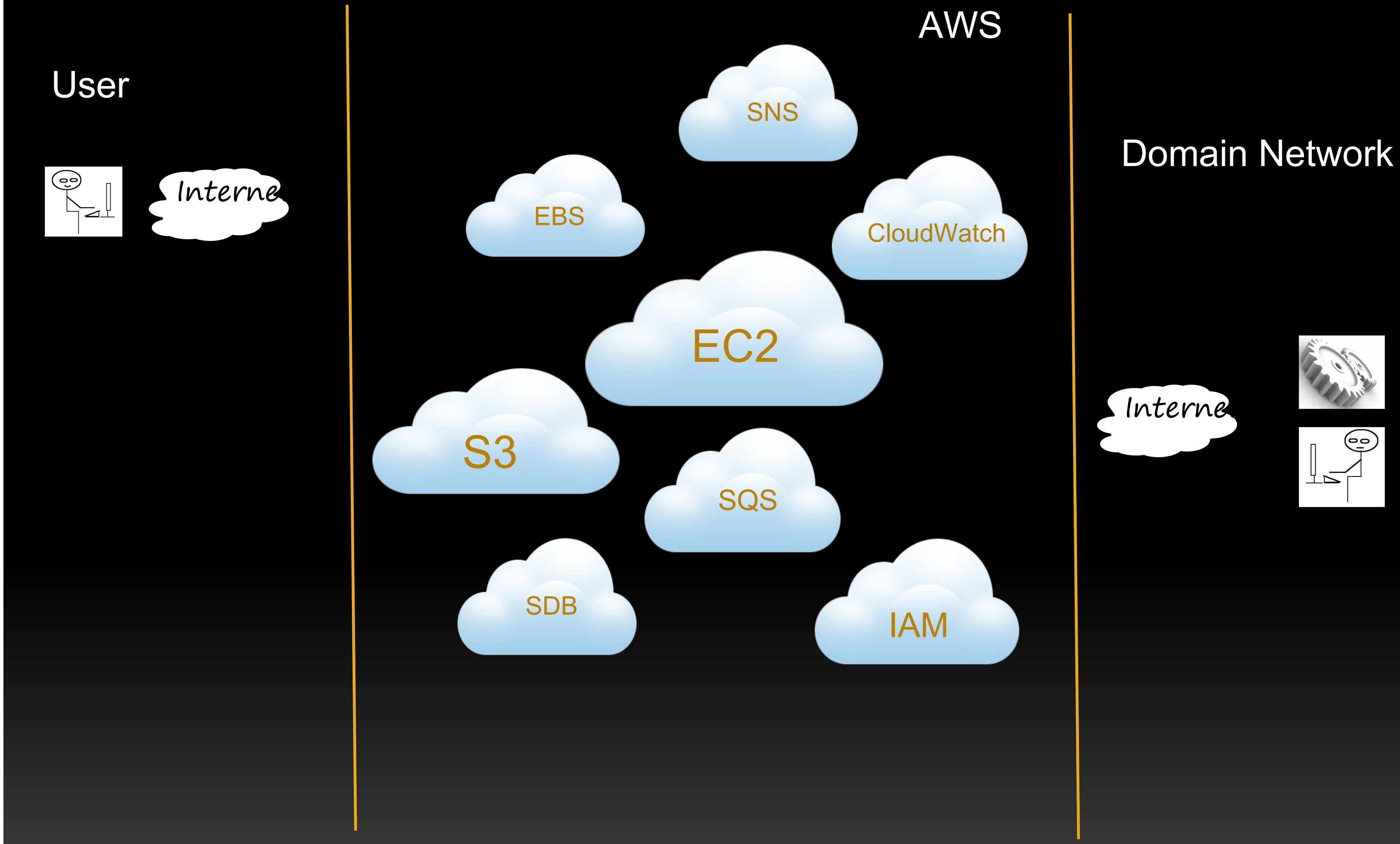
AWS Cloud Design Patterns & Practice

丁建　　2014/06/07

Our Service



User

AWS

Domain Network

SNS

EBS

CloudWatch

EC2

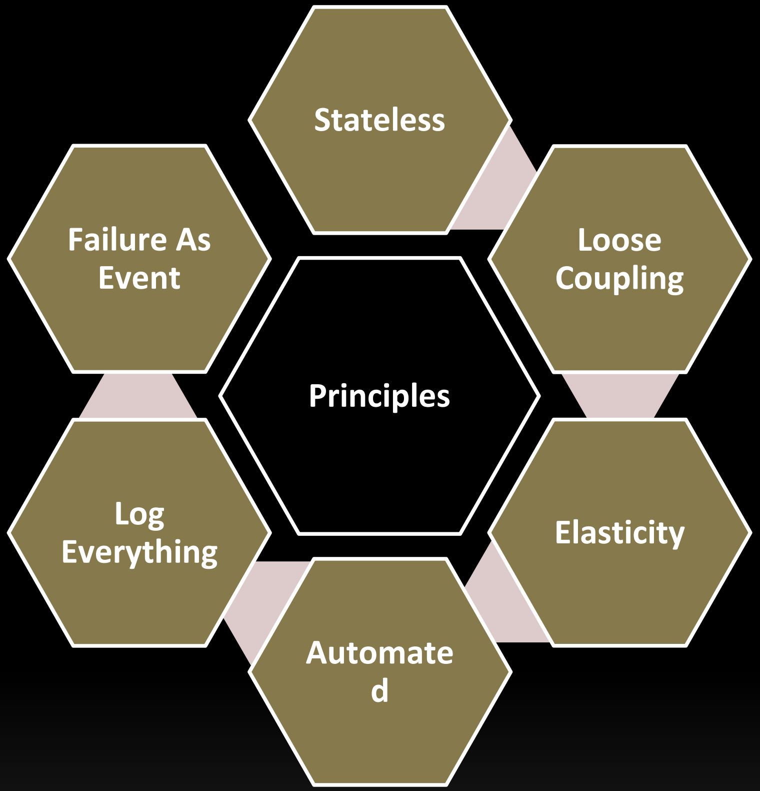
S3

SQS

SDB

IAM

Design Principles



Principles

Stateless

Loose Coupling

Elasticity

Automate
d

Log Everything

Failure As Event

- Stateless Image
- Stateless EC2 instance

- For scalable purpose
- Decouple from location, platform, time, and data format

- Each layer, service, component should be elasticity
- Bottleneck will ruin the cloud advantage
- No SPF

- Scale frequently
- Manual operation is fallible

- No disk capacity limitation
- Not easy to debug in cloud environment, use log instead
- Resource disappear after scaling
- No log level, or level is just for reference

- Design for failure
- Can recover from disaster quickly

Cloud Design Pattern Concept

Design Patterns

http://en.wikipedia.org/wiki/Design_Pattern

A design pattern in architecture and computer science is a formal way of documenting a solution to a design problem in a particular field of expertise.

Software Design Patterns

http://en.wikipedia.org/wiki/Software_design_pattern

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design.

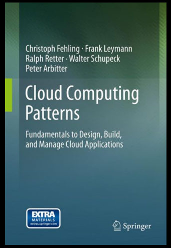
Cloud Design Patterns?

A general reusable solution to a commonly occurring problem within a given context in cloud computing system design.

Cloud Design Patterns

Patterns For Cloud Computing – Simon Guest
http://www.slideshare.net/simonguest/patterns-for-cloud-computing

Cloud Computing Design Patterns -- Thomas Erl, Amin Naserpour
http://cloudpatterns.org/


Christoph Fehling · Frank Leymann
Ralph Retter · Walter Schupeck
Peter Arbitter
Cloud Computing
Patterns
Fundamentals to Design, Build, and Manage Cloud Applications
EXTRA MATERIALS
Springer

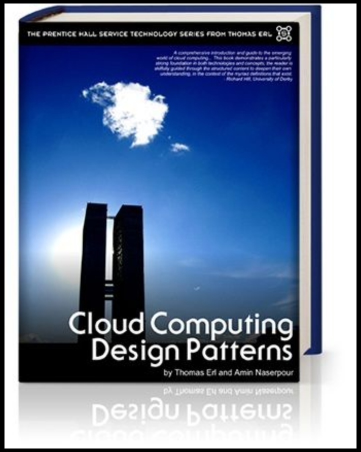
Cloud Computing Patterns -- Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, Peter Arbitter
http://www.cloudcomputingpatterns.org/

AWS Cloud Design Patterns -- Ninja of Three
http://en.clouddesignpattern.org

Patterns For Cloud Computing by Simon Guest

- Using the Cloud for **Scale**
- Using the Cloud for **Multi-Tenancy**
- Using the Cloud for **Compute**
- Using the Cloud for **Storage**
- Using the Cloud for **Communications**

Cloud Computing Design Patterns by CloudPatterns.org


THE PRENTICE HALL SERVICE TECHNOLOGY SERIES FROM THOMAS ERL
Cloud Computing
Design Patterns
by Thomas Erl and Amin Naserpour

- Prentice Hall/Pearson PTR
- Available: 2014
- Hardcover, ~ 550 pages

Infrastructure Patterns

- Physical Platform Patterns
- Virtualization Patterns
- Data and Storage Patterns
- Capacity Patterns

Scalability & Resource Pooling

- Dynamic Scaling & Elasticity Patterns
- High Availability Patterns
- Bursting & Balancing Patterns

Reliability & Recovery

- Failover & Reliability Patterns
- Disaster Protection & Recovery Patterns

Security

- Trust Boundary Patterns
- Hardening Patterns
- Privacy & Confidential Data Exchange Patterns

Monitoring

- Cloud Monitoring Patterns
- Usage Tracking & Billing Patterns
- Metric Collection & Triggers Patterns

Supplemental

- Common Compound Design Patterns
- Strategic Architecture Considerations

Example: Shared Resources

How can the capacity of physical IT resources be used to its potential?

Problem: Allocating dedicated IT resources to individual consumers can be wasteful and underutilize their collective capacity.

