



WHITE PAPER

AWS DEVELOPMENT & TESTING FOR CLOUD-BASED APPLICATIONS

KEY CONSIDERATIONS & BEST PRACTICES

apexon.com

INTRODUCTION

DEVELOPMENT & AWS TESTING FOR CLOUD APPLICATIONS

This white paper guides project managers, developers, testers, systems architects, enterprises and anyone involved in software applications development activities.

It covers aspects of the AWS as a development platform, AWS cloud products and solutions, building cloud native applications, advantages of AWS, support, pipelines, automation, testing tools and techniques.



Millions of customers leverage AWS cloud products and solutions to build sophisticated applications with increased flexibility, scalability and reliability.

AWS has established itself as a leading platform for enterprise development teams with a variety of command-line tools and software development kits (SDKs) to deploy and manage applications and services. AWS SDKs are available for any number of platforms and programming languages including Java, PHP, Python, Node.js, Ruby, C++, Android and iOS.

ADVANTAGES & EXPLOITATION

CLOUD NATIVE APPLICATIONS

Cloud native is an approach to building and running applications that fully exploit the advantages of the cloud computing model.

Developed and deployed properly, cloud native applications provide substantial benefits including high-level services, greater elasticity, on-demand delivery, and streamlined global deployment.

The AWS development platform is aligned entirely with the cloud native approach and all its benefits.



Packaged as lightweight containers



Developed with best-of-breed languages and frameworks



Designed as loosely coupled micro-services



Deployed on elastic cloud infrastructures



Managed through an Agile DevOps process



Dynamically orchestrated

“Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds.”

CLOUD NATIVE APPLICATIONS

WHY DEVELOPMENT TEAMS USE AWS TO BUILD

AWS has become the preferred platform for cloud native applications.

Some of its many advantages include:



PROGRAMMING BUILD AND INTEGRATION SUPPORT

AWS provides scalable platform, integration, development and deployment support to web centric programming languages like Ruby, NodeJS, JavaScript, CSS, HTML, Python and more. It also provides platforms like Elastic Bean Stalk to build and run this code.



FULLY CONTROLLED SCALE- UP AND SCALE-DOWN

Cloud applications require a mechanism to scale up and down as web traffic increases or decreases, thereby making it flexible and dynamic in nature. AWS-ECS, Autoscaling, and AWS-EKS provide full-featured functionality to address dynamic scalability in a cloud native stack.



NOTIFICATION, USER AUTHENTICATION AND MESSAGE QUEUING

AWS SNS helps to coordinate the delivery of push notifications for apps, to either subscribing endpoints, or clients. AWS SQS is a message queuing system that helps developers de-couple as well as scale distributed systems, serverless apps, and microservices that are deployed on cloud native stacks. AWS Cognito facilitates the control of user authentication on the cloud native stack.



COST EFFECTIVE STORAGE AND SECURED NETWORK LAYER SUPPORT

Cloud Native Applications require storage and network capability that is mutable and flexible in nature. AWS S3 storage helps to provide online backup as well as archive data and application programs. AWS VPC, Subnets, and Elastic IP help to build flexible network segmentation over cloud space to keep it secure and safe from external non-authorized access.



FAULT-TOLERANT MICROSERVICES

Amazon ECS (Elastic Container Service) is a highly scalable, high-performance container orchestration service that supports Docker containers that enable run and scale containerized cloud applications on the cloud native stack. It is containers without servers making it secure and cost effective. In addition, Amazon EKS runs the Kubernetes management infrastructure across multiple availability zones to eliminate a single point of failure.



CODE MANAGEMENT

AWS also offers CodeDeploy, CodeCommit and CodePipeline to manage code in the cloud native stack. CodeDeploy helps to deploy code at scale, with a focus on rapid development and rapid deployment in mission-critical situations where the cost of failure is high. Code Commit is a managed revision control service that hosts Git repositories and works with all Git-based tools and CodePipeline is used to model and automate software release process on the cloud native stack.

AWS BENEFITS

12-FACTOR APP SUPPORT

1 CODEBASE

Main objective of this factor is to have all cloud application code in revision control – If events are shared (such as a common Amazon API Gateway API), then the Lambda function code can be used for those events to put in the same repository. Otherwise, AWS Lambda break functions can be used along with event sources into their own repositories.

2 DEPENDENCIES

To build special processing or business logic the best solution may be to create a purposeful library using following languages – Node.js, Python: pip, Java: Maven, C#: NuGet and Go: Go get Packages.

3 CONFIG

AWS Lambda and AWS API Gateway can be used to set configuration information using the environment in which each service runs.

4 BACKING SERVICES

AWS Lambda has a default model for this factor. Typically, it is possible to reference any database or data store as an external resource via HTTP endpoint or DNS name.

5 BUILD, RELEASE, RUN

AWS CodeDeploy, CodeCommit and CodePipeline can be utilized for this factor.

6 PORT BINDING

AWS Lambda can be used with either of these three invocation modes – Synchronous, Asynchronous and Stream-based.

7 CONCURRENCY

AWS Lambda can be used which has massive concurrency and scaling capacity.

8 STATELESS PROCESSES

AWS Lambda functions can be used and treated as being stateless, despite the ability to potentially store some information locally between execution environment re-use.

9 DISPOSABILITY

AWS best practices help to identify where to place certain logic, how to re-use execution environments, and how by configuring function for more memory to get a proportional increase in CPU available. With AWS X-Ray, it is possible to gather some insight as to what a function is doing during an execution and to make adjustments accordingly.

10 ENVIRONMENT PARITY

AWS serverless application model allows users to model serverless applications in greatly simplified AWS CloudFormation syntax. With this model, it can use CloudFormation's capabilities – such as Parameters and Mappings to build dynamic templates.

11 LOGS

AWS Cloud watch logs as well as API Gateway provides two different method of logs – 1) Execution logs 2) Access logs.

12 ADMIN PROCESSES

Typically, cloud functions are scoped down to single or limited use cases and have individual functions for different components of the web application. Even if they share a common invoking resource, such as an API Gateway endpoint and stage, it is possible to separate the individual API resources and actions to their own Lambda functions, so it is possible to build design that meets the requirement.

SYSTEMATIC HANDLING OF TEST ENVIRONMENTS

DEFINING & CREATING PIPELINE FOR CLOUD NATIVE

One of the challenges to testing in the cloud lies in finding ways to create a test environment that contains all of the necessary app configurations and test data. This process can be time-consuming for testers and developers alike. The model below presents a systematic way of handling the different test environments in the cloud to overcome these challenges.

Functionally, there should be no change in the test approach between web-based and cloud environment. However, more focus may be required in certain aspects such as performance, security, environment availability, integration with interfaces, customer experience and adherence to SLAs when testing in a **cloud-based environment**.

The test strategy and test plan document may also change depending on the service models and implementation models selected. Choice of tools and techniques should be similar to that of web-based applications, however, during tool selection it is important to make sure that they are available on cloud and that they support the pay-as-you-use model.

LOCAL ENVIRONMENT

- Build
- Locally deploy
- Manual verification
- Static code analysis

PRE-POD ENVIRONMENT

- E2E tests
- Load tests
- Security tests

STAGING ENVIRONMENT

CI Job:

- Build and deploy
- Regression tests
- Manual verification
- Regression, exploratory

INTEGRATION ENVIRONMENT

CI Job:

- Build and deploy
- Integration tests
- Security and performance tests

DEV ENVIRONMENT

CI Job:

- Build and deploy
- Smoke tests
- Static code analysis

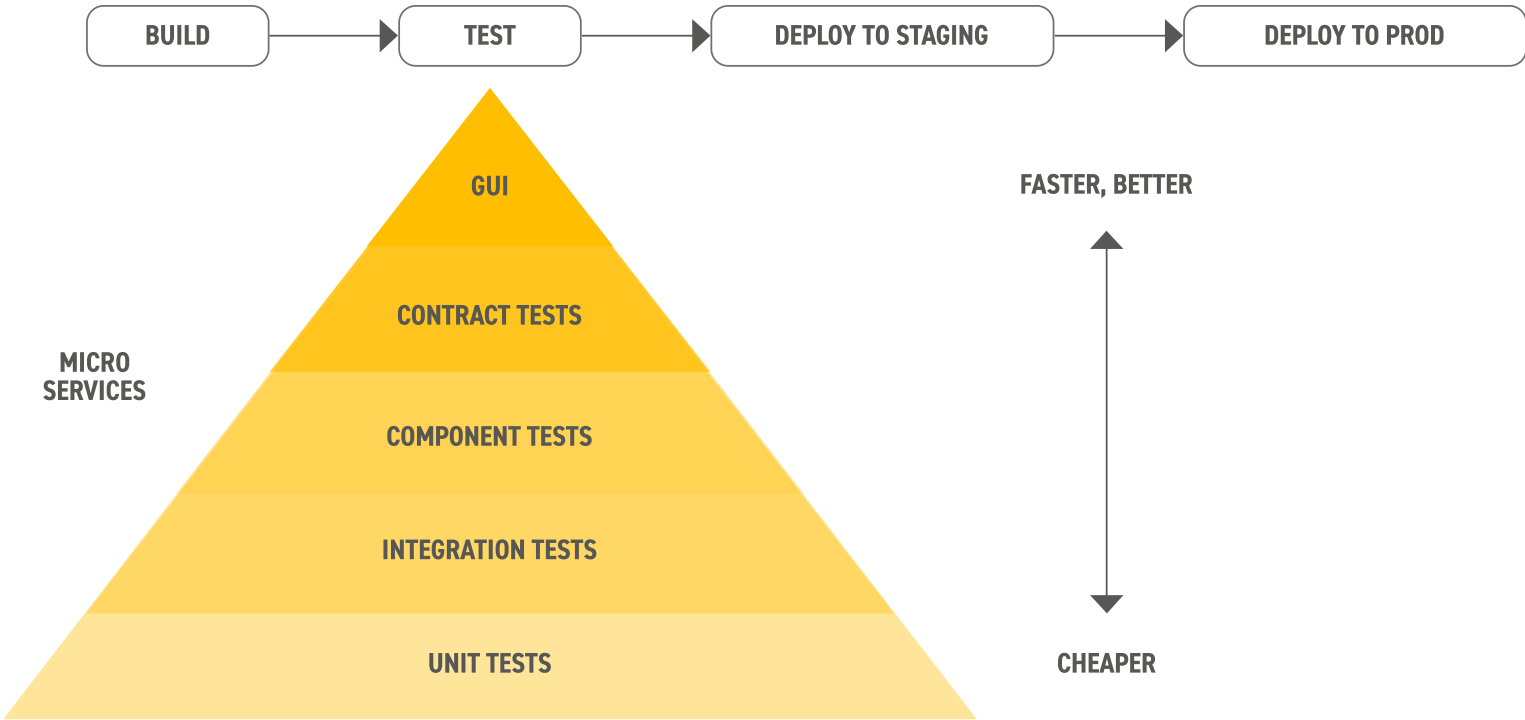
PRODUCTION ENVIRONMENT

- Release tests
- Smoke tests
- Metrics verification

TESTING & VERIFICATION

ROBUST AUTOMATION

Cloud native applications
lend themselves to robust
test automation.



Almost all rounds of testing
can be automated



There should be more focus on verifying
the communication paths and interactions
between components to detect interface
defects for applications on cloud



Even the smallest piece of
testable software is in scope



It is also critical to verify interactions at
the boundary of an external service to
assure that it meets the contract expected
by a consuming relevant cloud service

OUR GUIDE

TESTING TOOLS & TECHNIQUES FOR CLOUD NATIVE

The right tools and thoughtfully designed acceptance criteria are key success factors for cloud native testing. The table below is a good guide based on our experience:

TESTING TYPE	TOOLS	ACCEPTANCE CRITERIA DESIGNED FOR CN APPLICATIONS
Unit/Component Testing	Testing Framework: JUnit, NUnit, RSpec Automation Tools: Pact/QMetry Automation Framework	<ol style="list-style-type: none">1. Boundary condition2. Status code and Status message verification3. Positive and Negative business flows
Integration Testing	Testing Framework: JUnit, NUnit, RSpec Automation Tools: Pact/QMetry Automation Framework	<ol style="list-style-type: none">1. Response chaining2. Status code and Status message verification at the end level3. Session management4. Positive and Negative scenarios with respect to third party5. Consumer scenarios to verify nothing is breaking after service deployment
Contract Testing	Automation Tools: Pact/QMetry Automation Framework	<ol style="list-style-type: none">1. Each consumer needs to verify each contract<ul style="list-style-type: none">• Key and value pair• Value type2. Positive and Negative contact verification

AWS CLOUD-BASED APPLICATIONS

AWS TESTING APPROACHES

Picking the right testing strategy to address specific challenges.

FUNCTIONAL PERFORMANCE TESTING (FPT)

Functional Performance Testing (FPT) means conducting functional testing of the business use cases under the application load conditions. It includes a suite of tests all under server load conditions; e.g.

- ✓ End-user business scenarios
- ✓ Validation of the data breach (data swap)
- ✓ Multi-user/Concurrent testing
- ✓ Validation of the session continuity
- ✓ Validation of auto scaling



PROPER FPT TESTING

The table below compares the differences and impact of a proper FPT approach in most enterprises.

BEFORE (CURRENT ACTIVITIES)

- 1 Very limited functional testing under load (eg. Data Swap)
- 2 Very limited multi user testing (in Appt Center Booking Flow)
- 3 Lack of awareness among team members on edge conditions/corner conditions that resulted
 - a) Potential high risk for bad member experience
 - b) Compromise on member PHI data
- 4 Lack of enforcement as a practice across the squad members

AFTER - SUCCESSFUL FPT ROLLOUT

- 1 Formalized and documented approach for Functional Performance Testing (FPT)
- 2 FPT Suite contains collection of tests including data breach validations
- 3 Functional verification of business use cases while application under load
- 4 Enforcement of FPT via DoD Checklist (Squad level and Release QA checklist)

OUTCOME - BUSINESS BENEFITS Increased test coverage that results in reduced risk level data swap conditions



FPT TESTING TYPES

TYPE OF TESTING	OBJECTIVE	DESCRIPTION
<div>1</div> Data Swap Validation (multi-user, negative concurrency)	Helps in validating the data integrity for the end-user data	<p>Validate the data integrity of the end-users with single user or multi-user.</p> <p>Multi User Testing → Up to 5 TPS for Service Layer via Live Services</p> <p>Concurrent Users (High Traffic) → Up to 50 TPS for Service Layer via Virtualized Services</p>
<div>2</div> Auto-Scaling Policy Validation	Helps in identifying the potential issues related to data loss, session management, session continuity, performance degradation during the auto scaling of the infrastructure	Verify the application behavior is working as expected when infrastructure resources are updated via auto scaling (scale in or scale out) policy

SECURITY TESTING

Security is absolutely key when working in the cloud and there are a number of important factors when deciding how to approach it.

- ✓ The more the security, the more load on the performance, so decisions about what to secure need to be made at the architecture level
- ✓ It is not possible to attain 100% security
- ✓ Never presume anything in regard to security
- ✓ Never trust the dependencies of the application
- ✓ Set traps in code and in repositories (Canary Tokens)
- ✓ Always identify the communication method used between:
 - Users and Resources
 - Between different Resources
- ✓ Always audit the logs and API calls (Cloud Trail and Cloud Watch)



ENCRYPTION IN AWS

The table below compares the differences and impact of a proper FPT approach in most enterprises.



ENCRYPTION IN FLIGHT

Data should be encrypted before sending and after receiving, but only server and sender should know how to do this.

- SSL Certificates help with Encryption (HTTPS)
- It ensures that no MITM (Man-in-the-middle) attack can happen



ENCRYPTION AT REST

Server-side encryption

- Data should be encrypted after received so, even in case the server is hijacked, the hacker could be able to read the data
- Data should be decrypted before being sent back to the client
- Encryption/decryption keys should be managed somewhere (KMS) and the server should have rights to access them



CLIENT-SIDE ENCRYPTION

Data is encrypted by the client and never decrypted by the server.

- Data should be always decrypted by the client only in this case (Envelope Encryption)



TYPES OF SECURITY

There are two types of Security to focus on – User-Based and Resource-Based.



USER-BASED SECURITY

Is managed by IAM, which helps determine what kinds of API calls should be allowed from a specific user according to their authorization level.

- IAM acts as the center of the AWS environment and is integrated with all the AWS services, so it plays the role of central security provider for the whole AWS environment.
- There is typically only one IAM role per application.
- Best practices include never writing IAM credentials in code, setting up Multi-Factor Authentication (MFA) for the root user, and using least privilege principals.



RESOURCE-BASED SECURITY

Depends on every resource, according to its communication method and data usage method.

LOAD & PERFORMANCE TESTING

Load and performance testing is intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload. It typically involves:

- ✓ Assessing production readiness
- ✓ Evaluating against performance criteria
- ✓ Comparing performance characteristics of multiple systems or system configurations
- ✓ Finding the source of performance problems
- ✓ Supporting system tuning
- ✓ Finding throughput levels

Why Load & Performance Test?

There are two types of Security to focus on - User-Based and Resource-Based.



Speed

Does the application respond quickly enough for the number of users?



Stability

Is the application stable under expected and unexpected user load?



Scalability

Will the application handle load beyond expected user demand?



Confidence

Do we believe the application will handle the required amount of load?

TESTING CONSIDERATIONS

FIVE DISTINCT TYPES OF LOAD & PERFORMANCE TESTING

1. AWS LOAD TESTING

Tests the application for its performance on normal and peak usage by checking the response to user requests consistently within accepted tolerance.

Key Considerations

- What is the max load the application is able to hold before the application starts behaving unexpectedly?
- How much data the database is able to handle before system slowness or the crash is observed?
- Any network related issues to be addressed?

4. AWS VOLUME TESTING

Involves huge volumes of data exchanged in database and testing its relation to performance of application. This can be performed in short and long duration.

Key Considerations

- Are the database queries tuned to be performant?
- What is the effect of heavy database on application performance?

2. AWS STRESS TESTING

Involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. This kind of test is started with good load test for which the application has already been tested. The load is increased slowly up to the point where the system fails to respond.

Key Considerations

- What is the max load a system can sustain before it breaks down?
- How the system breaks down?
- Is the system able to recover once it's crashed?

5. AWS ENDURANCE TESTING

Also known as Soak Testing, is typically done to determine if the system can sustain the continuous expected load. During tests, memory utilization is monitored to detect potential leaks. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.

3. AWS SPIKE TESTING

Involves increasing the system load rapidly using bursts of concurrent users to check behavior of the system. The goal is to determine whether performance will suffer, the system will fail, or if it is able to handle dramatic changes in load.

Key Considerations

- Is the application able to handle sudden burst of user's load?
- Working of system fallback mechanism due to unexpected load?

AWS Load and Performance Testing: Parameters

Response Time – the total amount of time it takes to respond to a request for service. That service can be anything from a memory fetch, to a disk IO, to a complex database query, or loading a full web page.

