**Architecture:** Fundamental organization of a system, embodied in its <u>components</u>, their <u>relationships</u> to each other & the environment, and principles guiding its design & evaluation

**ISO quality requirements:** Functionality, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability

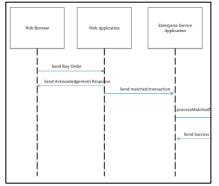
Scheduling algos: Round robin, priority, FCFS, shortest job first, shortest remaining time next, guaranteed scheduling

**Demilitarized zone**: Separates internal network from external network

Proxies: Reverse proxy hides identity of web server, forward proxy hides identity of client

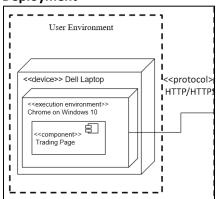
## Diagrams

## Sequence



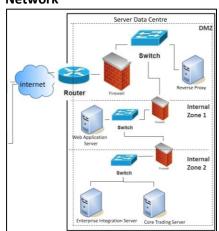
<u>Components</u>: sync msg, async msg, return msg

# Deployment



<u>Components</u>: Location, node, artifact, communication path

#### Network



<u>Components</u>: Grouping, node, communication path

### **Networking and security**

				,				
ARP	Link layer	<u>Link layer</u> protocol to find MAC address of a node within the subnet, given its IP address						
IP	Network	Network layer protocol, allows for subnetting with CIDR notation						
DNS	Application	Application layer protocol to query the IP address of a domain name. Servers can be local(router) or						
authoritative(NS)								
TCP	Transpor	<u>Transport layer</u> protocol that provides reliable data transfer						
SSL/TLS Transport layer protocol that adds a layer of security through encryption (key exchange)				change)				
HTTP	<u>Application</u>	Application layer, allows GET/POST/PUT/DELETE/PATCH						
Threat	Potential danger to information							
Vulnerabil	ity Weaknes	y Weakness within a system that can cause a breach/violation						
Asset	Anything of value to the organization							
Control Safeguards to prevent/minimize loss								
Risk	Probability of a threat exploiting a vulnerability and impacting an asset							
Threat ex.	SQL injec	SQL injection, HTTP tampering, packet sniffing, DNS spoofing, MITM, HTTPS downgrade						
Security Architecture design principles								
Mitigate	Isolate compo	solate components based on security levels, protect data in transit, storage & renewal, design for			al, design for			
	backups, limit	kups, limit attack surfaces, fail secure/safe, defense-in-depth, secure the weakest link						
Detect	Monitor for m	onitor for malicious traffic, collect logs, use tools (OWASP ZAP)						
Recovery	Data and syste	ta and system recovery methods						
Firewall	Can be statele	nn be stateless/stateful, sets inbound/outbound rules for packet filtering – one table required for each						
Rule	Interface	Src Addr	Src port	Dst Addr	Dst port	Action		
no.								
#	eth0	X.X.X.X	<port(s)></port(s)>	X.X.X.X	<port(s)></port(s)>	ALLOW/DENY		
Maintainability: Modularity. Analyzability. Testability. Reusability. Modifiability								

	Maintainability: Modularity, Analyzability, Testability, Reusability, Modifiability	
Layered archi	Logical separation of software components – layers are dependent on those below and	
	independent of those above. Can be open(bypassable) or closed(non-bypassable)+(Easy to debug,	
	separation of responsibilities, reusability) –(Performance overhead, redundant code)	
Client-server	Client sends requests, server returns responses. Can be strict client-server or p2p	
Tiered archi	Tiers are treated as "layers", but components <u>physically separate</u> . Each tier is a server for its caller	
	and client for its callee	
CAP theorem	Consistency, availability, tolerance to partitions	

