Software Architecture 2

Software Engineering
Prof. Maged Elaasar

Categories of Architecture Patterns

1

Application Landscape

2

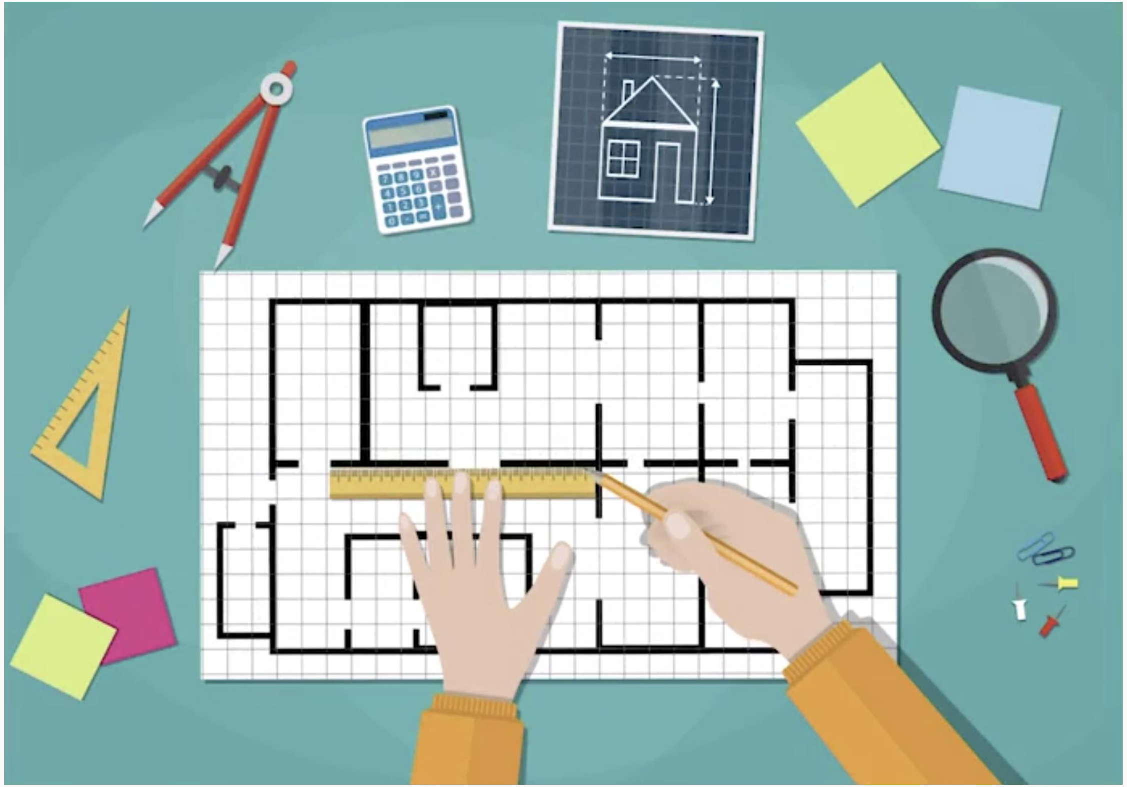
Application Structure

3

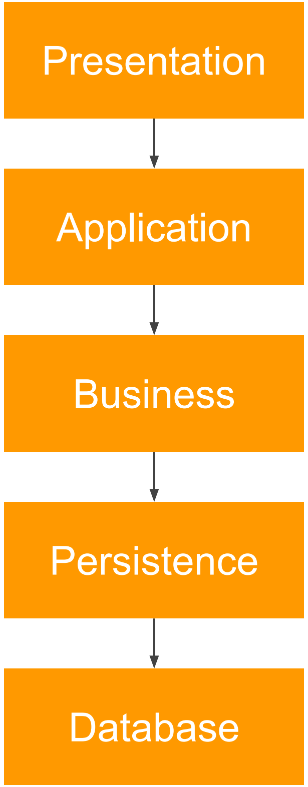
User Interface

2. Application Structure Patterns

- Layered
- Microkernel
- CQRS
- Event Sourcing



Layered



Presentation	User Interface
Application	Translation between UI and business
Business	Business logic (APIs)
Persistence	Code to map from database to business objects
Database	Storage

Layered

1

Advantages

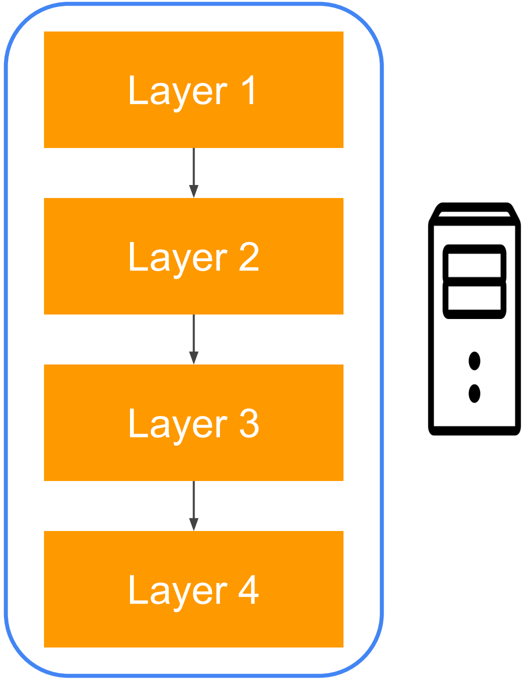
- Well-known among developers
- Easy to recognize
- Separation of concerns

2

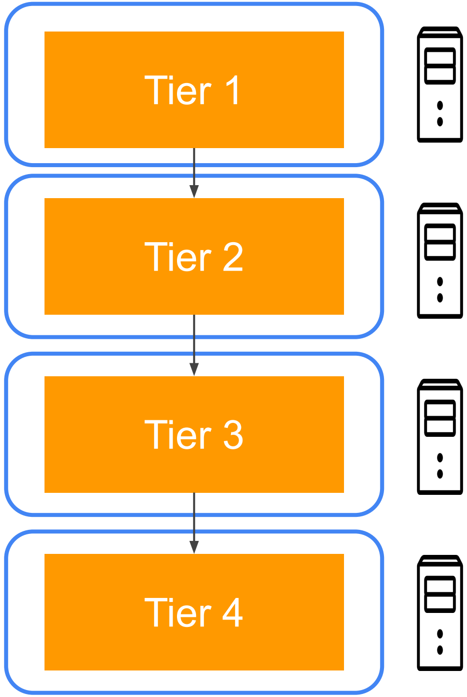
Disadvantages

- Can lead to monolith applications
- Need to write lots of code to hand off between layers
- Can suffer form sinkhole anti-pattern