Throughput

Calculated as requests/unit of time. The time is calculated from the start of the first sample to the end of the last sample. This includes any intervals between samples, as it is supposed to represent the load on the server.

The formula is:

$\text{Throughput} = (\text{number of requests}) / (\text{total time}).$

Latency

The measure of responsiveness that represents the time taken to complete execution of a request. It can also represent sum of several latencies or subtasks.

90% Line (Percentile)

The 90th percentile is the value for which 90% of the data points are smaller. This is a standard statistical measure.

Baseline

A set of tests run to capture performance metric data for the purpose of evaluating the effectiveness of subsequent tuning activities performed to the application.

Benchmarking

A process of comparing system performance against the baseline that is created internally or against an industry standard recognized by another organization.

Transaction Response Time

Represents the time taken for the application to complete a defined transaction or business process.

Hits per Seconds

Hits are requests of any kind made from the virtual client to the application being tested (Client to Server). It is measured by number of Hits and the higher the Hits Per Second, the more requests the application is handling per second.

Throughput

The amount of data transferred across the network is called throughput. It considers the amount of data transferred from the server to client only and is measured in bytes/sec.

PERFORMANCE TESTING

METHODOLOGY

1. REQUIREMENTS GATHERING

Collecting key project information including performance testing goals (expected users, response time, etc.), required credentials, application URL, tool preference, etc.

2. TEST PLANNING

Creating complete test plans that incorporate objectives, scope, approach, and focus of the software testing effort. Test planning includes planning load test with the reference of Capacity Planning conducted at the time of Application Development. Capacity Planning helps determine what type of hardware and software configuration is required to meet application needs, and how many resources an application uses. The main goal is to identify the right amount of resources required to meet service demands now and in the future.

3. SCRIPT CREATION

In this step, actual script is created per the business flows using Performance Testing tools such as JMeter, Load Runner, etc. A sample execution with 1-5 users is typically done to verify created script.

4. LOAD TEST EXECUTION

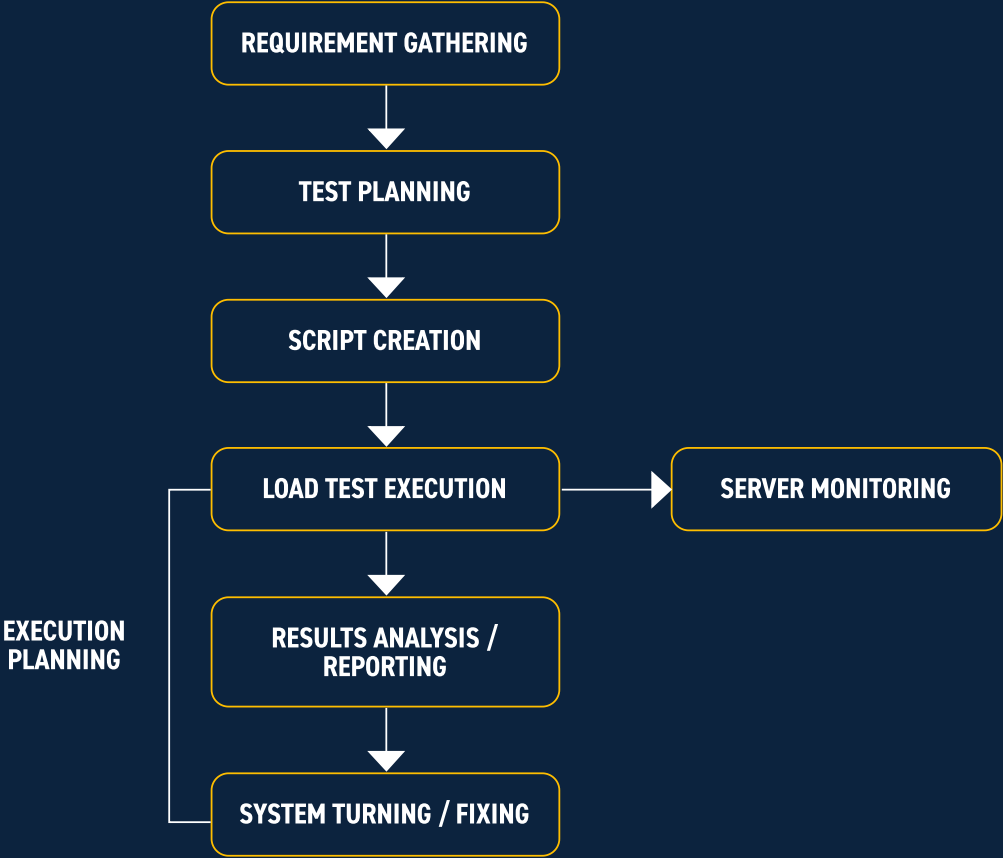
Users execute created script with provided parameters (such as # of user's, ramp-up, ramp-down time, etc.). It requires monitoring servers for memory utilization, CPU utilization, Disk I/O, etc.

5. RESULT ANALYSIS AND REPORTING

In this phase, users take the test results and perform analysis and create a report which identifies performance bottlenecks such as memory leak, server errors, response time, etc.

6. TUNING AND FIXING

The above reports enable users to fix these bottlenecks and continue to tune the application until specified goals are met.



LOAD & PERFORMANCE TESTING

TOOLS USED



OPEN SOURCE

Open STA
Grinder
Rubis
Locust



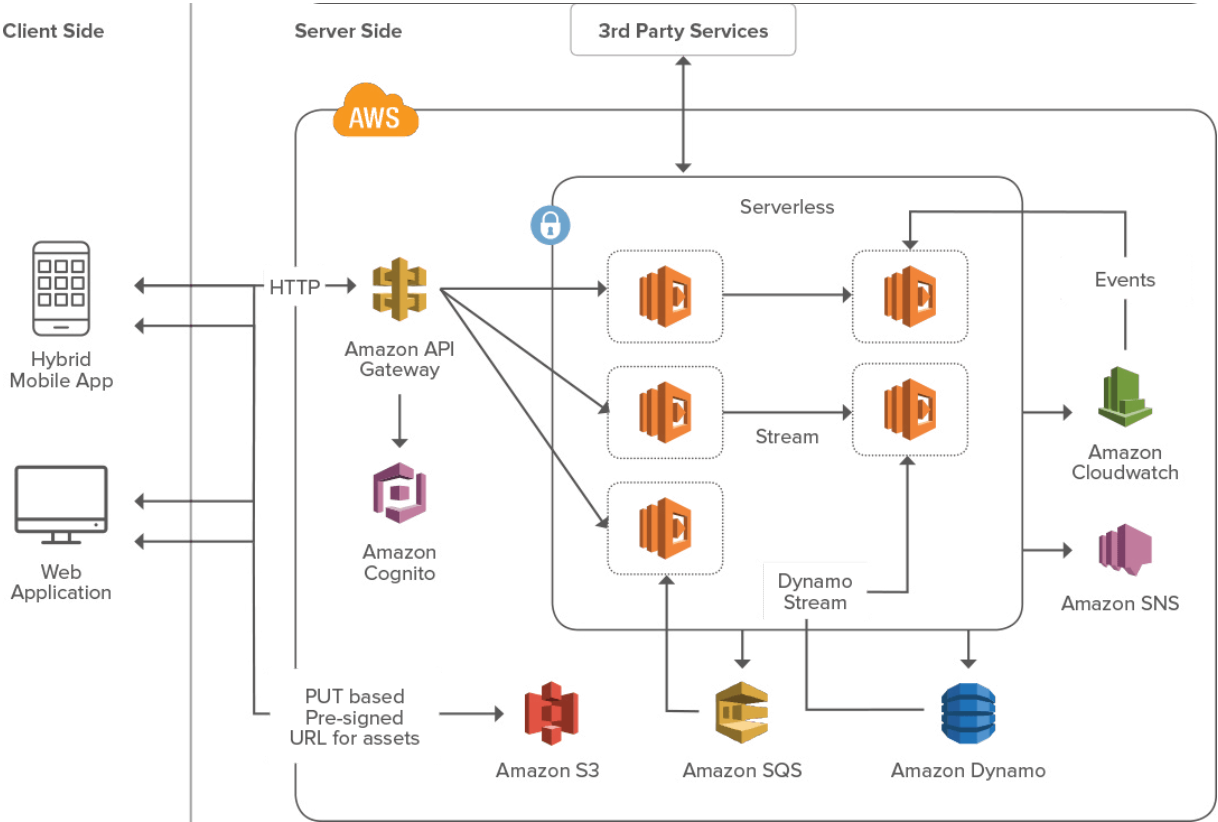
PAID TOOLS

LoadRunner (most popular)
Neo Load
Skill Performer
Blazemeter

CLOUD-BASED APPLICATIONS

TOOLS TO COMPLEMENT & ACCELERATE

AWS Architecture – In General



MIGRATION SEQUENCE	COMPONENTS	TOOLS
Database Server	Tables, users, functions, procedures, triggers etc.	Datapump, Import, Export, SQL developer
Data Transfer	All tables data	AWS DMS, PL/SQL script, Talend etc.
Application Server	Forms fmb, fmx, configuration and setup files	SCP, Oracle 11g forms converter and engine
WebLogic	Web API	Oracle fusion middleware
Roles	Application users, roles, DB roles	AWS DMS or third party tool
Security	Groups, users, roles, privileges	AWS console
Backup	Application and database	Amazon S3 buckets and RMAN

DISASTER RECOVERY PLANNING IN AWS



NATURAL TYPES OF DISASTERS

- Agricultural diseases and pests
- Damaging winds
- Drought and water shortage
- Earthquakes
- Emergency diseases (pandemic influenza)
- Extreme heat
- Floods
- Hurricanes and tropical storms
- Landslides and debris flow
- Thunderstorms