Activity   Illustrates various stages in the software development strategy, (e.g. develop >> test >> compile >> delogy to staging >> STE & LAT >> deploy to production)			Maintainability – Development Strategy				
delegory to staging > SIT & UAT > deploy to production)  Cloudeformation  Comprises description, metadata, parameters, mappings, conditions, resources and outputs  Grouping  Break up a template into smaller ones based on ownership and responsibility  Enterprise integration patterns  File transferred in batch mode with defined format, protocol, frequency and server architecture.  (+) universal integration, simple, high abstraction, (-) error processing, data sync, data-only/no metadata), one-way communication  Database  Integration  Broak up a low-eway communication  Database  Supports batch or record/hor mode, guarantees atomicity, consistency, isolation, durability (ACID), allows shared/replicated DB configs  Integration  Integration  Integration  Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence/whether messages are stored on disk), durability/whether messages are still received when subscribes in sor turning) and exception handling. Define model[point-point/pubsub) and communication (polling/event-driven, sync/async)  Define functions, input/output params and exception handling. Expose function hrough a protocol. SQAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Brokering  Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services, consider tradeoffs in speed, security, scalability  Availability  Tradeoff  Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Architectures: Thin client, thick client, native passive: How redundant nodes are configured to handle traffic failure detection: Can be indouncel/pinging) or outbound(flerentbeat)  Failured etcliention: A can be replicated through DB/cache, client, or in-memory  Database: C	Activity	Illust					
CloudFormation   Comprises description, metadata, parameters, mappings, conditions, resources and outputs	-						
Break up a template into smaller ones based on ownership and responsibility	CloudFormation						
File transfer File transfer File transfer Files transferred in batch mode with defined format, protocol, frequency and server architecture. (+) universal integration, simple, high abstraction, (-) error processing, data sync, data-only(no metadata), one-way communication Supports batch or record/tox mode, guarantees atomicity, consistency, isolation, durability (ACID), allows shared/replicated DB configs Integration Interface Screen/web scraping presentation layer, (+) minimal changes (-) consistency, formatting, preprocessing, speed Messaging Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)  Perform transformations on data payloads from one endpoint to another. (+) Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal failure active vs active vs active passive: How redundant nodes are configured to handle traffic failure active vs active vs active passive: How redundant nodes are configured to handle traffic failure active vs active passive: How redundant nodes are configured to handle traffic failure active vs active passive: How redundant nodes are configured to handle traffic failure active received by rounning multiple artifacts in a single node, horizontal redundancy. Process of redirecting traffic/workload after a m							
(+) universal integration, simple, high abstraction, (-) error processing, data sync, data-only(no metadata), one-way communication  Database Supports batch or record/txn mode, guarantees atomicity, consistency, isolation, durability (ACID), allows shared/replicated DB configs Interface Interface Screen/web scraping presentation layer, (+)minimal changes (-)consistency, formatting, preprocessing, speed  Messaging Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsib) and communication (polling/event-driven, sync/async)  APIS  Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDI Interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Brokering Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (+) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Sessions: Can be replicated through DB/cache, client, or in-memory  Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where bo							
(+) universal integration, simple, high abstraction, (-) error processing, data sync, data-only(no metadata), one-way communication  Database Supports batch or record/txn mode, guarantees atomicity, consistency, isolation, durability (ACID), allows shared/replicated DB configs Interface Interface Screen/web scraping presentation layer, (+)minimal changes (-)consistency, formatting, preprocessing, speed  Messaging Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsib) and communication (polling/event-driven, sync/async)  APIS  Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDI Interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Brokering Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (+) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Sessions: Can be replicated through DB/cache, client, or in-memory  Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where bo	File transfer File		· · ·				
Database   Integration   Integration   Integration   Integration   Screen/web scraping presentation layer, (+)-minimal changes (-) consistency, formatting, allows shared/replicated DB configs   Integration   Screen/web scraping presentation layer, (+)-minimal changes (-) consistency, formatting, preprocessing, speed   Handles transfer of records/txms and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (poling/event-driven, sync/async)   Define functions, input/output params and exception handling. Expose function through a protocol. SQAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods   Perform transformations on data payloads from one endpoint to another. (+)Can link any integration patern. (-) incurs performance overhead   Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability   Tradeoff   Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)   Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes   Active-active vs active-passive: How redundant nodes are configured to handle traffic   Ealiure detection: Can be inbound(pinging) or outbound(heartbeat)   Ealiove: Process of redirecting traffic/workload after a machine fails. Can be cellent-based, done by load balancer, DNS or virtual IPs   Sessions: Can be replicated through DB/cache, client, or in-memory   Database: Can be done in master-slave (where slave acts as a backup wread permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Separation of c							