Layered vs. N-Tier Architectures



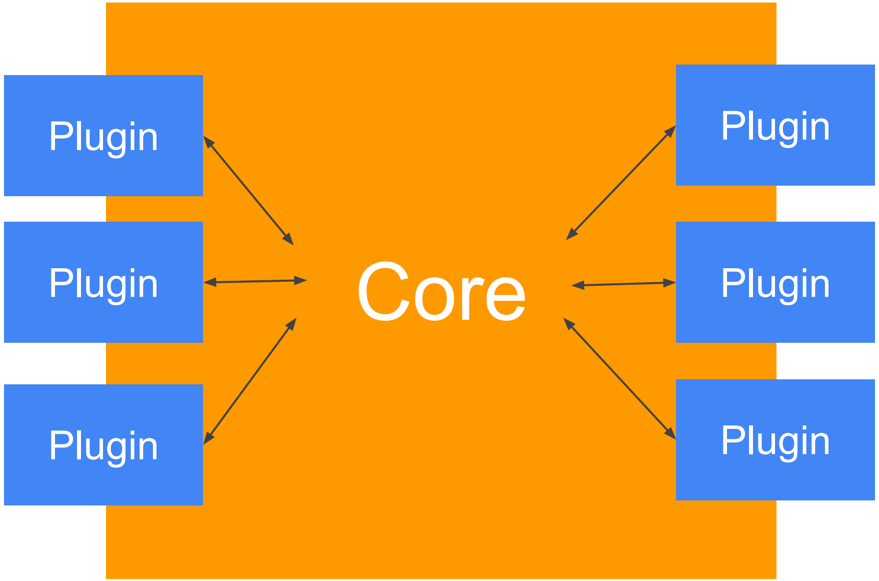
Layered Architecture



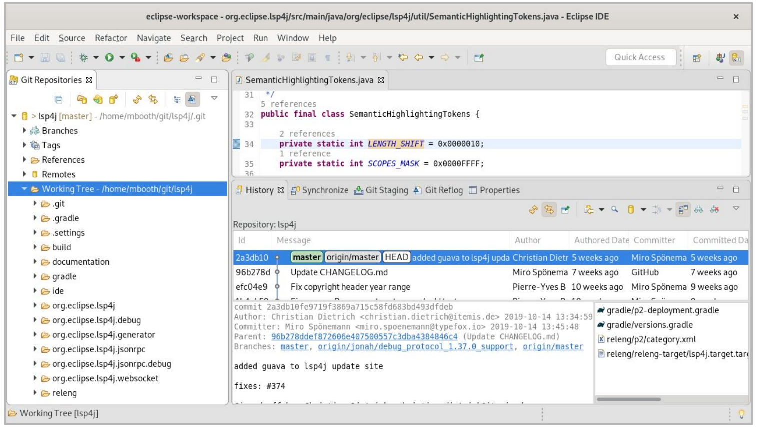
N-Tier Architecture

Microkernel

- Task scheduler (tasks)
- Workflow (steps)
- Browser (extensions)
- Graphic designer (e.g., filters)
- IDE (for different languages)



Microkernel Example: Eclipse


eclipse-workspace - org.eclipse.lsp4j/src/main/java/org/eclipse/lsp4j/util/SemanticHighlightingTokens.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Git Repositories
SemanticHighlightingTokens.java
History
Synchronize
Git Staging
Git Reflog
Properties

> lsp4j [master] - /home/mbooth/git/lsp4j/.git
- Branches
- Tags
- References
- Remotes
- Working Tree - /home/mbooth/git/lsp4j
  - .git
  - .gradle
  - .settings
  - build
  - documentation
  - gradle
  - ide
  - org.eclipse.lsp4j
  - org.eclipse.lsp4j.debug
  - org.eclipse.lsp4j.generator
  - org.eclipse.lsp4j.jsonrpc
  - org.eclipse.lsp4j.jsonrpc.debug
  - org.eclipse.lsp4j.websocket
  - releng

Working Tree [lsp4j]

```
 */
5 references
public final class SemanticHighlightingTokens {

    2 references
    private static int LENGTH_SHIFT = 0x0000010;
    1 reference
    private static int SCOPES_MASK = 0x0000FFFF;
```

Repository: lsp4j

Id	Message	Author	Authored Date	Committer	Committed Da
2a3db10	master origin/master HEAD added guava to lsp4j upda	Christian Dietr	5 weeks ago	Miro Spönema	5 weeks ago
96b278d	Update CHANGELOG.md	Miro Spönema	7 weeks ago	GitHub	7 weeks ago
efc04e9	Fix copyright header year range	Pierre-Yves B	10 weeks ago	Miro Spönema	9 weeks ago

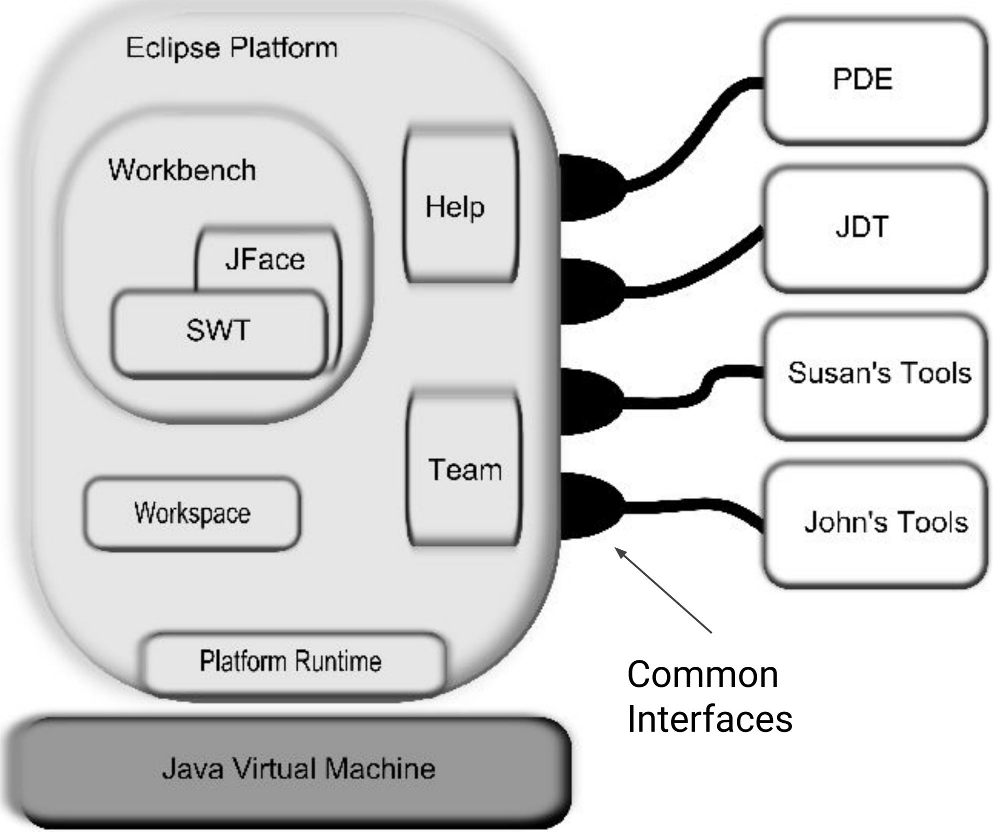
```
commit 2a3db10fe9719f3869a715c58fd683bd493dfdeb
Author: Christian Dietrich <christian.dietrich@itemis.de> 2019-10-14 13:34:59
Committer: Miro Spönemann <miro.spoenemann@typefox.io> 2019-10-14 13:45:48
Parent: 96b278ddef872606e407500557c3dba4384846c4 (Update CHANGELOG.md)
Branches: master, origin/jonah/debug_protocol_1.37.0_support, origin/master

added guava to lsp4j update site

fixes: #374
```

- gradle/p2-deployment.gradle
- gradle/versions.gradle
- releng/p2/category.xml
- releng/releng-target/lsp4j.target.tar

Eclipse



Microkernel

1

Advantages

- Flexibility
- Clean separation
- Separate teams possible
- Add/remove functionality at runtime