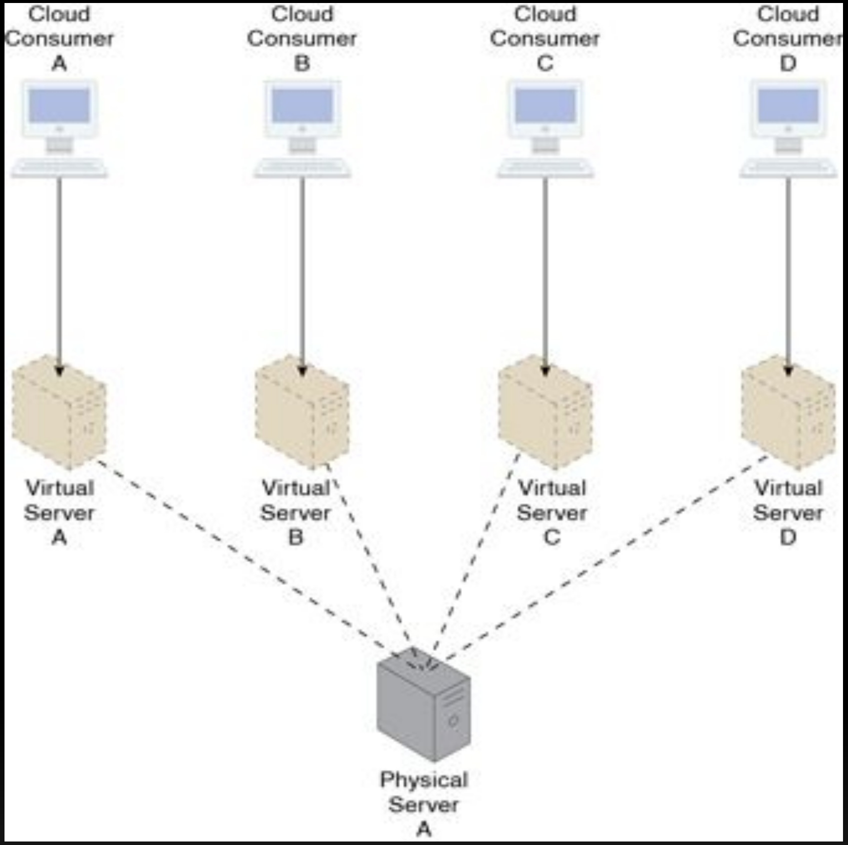
Solution : Physical IT resources are shared by partitioning them into lower capacity virtual IT resources that are provisioned to multiple cloud consumers.

Application: Virtualization technology is used to create virtual instances of physical IT resources. Each virtualized IT resource can be assigned to a cloud consumer, while the underlying physical IT resource is shared.

Mechanisms: Audit Monitor, Cloud Storage Device, Cloud Usage Monitor, Hypervisor, Logical Network Perimeter, Resource Replication, Virtual Server

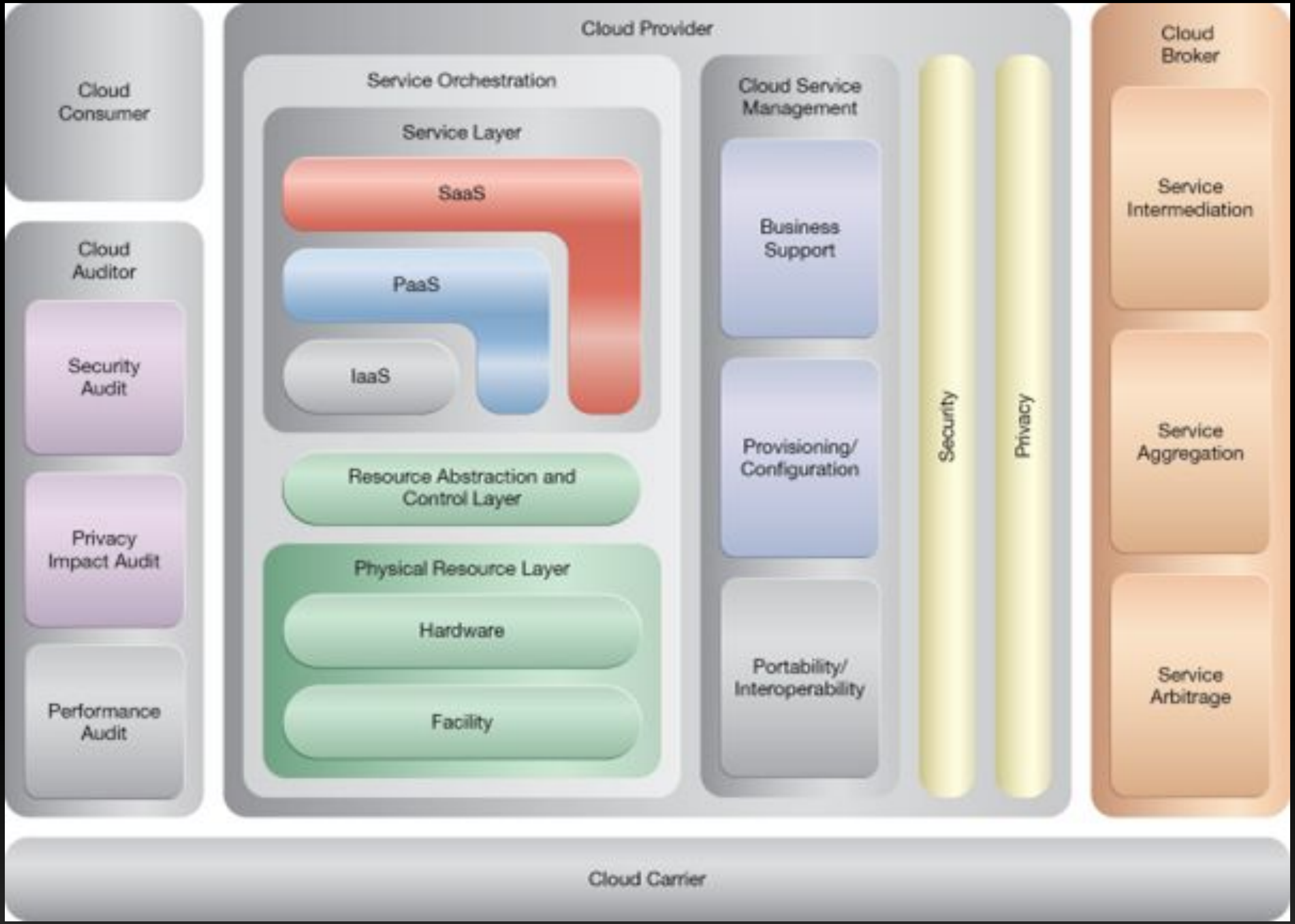
Compound Patterns: Burst In, Burst Out to Private Cloud, Burst Out to Public Cloud, Elastic Environment, Infrastructure-as-a-Service (IaaS), Multitenant Environment, Platform-as-a-Service (PaaS), Private Cloud, Public Cloud, Resilient Environment, Software-as-a-Service (SaaS)

Each cloud consumer is allocated a virtual server instance of a single underlying physical server. In this case, the physical server is likely greater than if each cloud consumer were given its own physical server. However, the cost of one high-capacity physical server is lower than four medium-capacity physical servers and its processing potential will be utilized to a greater extent.

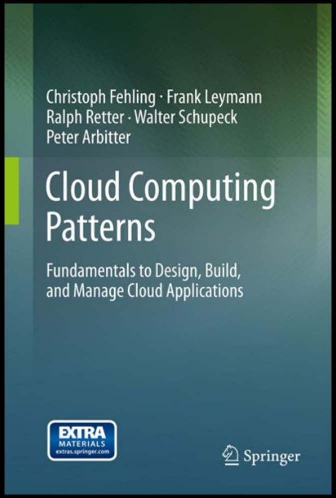


NIST Cloud Computing Reference Architecture Mapping

This pattern relates to the highlighted parts of the NIST reference architecture, as follows:



Cloud Computing Patterns by Frank, et al



Christoph Fehling · Frank Leymann
Ralph Retter · Walter Schupeck
Peter Arbitter

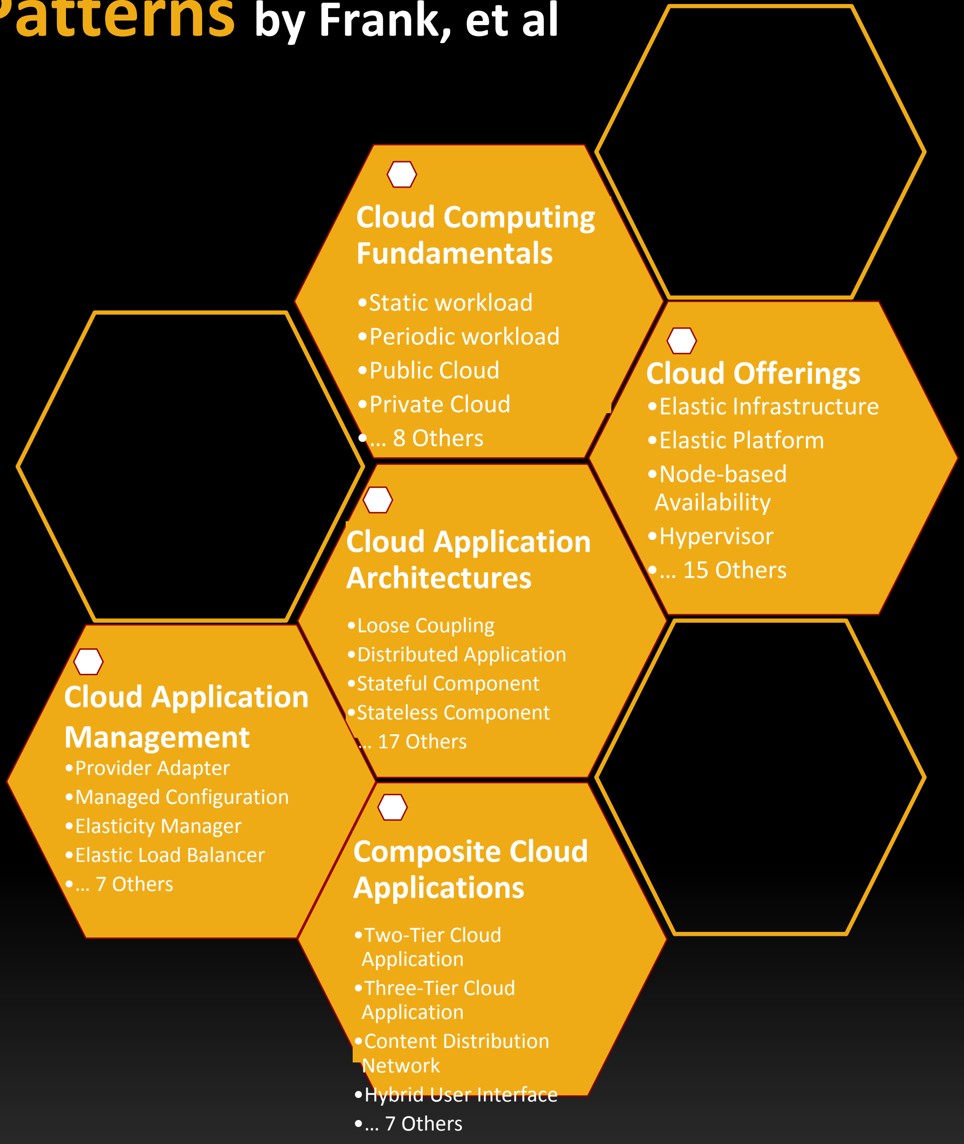
Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud Applications