Integration allows shared/replicated DB configs  Interface Integration Screen/web scraping presentation layer, (+) minimal changes (-) consistency, formatting, preprocessing, speed  Messaging Handles transfer of records/txns and can be centralized/decentralized. Define file format. selectivity. persistence(whether messages are stored on disk), durability (whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)  APIS Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Brokering Perform transformations on data payloads from one endpoint to another. (+) Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monoithic, services. Consider tradeoffs in speed, security, scalability  Availability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering Active-active vs active-assive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer. DNS or virtual IPS  Segsarion of concern: loose coupling be tween critical & non-critical components  Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state  Segaration of concern: loose coupling be tween critical & non-critical components  Fault tolerance; Ensure that errors do not terminate a s		meta					
Interface   Screen/web scraping presentation layer, (+)minimal changes (-)consistency, formatting, preprocessing, speed   Messaging   Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)   Define functions, input/output params and exception handling. Expose functionality. REST organizes API into resources, and defines operations using HTTP methods   Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead   Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability   Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)   Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)   Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPS   Replication   Sessions: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Separation of concern: Loose coupling between critical & non-critical components   Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state   Parallel vs series: Having services in series always increases failure rate   Parallel vs series: Having services in series always increases failure rate   Parallel vs series: Having services in se	Database	Supp	orts batch or record/txn mode, guarantees atomicity, consistency, isolation, durability (ACID),				
Integration   preprocessing, speed   Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability(whether messages are stored on disk), durability (befine methods in XMIL, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Brokering   Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability	integration	allow	s shared/replicated DB configs				
Handles transfer of records/txns and can be centralized/decentralized. Define file format, selectivity, persistence(whether messages are stored on disk), durability (whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)    APIS   Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSD. interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods    Brokering	Interface						
Selectivity, persistence(whether messages are stored on disk), durability/whether messages are still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)    APIS	integration						
still received when subscriber is not running) and exception handling. Define model(point-point/pubsub) and communication (polling/event-driven, sync/async)  Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern. (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Tradeoff  Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering  Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs  Sessions: Can be replicated through DB/cache, client, or in-memory  Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design  Separation of concern: Loose coupling between critical & non-critical components techniques  Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state  Parallel vs series: Having services in series always increases failure rate  AWS archi  Liss to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilizati	Messaging Handle						
point/pubsub) and communication (polling/event-driven, sync/async)  Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods  Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Availability  Tradeoff  Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering  Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs  Replication  Sessions: Can be replicated through DB/cache, client, or in-memory  Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design  Separation of concern: Loose coupling between critical & non-critical components fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state  Parallel vs series: Having services in series always increases failure rate  AWS archi  Elßs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  A class should have one & only one reason to chan			<del></del>				
Define functions, input/output params and exception handling. Expose function through a protocol. SOAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods in the protocol. SOAP defines methods in XML, uses WSDL interfaces to expose functionality. REST organizes API into resources, and defines operations using HTTP methods in the gration pattern, (-) incurs performance overhead integration and performance overhead integration of considering the protocological protocological performance integration and performance integration and performance integration and performance integration and performance integration of concern: Loose coupling between critical & non-critical components fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state parallel vs series: Having services in series always increases failure rate  AWS archi Elbs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics Time behaviour, resource utilization, capacity  Techniques Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling (AWS)Use ElB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  SOLID principles and design patterns  SOLID principles and design patterns  AC ac							
protocol. <u>SOAP</u> defines methods in XML, uses WSDL interfaces to expose functionality. <u>REST organizes API into resources, and defines operations using HTTP methods</u> Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Tradeoff  Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering  Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)  Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs  Replication  Sessions: Can be replicated through DB/cache, client, or in-memory  Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design  Separation of concern: Loose coupling between critical & non-critical components  Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state  Parallel vs series: Having services in series always increases failure rate  AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  Load balancing, parallel execution, caching and pre-fetching  (AWS) use ELB, autoscaling and cloudwatch to configure EC2 instances to s							