2

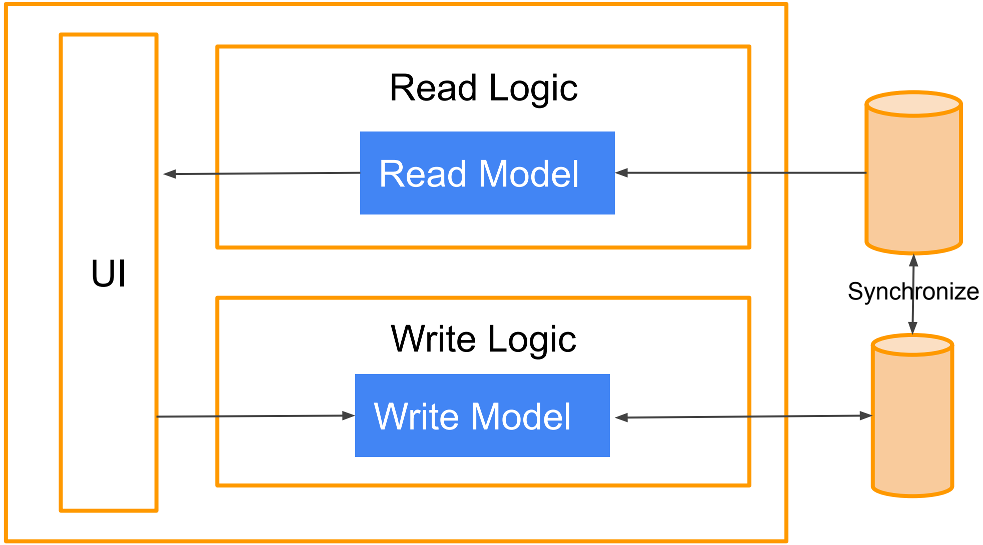
Disadvantages

- Core API might not fit future plugins
- Plugins may not be trusted
- Not always clear what should belong to the core vs. plugins

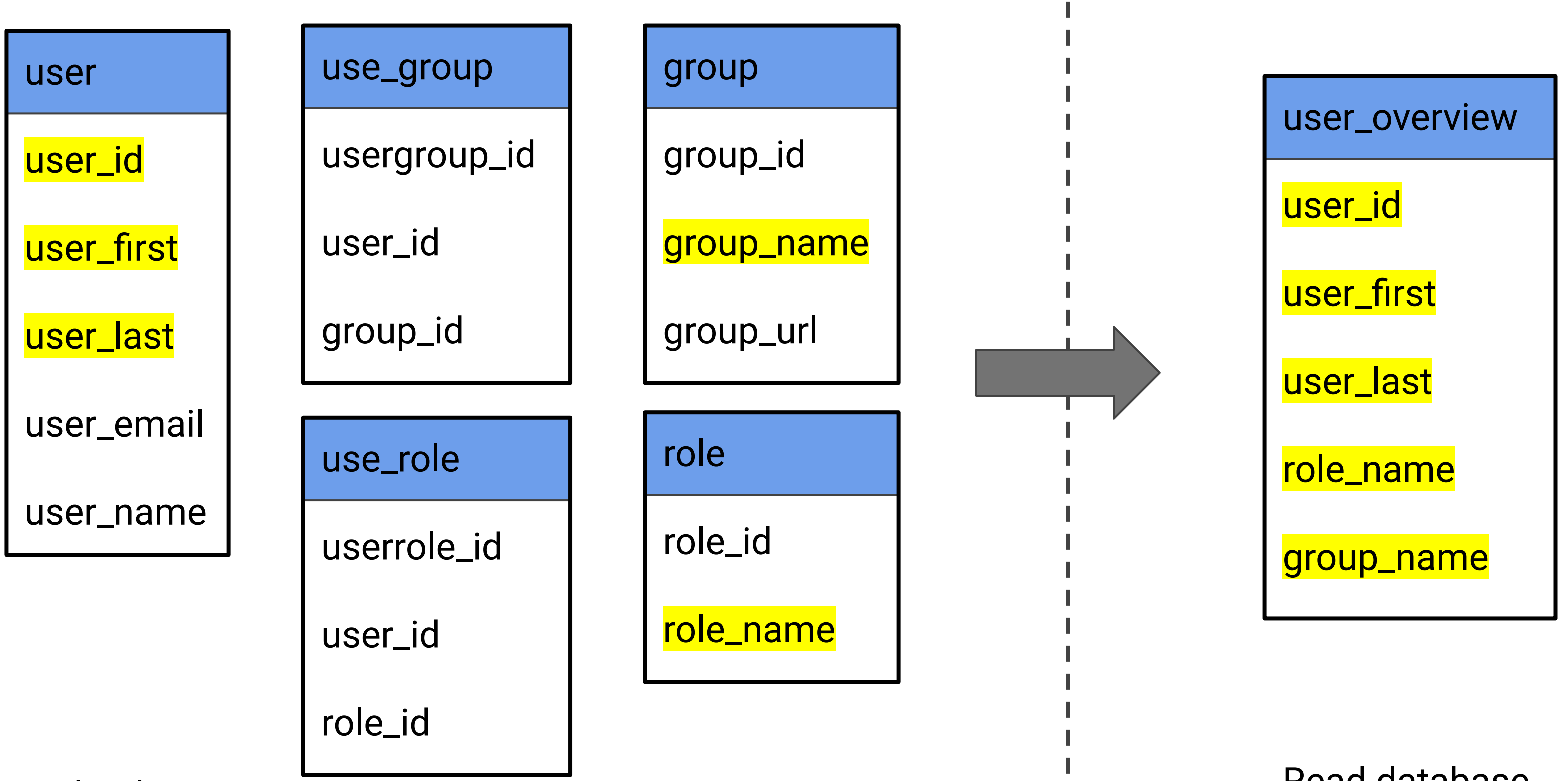
CQRS

- **C**ommand **Q**uery **R**esponsibility **S**egregation
- Separates read and update operations for a data store
- Optimizes scenario-specific queries
- Synchronization is required

updates and queries have different requirements for throughput, latency, or consistency



CQRS Simpler Queries



user
user_id
user_first
user_last
user_email
user_name

use_group
usergroup_id
user_id
group_id

use_role
userrole_id
user_id
role_id

group
group_id
group_name
group_url

role
role_id
role_name

user_overview
user_id
user_first
user_last
role_name
group_name

Write database

Read database

CQRS

1

Advantages

- Simple read queries
- Faster/more scalable read queries

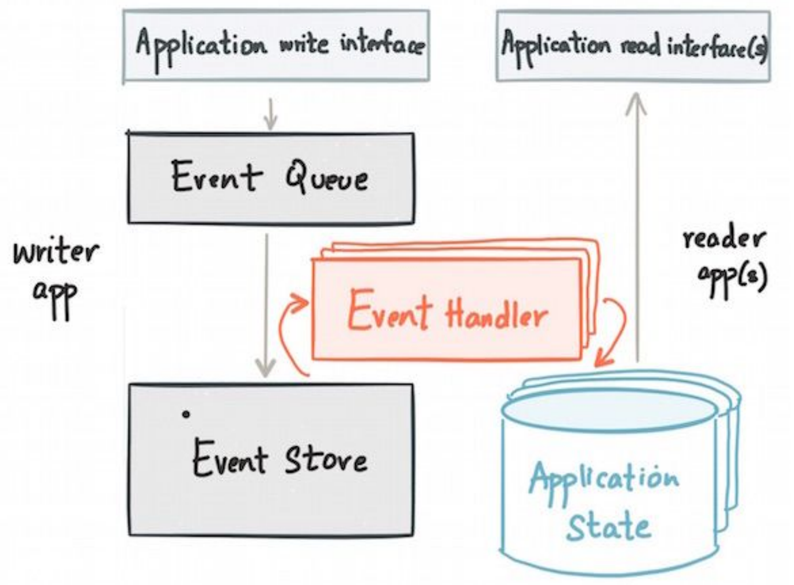
2

Disadvantages

- Added complexity
- Learning curve
- Possibility of data inconsistencies
- Eventual consistency