EXTRA MATERIALS

Springer

- Springer Vienna
- Available: 2014
- ISBN: 978-3-7091-1567-1 (Print) 978-3-7091-1568-8 (Online)
- 367 pages



Cloud Computing Fundamentals

- Static workload
- Periodic workload
- Public Cloud
- Private Cloud
- … 8 Others

Cloud Offerings

- Elastic Infrastructure
- Elastic Platform
- Node-based Availability
- Hypervisor
- … 15 Others

Cloud Application Architectures

- Loose Coupling
- Distributed Application
- Stateful Component
- Stateless Component
- … 17 Others

Cloud Application Management

- Provider Adapter
- Managed Configuration
- Elasticity Manager
- Elastic Load Balancer
- … 7 Others

Composite Cloud Applications

- Two-Tier Cloud Application
- Three-Tier Cloud Application
- Content Distribution Network
- Hybrid User Interface
- … 7 Others

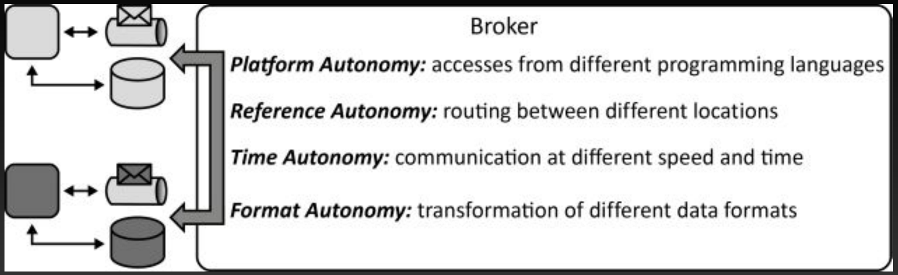
Example: Loose Coupling

A communication intermediary separates application functionality from concerns of communication partners regarding their location, implementation platform, the time of communication, and the used data format.

How can dependencies between Distributed Applications and between individual components of these applications be reduced?

Context: Information exchange between applications and their individual components as well as associated management tasks, such as scaling, failure handling, or update management can be simplified significantly if application components can be treated individually and the dependencies among them are kept to a minimum.

Solution: Communicating components and multiple integrated applications are decoupled from each other by interacting through a broker. This broker encapsulates the assumptions that communication partners would otherwise have to make about one other and, thus, ensures separation of concerns.



Broker

Platform Autonomy: accesses from different programming languages

Reference Autonomy: routing between different locations

Time Autonomy: communication at different speed and time

Format Autonomy: transformation of different data formats

AWS Cloud Design Patterns by NoT

Ninja of Three

- @KenTamagawa
- @c9katayama
- @suz_lab

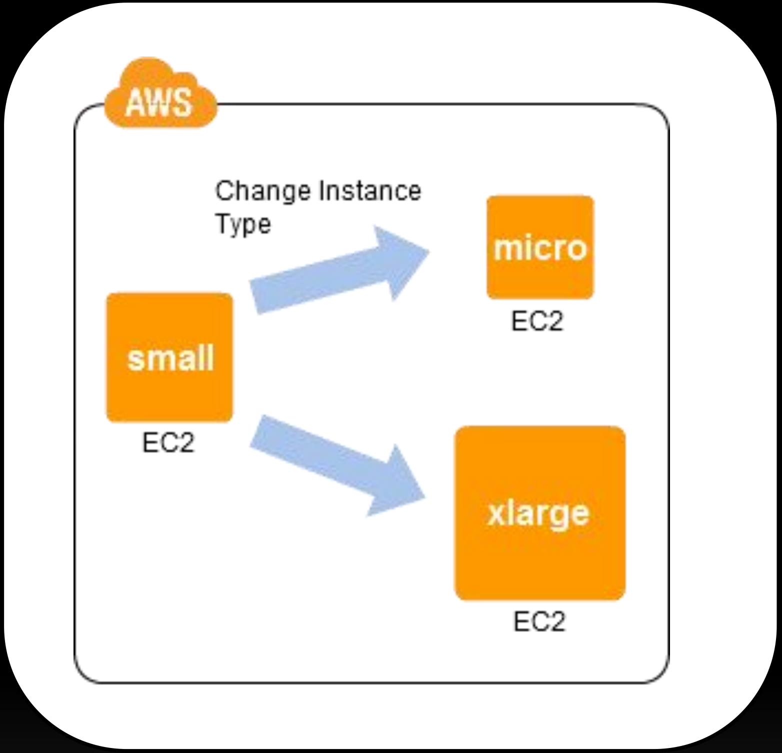
9 Categories, 48 Patterns

- Basic Patterns
- Patterns for Processing Static Content
- Patterns for Batch Processing
- Patterns for High Availability
- Patterns for Processing Dynamic Content
- Patterns for Uploading Data
- Patterns for Relational Database
- Patterns for Operation and Maintenance
- Patterns for Network

Basic Patterns

- Scale Up Pattern --- Dynamic Server Spec Up/Down
- Scale Out Pattern --- Dynamically Increasing the Number of Servers
- Scheduled Scale Out Pattern --- Increasing or Decreasing the Number of Servers Following a Schedule
- On-demand Disk Pattern --- Dynamically Increasing/Decreasing Disk Capacity
- Snapshot Pattern --- Data Backups
- Stamp Pattern --- Server Replication

Scale Up Pattern Dynamic Server Spec Up/Down



Benefits:

- ✓ Eliminates the need for precise estimation of server specifications at the time of system design/development.
- ✓ Reduces opportunity costs due to system stoppages and the inability to provide services to customers caused by inadequate resources
- ✓ Enables reduction in waste, in terms of expenses

Cautions:

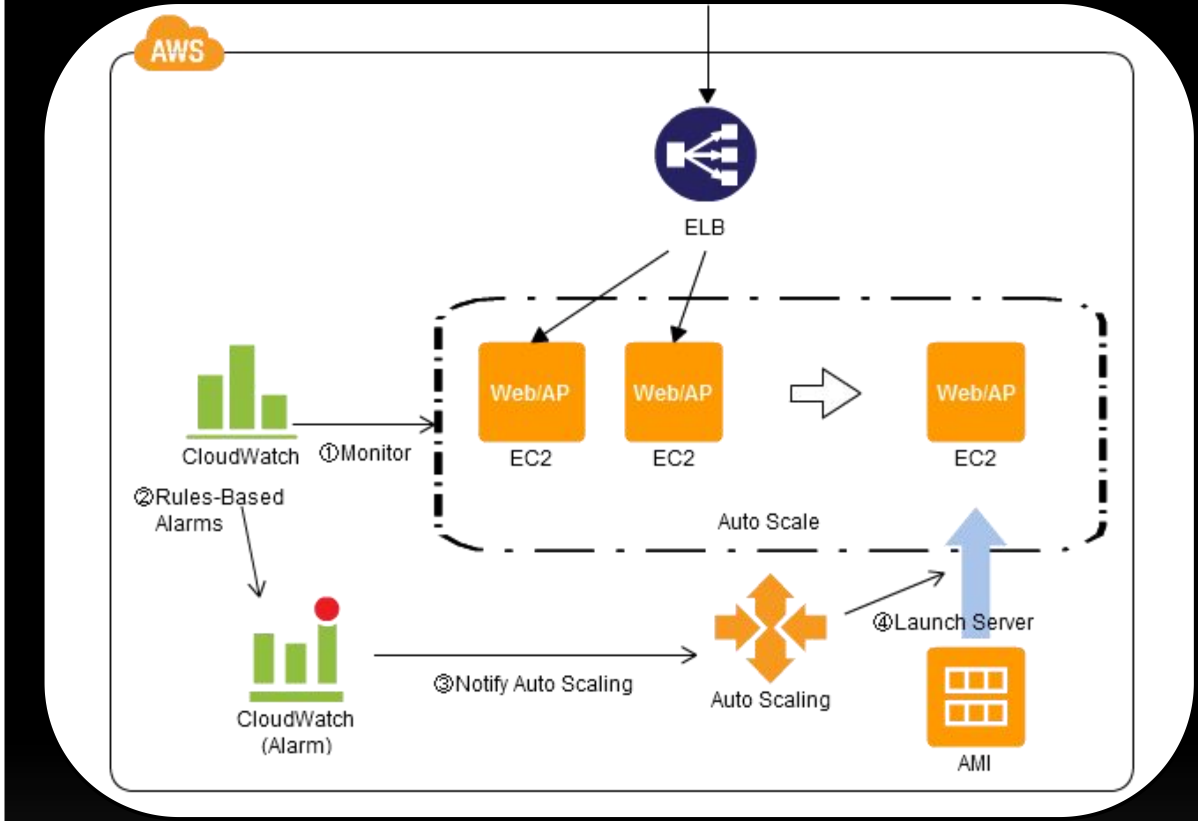
- ✓ Need time to adjust (30 seconds ~ several minutes)
- ✓ The server specification is not limitless

