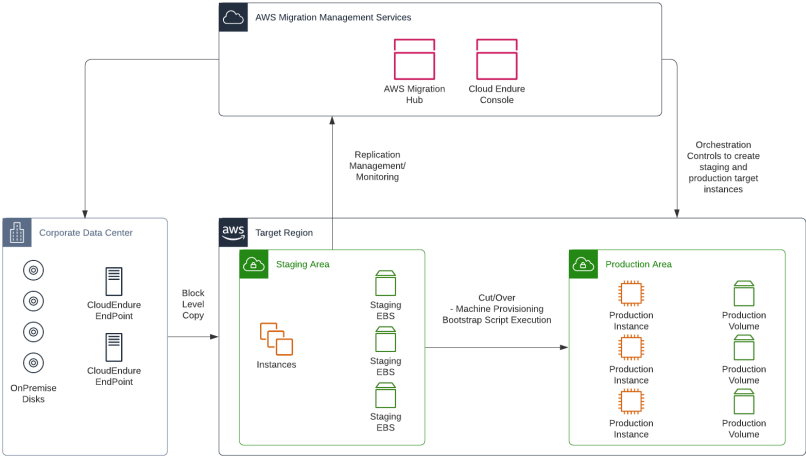


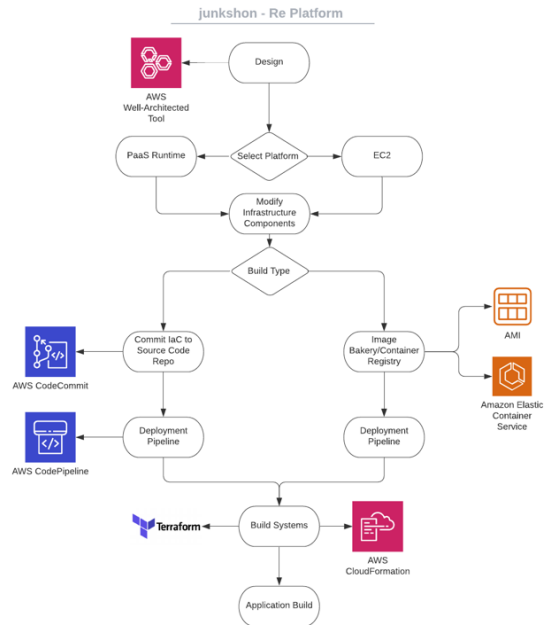
Migration Treatments

The treatments section provides an overview of the different technical patterns available for the migrations.

Migration Solution	
Re-host Cloud Endure	
<div><p>junkshon - ReHost CloudEndure</p></div>	<p>Overview</p> <p>Cloud Endure Migration works through an agent that you install on your source machines. The Cloud Endure Agent connects to the Cloud Endure User Console, which issues an API call to your target AWS Region. The API call creates a Staging Area in your AWS account that is designated to receive replicated data.</p> <p>Cloud Endure Migration</p> <p>Cloud Endure Migration automated rehosting consists of three main steps:</p> <ol style="list-style-type: none">1. Installing the Agent: The Cloud Endure Agent replicates entire machines to a Staging Area in your target.2. Configuration and testing: You configure your target machine settings and launch non-disruptive tests.3. Performing cutover: Cloud Endure automatically converts machines to run natively in AWS. <p>The Staging Area comprises both lightweight EC2 instances that act as replication servers and staging EBS volumes.</p> <p>Each source disk maps to an identically sized EBS volume in the Staging Area.</p> <p>The replication servers receive data from the Cloud Endure Agent running on the source machines and write this data onto staging EBS volumes.</p> <p>After all source disks copy to the Staging Area, the Cloud Endure Agent continues to track and replicate any changes made to the source disks. Continuous replication occurs at the block level.</p> <p>When the target machines launch for testing or cutover, Cloud Endure automatically converts the target machines so that they boot and run natively on AWS.</p> <p>Scalability Characteristics</p> <p>One replication server can handle multiple source machines replicating concurrently.</p>

Migration Solution

Re-platform



Overview

The re-platform architecture blueprint moves the application from the existing runtime to a new platform, this approach is different from rehost because the application and its components are reinstalled.

Re Platform Flow Design: The design phase uses the AWS well architected framework and tool to evaluate the existing workload and define the action plan/blueprint for the application.