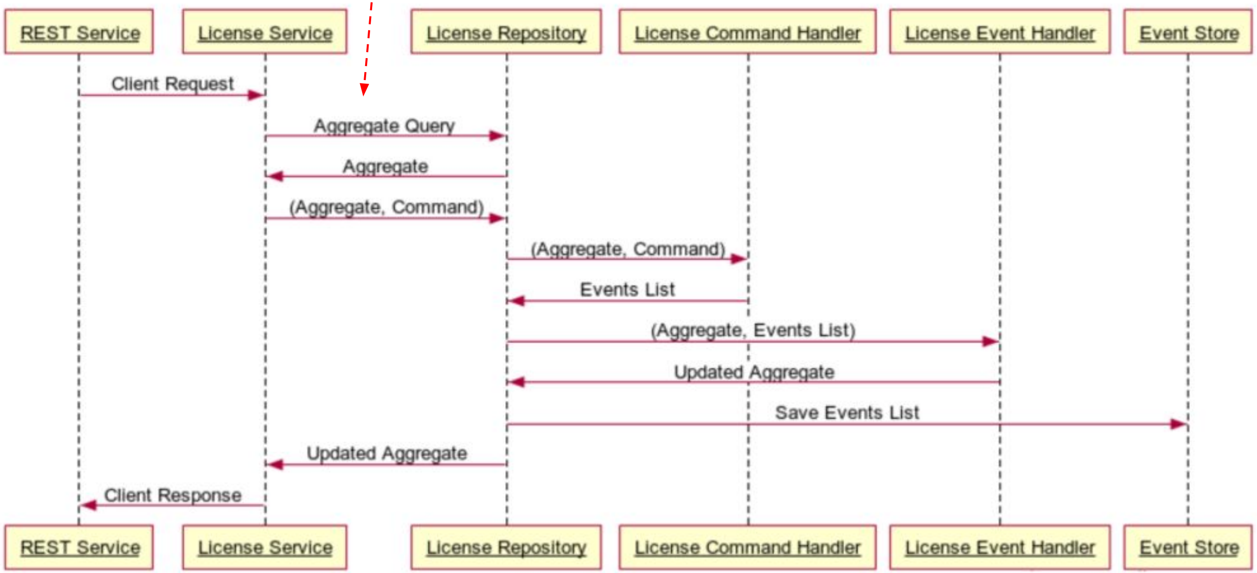
Event Sourcing

- Store events instead of current state
- Event = the result of executing commands
- Construct read database from events
- Can replay of events from event store



Event Sourcing Example: Netflix

"Aggregate" is the read state (the license)



NETFLIX

License Acquisition Scenario

Event Sourcing

1

Advantages

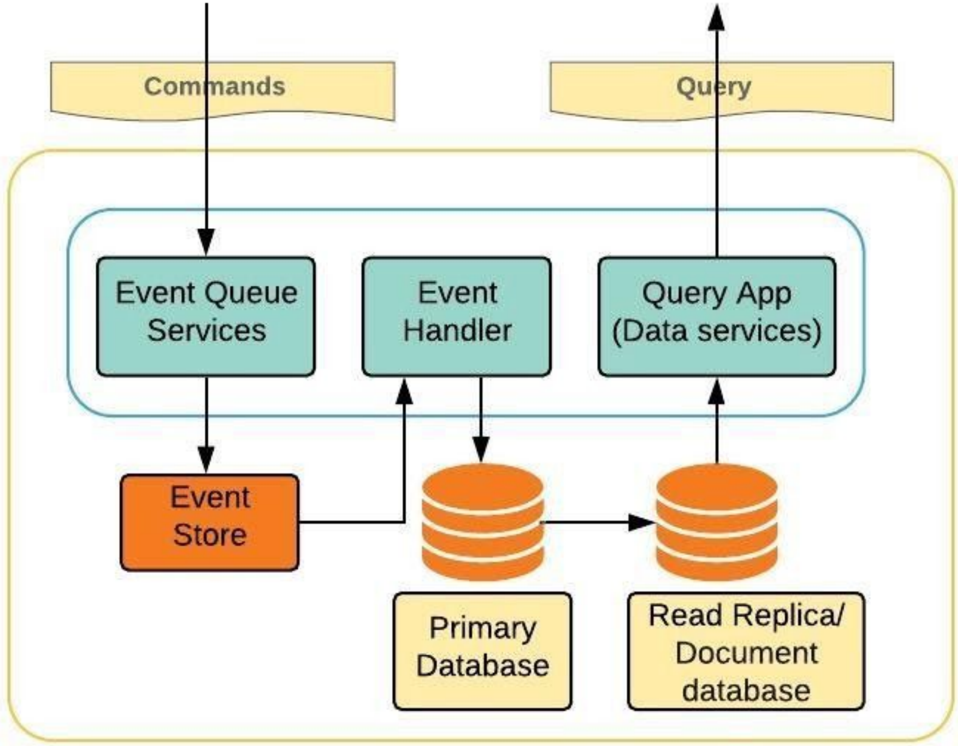
- Audit trail: trace of events
- Event replay

2

Disadvantages

- When event structure changes
- Having to take snapshots
- Learning curve

CQRS and Event Sourcing Combined



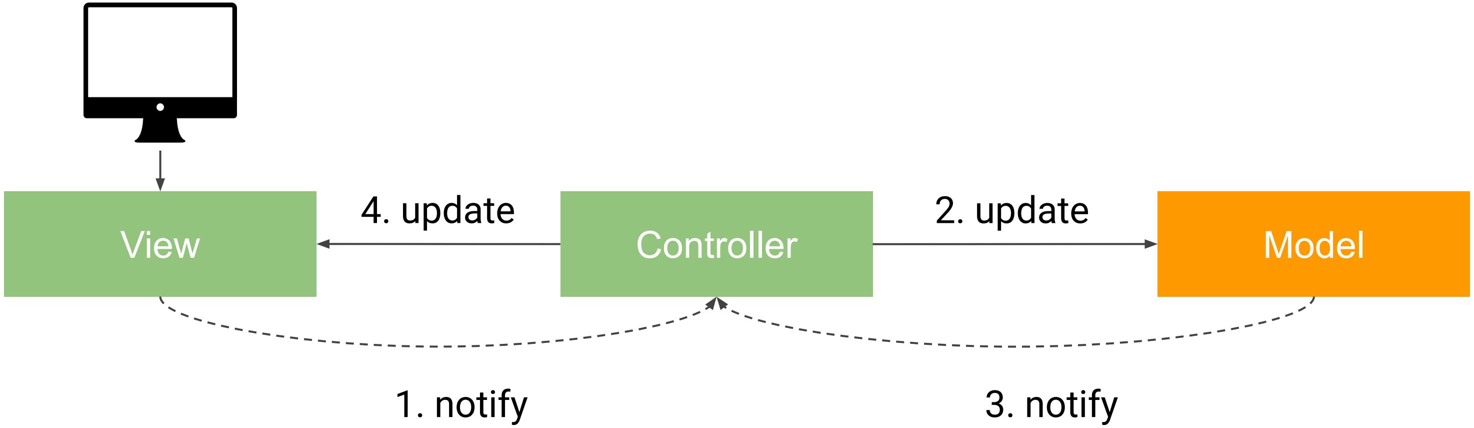
Event Sourcing

CQRS

3. User Interface Patterns

- Model-View-Controller (MVC)
- Model-View-Presenter (MVP)
- Model-View-ViewModel (MVVM)