Brokering Perform transformations on data payloads from one endpoint to another. (+)Can link any integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Availability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat) Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs  Replication Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design Separation of concern: Loose coupling between critical & non-critical components techniques  Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state  Parallel vs series: Having services in series always increases failure rate  AWS archi Elbs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics Time behaviour, resource utilization, capacity  Techniques Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility Operation of patterns  Single responsibility Operation of	APIs						
Perform transformations on data payloads from one endpoint to another. (+) Can link any integration pattern, (-) incurs performance overhead   Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability   Availability   Specified functions, conditions and time frames   Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple ondes   Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be elient-based, done by load balancer, DNS or virtual IPs   Sessions: Can be replicated through DB/cache, client, or in-memory   Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Separation of concern: Loose coupling between critical & non-critical components   Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state   Parallel vs series: Having services in series always increases failure rate   AWS archi   ElBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions   Performance   Pe		-	<del></del>				
integration pattern, (-) incurs performance overhead  Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy  Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Clustering  Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat) Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs  Replication  Sessions: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design  Separation of concern: Loose coupling between critical & non-critical components fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state Parallel vs series: Having services in series always increases failure rate  AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  AWS autoscaling (AWS) Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Derived classes must be able to substitute their base classes  Interface segregation  Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require t			· · · · · · · · · · · · · · · · · · ·				
Architectures: Thin client, thick client, native, monolithic, services. Consider tradeoffs in speed, security, scalability  Availability  Tradeoff Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)  Redundancy Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes  Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat) Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DIS or virtual IPs  Replication Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design Separation of concern: Loose coupling between critical & non-critical components Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state Parallel vs series: Having services in series always increases failure rate  AWS archi ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics Time behaviour, resource utilization, capacity  Techniques Load balancing, parallel execution, capacity  Techniques AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification Derived classes must be able to substitute their base classes  Depend on abstractions, not concrete implementations. Dependencie	Brokering						
Availability   Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)   Redundancy   Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes   Clustering   Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat) Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs	A selection of the Thirt						
Availability (operational & accessible) vs reliability (specified functions, conditions and time frames)   Redundancy   Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes   Clustering   Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs   Replication   Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Design   Separation of concern: Loose coupling between critical & non-critical components fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state	Architectures: Ini	n clier					
Redundancy   Vertical redundancy achieved by running multiple artifacts in a single node, horizontal redundancy achieved by running multiple nodes   Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs	Tradeoff	A	,				
Vertical redundancy   Vertical reduced   Vertical reduced   Vertical redundancy   Vertical redundancy   Vertical reduced	Tradeoff						
Clustering	Podundancy	_	•				
Active-active vs active-passive: How redundant nodes are configured to handle traffic Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs   Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Separation of concern: Loose coupling between critical & non-critical components Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state Parallel vs series: Having services in series always increases failure rate   AWS archi   ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions   Performance   Metrics   Time behaviour, resource utilization, capacity   Techniques   Load balancing, parallel execution, caching and pre-fetching   (AWS) Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage   SOLID principles and design patterns	Reduituality						
Failure detection: Can be inbound(pinging) or outbound(heartbeat)   Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs	Clustering						
Failover: Process of redirecting traffic/workload after a machine fails. Can be client-based, done by load balancer, DNS or virtual IPs   Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Design	_						
By load balancer, DNS or virtual IPs   Replication   Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Design   Separation of concern: Loose coupling between critical & non-critical components							
Sessions: Can be replicated through DB/cache, client, or in-memory Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences   Design							
Database: Can be done in master-slave (where slave acts as a backup w read permissions) or master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences    Design	Replication						
master-master (where both DBs can write). Important to implement precedence rules, primary key segregation or domain-specific differences  Design Separation of concern: Loose coupling between critical & non-critical components Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state Parallel vs series: Having services in series always increases failure rate  AWS archi ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics Time behaviour, resource utilization, capacity Techniques Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes  Interface segregation Make client-specific interfaces  Dependency inversion Dependency inversion Dependency inversion Dependency inversion	•		<del></del>				