MAN-MADE & TECHNOLOGICAL TYPES OF DISASTERS

- Power service disruption and blackout
- Nuclear power plant and nuclear blast
- Radiological emergencies
- Chemical threat and biological weapons
- Cyber attacks
- AWS console
- Civil unrest

PREVENTIVE MEASURES

Aimed at preventing an event from occurring

CORRECTIVE MEASURES

Aimed at fixing a system in case of a negative event or disaster

DETECTIVE MEASURES

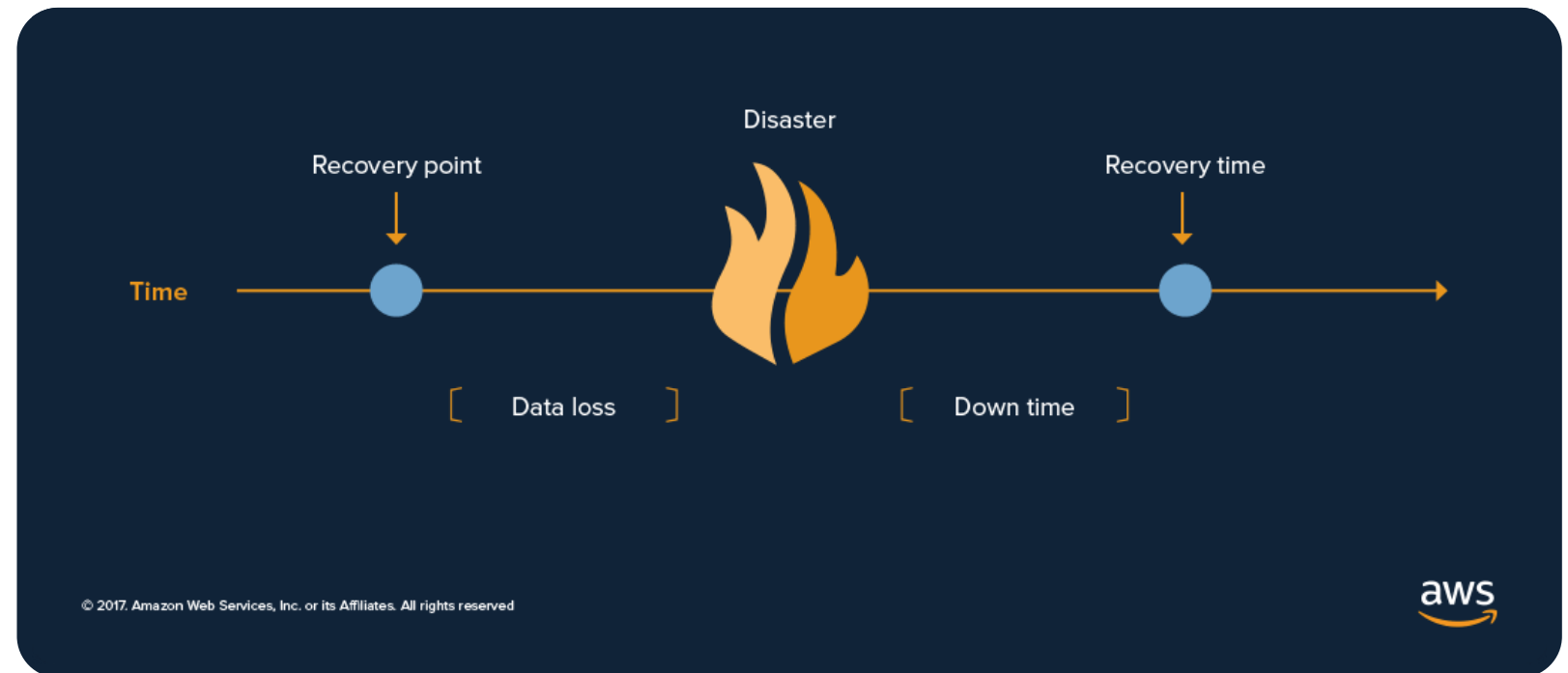
Aimed at detecting and discovering negative events

KEY FACTORS

RECOVERY TIME OBJECTIVE (RTO) & RECOVERY POINT OBJECTIVE (RPO)

The right DR plan for an organization involves many factors, but few are as important as RTO and RPO.

The terms are similar, but fundamentally different. RTO is the maximum length of time after an outage that your company is willing to wait for the recovery process to finish. RPO is the maximum amount of data loss your company is willing to accept as measured in time.



THERE ARE FOUR PRIMARY APPROACHES FOR DR ON AWS:

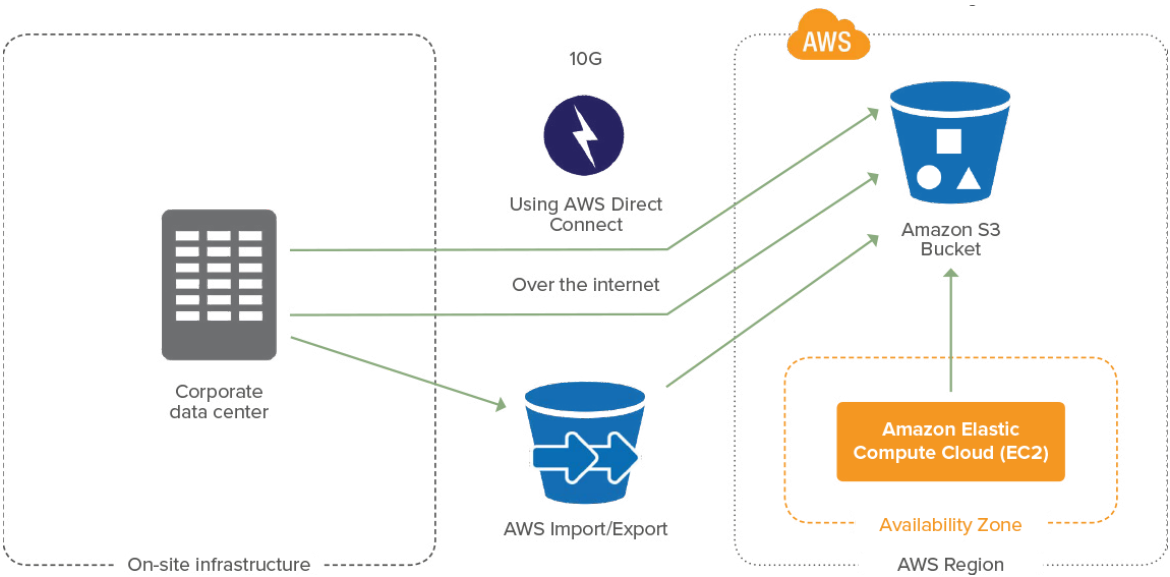
1

BACKUP AND RECOVERY

- ✓ In most traditional environments, data is backed up to tape and sent off-site regularly
- ✓ It can take a long time to restore your system in the event of a disruption or disaster
- ✓ Amazon S3 is an ideal destination for backup data that might be needed quickly to perform a restore
- ✓ Transferring data to and from Amazon S3 is typically done through the network, and accessible from any location
- ✓ You can use AWS Import/Export to transfer very large data sets by shipping storage devices directly to AWS
- ✓ For longer-term data storage where retrieval times of several hours are adequate, there is Amazon Glacier, which has the same durability model as Amazon S3
- ✓ Amazon Glacier and Amazon S3 can be used in conjunction to produce a tiered backup solution

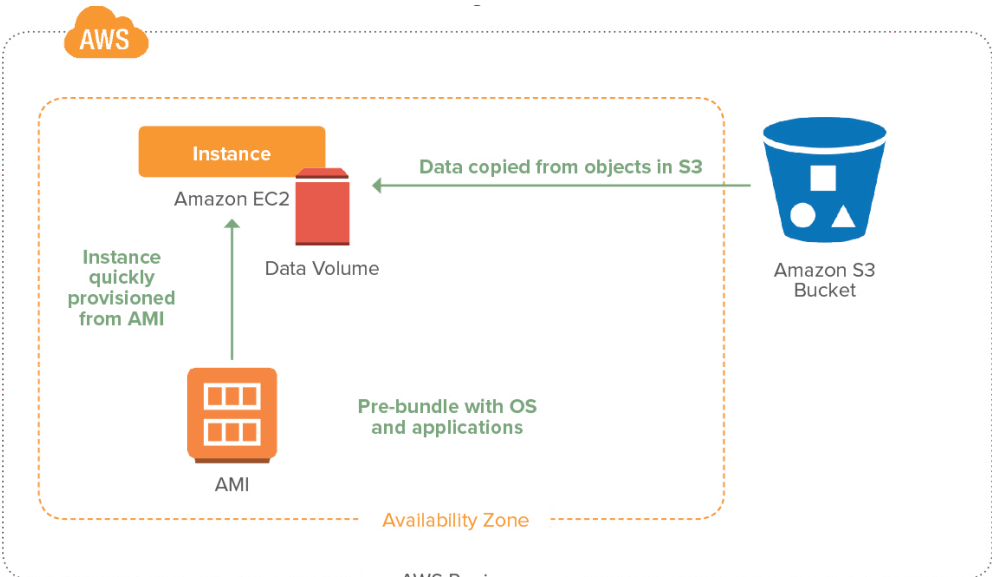
PREPARATION PHASE

The figure below shows data backup options to Amazon S3, from either on-site infrastructure or from AWS.