Model-View-Controller



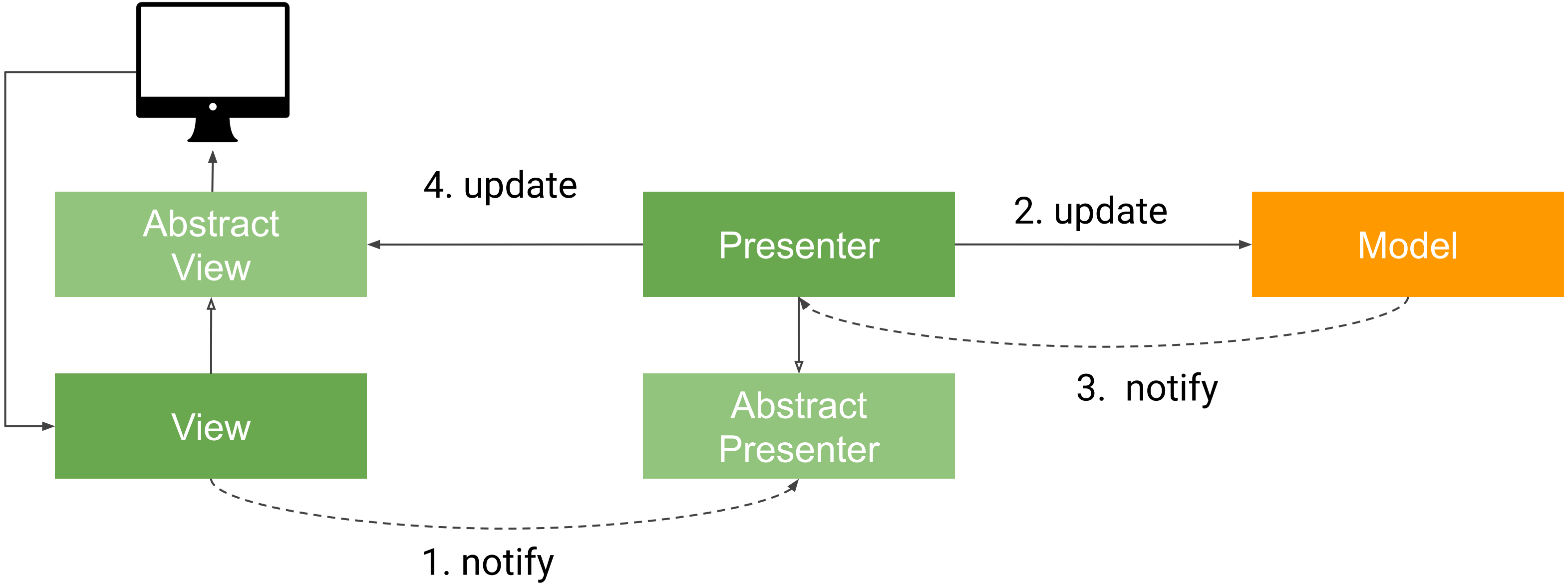
Advantages

- Separation of concerns
- Parallel development
- Popular in web frameworks

Disadvantages

- Controllers can become bloated
- Controller is coupled with view

Model-View-Presenter



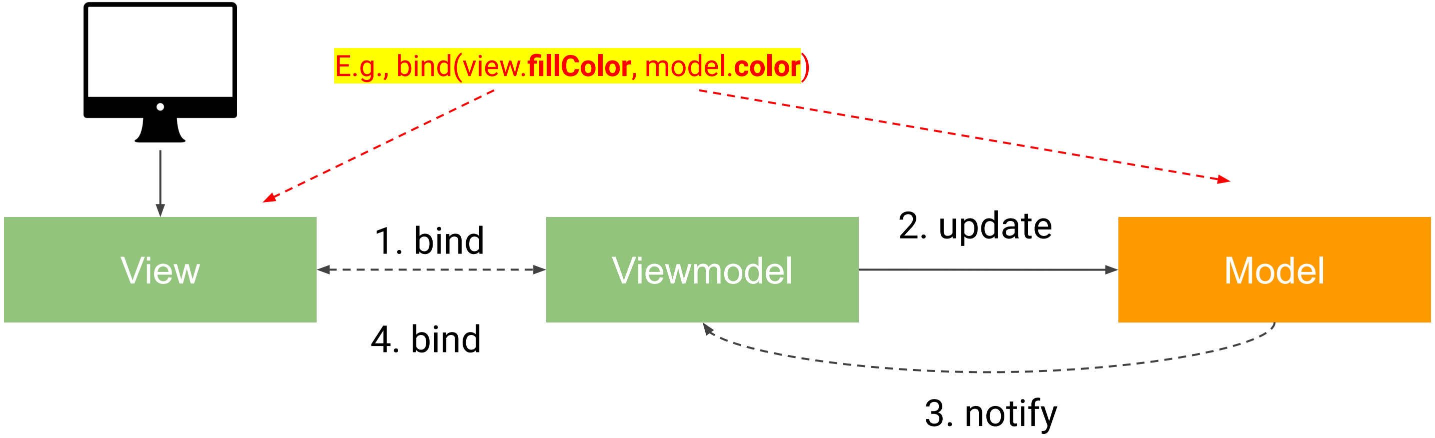
Advantages

- Decoupling of viewer and presenter

Disadvantages

- Presenters can become bloated

Model-View-Viewmodel



Advantages

- Simpler to develop, maintain & test

Disadvantages

- Overkill for simple user interfaces
- Hard to use for complex mappings

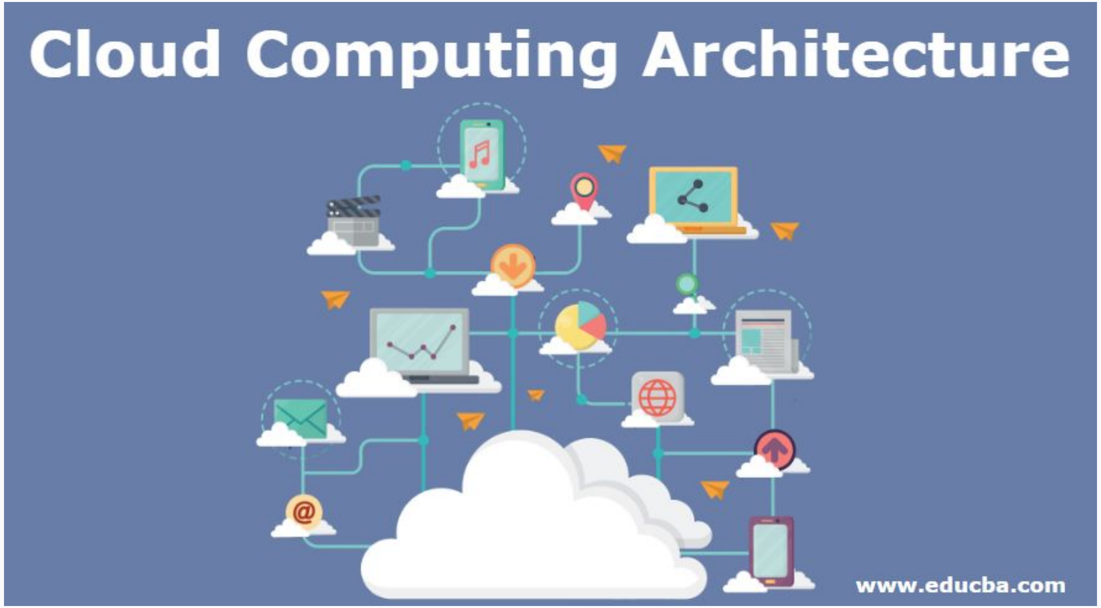
Application Structure & UI Patterns Quiz