Select Platform:

The select platform decision box represents the decision to consume either a higher-level AWS application service like containerisation or run-time solution like AWS Beanstalk or Container Solution

Modify Infrastructure Components:

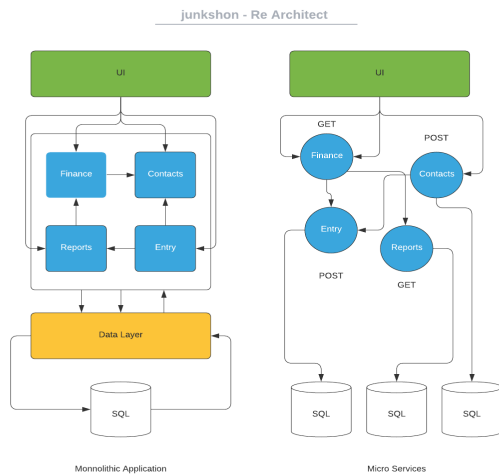
This step is where infrastructure software is modified for the new build. e.g. agent software packages/application libraries and dependencies.

Build Type:

Select the build type depending on the platform type selected this can either be an immutable build process based upon infrastructure and application build scripts stored in a repository such as code commit or a machine image. Deployment Pipeline: The deployment pipeline will either build the application stack onto the targeted PaaS solution or a machine image is made available for the application teams to consume.

Migration Solution

Re-factor



Overview

The re architect blueprint takes a legacy monolithic application and re-architect it to use a combination of cloud services to increase scalability, performance and manageability.

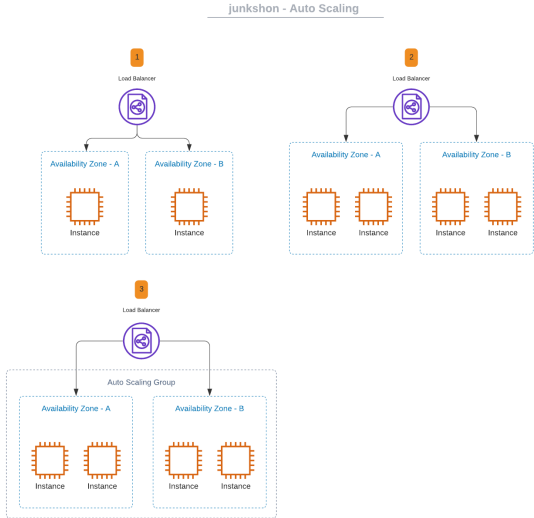
Re Architect Pattern Monolithic Application:

In software engineering, a monolithic application describes a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform. A monolithic application is self-contained, and independent from other computing applications.

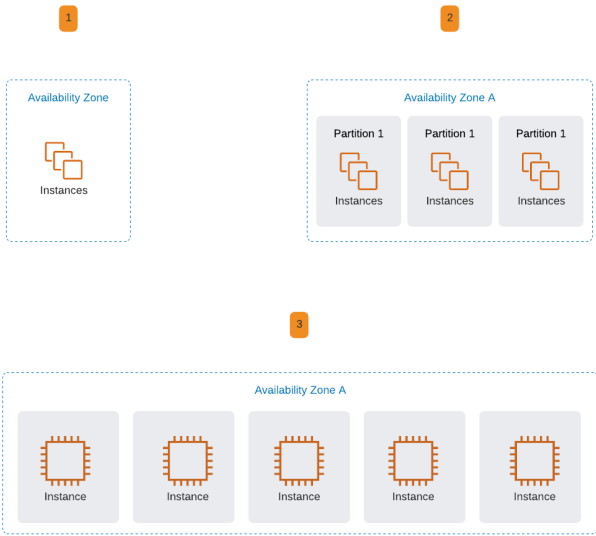
Microservices Application:

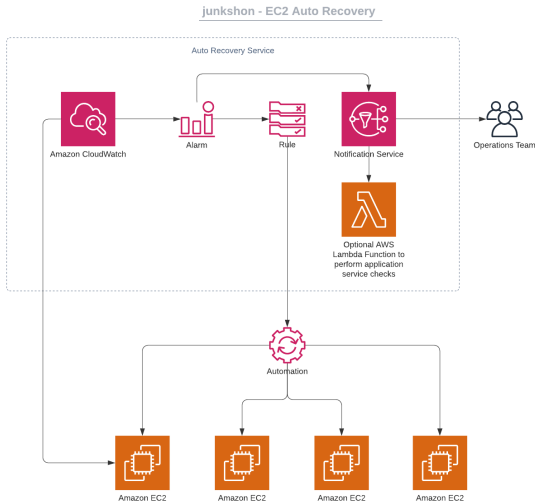
Microservices- also known as the microservice architecture- is an architectural style that structures an application as a collection of services that are. Highly maintainable and testable. Loosely coupled. Independently deployable. Organized around business capabilities.