Other:

- ✓ Can adjust the server periodically
- ✓ Suitable for the application which is hard to be scale out

Scale Out Pattern

Dynamically Increasing the Number of Servers



Benefits:

- ✓ Provide service continuity
- ✓ Reduce cost
- ✓ Reduce the workload (automate the scale)
- ✓ The limit is smaller than Scale Up Pattern

Cautions:

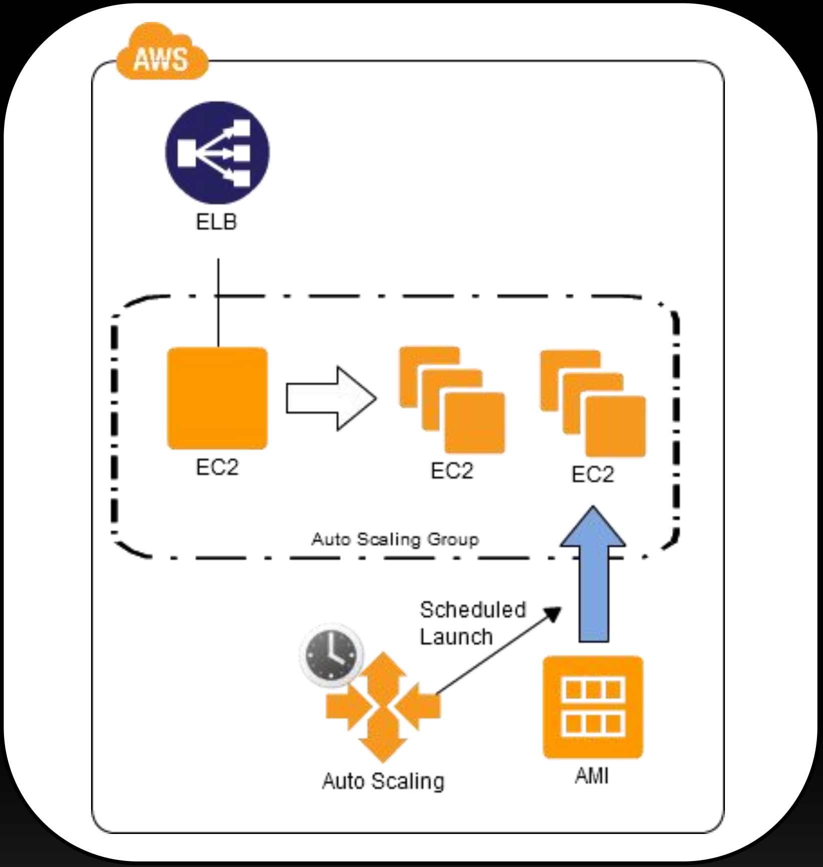
- ✓ Cannot handle severe or rapid variations in traffic
- ✓ States (HTTP session, SSL processes, and the like) should be controlled by dedicated server

Other:

- ✓ See State Sharing Pattern for session administration

Scheduled Scale Out Pattern

Increasing or Decreasing the Number of Servers Following a Schedule



Benefits:

- ✓ Use prediction to resolve the issue of severe or rapid variations in traffic
- ✓ Reduces the cost because the instances number is reduced when there is little traffic
- ✓ The limit is smaller than Scale Up Pattern

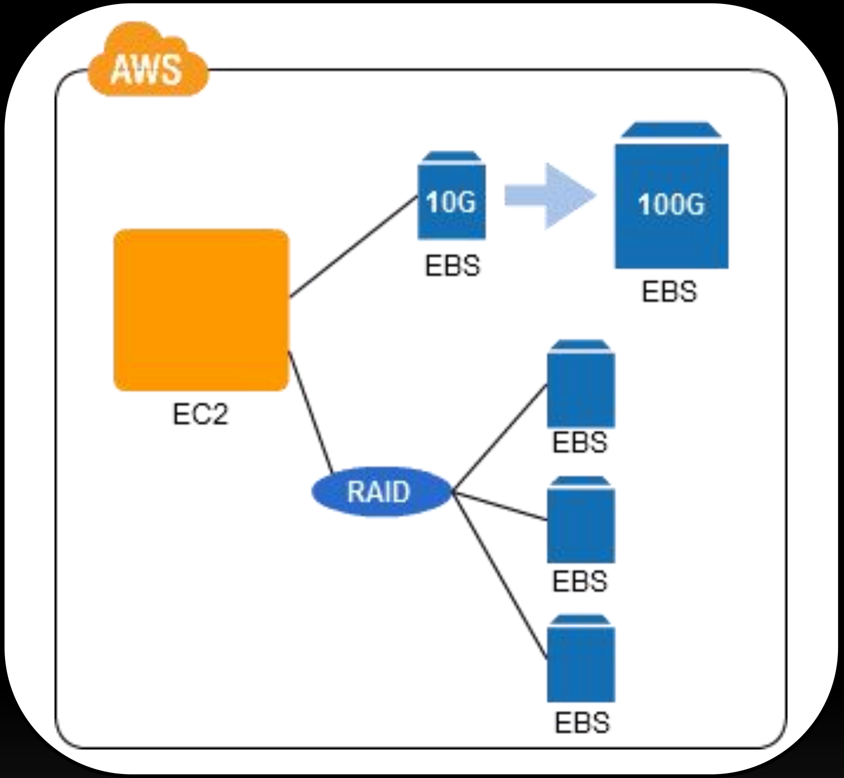
Cautions:

- ✓ Not work for un-predicted load increase/derease

Other:

On-demand Disk Pattern

Dynamically Increasing/Decreasing Disk Capacity



Benefits:

- ✓ No need to buy the storage which exceeds your requirement
- ✓ Striping can improve the disk I/O

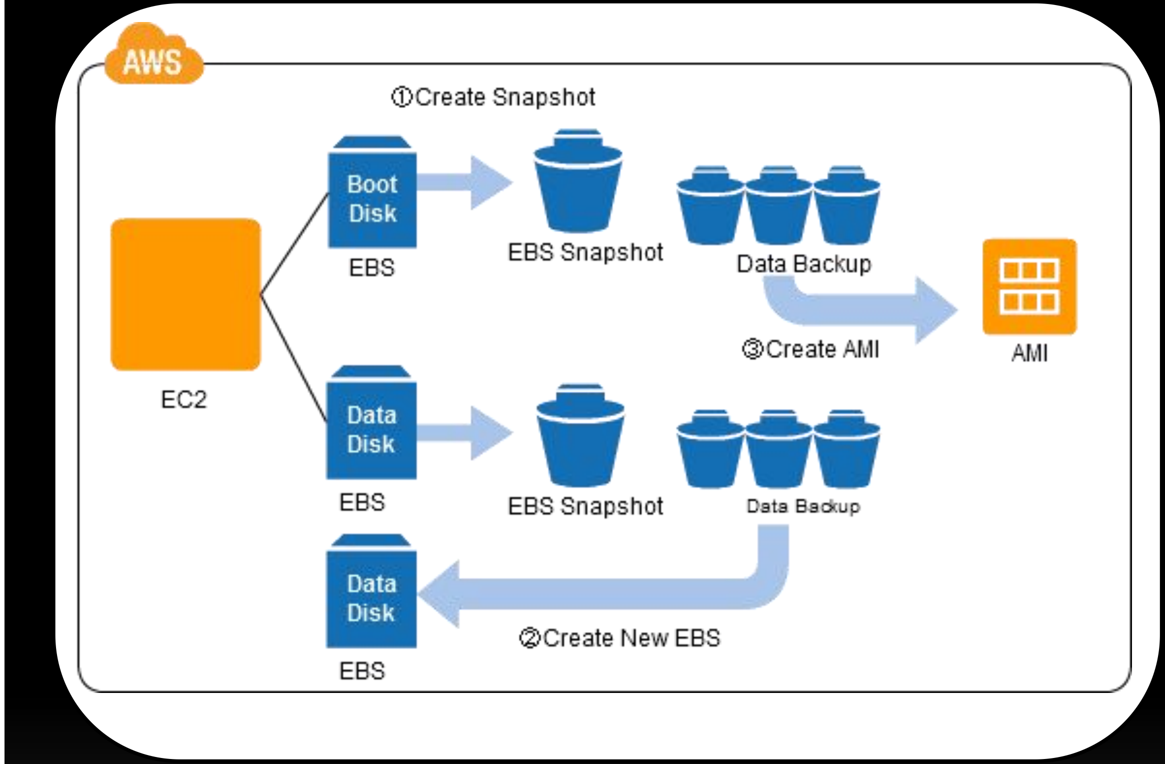
Cautions:

- ✓ Is charged by capacity instead of usage
- ✓ 1TB capacity limit for single EBS, larger one needs striping
- ✓ The amount of EBS is limited (10?)