4. Cloud Based Architecture

- Scriptable infrastructure
- Improved development life cycle
- Unconstrained resources
- On-demand scaling
- High availability and disaster recovery
- Shared security model
- Optimized cost

IBM Cloud

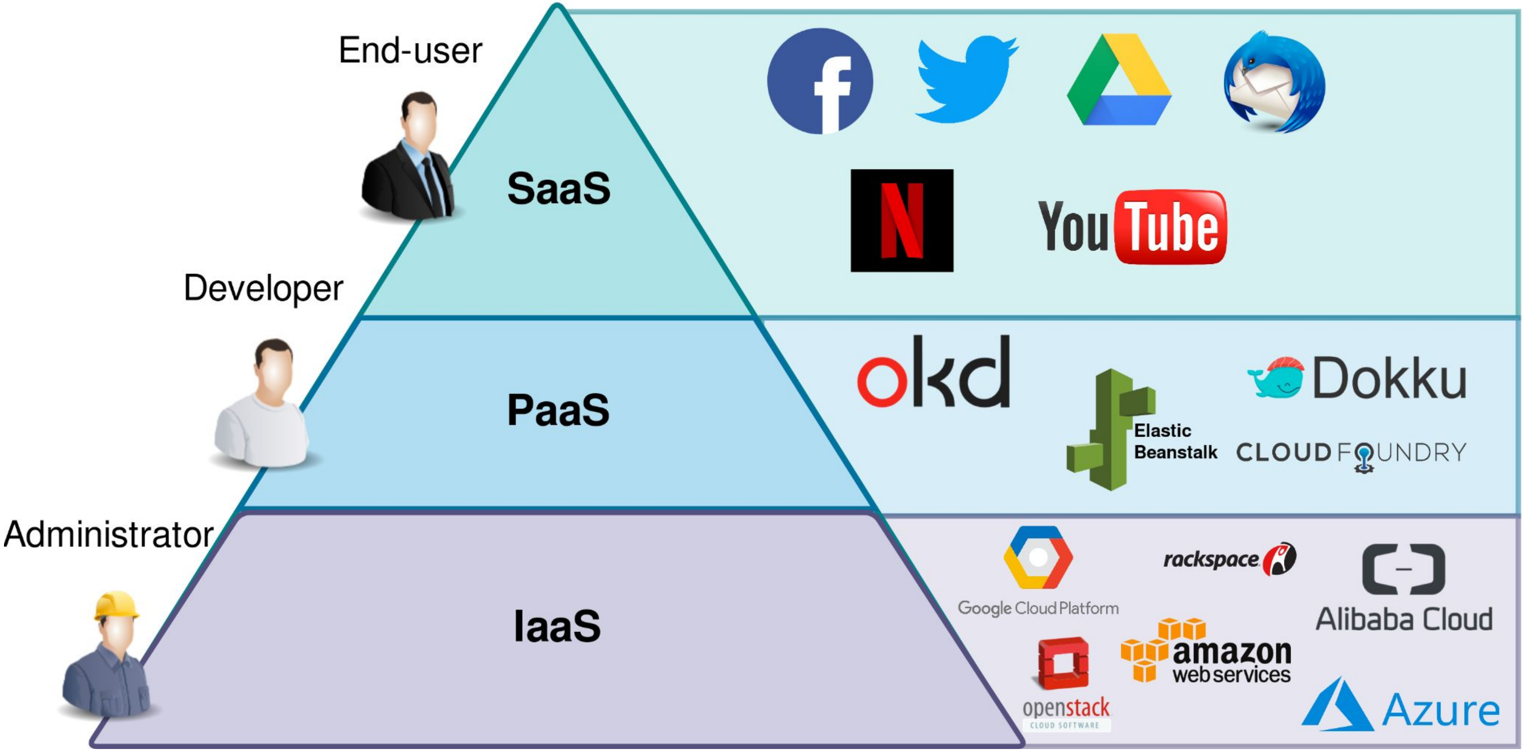
Google Cloud



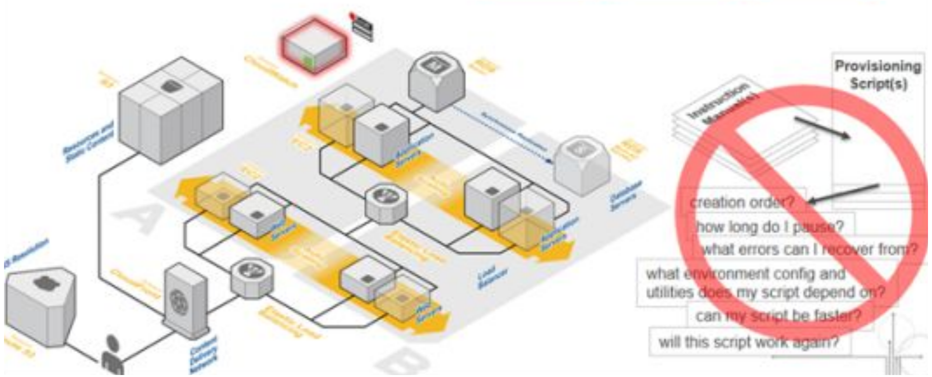
aws

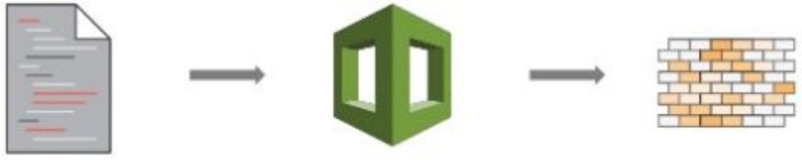
Azure

Cloud Services

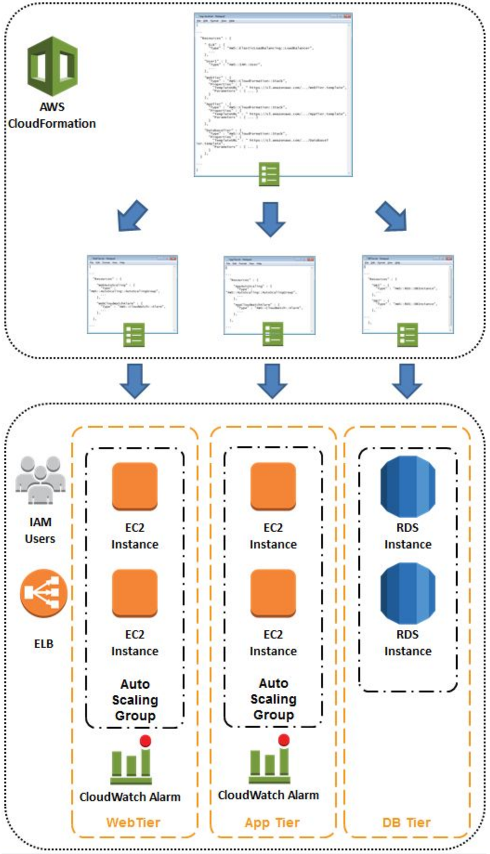


Scriptable Infrastructure

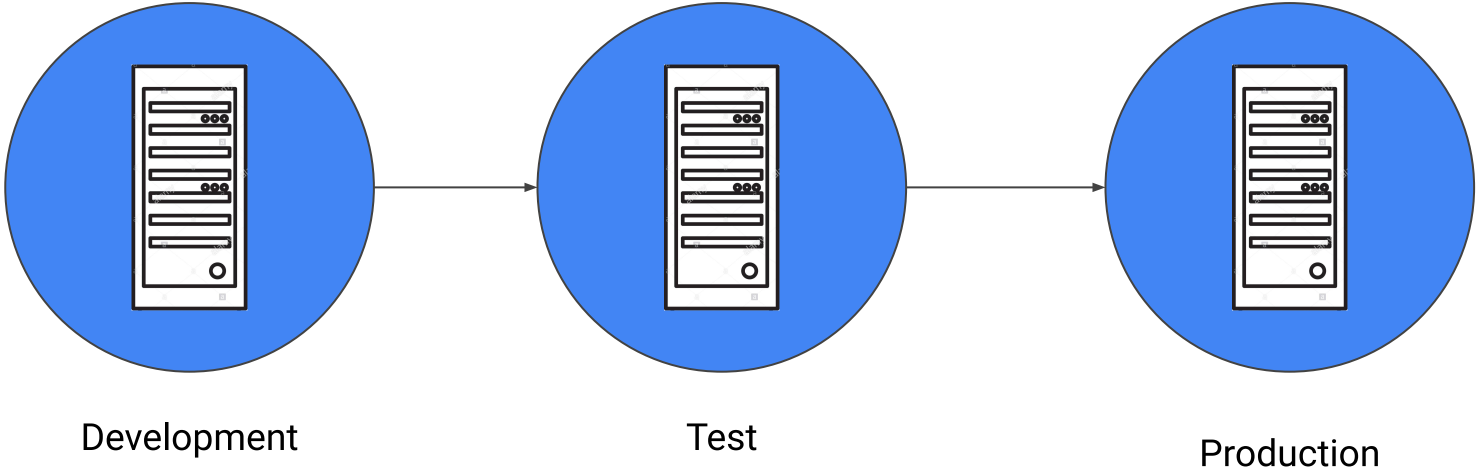

Provisioning Script(s)
Instruction Manual(s)
creation order?
how long do I pause?
what errors can I recover from?
what environment config and utilities does my script depend on?