Compute Target State Patterns

Target Solution	Treatment
Auto Scaling	Re Host Re Platform Re Architect Re Purchase
<div><p>junkshon - Auto Scaling</p></div>	<p>Overview</p> <p>Auto scaling is the use of technology to scale up and down workload automatically based upon rules and historical performance data. Using auto scaling you should not over-provision the resources to meet the peak demand.</p> <p>Auto Scaling Architectural Patterns</p> <p>Dynamic scaling - the biggest advantage of auto scaling is the scaling of resources based upon demand.</p> <p>Health check - you can monitor the health of the EC2 instances if your application runs on more than once EC2 instance also known as a <i>fleet</i>.</p> <p>Load balancing - is used to dynamically scale up and down resources, the load balancer takes care of balancing the workload across multiple EC2 instances.</p> <p>Target tracking - you can use auto scaling to run on a particular target and scale resources on metrics e.g. cpu utilization or network utilization.</p> <p>Auto Scaling Groups</p> <p>Auto scaling groups is a feature where logic is defined for scaling up and down instances, an auto scale group is a collection of EC2 servers running together as a group. You can define auto scale groups to scale up and down in the following ways:</p> <p>Maintaining instance levels: This is a default scaling plan; in this policy you define a minimum number of instances.</p> <p>Manual scaling: instances are scaled up and down manually either via the console or the API or CLI.</p> <p>Scaling per demand: another auto scaling group option is to scale based upon on-demand based upon CloudWatch metrics CPU, disk reads, disk writes and network.</p> <p>Scaling as per schedule: scaling instances based upon pre-defined workload schedule e.g. during Monday morning peak.</p> <p>Patterns</p> <p>Single server instances running in two different availability zones.</p> <p>Use Cases: web/application servers requiring high availability but no-scaling requirements.</p> <p>Multiple server instances running in two different availability zones.</p> <p>Use Cases: web/application servers that require additional nodes to support the specific application performance requirements, application must be capable of providing appicate state management.</p> <p>Multiple servers running in two availability zones, servers placed in auto scaling group to provide dynamic and scheduled scaling.</p>

	Use Case: web/application servers that either require or can support dynamic scaling solution.
--	---

Target Solution	Treatment
Placement Groups	Re Host Re Platform Re Architect Re Purchase
<p style="text-align: center;"><u>junkshon - Placement Groups</u></p>  <p>The diagram shows three numbered examples of placement groups within an Availability Zone:</p> <ul style="list-style-type: none"> 1. Availability Zone: A dashed box labeled 'Availability Zone' containing a single group of three 'Instances' represented by orange squares. 2. Availability Zone A: A dashed box labeled 'Availability Zone A' containing three separate 'Partition 1' boxes, each containing three 'Instances' (orange squares). 3. Availability Zone A: A dashed box labeled 'Availability Zone A' containing five individual 'Instance' boxes, each represented by a square with a chip icon. 	<p>Overview Placement groups - when launching EC2 instances, the underlying EC2 control plane will attempt to spread instances across the underlying hardware to minimize the blast radius in the event of an underlying hardware failure.</p> <p>Placement Groups Architectural Patterns Cluster – A cluster placement group is a logical grouping of instances within a single availability zone, the cluster placement group have access to higher throughput up 10Gbps of TCP/IP traffic.</p> <p>Partition A partition placement groups helps to reduce the likelihood of a hardware failure that impacts the application. EC2 group into a partition that ensures each partition has its own set of racks which have power and network services.</p> <p>Spread A spread placement group is a group of instances that are each placed on distinct racks with its own power and network services.</p> <p>Patterns</p> <ol style="list-style-type: none"> 1. Cluster placement groups are recommended for applications that benefit from low network latency, high network throughput, or both. They are also recommended when the majority of the network traffic is between the instances in the group. To provide the lowest latency and the highest packet-per-second network performance for your placement group, choose an instance type that supports enhanced networking. 2. Partition placement groups can be used to deploy large distributed and replicated workloads, such as HDFS, HBase, and Cassandra, across distinct racks. When you launch instances into a partition placement group, Amazon EC2 tries to distribute the instances evenly across the number of partitions that you specify. 3. Spread placement groups are recommended for applications that have a small number of critical instances that should be kept separate from each other. Launching instances in a spread placement group reduces the risk of simultaneous failures that might occur when instances share the same racks. Spread placement groups provide access to distinct racks and are therefore suitable for mixing instance types or launching instances over time.