Other:

Snapshot Pattern

Data Backups



Benefits:

- ✓ Automate the backup process
- ✓ High durability
- ✓ Backup both data and environment
- ✓ Easy to recover

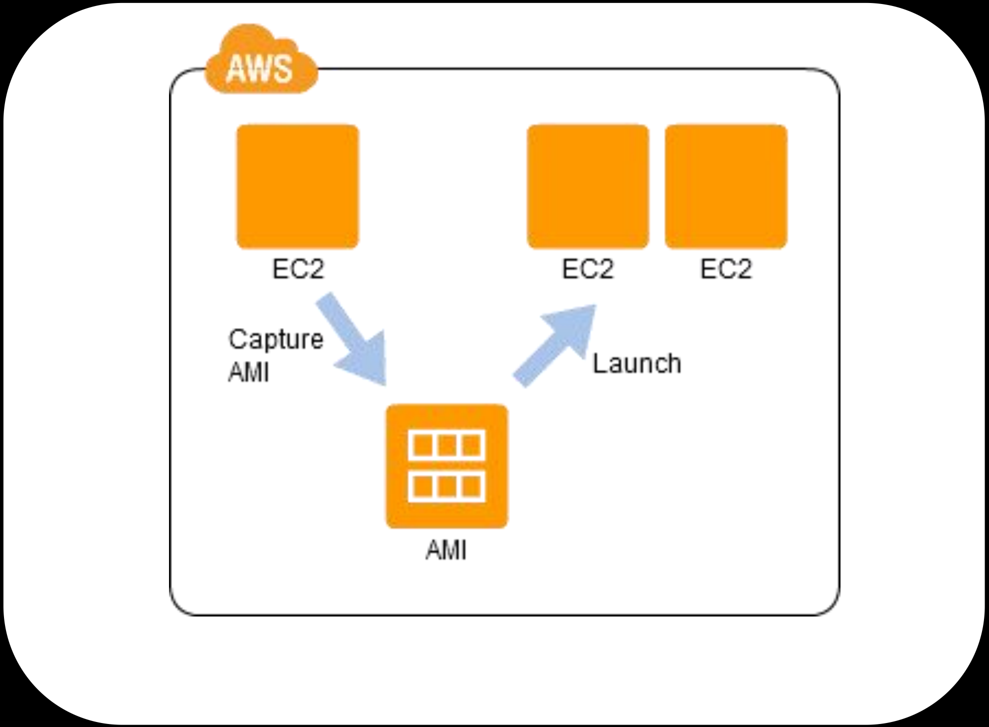
Cautions:

- ✓ Maintain the logical consistency by yourself (flush data, finish the transaction)

Other:

- ✓ Spit EBS to be root and data partitions can improve the backup speed

Stamp Pattern Server Replication



Benefits:

✓ Automate the deployment

✓ Easy to share

Cautions:

✓ Need to recreate AMI if the environment needs to be changed (but we can use other way to mitigate)

Other:

✓ See bootstrap pattern to provide more flexibility

Practice 1 - Use our own scaling server

- Compound scaling modes: Scale Out, Scale Up and Scheduled Scale
- Support multiple ami types
- Support safe scaling down for long session
- Support hourly scaling
- Support reserved instance

Practice 2 - Make EC2 instance stateless

- S3/SDB for configuration
- SDB/DynamoDB for NoSQL records & Key-Value data
- EBS for dependency & procedure data
- SQS for task dispatch
- Elasticache for high speed cache
- RDS for traditional relation data

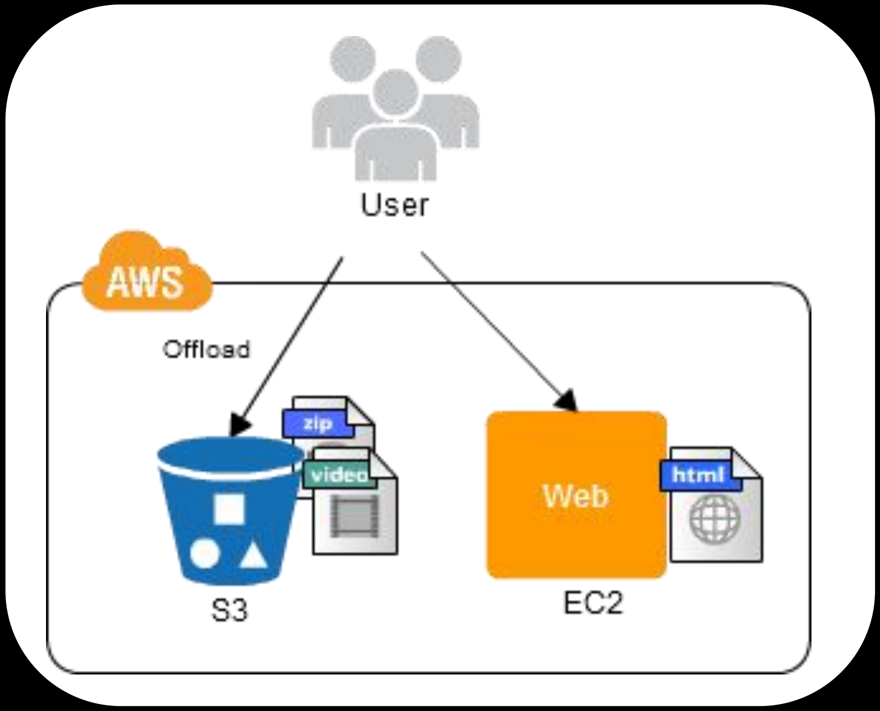
Practice 3 – Create our own load balancer

- Complex proxy logic supporting
- Bind the SSL certificate of our company
- Multiple instance types support

Patterns for Processing Static Content

- Web Storage Pattern --- Use of High-Availability Internet Storage
- Direct Hosting Pattern --- Direct Hosting Using Internet Storage
- Private Distribution Pattern --- Data Delivery to Specified Users
- Cache Distribution Pattern --- Locating Data in a Location That Is Physically Near to the User
- Rename Distribution Pattern --- Delivery Without Update Delay

Web Storage Pattern Use of High-Availability Internet Storage

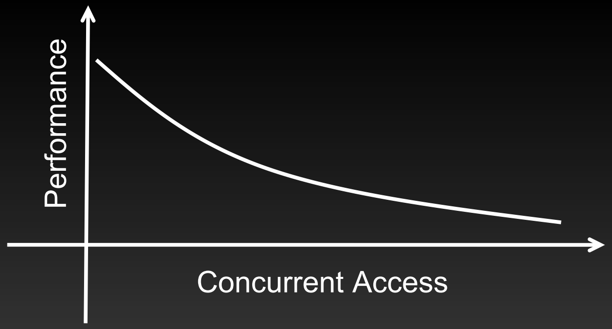


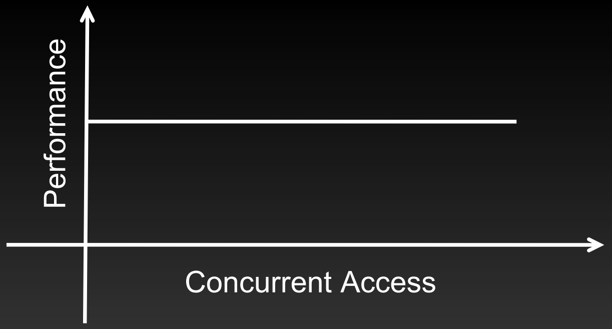
Benefits:

- ✓ High load support and limitless capacity
- ✓ High durability
- ✓ Since URL is issued for each content object, it can be used for a broad range of purposes (file sharing, state sharing, configuration sharing…)

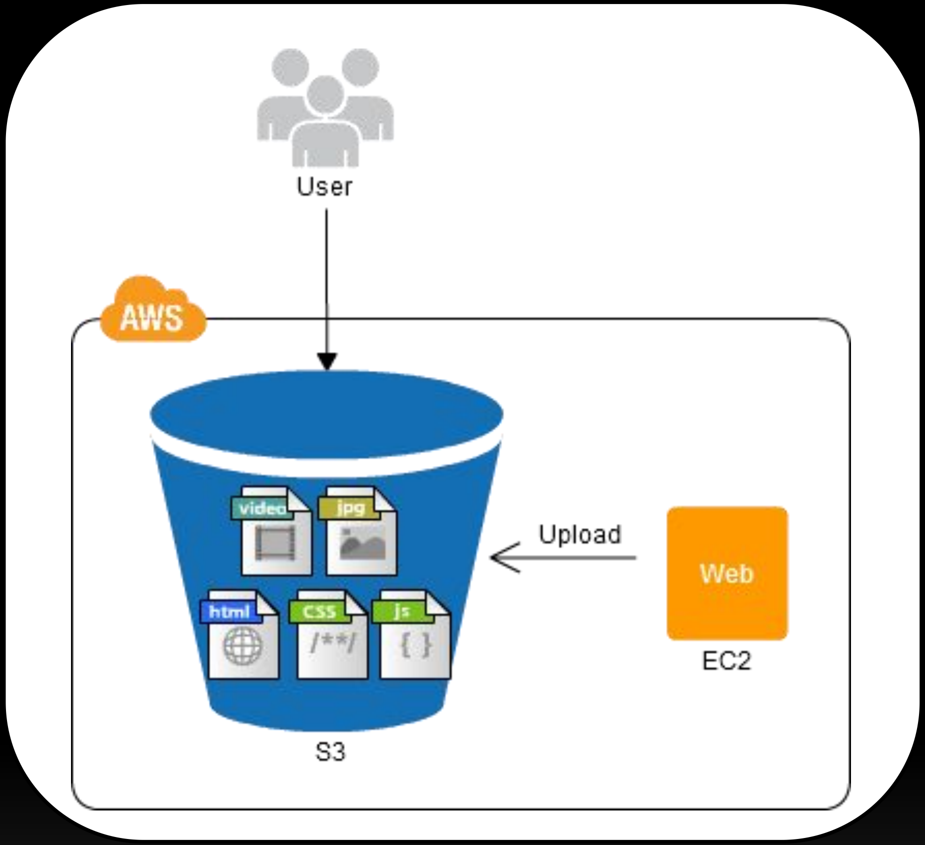
Cautions:

Other:





Direct Hosting Pattern
Direct Hosting Using Internet Storage



Benefits:

- ✓ Increase the availability and durability of the web system

Cautions:

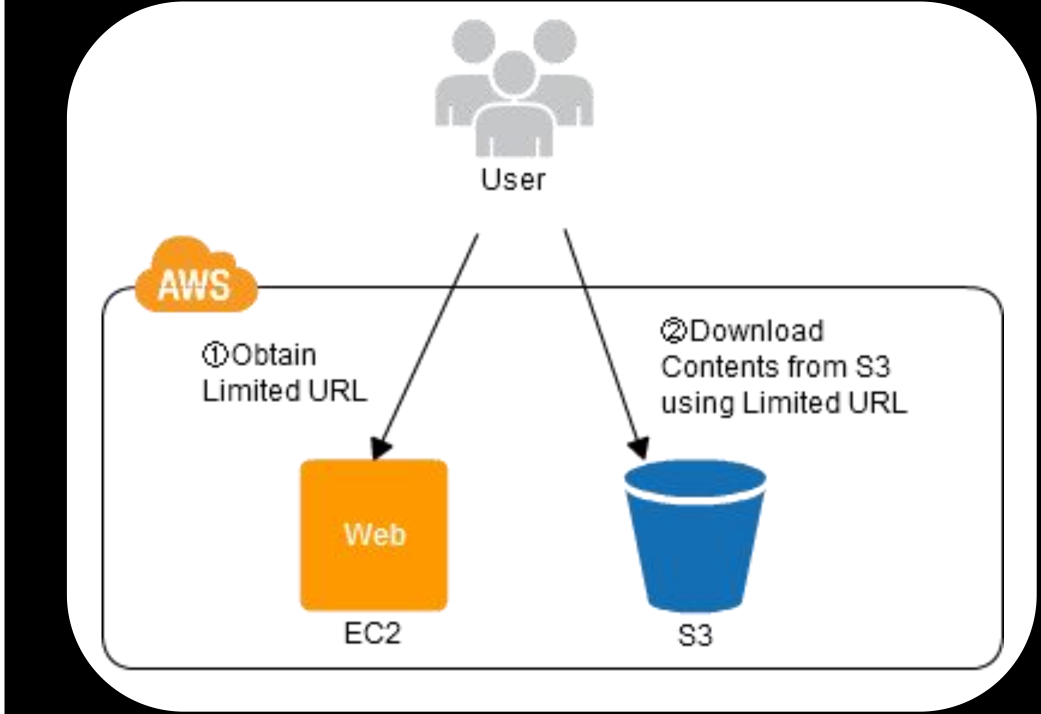
- ✓ Hard to do user authentication
- ✓ Can’t run server side program in S3
- ✓ JavaScript issue

Other:

- ✓ Can still be used even in a dynamic site (Such as blog)
- ✓ Can provide limited user access control (signed URLs, bucket policy)
- ✓ 900 billion objects, 700000 requests per second (March 2012)

Private Distribution Pattern

Data Delivery to Specified Users



Benefits:

- ✓ Enables delivery of private content through time-limited use by specified users only

Cautions:

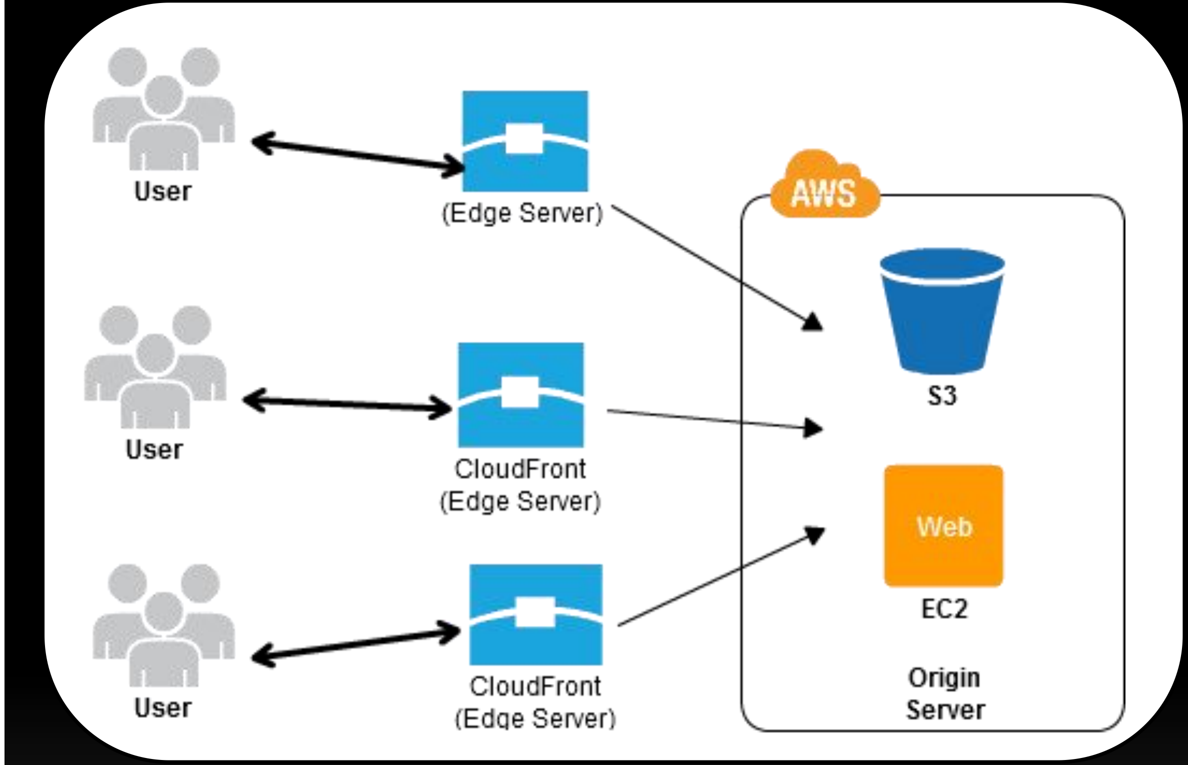
- ✓ Must provide a validation system and a server for issuing time-limited URLs
- ✓ Even if the user validation has not expired, the term of effectiveness of the URL will expire, preventing downloading

Other:

- ✓ Can use this pattern in combination with an application validation system

Cache Distribution Pattern

Locating Data in a Location That Is Physically Near to the User



Benefits:

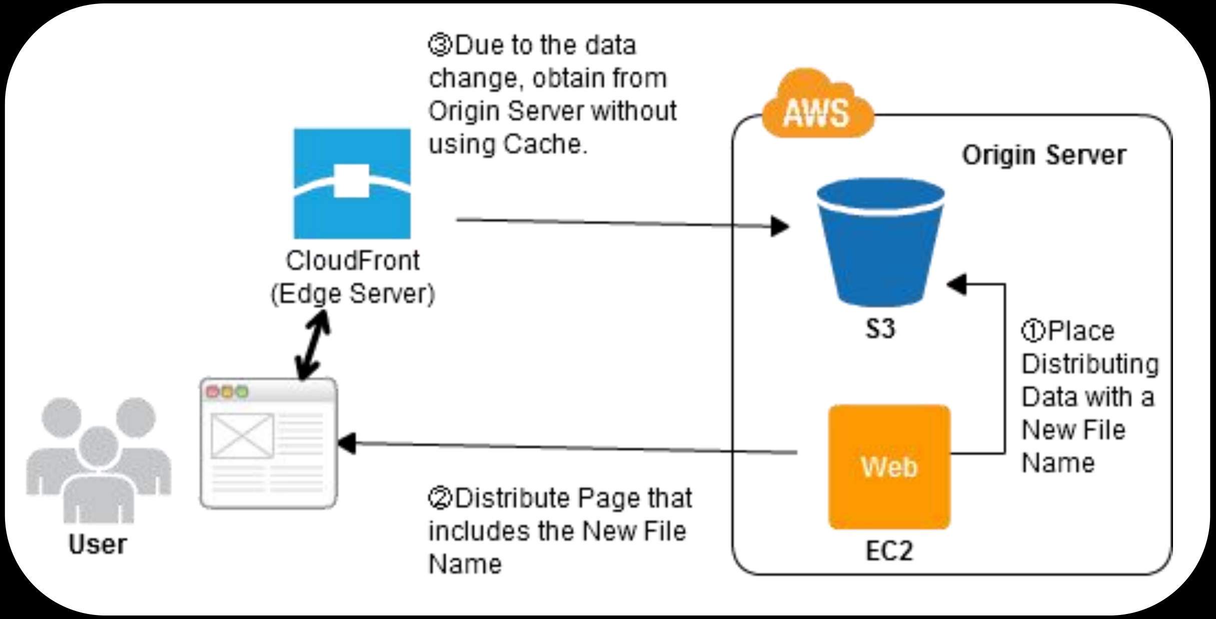
- ✓ Better experience in geographically distant places
- ✓ Can distribute the file download processes
- ✓ Can use an existing server as origin server
- ✓ Can use S3 as an origin server directly

Cautions:

- ✓ Be careful about the timeout issue of cache

Other:

Rename Distribution Pattern Delivery Without Update Delay


③Due to the data change, obtain from Origin Server without using Cache.
AWS
Origin Server
CloudFront (Edge Server)
S3
①Place Distributing Data with a New File Name
Web
User
②Distribute Page that includes the New File Name
EC2

Benefits:

- ✓ Can deliver new content without waiting for the cache timeout

Cautions:

- ✓ This would be ineffective if the cache timeout for the base content itself were too long
- ✓ Old file will remain on the edge server until the cache timeout

Other:

Practice 1 - Direct hosting

- Publish installer & patch

  https://s3.amazonaws.com/installer-bucket-name/installer/setup.exe

- Client side configuration

  https://s3.amazonaws.com/configuration-bucket-name/configure.xml

- Publish information

  https://s3.amazonaws.com/message-bucket-name/message.txt

- Huge data accessing

  https://s3.amazonaws.com/huge-data-bucket-name/somehugedata.data

Practice 2 - Use S3 accessing log to collect metrics

- Distributed
- Stable
- Simple
- A little bit latency

```
6eb14f6a410a8f6a8b0405842cf4b7e77a8db7d160793d2482245bebafc9b070 bucket-for-tracking
[23/Oct/2013:08:11:04 +0000] 184.22.72.192 - 15B237D8C860633D REST.GET.OBJECT t.gif "GET /bucket-
for-tracking/t.gif?p1=xx&p2=xx&p3=xx&p4=ec2-184-124-219-1.us-west-
1.compute.amazonaws.com&un=xx&id=A2URKEKJCE2T HTTP/1.1" 200 - 807 807 162 161 "-" "-" -
```