can my script be faster?
will this script work again?



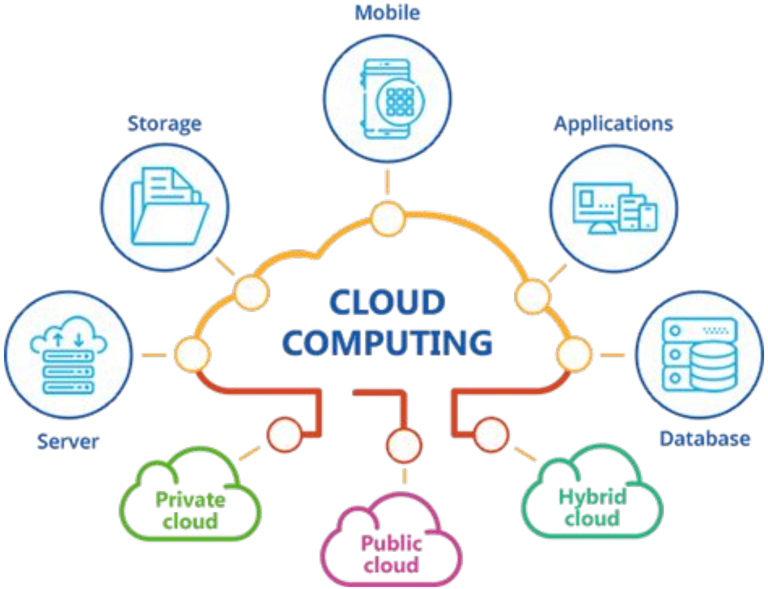
Template	CloudFormation	Stack
JSON/YAML formatted file	Framework	Configured AWS resources
Parameter definition	Stack creation	Comprehensive service support
Resource creation	Stack updates	Service event aware
Configuration actions	Error detection and rollback	Customizable

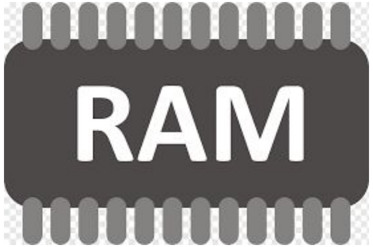


Improved Development Lifecycle



Unconstrained Resources





Not enough memory?
Consider in memory database

Paying too much for DB?
Consider serverless DBs

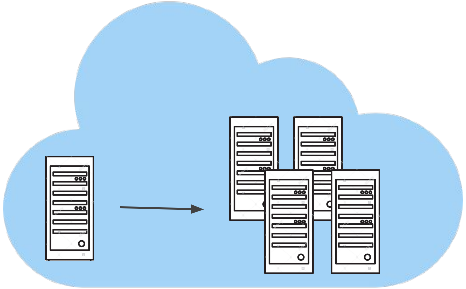
Need more processing powers?