RECOVERY PHASE

The following diagram shows how you can quickly restore a system from Amazon S3 backups to Amazon EC2



Key steps for backup and recovery:

- ✓ Selecting an appropriate tool or method to back up data into AWS
- ✓ Ensuring an appropriate retention policy for the data
- ✓ Ensuring appropriate security measures are in place for the data, including encryption and access policies
- ✓ Regularly testing the recovery of data and system restoration

2

PILOT LIGHT

The term describes a DR scenario in which a minimal version of an environment is always running in the cloud:

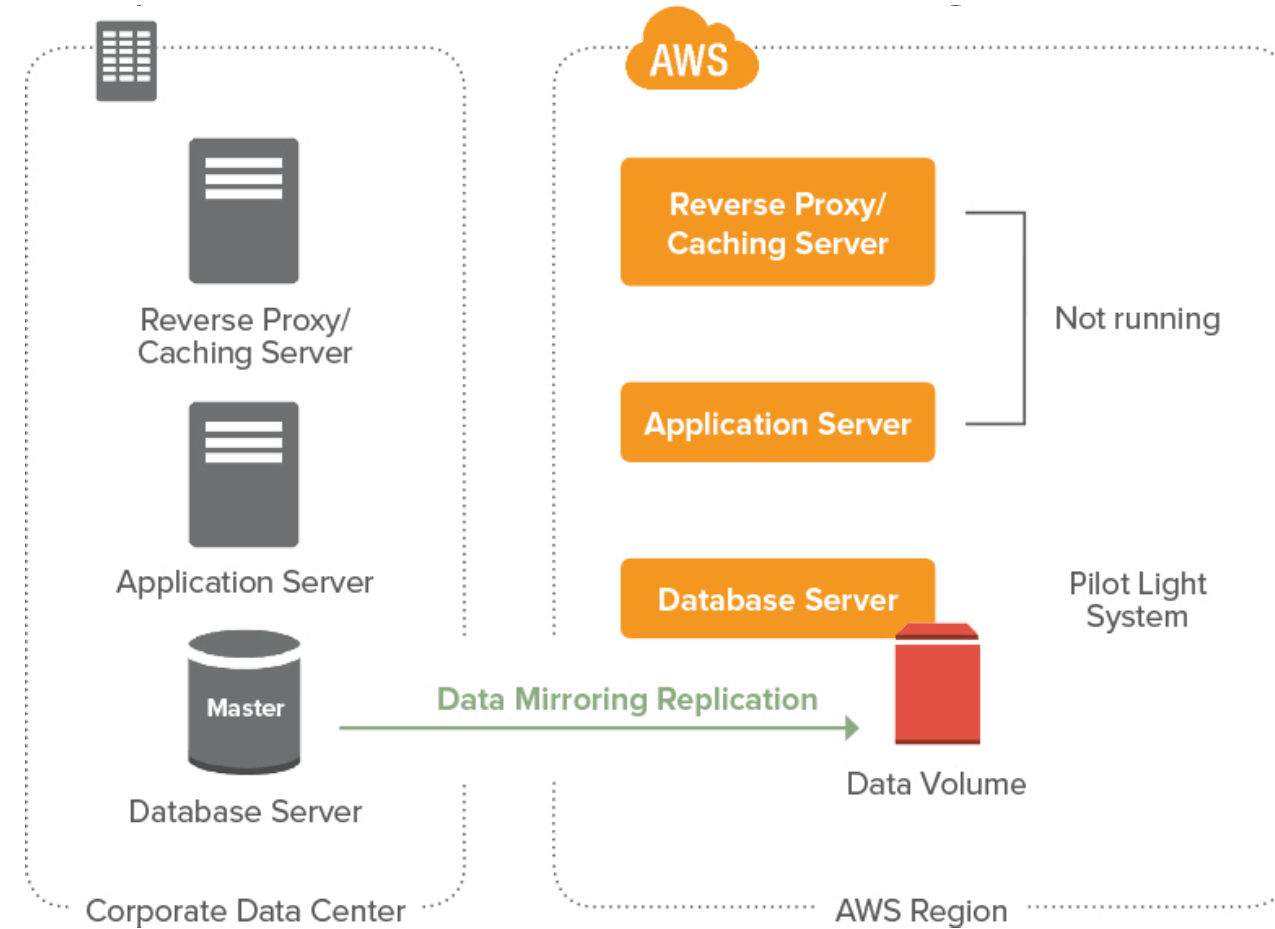
- ✓ The on-premise database server mirrors data-to-data volumes on AWS
- ✓ The database server on cloud is always activated for frequent or continuous incremental backup
- ✓ The application and caching server replica environments are created on cloud and kept in standby mode — very few changes take place over time
- ✓ AMIs can be updated periodically
- ✓ If the on-premise system fails, the application and caching servers get activated
- ✓ Users are re-routed using elastic IP addresses to the ad hoc environment on cloud
- ✓ Recovery should take just a few minutes

Key steps include:

- ✓ Setting up Amazon EC2 instances to replicate or mirror data
- ✓ Ensuring access to all supporting custom software packages available in AWS
- ✓ Creating and maintaining AMIs of key servers where fast recovery is required
- ✓ Regularly running and testing servers, and applying software updates and configuration changes
- ✓ Automating the provisioning of AWS resources where possible

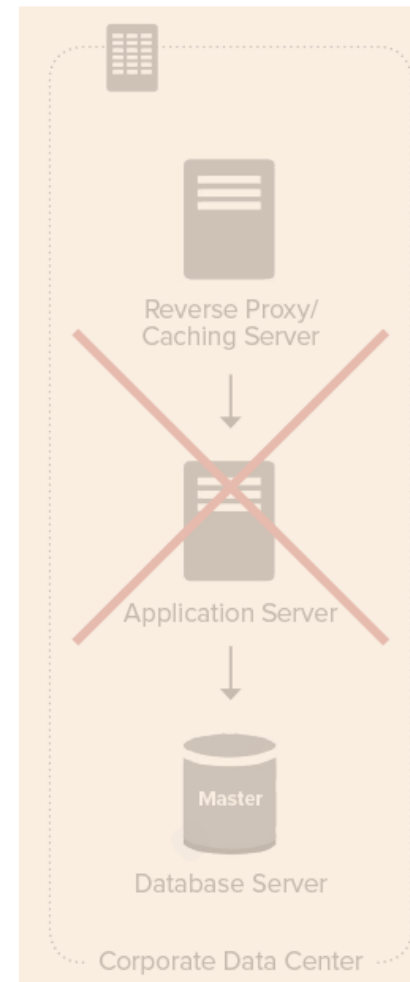
PREPARATION PHASE

The figure below shows the preparation phase.

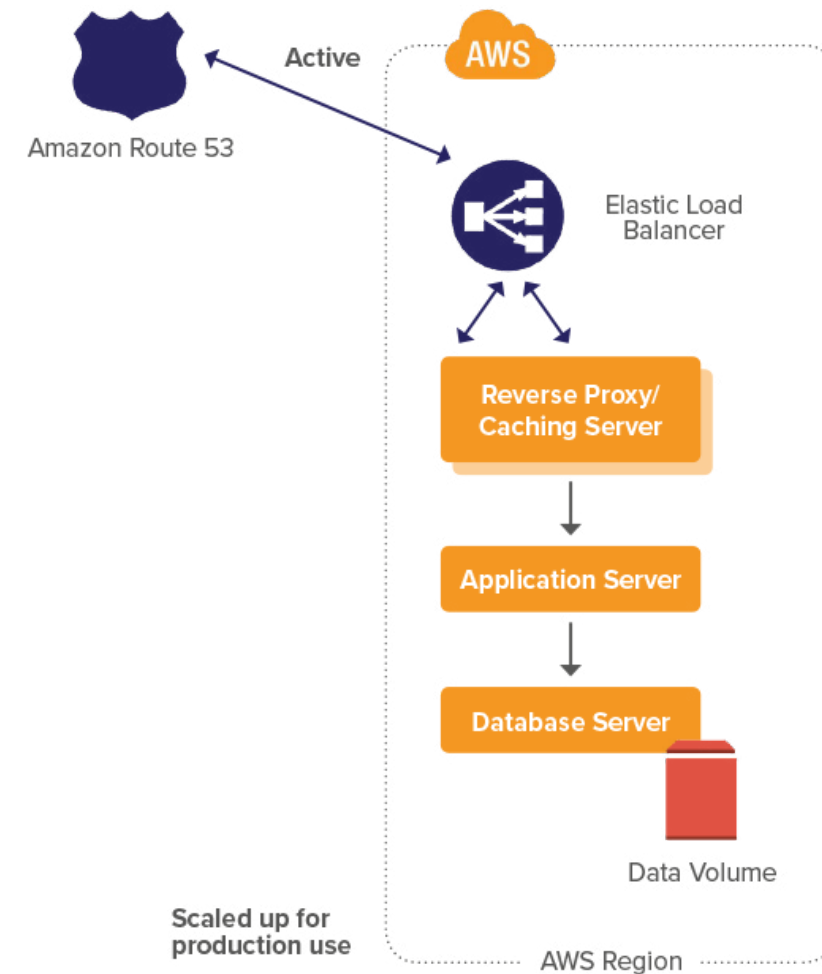


Key steps for the Pilot Light Recovery Phase include:

- ✓ Starting the application Amazon EC2 instances from custom AMIs
- ✓ Resizing existing database/data store instances to process the increased traffic
- ✓ Adding additional database/data store instances to give the DR site resilience in the data tier
- ✓ Changing DNS to point at the Amazon EC2 servers
- ✓ Installing and configuring any non-AMI based systems, ideally in an automated way



