Qalius CONSULTING INC.

WHITEPAPER:

Twelve Opportunities in Quick Migration Projects

Abstract

Application migrations often get compressed into a short intense project when a long decision-making process concludes with approval only a few weeks before a deadline. In these conditions one is tempted to simply lift-and-shift the application into AWS while changing as little as possible. This rehost migration pattern misses an opportunity to improve the performance, security, and management of the application. In this whitepaper we challenge you to consider twelve potential improvements in the application hosting that you may be able to implement during your migration project. These are high-impact and low-risk improvements to consider even when time is short and risk tolerance is low.

Introduction

Often a specific milestone—such as a product launch, hardware end-of-life, or security mandate—compels a business to approve an application migration into AWS with a firm deadline. A natural instinct in this circumstance is to change as little as possible during the migration. In the very short term this might minimize risk, but it also misses an opportunity to use the tremendous capabilities of AWS to improve the value the application delivers to the business. This paper is about how best to get your application into AWS while simultaneously improving application performance, security, or management. Often you can make substantial improvements even when time is tight and risk tolerance is low.

If you migrate your application to AWS and you don't make a significant improvement in application performance, security, or management you've missed a big opportunity to retire technical debt.

In fact, the application migration project is often a particularly good time to make improvements. It is a technical opportunity because the AWS platform presents a broad and deep set of tools, many of which were probably not available in the previous on-premises installation. It is typically also a practical opportunity because a cross-disciplinary team is made available to support the migration. A team that includes development, solution architecture, and end users can support implementation tasks and thorough testing. Consider how hard it is to assemble such a team when the focus of the business has moved on to other priorities.

The 6R Migration Patterns

AWS have found that application migrations into AWS fall into six migration patterns: Retain, Retire, Rehost, Replatform, Repurchase, and Re-Architect. These "6R's" help inform the lifecycle planning for an application portfolio. See the panel The 6R Migration Patterns for an explanation of the six patterns.

When we meet project teams facing pressure to migrate an application quickly to AWS, they often think of Rehosting, and with good reason. Rehosting aims to change nothing other than switching to AWS virtualized infrastructure. It may be the only option when you can't modify or recompile the application code.

However, with a little bit of support from developers or solution architects, often the migration team can make small changes to the application that pay huge dividends in improved application performance, security, and maintainability. AWS call this the Replatform migration pattern. A Replatform migration incorporates small modifications to the application so it can better take advantage of AWS. The one-time investment in IT resources during the migration pays dividends over the life of the application.

AWS provide extensive documentation to inform these architectural decisions. For example, the AWS Well-Architected Framework distills experience from thousands of application deployments into a set of best practices organized in five pillars: operational excellence, performance efficiency, security, reliability, and cost optimization. When you have the luxury of re-architecting your application you can comprehensively apply the well-architected framework. But even when you have a deadline bearing down, the application migration is an opportunity to make substantial improvements by adopting some key AWS services provisioned according to best practices. By carefully selecting which AWS services you adopt you'll have minimal impact on your project timelines while making improvements with big impact.

The 6R Migration Patterns

AWS analyzed countless application migrations to build a classification of migration types. The result is the 6R Migration Patterns listed below. Migration planning and execution are streamlined by assigning each application in a portfolio to one of the migration patterns.

Retain

The application continues to be operated on premises.

Retire

The application is decommissioned without migration.

Rehost

The application components undergo no change or very minimal change. The components are moved without recompiling, altering the code, or updating operating system and application services stacks.

> *This paper asks you to consider elements of the Replatform pattern when you might otherwise be tempted to settle on the Rehost pattern.*

Replatform

The application components undergo limited updates that allow them to take significant advantage of the AWS platform services. Expect to update the operating system, application services, and database services while possibly making small changes to the code.

Repurchase

The application is replaced with a commercial solution, typically in a software-as-a-service (SaaS) model.

Re-architect

The application functions are implemented with broad adoption of AWS cloud-native services. Sometimes also called the Refactor pattern, this migration pattern is the most impactful but takes the greatest investment of developer and solution architect resources.

> *Re-architecting can turn an existing conventional application into a cloud-native serverless application. Re-architecting often starts with a Replatform migration followed by refactoring of one application component at a time.*

This description of the 6R Migration Patterns is adapted from the AWS whitepaper "Understanding Your Application Readiness when Migrating to AWS".

Migration Case Example

Let's look at a specific example.

Application

Three tiers on LAMP stack.

Business Context

Investment in this application hasn't kept up with the growth in its importance to the business so the application stakeholders are eager to make improvements. System integration testing and user acceptance testing will be available. Risks need to be carefully assessed to ensure that the migration doesn't miss the launch deadline.