Target Solution	Treatment
AWS EC2 Auto Recovery	Re Host Re Platform Re Architect Re Purchase
 <p>The diagram, titled 'Junkshon - EC2 Auto Recovery', illustrates the workflow for recovering failed EC2 instances. It features an 'Auto Recovery Service' box containing 'Amazon CloudWatch', 'Alarm', 'Rule', and 'Notification Service'. An 'Optional AWS Lambda Function to perform application service checks' also feeds into the 'Rule'. An 'Operations Team' icon is connected to the 'Notification Service'. Below the service box, an 'Automation' icon (AWS Step Functions) is connected to four 'Amazon EC2' instance icons. Arrows indicate the flow from CloudWatch to Alarm, then to Rule, which triggers the Notification Service and the Automation workflow. The Lambda function also triggers the Rule. The Automation workflow then acts on the EC2 instances.</p>	<p>Overview EC2 auto recovery is a solution that uses a number of AWS components to provide a compute recovery service in the event of an underlying hardware failure or problem that requires AWS involvement to repair.</p> <p>Auto Recovery Architectural Pattern When the StatusCheckFailed_System alarm is triggered, and the recover action is initiated, you will be notified by the Amazon SNS topic that you selected when you created the alarm and associated the recover action.</p> <p>During instance recovery, the instance is migrated during an instance reboot, and any data that is in-memory is lost.</p> <p>When the process is complete, information is published to the SNS topic you've configured for the alarm.</p> <p>In addition to the infrastructure restore it is also possible to execute additional workflows to validate the application status.</p> <p>Examples of problems that cause system status checks to fail include:</p> <ul style="list-style-type: none"> • Loss of network connectivity • Loss of system power • Software issues on the physical host Hardware issues on the physical host that impact network reachability <p>Requirements The recover action is supported only on instances with the following characteristics: -</p> <p>Runs in a virtual private cloud (VPC) - Uses default or dedicated instance tenancy</p> <p>Has only EBS volumes (do not configure instance store volumes)</p>

Target Solution	Treatment
<div data-bbox="193 293 1010 331"> <p>AWS RDS</p> </div> <div data-bbox="225 342 948 864"> <p>The diagram illustrates the AWS RDS Multi-AZ architecture. It shows two Availability Zones, each containing a Private Subnet. In the left Availability Zone, there is an RDS DB Instance Primary and an RDS DB Read Replica. In the right Availability Zone, there is an RDS DB Instance Standby. Arrows indicate 'Async data replication' from the Primary to the Read Replica and 'Sync data replication' from the Primary to the Standby. Below the Availability Zones, a section titled 'Architecture Features' lists five capabilities: Scale Database Instances using API, Simple Database Patching, Choice of Storage Services Software Defined, Integrated Backup, and Optional Snapshots.</p> </div>	<div data-bbox="1010 293 1565 331"> <p>Database Re-platform</p> </div> <div data-bbox="1010 360 1565 1507"> <p>Overview</p> <p>Amazon Web Services (AWS) customers bet their businesses on their data store and highly available access to it. For these customers, Multi-AZ configurations provide an easy-to-use solution for high availability (HA).</p> <p>Architectural Pattern</p> <p>The Multi-AZ feature is implemented using a replication layer installed between the database application and the EBS Elastic Block Store volumes.</p> <p>This layer handles application read and write requests and applies them in an environment where two discrete EBS volume copies are maintained done accessed locally and one accessed remotely.</p> <p>There are two RDS DB instances in this Multi-AZ instance: the primary instance and the standby instance on the right-hand side. The two instances are connected, and the synchronous data replication takes place to write data to the standby instance. In situations where the RDS service detects a problem the failover is initiated by automation; it is also possible to trigger failover via the RDS API. The replication layer has limited visibility above itself and is therefore incapable of making additional decisions for example: connectivity issues, local or regional outages. The availability and durability improvements provided by the Multi-AZ come at minimal performance cost, in normal operation the replication layers are connect and synchronous write operations to the standby EBS volume.</p> <p>RDS Read Replica</p> <p>In addition to a standby instance it is also possible to establish an RDS read replica that is an option for applications that have a heavy read use-case, read actives can be off loaded to a replica. Typical application use case maybe a Business Intelligence application.</p> <p>Architecture Features</p> <p>Ability to scale up and down database instances via the integrated API. Reduction of operational tasks by utilising the simple patching process provided by the RDS service.</p> <p>RDS offers different storage service tiers to allow for optimisation of the underlying storage technology to the specific application use-case.</p> <p>Integrated backup and snapshots, database backup provided by the RDS service and simple API to execute automated snapshots.</p> </div>