RECOVERY PHASE



3

WARM STAND-BY

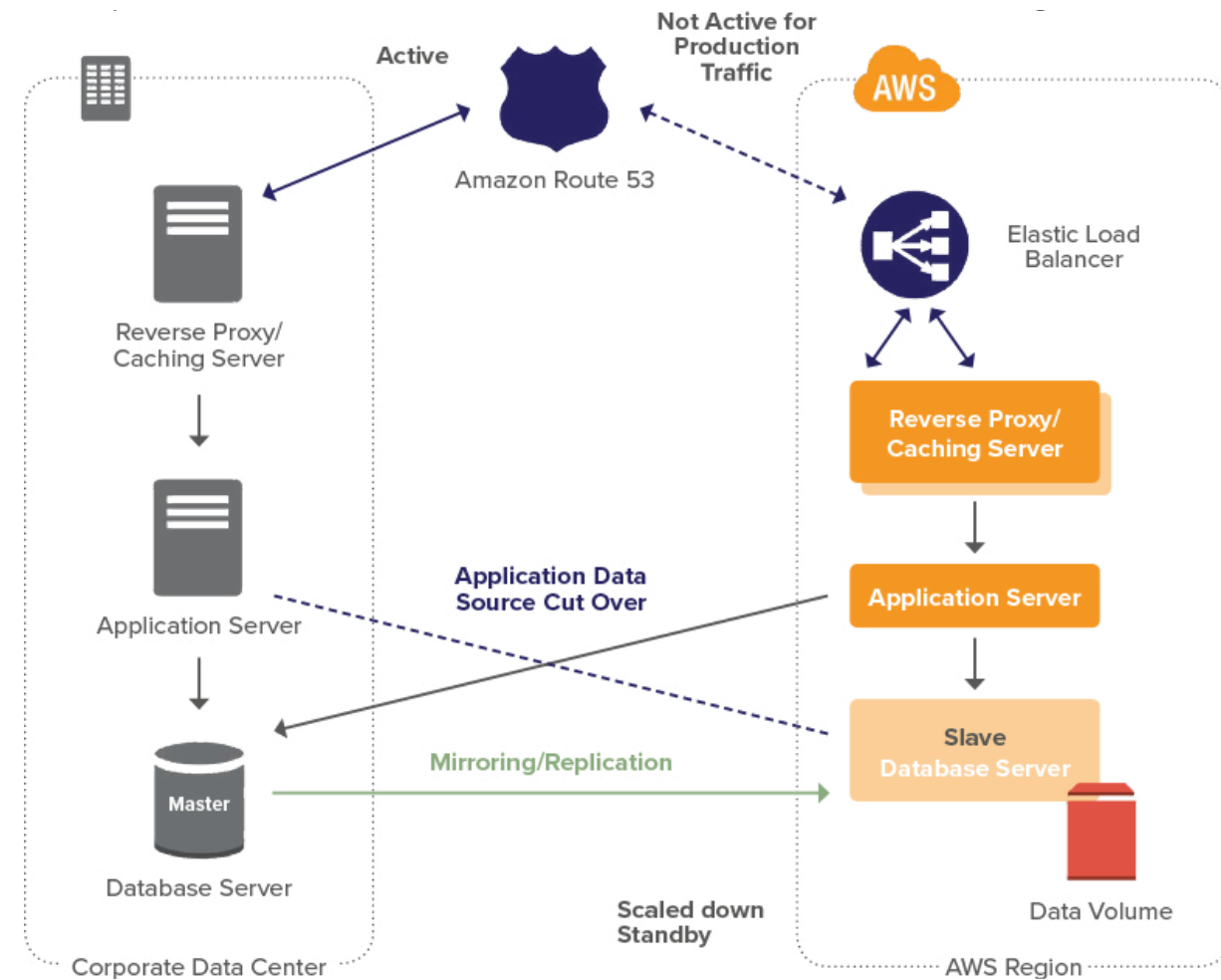
This approach is a scaled-down version of a fully functional environment and is always running in the cloud. It extends the Pilot Light elements and preparation and further shortens the recovery time. By identifying business-critical systems, users can fully duplicate them on AWS and have them always on – running on a minimum-sized fleet of Amazon EC2 instances on the smallest sizes possible. In a disaster, the system can scale up quickly to handle the production load.

Key steps in the Preparation Phase include:

- ✓ Setting up Amazon EC2 instances to replicate or mirror data
- ✓ Creating and maintaining AMIs
- ✓ Running an application using a minimal footprint of Amazon EC2 instances or AWS infrastructure
- ✓ Patching and updating software and configuration files in line with the live environment

PREPARATION PHASE

The following figure shows the preparation phase for a warm standby solution, in which an on-site solution and an AWS solution run side-by-side.

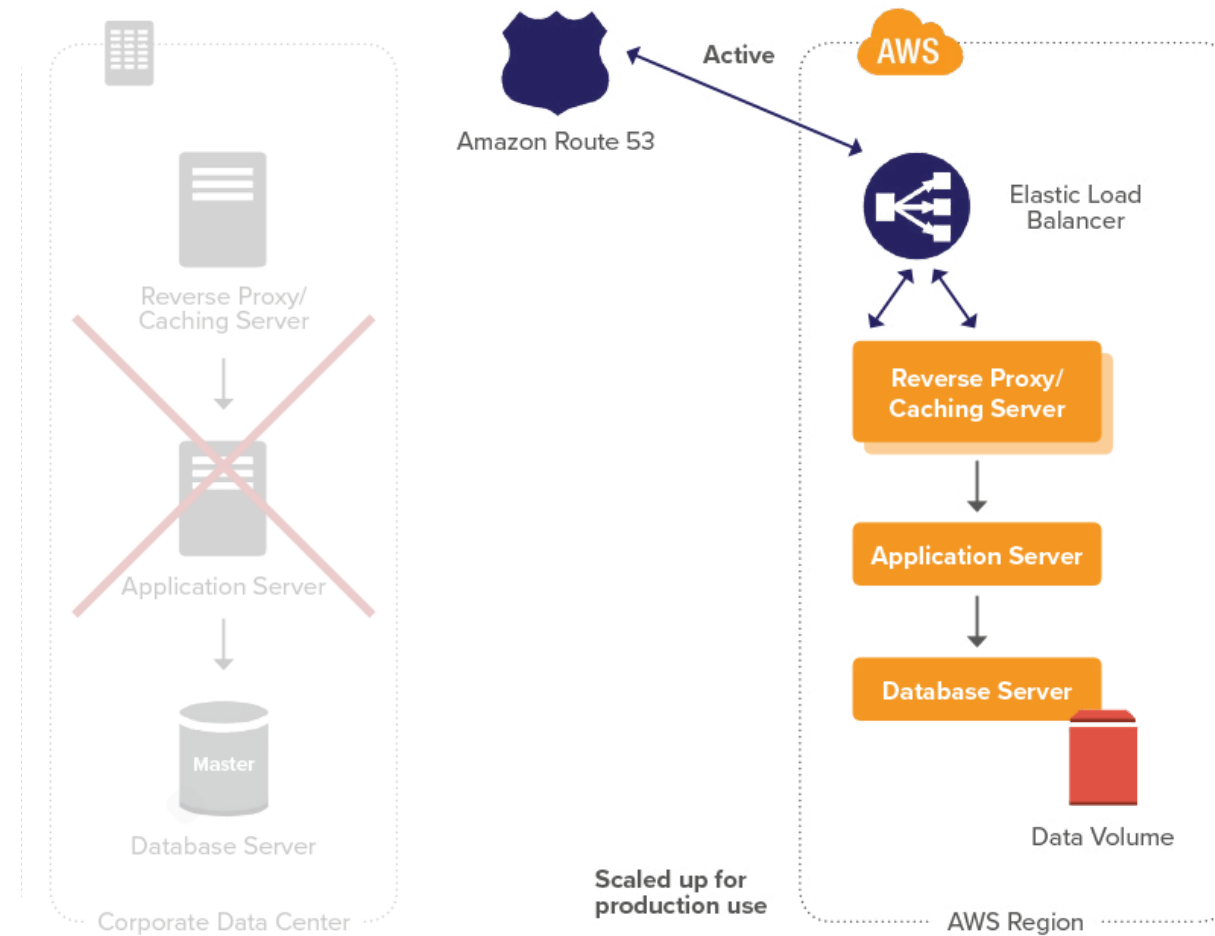


Key steps include:

- ✓ Increasing the size of Amazon EC2 fleets in service with the load balancer
- ✓ Starting applications on larger Amazon EC2 instance types as needed
- ✓ Manually changing the DNS records, or using Amazon Route 53 automated health checks so that all traffic is routed to the AWS environment
- ✓ Using auto scaling to right-size the fleet or accommodate the increased load
- ✓ Adding resilience or scale up to the database

RECOVERY PHASE

In the case of failure of the production system, the standby environment can scale up quickly for production load, and DNS records are changed to route all traffic to AWS.



