**Well Architected Framework General Guiding Principles**

* Stop guessing your capacity needs
* Test systems at production scale(by testing you can modify at intial stage)
* Automate to make architectural experimentation easier(improve the architecture so that we can complete the task fast)
* Allow for evolutionary architectures (we should allow architecture to adapt to changing requirement)
  + Design based on changing requirements
* Drive architectures using data(identifying what actually you need)
* Improve through game days
  + Simulate applications for flash sale days

**AWS Cloud Best Practices - Design Principles**

* **Scalability**: vertical & horizontal
* **Disposable Resources**: servers should be disposable & easily configured
* **Automation**: Serverless, Infrastructure as a Service, Auto Scaling…
* **Loose Coupling**:
  + Monolith are applications that do more and more over time, become bigger
  + Break it down into smaller, loosely coupled components
  + A change or a failure in one component should not cascade to other components
* **Services, not Servers**:
  + Don’t use just EC2
  + Use managed services, databases, serverless, etc..

 **Scalability (think growing bigger or smaller):**

* Vertical scaling: Add more power to existing blocks (like adding more cores to a CPU).
* Horizontal scaling: Add more blocks of the same type (like adding more servers).

 **Disposable Resources (think LEGO bricks):** Treat your servers like LEGO bricks. Easily build, configure, and replace them when needed.

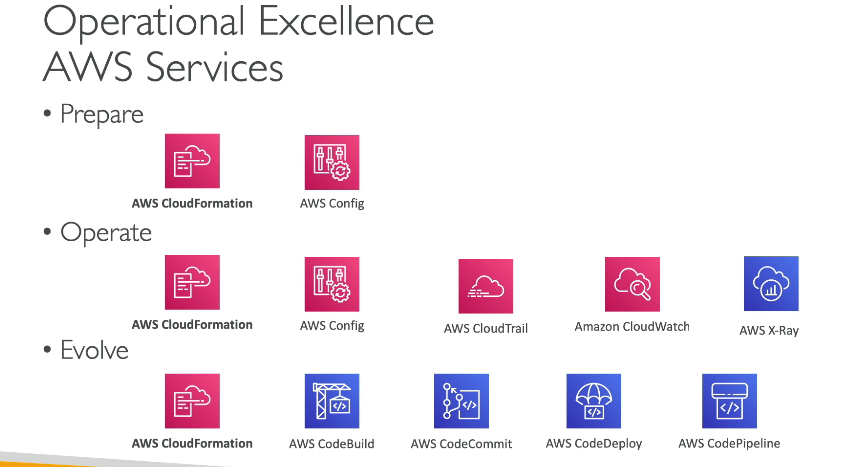
 **Automation (think following instructions):** Use tools to automatically build and manage your infrastructure, like using instructions for a LEGO set. This saves time and reduces errors.

 **Loose Coupling (think independent blocks):** Break your large application (monolith) into smaller, independent services. Like separate LEGO creations, a failure in one service doesn't affect others.

 **Services, not Servers (think using pre-built components):** Don't just use basic building blocks (EC2). Use AWS's pre-built services like managed databases and serverless options. Think of them like pre-built LEGO modules that simplify building and improve security.

### 1. Operational Excellence

* Includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures
* Design Principles
  + **Perform operations as code** - Infrastructure as code
  + **Annotate documentation** - Automate the creation of annotated documentation after every build
  + **Make frequent, small, reversible changes** - So that in case of any failure, you can reverse it
  + **Refine operations procedures frequently** - And ensure that team members are familiar with it
  + **Anticipate failure**
  + **Learn from all operational failures**



### 2. Security

* Includes the ability to protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies
* Design Principles
  + **Implement a strong identity foundation** - Centralize privilege management and reduce (or even eliminate) reliance on long-term credentials - Principle of least privilege - IAM
  + **Enable traceability** - Integrate logs and metrics with systems to automatically respond and take action
  + **Apply security at all layers** - Like edge network, VPC, subnet, load balancer, every instance, operating system, and application
  + **Automate security best practices**
  + **Protect data in transit and at rest** - Encryption, tokenization, and access control
  + **Keep people away from data** - Reduce or eliminate the need for direct access or manual processing of data
  + **Prepare for security events** - Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery
  + **Shared Responsibility Mode**

### 3. Reliability

* Ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues
* Design Principles
  + Test recovery procedures - Use automation to simulate different failures or to recreate scenarios that led to failures before
  + Automatically recover from failure - Anticipate and remediate failures before they occur
  + Scale horizontally to increase aggregate system availability - Distribute requests across multiple, smaller resources to ensure that they don't share a common point of failure
  + Stop guessing capacity - Maintain the optimal level to satisfy demand without over or under provisioning - Use Auto Scaling
  + Manage change in automation - Use automation to make changes to infrastructure

### 4. Performance Efficiency

* Includes the ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve
* Design Principles
  + **Democratize advanced technologies** - Advance technologies become services and hence you can focus more on product development
  + **Go global in minutes** - Easy deployment in multiple regions
  + **Use serverless architectures** - Avoid burden of managing servers
  + **Experiment more often** - Easy to carry out comparative testing
  + **Mechanical sympathy** - Be aware of all AWS services

### 5. Cost Optimization

* Includes the ability to run systems to deliver business value at the lowest price point
* Design Principles
  + **Adopt a consumption mode** - Pay only for what you use
  + **Measure overall efficiency** - Use CloudWatch
  + **Stop spending money on data center operations** - AWS does the infrastructure part and enables customer to focus on organization projects
  + **Analyze and attribute expenditure** - Accurate identification of system usage and costs, helps measure return on investment (ROI) - Make sure to use tags
  + **Use managed and application level services to reduce cost of ownership** - As managed services operate at cloud scale, they can offer a lower cost per transaction or service

### 6. Sustainability

* The sustainability pillar focuses on minimizing the environmental impacts of running cloud workloads.
* Design Principles
  + **Understand your impact** – establish performance indicators, evaluate improvements
  + **Establish sustainability goals** – Set long-term goals for each workload, model return on investment (ROI)
  + **Maximize utilization** – Right size each workload to maximize the energy efficiency of the underlying hardware and minimize idle resources.
  + **Anticipate and adopt new, more efficient hardware and software offerings** – and design for flexibility to adopt new technologies over time.
  + **Use managed services** – Shared services reduce the amount of infrastructure; Managed services help automate sustainability best practices as moving infrequent accessed data to cold storage and adjusting compute capacity.
  + **Reduce the downstream impact of your cloud workloads** – Reduce the amount of energy or resources required to use your services and reduce the need for your customers to upgrade their devices