Separation of concern: Loose coupling between critical & non-critical components   Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state   Parallel vs series: Having services in series always increases failure rate			<del></del>				
Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded state Parallel vs series: Having services in series always increases failure rate  AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling  (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Liskov substitution  Derived classes must be able to substitute their base classes  Interface segregation  Make client-specific interfaces  Dependency inversion  Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them		key	segregation or domain-specific differences				
state Parallel vs series: Having services in series always increases failure rate  AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling  (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Liskov substitution  Derived classes must be able to substitute their base classes  Interface segregation  Make client-specific interfaces  Dependency inversion  Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them			aration of concern: Loose coupling between critical & non-critical components				
Parallel vs series: Having services in series always increases failure rate  AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics  Time behaviour, resource utilization, capacity  Techniques  Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling  (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Liskov substitution  Derived classes must be able to substitute their base classes  Interface segregation  Make client-specific interfaces  Dependency inversion  Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	techniques	<u>Fau</u>	Fault tolerance: Ensure that errors do not terminate a service but cause it to enter a degraded				
AWS archi  ELBs to provide health-checks, security, TLS, balancing, elastic IP for virtual IP configs, route 53 for as an authoritative DNS for availability cross regions  Performance  Metrics Time behaviour, resource utilization, capacity  Techniques Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes Interface segregation Make client-specific interfaces  Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	sta		re				
Ferformance  Metrics Time behaviour, resource utilization, capacity  Techniques Load balancing, parallel execution, caching and pre-fetching  AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes  Interface segregation Make client-specific interfaces  Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them		_					
Metrics   Time behaviour, resource utilization, capacity     Techniques   Load balancing, parallel execution, caching and pre-fetching     AWS autoscaling   (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage	I		<del>-</del> •				
Time behaviour, resource utilization, capacity	for		·				
Techniques  AWS autoscaling  (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Liskov substitution  Derived classes must be able to substitute their base classes  Interface segregation  Make client-specific interfaces  Dependency inversion  Dependency inversion  Dependency them		1					
AWS autoscaling (AWS)Use ELB, autoscaling and cloudwatch to configure EC2 instances to scale out and scale in according to traffic with step adjustments according to CPU usage  SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes  Interface segregation Make client-specific interfaces  Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them		_					
SOLID principles and design patterns  Single responsibility A class should have one & only one reason to change  Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes  Interface segregation Make client-specific interfaces  Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	•						
SOLID principles and design patterns  Single responsibility  A class should have one & only one reason to change  Open-closed  Open for extension, closed for modification  Liskov substitution  Derived classes must be able to substitute their base classes  Interface segregation  Make client-specific interfaces  Dependency inversion	AWS autoscaling						
Single responsibilityA class should have one & only one reason to changeOpen-closedOpen for extension, closed for modificationLiskov substitutionDerived classes must be able to substitute their base classesInterface segregationMake client-specific interfacesDependency inversionDepend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them		acc					
Open-closed Open for extension, closed for modification  Liskov substitution Derived classes must be able to substitute their base classes  Interface segregation Make client-specific interfaces  Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	Single reer small-!!	ity	· · · · · · · · · · · · · · · · · · ·				
Liskov substitution         Derived classes must be able to substitute their base classes           Interface segregation         Make client-specific interfaces           Dependency inversion         Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	Single responsibil	ity	A class should have one & only one reason to change				
Liskov substitution         Derived classes must be able to substitute their base classes           Interface segregation         Make client-specific interfaces           Dependency inversion         Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	Onen-closed		Open for extension, closed for modification				
Interface segregation         Make client-specific interfaces           Dependency inversion         Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them	•		·				
Dependency inversion Depend on abstractions, not concrete implementations. Dependencies should be injected into classes that require them							
into classes that require them	Interface segregation		Make client-specific interfaces				
	Dependency inversion		Depend on abstractions, not concrete implementations. Dependencies should be injected				
Singleton Create 1 private, static instance of the class and provide global access to all other classes			into classes that require them				
	Singleton		Create 1 private, static instance of the class and provide global access to all other classes				

Builder	Create classes in a step-by-step process of construction with a builder object
Factory	Define an interface for creating an object but let classes decide
Adapter	Convert existing interface into another interface that clients/users expect
Façade	Provide an interface to a set of interfaces that make the subsystem easier to use
Chain of	Pass requests along a chain of objects (linked list) to avoid excessive coupling between
Responsibility	sender and receiver
Observer	Create a one-many dependency between observers and subject, so that changes in subject are shared with all observers (push model)