Business Objectives

The application must be migrated to AWS and ready to perform critical business functions within a few weeks. Within that constraint, seize every practical opportunity to improve performance, security, and management.

We're going to consider twelve practical ideas for how to achieve this business objective, but first let's look at a pure rehosting migration that will provide an architectural comparison.

The Rehost Migraton

A simple rehosting of the three-tier web application would make limited use of AWS. Application components that are running on physical or virtual hardware on-premises can be usually be installed on Amazon EC2 virtual servers with minimal changes. Architecturally, the resulting configuration in AWS could be as simple as Figure 1. The application will benefit from the flexibility and performance of Amazon EC2 but fail to exploit the broader capabilities of the AWS platform.

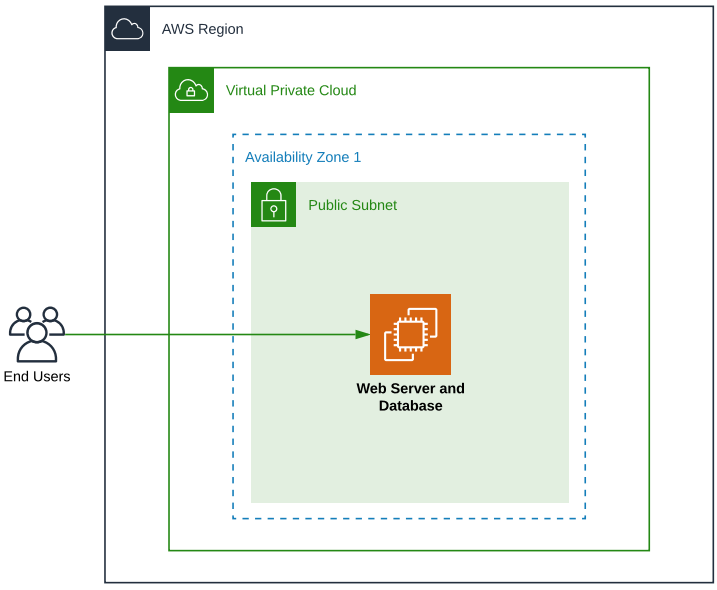

Figure 1: *Rehost Migration. Application components that are running on physical or virtual hardware on-premises are installed on Amazon EC2 virtual servers with minimal changes. The application will benefit from the flexibility and performance of Amazon EC2 but fail to exploit the broader capabilities of the AWS platform.*

In the short-term this migration may be low risk, but it is also low reward.

The Replatform Migration

We can do much better if we are willing to make some small changes. Figure 2 shows the application architecture that could result from applying all twelve recommendations that appear below. Veterans of AWS will recognize the icons in Figure 2 as representing AWS services, but here they are labelled according to the primary function they are delivering to the application. Numbers in grey squares refer to the twelve recommendations.