4

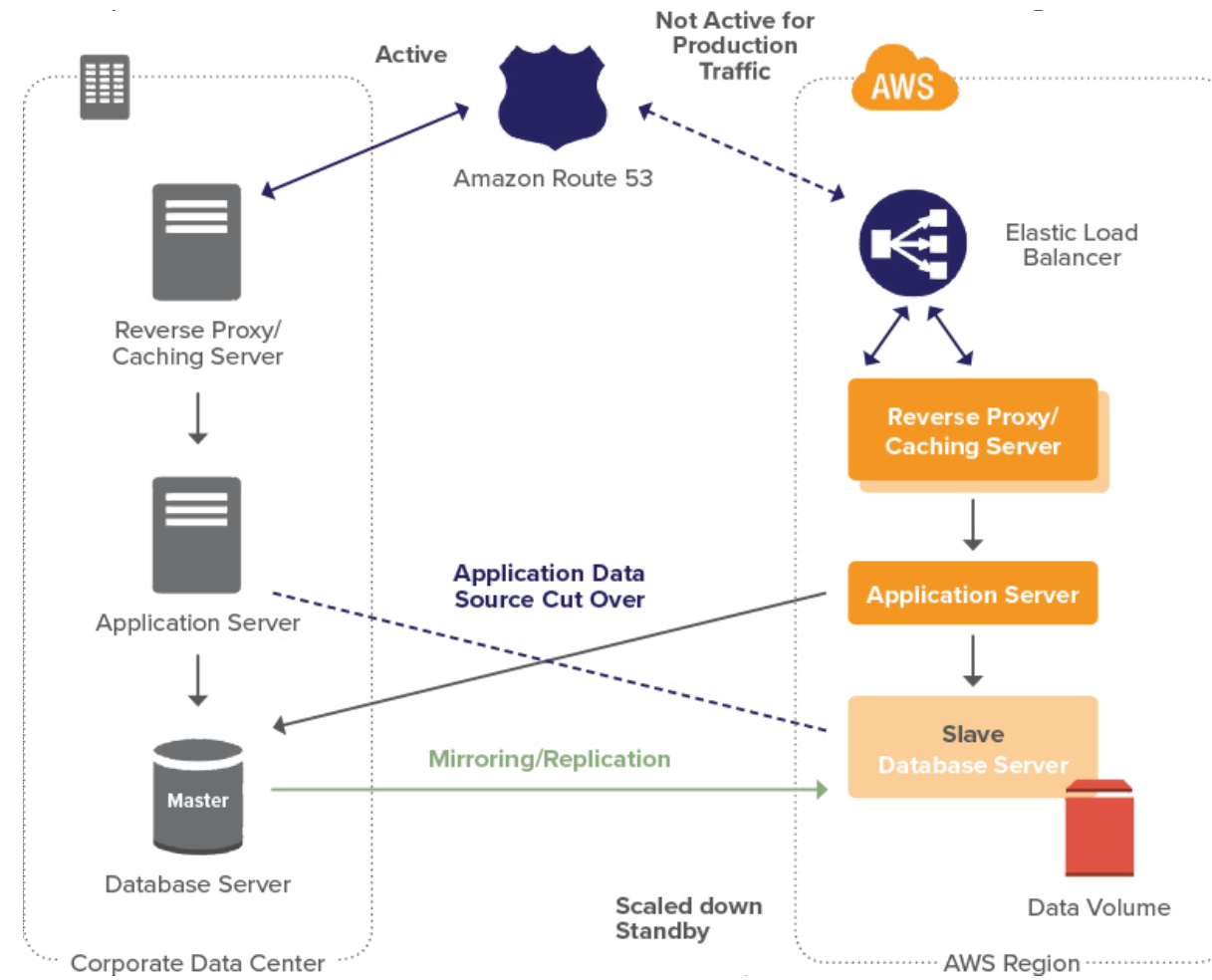
MULTI-SITE

Multi-Site is the optimal backup and DR approach. All activities in the preparatory stage are similar to a warm standby; except that AWS backup on Cloud is also used to handle some portions of the user traffic using Route 53. When a disaster strikes, the rest of the traffic that was pointing to the on premise servers is re-routed to AWS. Using auto scaling techniques, multiple EC2 instances are deployed to handle full production capacity.

In a Multi-Site Approach, preparation includes:

- ✓ Setting up the AWS environment to duplicate the production environment
- ✓ Setting up traffic routing technology to distribute incoming requests to both sites
- ✓ Configuring automated failover to re-route traffic away from the affected site

PREPARATION PHASE

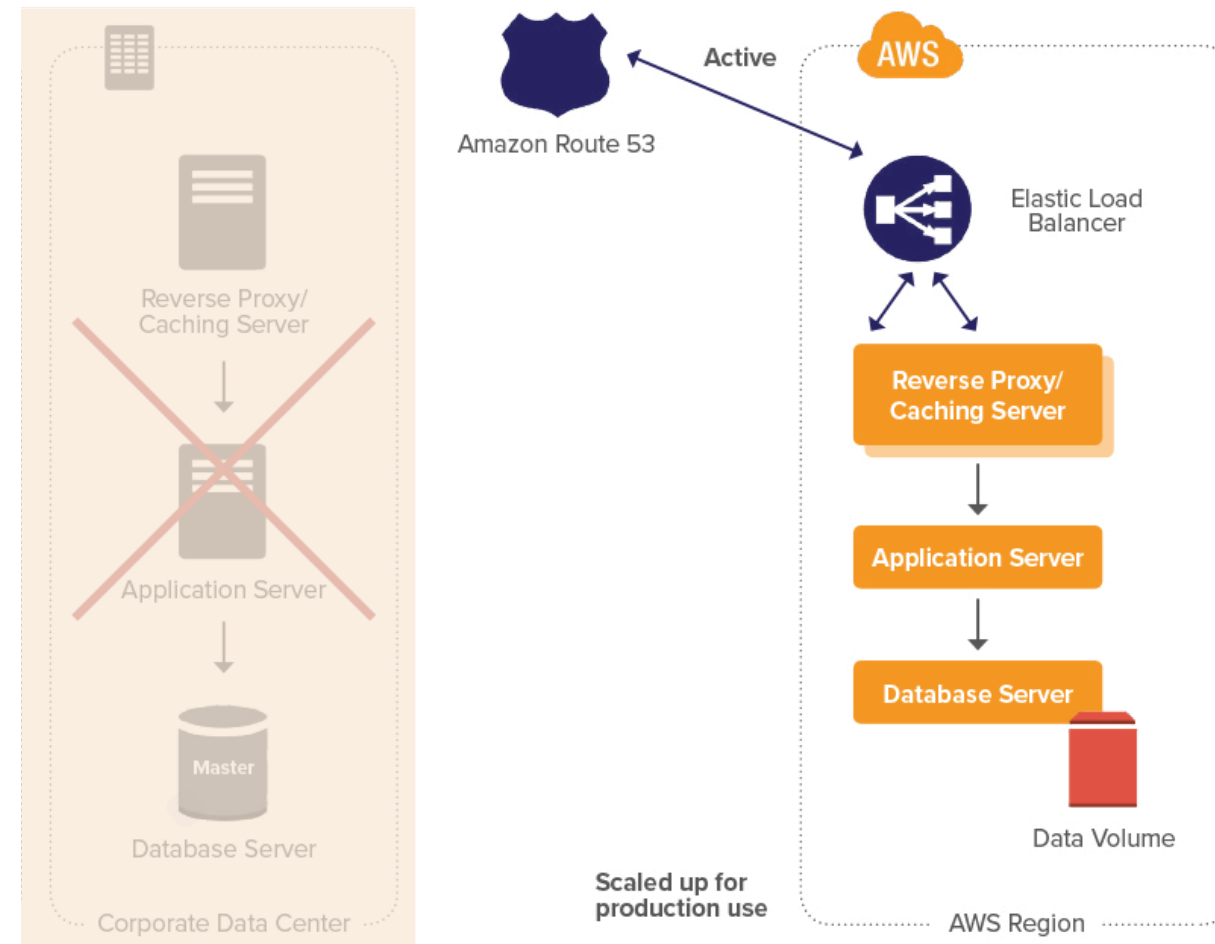


Key steps for recovery in a Multi-Site DR approach include:

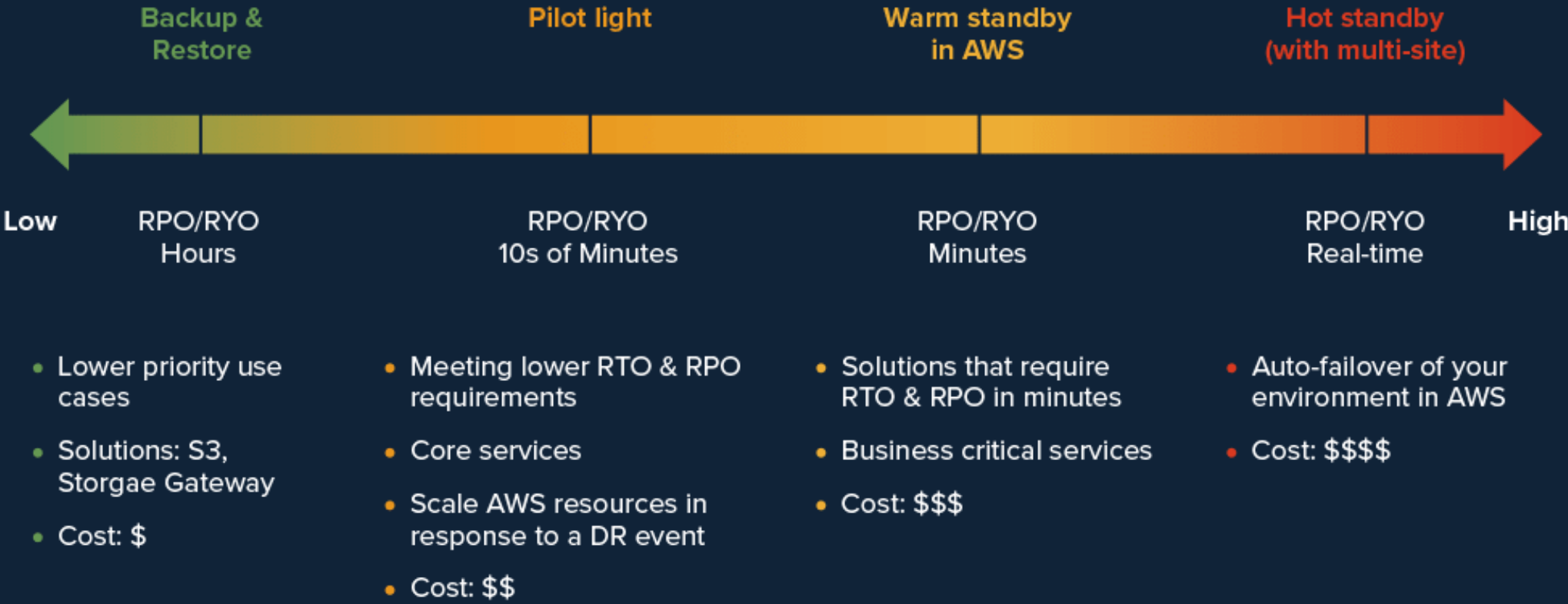
- ✓ Either manually, or by using DNS failover, changing DNS weighting so that all requests are sent to the AWS site
- ✓ Leveraging application logic for failover to use local AWS database servers for all queries
- ✓ Using auto scaling to automatically right-size the AWS fleet if possible

RECOVERY PHASE

The following figure shows the change in traffic routing in the event of an on-site disaster.