Use distributed computing

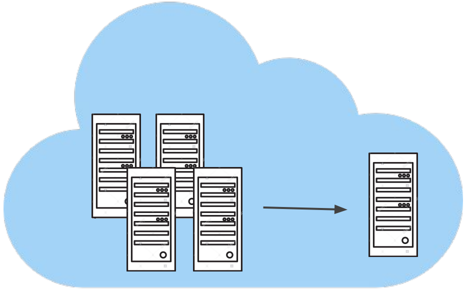
On-demand Scaling

Not enough capacity

Too much capacity

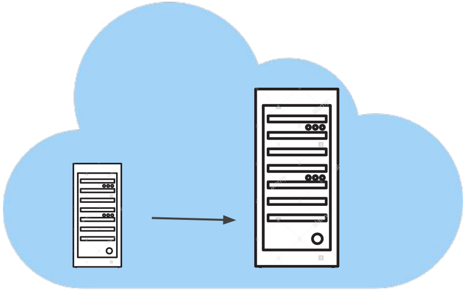


Scale Out

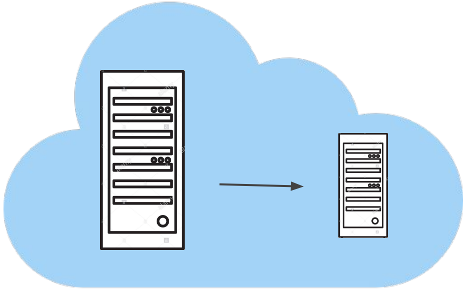


Scale In

Elasticity



Scale Up

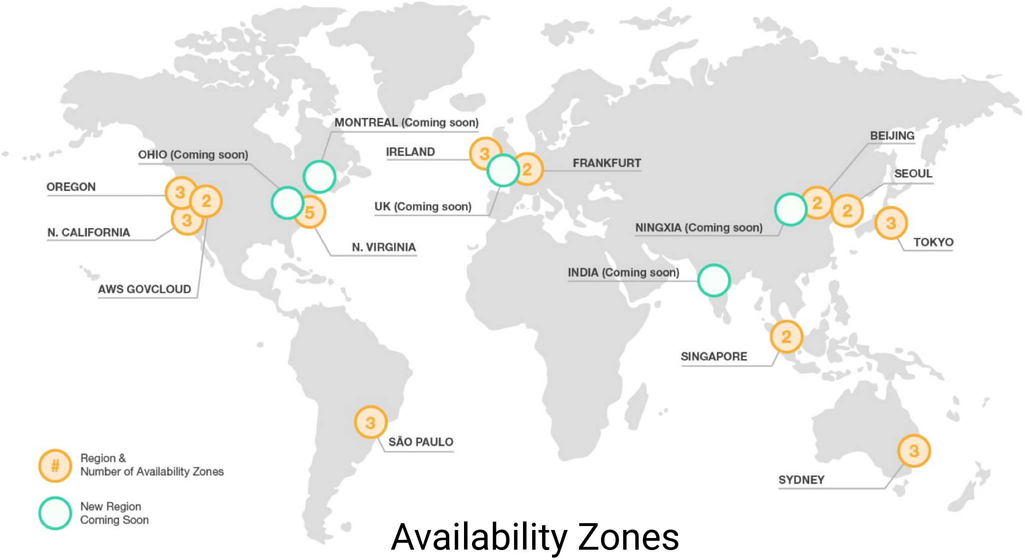


Scale Down

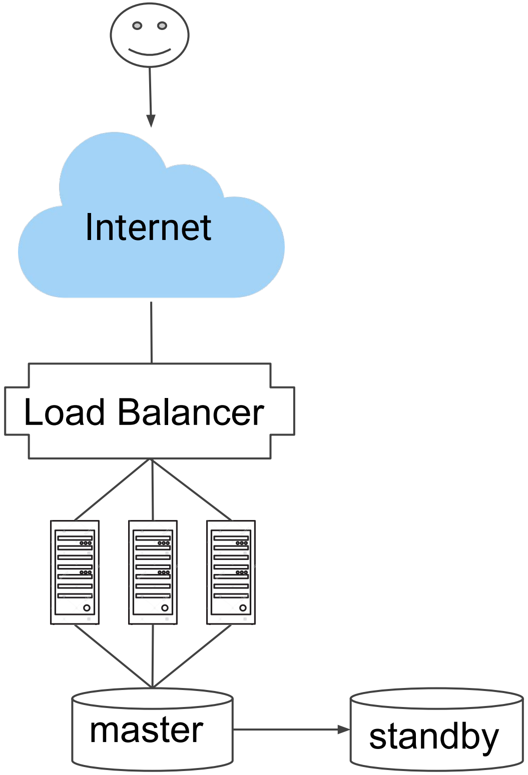
High Availability and Disaster Recovery

Everything fails, all the time
- Werner Vogels

Fail now → Failproof later

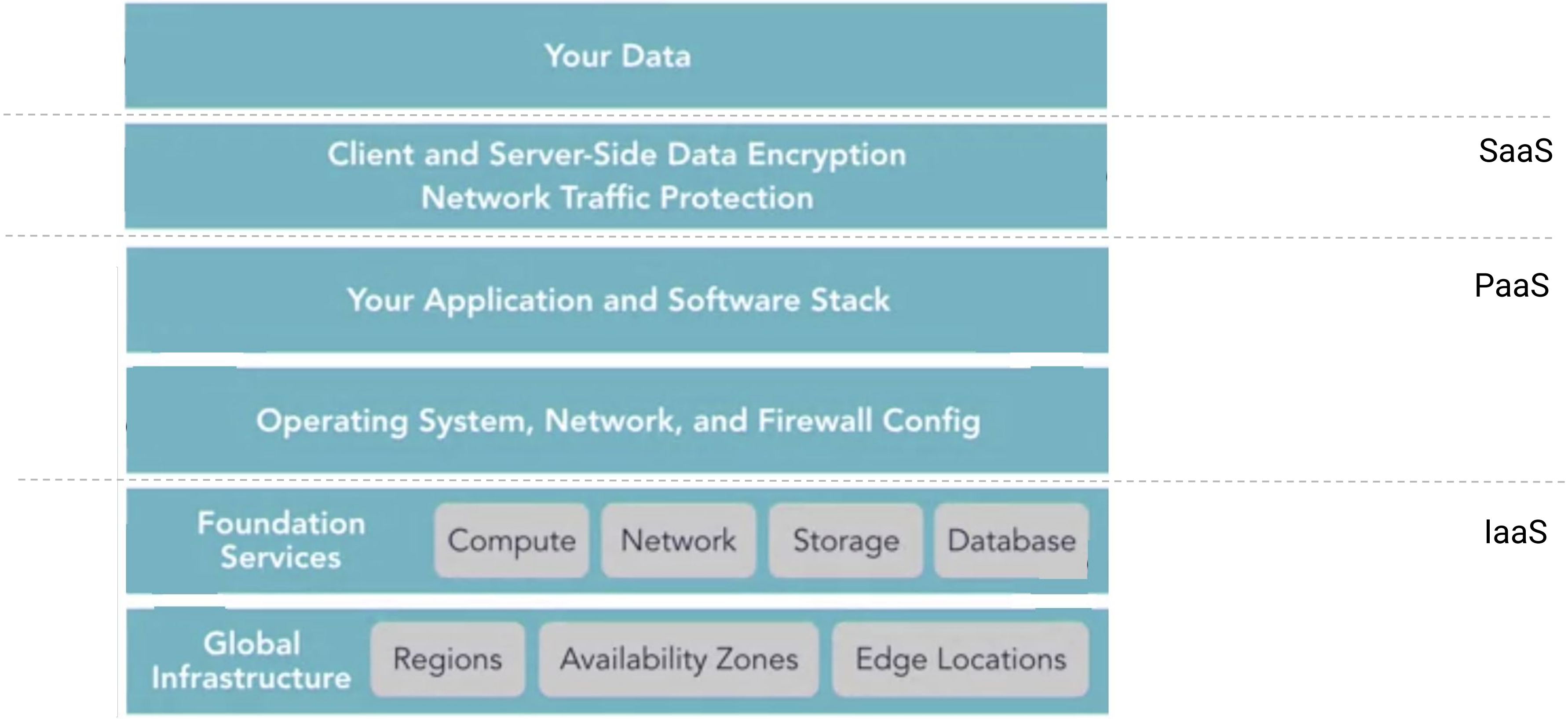


Availability Zones



Single points of failure = Bad!

Shared Security Model



Your Data

Client and Server-Side Data Encryption
Network Traffic Protection

Your Application and Software Stack

Operating System, Network, and Firewall Config

Foundation Services: Compute, Network, Storage, Database

Global Infrastructure: Regions, Availability Zones, Edge Locations

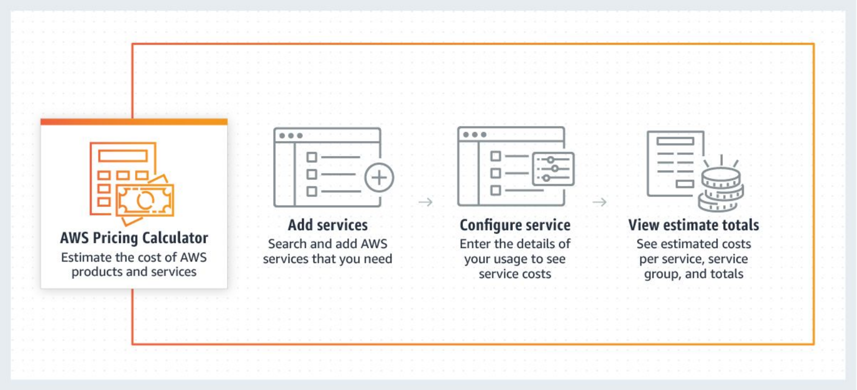
SaaS

PaaS

IaaS

Optimized Cost

Monthly Calculator



AWS Pricing Calculator
Estimate the cost of AWS products and services

Add services
Search and add AWS services that you need

→

Configure service
Enter the details of your usage to see service costs

→

View estimate totals
See estimated costs per service, service group, and totals

Pay-as-you-go

Save-when-you-reserve

Pay-less-by-using-more

Cloud Architectures Quiz

References

- Software Architectures: Patterns for Developers by Peter Morlion
- Amazon Web Services Essentials by Jeff Winesett
- Fundamentals of Software Architecture: An Engineering Approach by Marc Richards and Neal Ford
- Eclipse Architecture by Kim Moir
- Netflix Technology Blog Event Sourcing
- Event Sourcing and CQRS