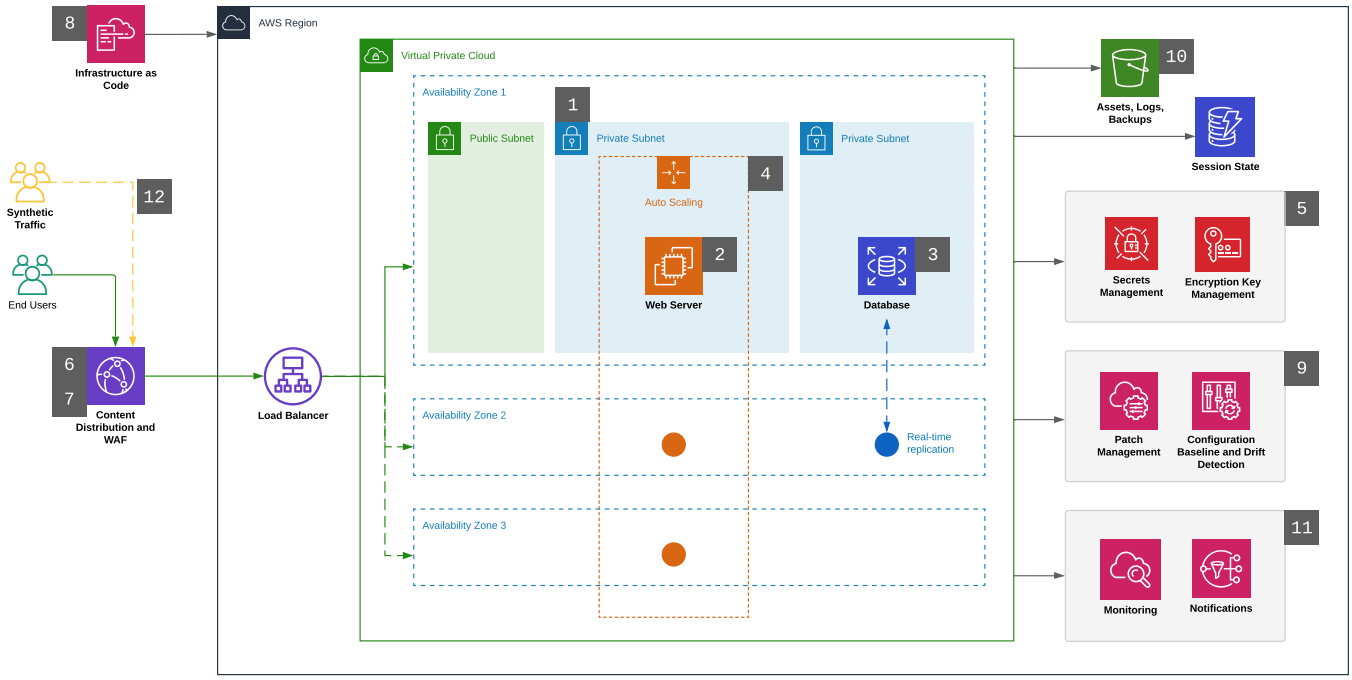

Figure 2: *Replatform Migration. This diagram helps to visualize the application hosting architecture that would result if all twelve recommendations (numbered in the grey squares) were applied. Icons represent specific AWS services, but they are labelled here according to the primary function they are delivering to the application.*

At the heart of Figure 2 are rectangles that represent three Availability Zones. Each Availability Zone is an independent cluster of one or more data centers: see the references for more information. In this example we assume a suitable landing zone has been configured. The landing zone is the foundational AWS account configuration: see the panel Landing Zones for more information.

At first glance Figure 2 may appear overwhelming, but don't despair. You can pick the recommendations that are best suited to your business and your application. As we review the suggestions refer back to Figure 2 to see how the pieces fit together.

Replatform the Core

The most popular way to Replatform the core is to adopt the Amazon Relational Database Service (RDS): an easy win in most migrations. Qalius recommends looking more widely at other ways to improve the application, especially when it comes to the simple changes below that improve security with minimal extra effort. The suggestions for encryption and secrets management may seem exotic at first glance, but the implementation on AWS is relatively straightforward, and can deliver substantial improvement for workloads that manage private data.

1. Network Segmentation

Configure subnets for each application tier and choose private subnets (without IP addressing via an internet gateway) for the application servers and database instances; use security groups (instance firewalls) to open only the specific ports required for the expected traffic from the expected sources

- Eliminates some paths by which application data could be exposed
- Prepares the installation in case future requirements call for additional networking restrictions

2. Operating System Hardening

Consider hardening the operating system and application services, perhaps to the standards of the Center for Internet Security (CIS) Benchmarks for Amazon Linux 2 and Apache

- Reduces the attack surface by turning off features that could be hijacked by an attacker
- May substantially improve application performance without increasing cost

3. Managed Database Services

Consider using the Amazon Relational Database Service (RDS) to deliver databases from fully managed infrastructure

- Delivers simplified management because AWS takes on much of the undifferentiated administration
- Improves disaster preparedness by easily deploying across multiple Availability Zones

4. Autoscaling

Consider updating the application code to store session state off the application server instances using services such as Amazon DynamoDB or Amazon ElastiCache; implement autoscaling to dynamically deploy application servers across the Availability Zones; use an Amazon Application Load Balancer to direct traffic to application servers

- Autoscaling provisions application servers in response to actual traffic levels
- Load balancing improves disaster preparedness and performance by directing traffic to healthy servers in any Availability Zone
- The off-instance session state management allows the application to provide a seamless user experience from any of the application servers

5. Encryption and Secrets

Encrypt the data stored in the data storage services such as Amazon RDS and Amazon S3 using either server-side encryption or encryption within your application using a customer-managed key stored in the AWS Key Management Service (KMS); consider storing secrets such as database credentials in AWS Secrets Manager instead of keeping them on the server instance or in the code

- Encryption is foundational for protecting application data
- Restricting access to database credentials helps secure your data

Replatform the Edge

AWS offers fantastic services that run at the edge to protect the application and accelerate performance. The suggestions below can be applied lightly or aggressively as circumstances warrant. For example, you can begin with simply counting firewall rule violations to gather data before you activate traffic blocking. Similarly, the content distribution network can initially be conservatively configured to cache only completely static content.

6. Web Application Firewall

Consider deploying AWS Web Application Firewall (WAF), perhaps with an AWS-managed or AWS Marketplace ruleset to filter out malicious traffic at the edge

- Delivers protection against common vulnerabilities with managed rulesets from AWS or the AWS Marketplace
- Provides a foundation on which additional rules can rapidly be deployed in response to changes in malicious traffic patterns

7. Content Distribution Network

Consider deploying the Amazon CloudFront content distribution network to accelerate the delivery of static content

- Accelerates application performance by caching static content close to end users
- May reduce cost by lowering the load on the application servers

Don't Forget Observability and Management

Preparing for the way the application will be observed and managed pays huge dividends over the application lifetime. To reduce risk when a business milestone is expected to drive unusual traffic levels, many organizations choose to stress-test the application and infrastructure with realistic synthetic traffic.