Practice 3 - Use S3 to collect logs

- Distributed
- Simple
- Stable
- High performance

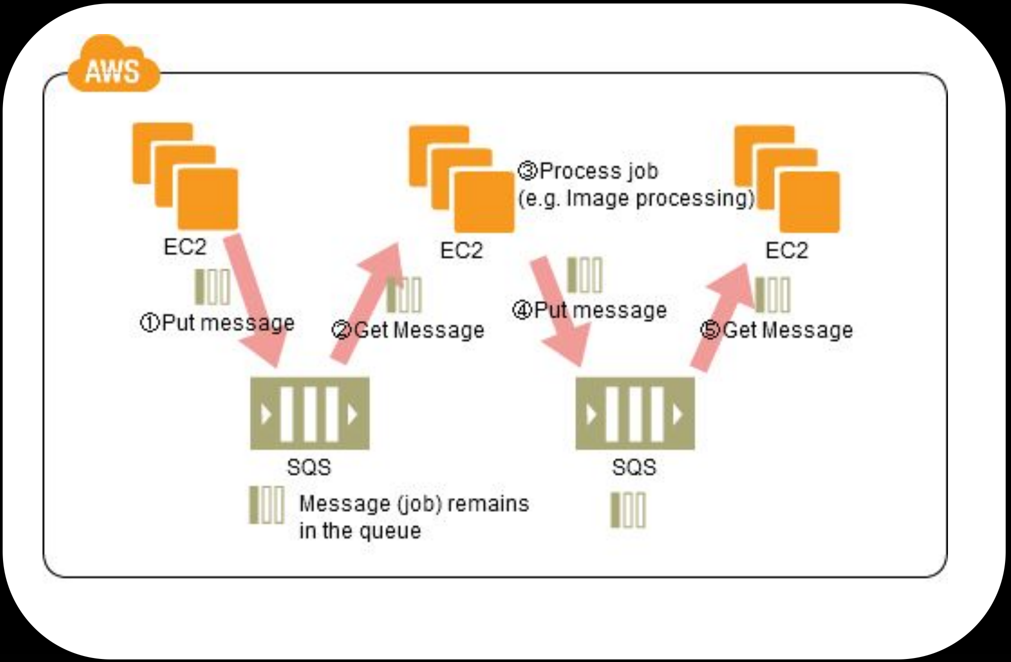
Practice 4 - Use S3 to store artifacts/scripts/binaries for auto-deployment

- High performance
- Stable
- Cross domain accessing

Patterns for Batch Processing

- Queuing Chain Pattern --- Loose-Coupling of Systems
- Priority Queue pattern --- Changing Priorities
- Job Observer Pattern --- Job Monitoring and Adding/Deleting Servers
- Scheduled Auto scaling Pattern --- Turning Batch Servers On and Off Automatically

Queuing Chain Pattern Loose-Coupling of Systems



Benefits:

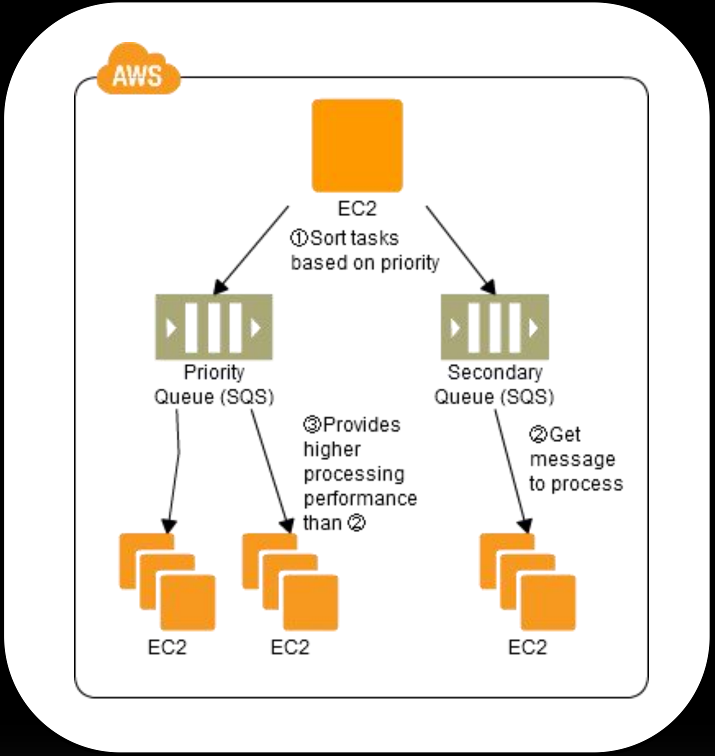
- ✓ Asynchronous processing
- ✓ Loose coupling the system
- ✓ Easy to scale in job processing
- ✓ High reliability (not influenced by the failure of EC2 instance and easy to recover)

Cautions:

- ✓ The sequence of messages is not completely guaranteed

Other:

Priority Queue pattern Changing Priorities



Benefits:

- ✓ Asynchronous processing
- ✓ Loose coupling the system
- ✓ Easy to scale in job processing
- ✓ High reliability
- ✓ Provide the different performance and service for different custom

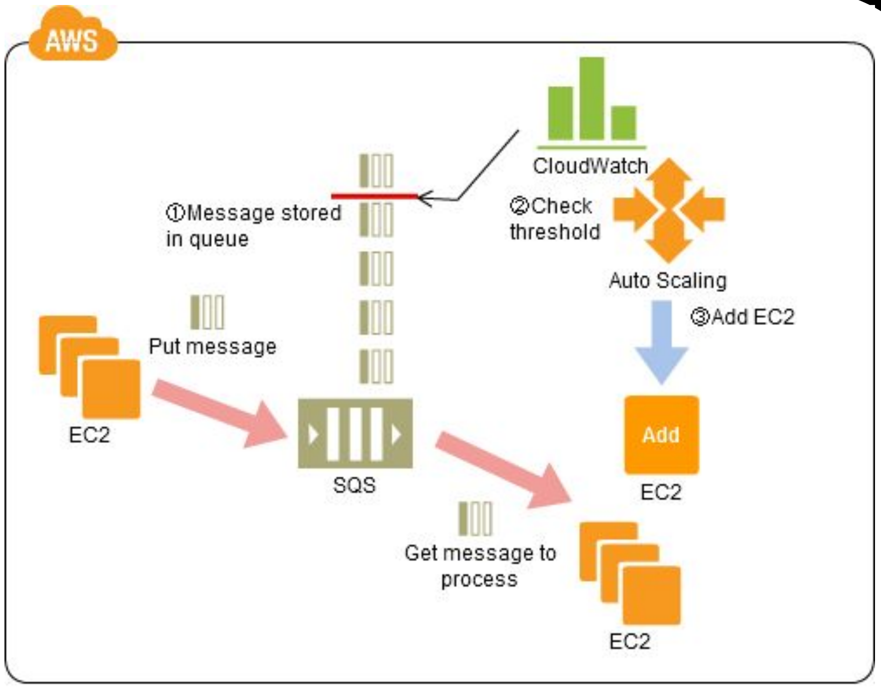
Cautions:

- ✓ You need to allocate the job processing ability by yourself

Other:

Job Observer Pattern

Job Monitoring and Adding/Deleting Servers



Benefits:

- ✓ Improving cost effectiveness
- ✓ Reduce the overall time for executing jobs through parallel processing
- ✓ Robust to failure

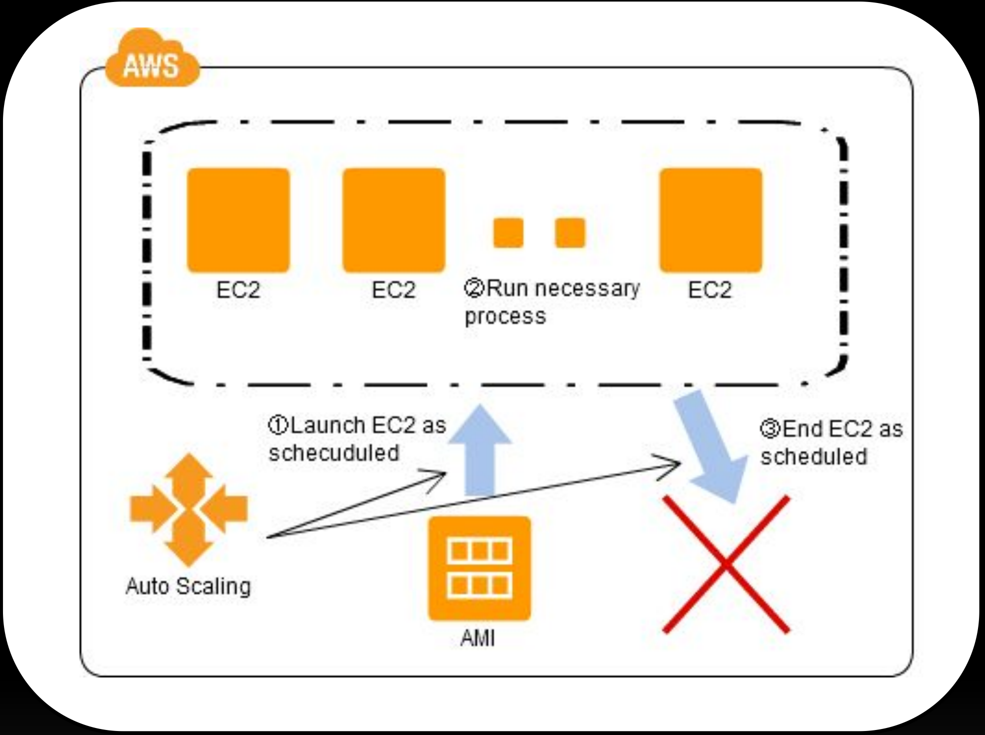
Cautions:

- ✓ EC2 instance is charged hourly

Other:

Scheduled Auto-scaling Pattern

Turning Batch Servers On and Off Automatically



Benefits:

- ✓ Reducing costs (no need to use dedicated server to scale)

Cautions:

- ✓ Be careful of when to shutdown the instance (leave instance itself to shutdown)

Other:

- ✓ EC2 instance is charged hourly

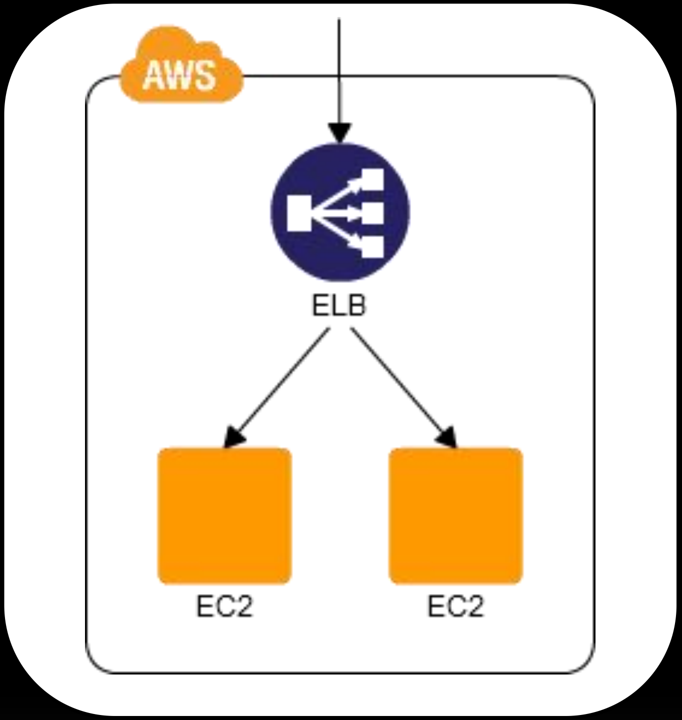
Practice 1

- Message transfer among servers and environments
  - Cross domain accessing in CI/CD
  - Decouple the components

Patterns for High Availability

- Multi-Server Pattern --- Server Redundancy
- Multi-Datacenter Pattern --- Redundancy on the Data Center Level
- Floating IP Pattern --- Floating IP Address
- Deep Health Check Pattern --- System Health Check

Multi-Server Pattern Server Redundancy



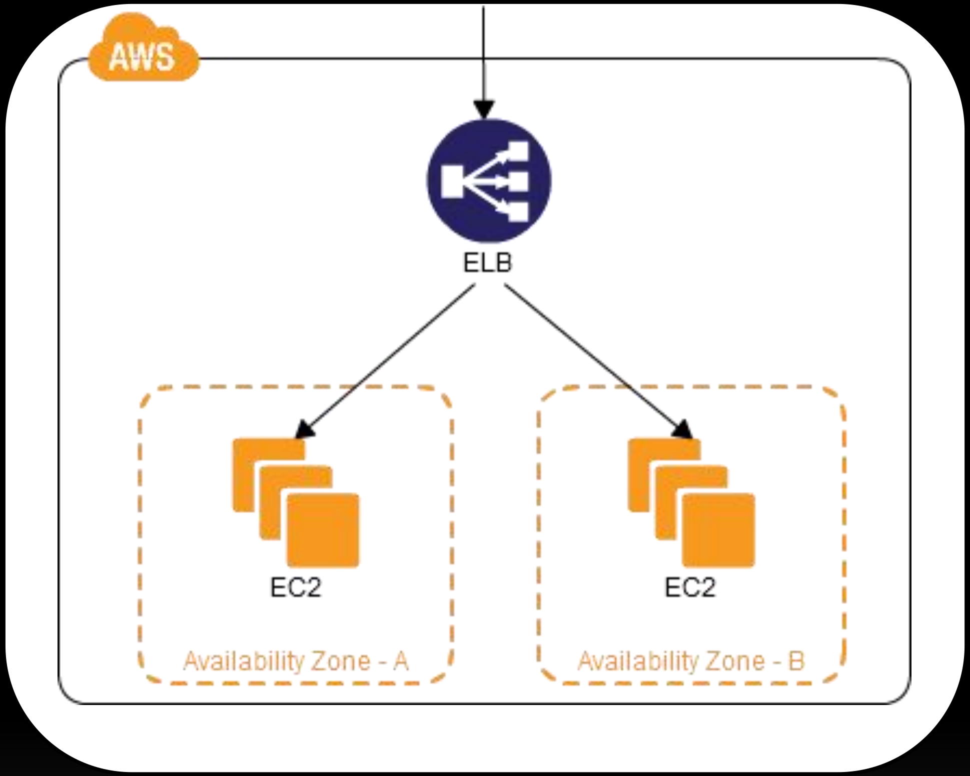
Benefits:
- ✓ More robust

Cautions:
- ✓ More cost
- ✓ The issue of sharing state
- ✓ Need to caution in using it as DB server

Other:

Multi-Datacenter Pattern

Redundancy on the Data Center Level



Benefits:

✓ More robust in data center level

Cautions:

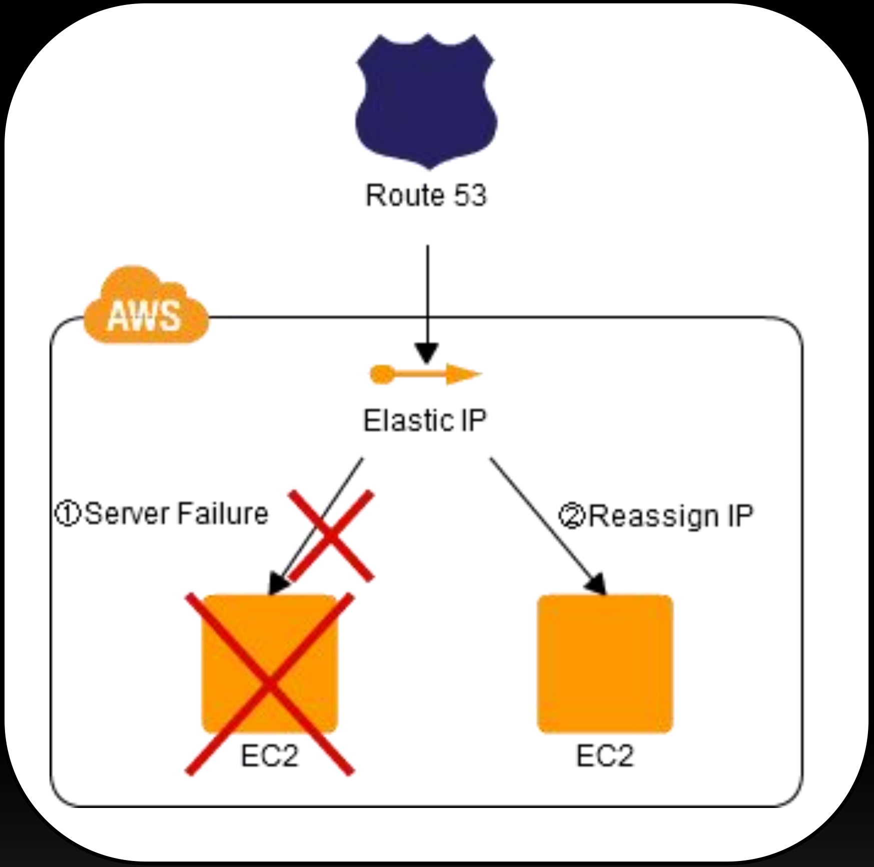
✓ DB synchronization and performance

Other:

✓ Need to pay for the communication between AZs (US$0.01 per GB)

Floating IP Pattern

Floating IP Address



Benefits:

- ✓ Avoid the influence of TTL

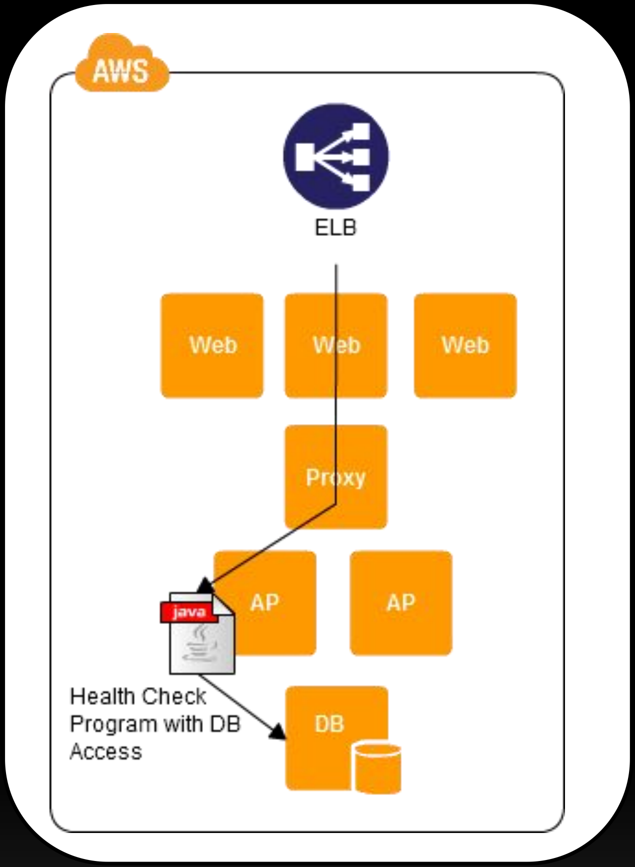
Cautions:

- ✓ Need several seconds to take effect
- ✓ SSH may be influenced

Other:

- ✓ Can also use this in parallel with Server Swapping Pattern to not only reassign EIP, but to swap EBS as well
- ✓ Can also use ELB

Deep Health Check Pattern System Health Check



Benefits:

- ✓ Check all the servers
- ✓ Easy to customize the error process

Cautions:

- ✓ Health check also contribute the traffic
- ✓ SPOF may take all the servers down

Other:

- ✓ DB Replication Pattern should be used in parallel so the DB server does not become a SPOF

Practice 1 - Multiple data centers deployment

- Failure tolerance
- Mitigate latency

Practice 2 - Leverage CloudWatch/SNS/S3 to monitor the instances

- Use CloudWatch to set up alarm rule
- Use SNS for subscribe notification
- Upload result to S3 for later analysis

Thank You