AWS offer four levels of DR support across a spectrum of complexity and time



EXECUTION & APPROACH

FAILING BACK

Once you have restored your primary site to a working state, you will need to restore your normal service, which is often referred to as a “failback.”

There are different ways to execute it based on which DR approach has been deployed.

FOR BACKUP & RESTORE

Freeze data changes to the DR site

Take a backup

Restore the backup to the primary site

Re-point users to the primary site

Unfreeze the changes

FOR PILOT LIGHT, WARM STANDBY, AND MULTI-SITE

Establish reverse mirroring/replication from the DR site back to the primary site, once the primary site has caught up with the changes.

Freeze data changes to the DR site.

Re-point users to the primary site.

Unfreeze the changes.

Data Replication

When replicating data to a remote location users should consider several factors:



Distance between the sites – longer distances are typically subject to more latency or jitter



Available bandwidth – the breadth and variability of the interconnections



Data rate required by the application – should be lower than the available bandwidth

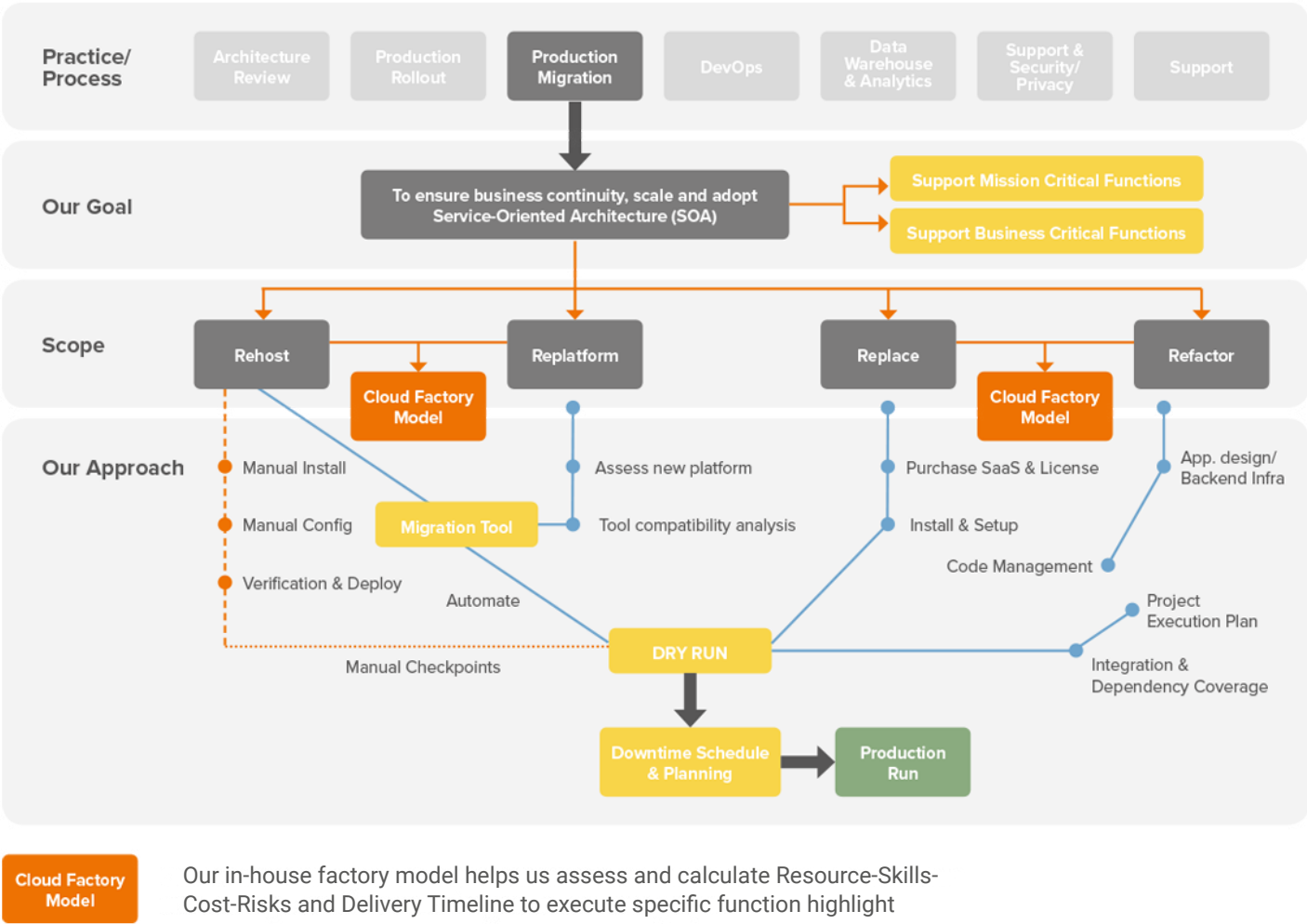


Replication technology – should be parallel (so that it can use the network effectively)

EXECUTION & APPROACH

DR/BUSINESS CONTINUITY PLANNING

Business Continuity Planning also is a key part of DR. The graphic below outlines how effective Business Continuity approach works.



Apexon is an advanced technology and consulting partner of AWS with extensive experience on the AWS platform in DevOpsSecTest and Agile environments.

Our team is fully equipped with the knowledge and skills to help organization take full advantage of the extensive capabilities of the AWS platform to accelerate digital initiatives.

Working together, Apexon and AWS are helping enterprises leverage the cloud to develop, test and deploy new digital products and services more quickly, flexibly, and efficiently while assuring the highest quality and service levels. We help delivery organizations unearth the full potential of cloud computing by enabling them to capitalize on the agility and security that DevOpsSecTest brings to accelerate their digital transformation initiatives.

Our close collaboration with our AWS partner team brings value to clients' cloud migration journey at every phase of the project covering design, build, test and delivery. We utilize time-tested frameworks, standards and procedures that are instrumental to realizing cloud migration objectives. Our experience and expertise covers a broad cross-section of industry verticals such as wearables, digital health, and IoT for the gaming industry to name a few and wide variety of technical domains such as DevOpsSecTest, IoT, and micro-services.


Apexon AWS Capabilities & Services


- ✓ Infrastructure Migration
- ✓ Platform and Application Migration
- ✓ Database Migration
- ✓ Micro-Services and Serverless Architectures
- ✓ Single Sign-On and Security
- ✓ DevOpsSecTest


Our AWS cloud hosting and software services are primarily focused on architecting and implementing in four key areas: Re-hosting, Re-Platforming, Replacing, and Re-factoring. We help customers plan their migration path to the AWS Cloud based on actual system conditions and criticality to operations by analyzing various aspects of their environment including security, networking, reliability and failover, services and support requirements, billing, licensing, transfer cost and 3rd- party tool support.


Our cloud migration plan is focused on identifying the mission and business-critical functions of the customer and relaying that to a delivery roadmap that can produce time-bound expected ROI.

The key pillars of our delivery roadmap include:

- 

Discovery identifying key gaps, challenges and inefficiencies.
- 

Cloud Factory Model gaining upfront visibility on resource, skill, effort and, timeline and risks for the migration.
- 

Alignment highlighting the value by quantifying business and IT value in \$ terms.
- 

On-Time Delivery cultivating a culture of “everything-is-delivered-as-a-service.”

AWS Certifications



Our project plan for AWS cloud hosting and related software services are classified into three areas: Gap Analysis; Migrate & Adopt; and Sustain & Scale. The planning process helps customers build collaboration between product-as-a-service, data, and cloud services to take full advantage of the scalability, cost and agility advantages of the AWS platform.

Apexon helps customers classify their applications on a spectrum of complexity and criticality based on direct impact to business-critical functions. We also develop migration plans that cater to each customer’s specific business needs. In addition, Apexon's cloud factory model and readily available accelerators help customers gain a holistic picture of “what it really takes” in terms of resources, skills, effort, timelines and risks as they progress on their AWS cloud journey.



APEXON IS A PURE-PLAY DIGITAL ENGINEERING SERVICES FIRM FOCUSED ON HELPING COMPANIES ACCELERATE THEIR DIGITAL INITIATIVES FROM STRATEGY AND PLANNING THROUGH EXECUTION.

We leverage deep technical expertise, Agile methodologies and data-driven intelligence to modernize systems of engagement and simplify human/tech interaction.

We deliver custom solutions that meet customers' technology needs wherever they are in their digital lifecycle. Backed by Goldman Sachs and Everstone Capital, Apexon works with both large enterprises and emerging innovators — putting digital to work to enable new products and business models, engage with customers in new ways, and create sustainable competitive differentiation.



info@apexon.com



www.apexon.com

FEELING SOCIAL?