8. Infrastructure as Code

Consider specifying the configuration of AWS services using AWS CloudFormation or the Terraform framework from HashiCorp

- Facilitates review by operations, security, and compliance stakeholders by providing a version-controlled record of the AWS service configurations
- Improves accuracy and accelerates future deployments of similar applications

9. Configuration Management

Consider deploying AWS Config to record a timeline of AWS service configuration changes or to control the configuration of critical resources; consider configuring the Patch Manager service within AWS Systems Manager for managing patch state of the application servers

- Facilitates ongoing assessment of how the infrastructure meets compliance requirements
- Improves security through visibility and control of patch state

10. Logging

Consider collecting logs from your application servers and aggregating them off-instance with your favorite log-management platform

- Ensures that logs will be available and secure even if an application server is destroyed and replaced

11. Monitoring

Configure Amazon CloudWatch to monitor the infrastructure, and consider using the CloudWatch Agent to get deeper insights into compute instances

- CloudWatch integrates very well with the AWS services to provide visibility of key service-health metrics
- Pushes notifications to operations teams through the Amazon Simple Notification Service (SNS)

12. Load Testing

Consider simulating production conditions by sending realistic synthetic traffic to the application while monitoring the application and infrastructure response

- Provides insight into likely application performance in future production traffic scenarios
- Support decisions on the trade-offs between performance and cost

What Next

Congratulations! By considering these twelve suggestions you have made architectural decisions in the context of your business requirements. You are well on your way to using the AWS Well-Architected Framework to improve how your application delivers value to your business. Over time you may revisit the list above or seek out other AWS services that are purpose-built to address specific challenges.

Organizations with the highest demands for application scalability and security, and organizations with expectations of very low per-transaction cost models, often turn to AWS cloud-native serverless technologies. Serverless technologies such as AWS Lambda, Amazon DynamoDB, and Amazon API Gateway abstract the underlying infrastructure to deliver secure and scalable functional capabilities to the application. Serverless technologies are usually applied in new application development projects, however sometimes Qalius customers adopt serverless technologies by re-architecting existing applications. In this model the application is migrated to AWS using some of the twelve recommendations above, and then over time the application is refactored to use the AWS serverless platform one microservice at a time.

Landing Zones

A landing zone is a secure AWS environment that has been prepared to receive application migrations. The concept is usually applied to environments with multiple AWS accounts, but the concept is also helpful in simpler cases. Here are some of the most important considerations when planning and configuring a landing zone.

Networking

Is integration required between on premises networks and AWS? What compliance programs may require specific network architecture, monitoring, and management?

Access Management

How will access to the AWS account be monitored and managed? Is federation to another identity provider required?

Cost Management

How will costs be observed and reported to decision makers to ensure sensible cost and performance trade-offs? How will account hygiene be managed to eliminate wasteful spend?

Security Operations

What security operations support the compliance programs, and where will the associated tooling reside?

Log Management and Observability

How are logs monitored and managed? What monitoring tools will provide transparency for operations teams?

This paper is focused on the application migration and it therefore assumes that the landing zone has been implemented. One source to learn more about landing zones is the documentation of AWS Control Tower, an AWS service for creating and managing multi-account environments.

References

Refer to the AWS documentation for additional information

WHITEPAPER:
Understanding Your Application Readiness when Migrating to AWS

October 2020

https://aws.amazon.com/whitepapers

WHITEPAPER:
AWS Well-Architected Framework

July 2020

https://aws.amazon.com/whitepapers/

AWS WEBSITE:
Regions and Availability Zones

Undated

https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

AWS WEBSITE:
AWS Control Tower

Undated

https://aws.amazon.com/controltower/

About Qalius

Specialists in Web Applications for AWS

At Qalius (sounds like "kale-eee-us") we are specialists in web applications that run on Amazon Web Services (AWS). We build applications to your specifications, and we update the applications you already have to take advantage of high-availability and high-performance architectures on AWS.

Custom Applications

Qalius builds web applications to your requirements in conventional LAMP or cloud-native serverless architectures.

Installation in AWS

Qalius uses infrastructure-as-code techniques to provision AWS services for application performance and data security.

Ongoing Support

Qalius delivers 99.99% application availability by configuring AWS platform automations to monitor and maintain the solution.

Tell Us What You Think

Write to us at wellarchitected@qalius.com with your feedback.