AWS Well Architected Framework

**Component** – code + configuration + AWS resource }=> Business Requirement

Workload – Set of components that together deliver a business value

[Component] = {Code, Configuration, AWS Resource}

[Component] <=> (Business Requirement)

[Workload] <=> (Business Value)

[Workload] = {[Component 1]∞[Component 2] …}

[Product Lifecycle] = {[Requirements]∞[Design]∞[Development]∞[Testing]∞[Deploy]∞[PostProduction]}

[Milestone] = {[Architectural Evolution]∞[Product Lifecycle]}

[Architecture] = {[Workload 1]∞[Workload 2] …}

[Technology Portfolio] = {[Workload 1], [Workload 2] …}

**General Design Principles**

1. **Stop guessing capacity needs** – you can start with as little capacity as you need and scale up or down automatically
2. **Test systems at production scale** – you can create production scale test environments on demand, complete your testing and decommission the resource
3. **Automate to make architectural experimentation easier**
4. **Allow for evolutionary architectures** – automate and test on demand lowers risk of impact from design changes. This allows systems to evolve over time so that businesses can take advantage of innovations as a standard practice
5. **Drive architectures using data** – collect data on how architectural choices affect the behavior of your workload. This makes fact-based decisions on how to improve your workload. Cloud infra is code, so you can use that data to inform your architectural choices and improvements over time.
6. **Improve through game days** – test how your architecture and processes perform by regularly scheduling game days to simulate events in production
7. **Operational Excellence**

* Ability to run and monitor systems to deliver business value
* Continually improve supporting processes and procedures

Design Principles

1. Perform operations as code
2. Annotate documentation
3. Make frequent small reversible changes
4. Refine operations procedures frequently
5. Anticipate failure
6. Learn from all operational failures

Best Practices

1. **Prepare** – **AWS Config and AWS Config rules** can be used to create standards for workloads and to determine if environments are compliant with those standards before being put into production
2. **Operate** – **Amazon CloudWatch** allows you to monitor the operational health of a workload
3. **Evolve** – **Amazon Elasticsearch** Service allows you to analyze your log data to gain actionable insights quickly and securely
4. **Security**

* **Ability to protect information, systems and assets while delivering business value through risk assessments and mitigation strategies**

1. **Implement a strong identity foundation**
2. **Enable traceability**
3. **Apply security at all layers**
4. **Automate security best practices**
5. **Protect data in transit and at rest**
6. **Keep people away from data**
7. **Prepare for security events**

**Best Practices**

1. **Identity and Access Management – IAM , MFA, AWS Organizations**
2. **Detective Controls – AWS CloudTrail, AWS Config, Amazon GuardDuty, Amazon CloudWatch**
3. **Infrastructure Protection – Amazon VPC, Amazon CloudFront, AWS Shield, AWS WAF, Application Load Balancer**
4. **Data Protection – ELB, Amazon EBS, Amazon S3, Amazon RDS, Amazon Macie, AWS KMS**
5. **Incident Response – IAM, AWS CloudFormation, Amazon CloudWatch**
6. **Reliability**

* **Ability of a system to recover from infrastructure or service disruptions**
* **Ability to dynamically acquire computing resources to meet demand**
* **Ability to mitigate disruptions such as misconfigurations or transient network issues**

1. **Test recovery procedures**
2. **Automatically recover from failure**
3. **Scale horizontally to increase aggregate system availability**
4. **Stop guessing capacity**
5. **Manage change in automation**

**Best Practices**

1. **Foundations – AWS IAM, Amazon VPC, AWS Trusted Advisor, AWS Shield**
2. **Change Management – AWS Cloud Trail, AWS Config, Amazon Auto Scaling, Amazon Cloud Watch**
3. **Failure Management – AWS Cloud Formation, Amazon S3, Amazon Glacier, AWS KMS**
4. **Performance Efficiency**

* **Ability to use computing resources efficiently to meet system requirements**
* **To maintain efficiency as demand changes and technologies evolve**

**Design Principles**

1. **Democratize advanced technologies**
2. **Go global in minutes**
3. **Use Serverless architectures**
4. **Experiment more often**
5. **Mechanical sympathy – Use technology approach that aligns best to what you are trying to achieve – consider data access patterns when selecting database or storage approaches**

**Best Practices**

1. **Selection**
   1. **Compute – Auto Scaling**
   2. **Storage – Amazon EBS, Amazon S3**
   3. **Database – Amazon RDS, Amazon DynamoDB**
   4. **Network – Amazon Route 53, Amazon VPC endpoints, AWS Direct Connect**
2. **Review – AWS Blog**
3. **Monitoring – AWS CloudWatch, AWS Lambda**
4. **Tradeoffs – Amazon ElastiCache, Amazon CloudFront, AWS Snowball, Read replicas in Amazon RDS**
5. **Cost Optimization**

* **Ability to run system to deliver business value at the lowest price point**

**Design Principles**

1. **Adopt a consumption model**
2. **Measure overall efficiency**
3. **Stop spending money on data center operations**
4. **Analyze and attribute expenditure**
5. **Use managed and application level services to reduce cost of ownership**

**Best Practices**

1. **Expenditure Awareness – AWS Cost Explorer, AWS Budget**
2. **Cost-Effective Resources – Cost Explorer for Reserved instance, Amazon CloudWatch, Trusted Advisor, Amazon Aurora on RDS, AWS Direct Connect and Amazon CloudFront**
3. **Matching supply and demand – Auto Scaling**
4. **Optimizing Over Time – AWS News Blog, AWS Trusted Advisor**