Call another services request (Eg: customer, cashflow...)
OS log
Syslog
Access, Error log
API Request, Audit...
Application Log
SQL Query log
Distributed tracing


docker
Logfiles
JMX Metrics
Notary
docker
Logfiles
JMX Metrics
Party A
docker
Logfiles
JMX Metrics
Party B
docker
beats
FileBeat Logfile Collector
docker
Prometheus
JMX Collector
docker
Stack
Elasticsearch - Logstash Kibana Monitoring
http://localhost:5601
docker
Grafana
Grafana Monitoring
http://localhost:3000

Reference


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
Bitbucket
GitBucket
GitLab
Testing
Selenium
Jasmine
NOCHA
GAUNTLT
OWASP ZAP
Gatling
JUnit
KARMA
FitNesse
N unit
QUnit
cucumber
Test NG
Galen Framework
LOAD IMPACT
JMeter
BlazeMeter
pytest
SERVERSPEC
Browsersync
Pa11y
specflow
Newman
xUnit.net
Deployment
Octopus Deploy
DEPLOY
RUNDECK
Capistrano
urban{code}
NOLIO
SSH
JUJU
ElasticBox
Spinnaker
Cloud / IaaS / PaaS
amazon web services
Heroku
Flynn
Docker
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
HU-BOT
CI
wercker
snap
TeamCity
Jenkins
Bamboo
circleci
Travis CI
go
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
Gulp
Nant
APACHE ANT
PACKER
MSBuild
Leiningen
Rake
Database Management
DBmaestro
DBDeploy
Flyway
redgate
Flocker
LIQUIBASE
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

Step 5. SecOps



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

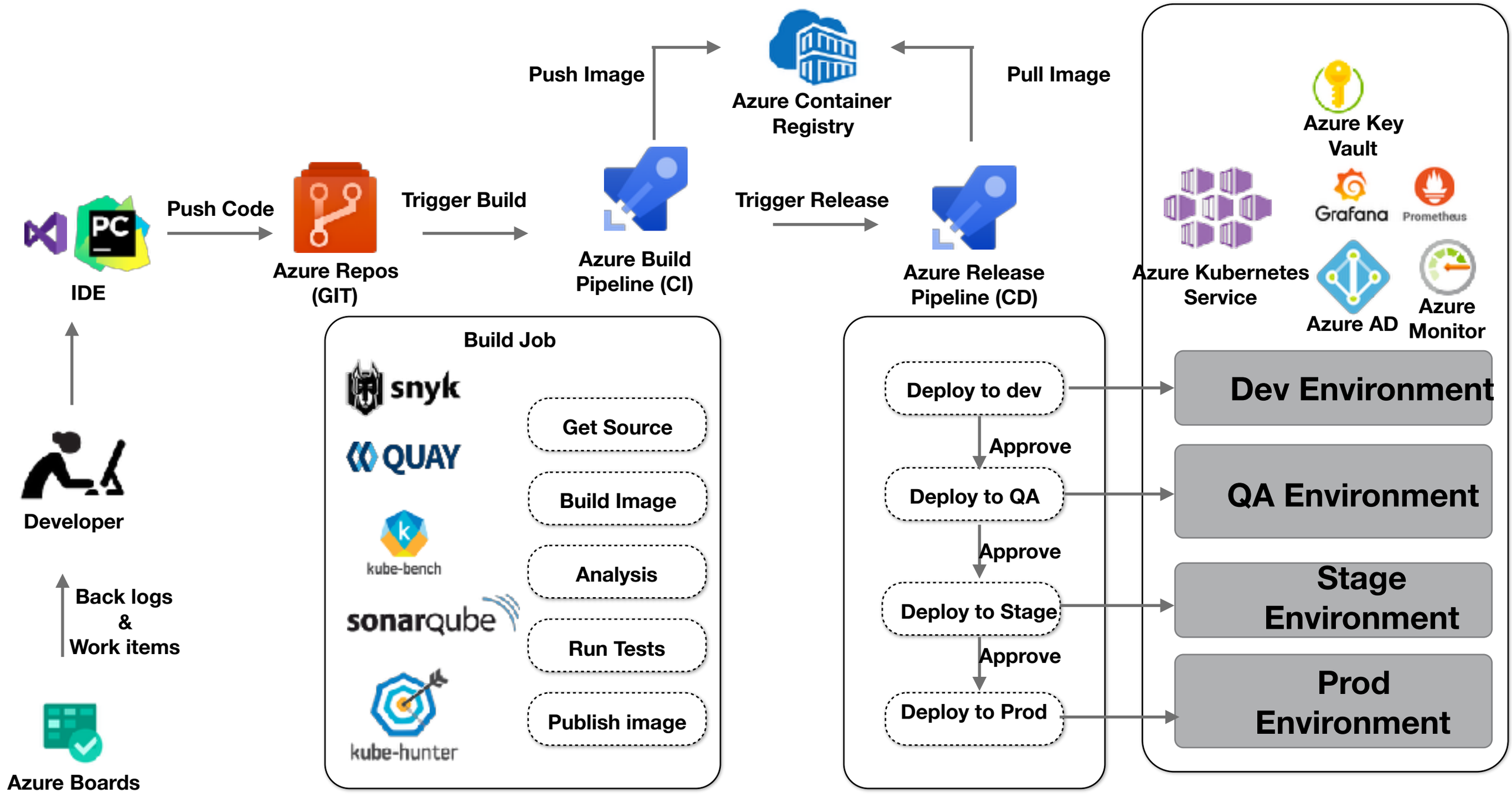
Chaos Engineering

Gremlin, ChaosToolkit

Vulnerability management

Hacker, ARCHERY

Reference model



Part 3

Arch Risks

	Risk	I	L	R	C	Notes
1	Performance of the View Todo is significantly impacted by large data volumes.	8	4	32	High	Testing to be undertaken to gauge impact.
2	Limitations of JPA API result in complex workarounds in repository implementation or workarounds in other architectural layers.	6	4	24	Medium	Can fall back to Hibernate which provides richer ORM functionality.
3	The Open Source tools and components selected lack the capabilities of equivalent commercial products limiting the features that can be implemented (e.g. GIS components).	6	3	18	High	Investigate alternative products.

- Risk – The description of the architectural risk.
- Impact to Project – The impact the risk will have on the project
    - (1 = Minor impact, 10 = Showstopper)
- Likelihood – The likelihood of the risk eventuating (1 = Low, 953 = Highly likely)
- Rating – The overall rating for the risk based on Impact * Likelihood (1 = Minor, 90 = Critical)
- Confidence – The confidence in resolving the issue should it eventuate (Low, Medium, High)