Customer Sample– Cloud Runbook

# Purpose

Abstract

Execution-level document specifying customer approach to deploying applications on Amazon Web Services. This is a living document detailing Customer current approach to architecture, provisioning, and operations on AWS.

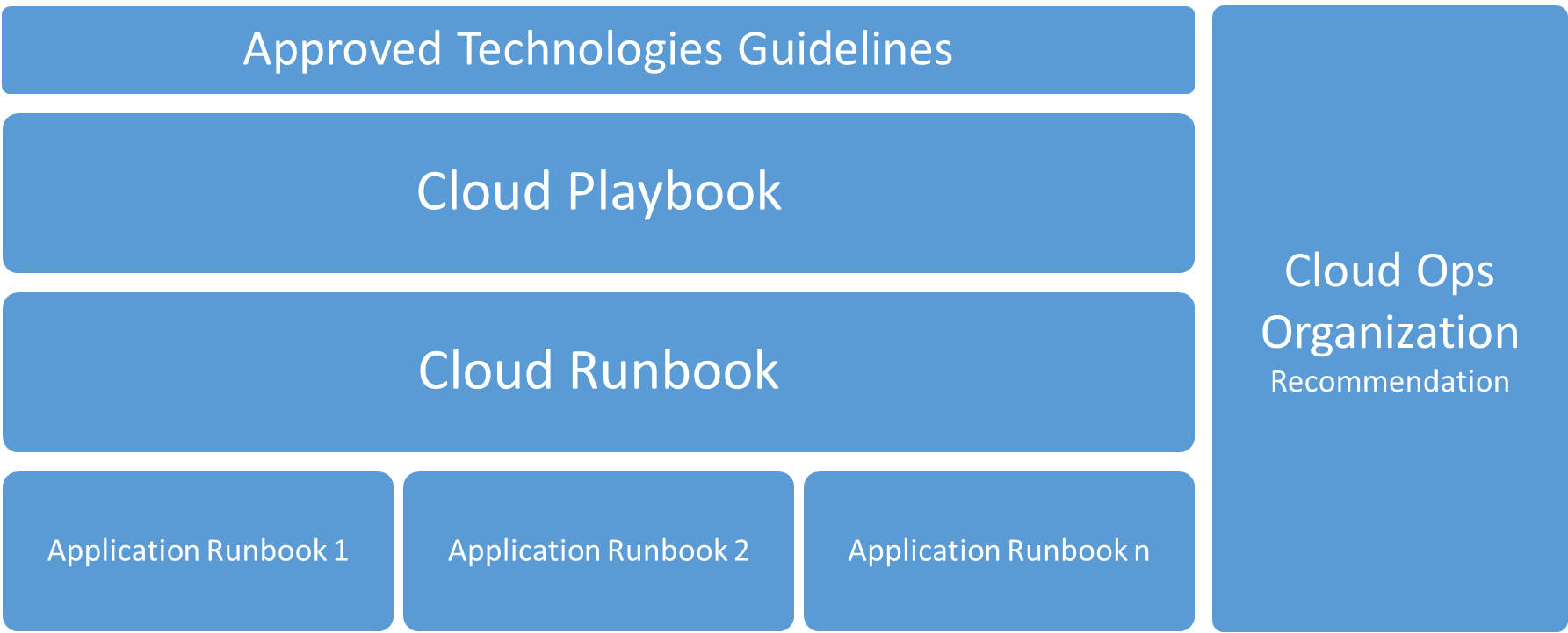
Delivered by:

Professional Services

This document should be used as a reference point for future choices and efforts in the operation of a cloud environment. It discusses many facets of the “how” and “why” of every day operations in the cloud and suggests steps that can be taken in the future. In many cases there are coded examples alongside of that suggestion, in some cases they will need to be created. While not specifically strategic, the intent of this document is to be used with a play book and it speaks to planned next steps in that document.

# Related Documents

The Cloud Runbook is part of a series of interrelated documents that address the topics of cloud adoption; including, technology selections, strategy formation, cloud execution, application-specific runbooks and even dealing with organizational considerations. The level of detail included in these documents is illustrated below, starting with the highest level strategy information at top, pushing down through the most specific and detailed at the bottom.



* Recommended Technologies Guidelines represent a framework to guide the organization in selection of technologies to insure an efficient use of resources, improve time to market and the ability to maintain service level operations.
* Cloud Playbook is a strategy-level document specifying CUSTOMER’s approach to deploying applications on and in a modern cloud architecture. Documents CUSTOMER’s current approach to architecture, provisioning, and operations in the cloud; and also provides the next steps to increase cloud adoption maturity.
* Cloud Runbook is an execution level document that provides the technical details of operating application and infrastructure in a cloud environment.
* Application Runbooks provide a view into application specific exceptions from the standard cloud guidelines as outlined in the Cloud Runbook.
* Cloud Team Organization outlines the resources, skills and teams that are involved in driving the cloud adoption, a Cloud Center of Excellence (COE) as well as those who manage the day to day provisioning, usage, monitoring, and cost of cloud resources (Cloud Engineering).

# Operations Overview

As organizations adopt cloud services, the organization must transform in order to fully realize the benefits of cloud technologies. Operationalizing your cloud service offering includes adapting the services offered, the manner in which they are offered as well as how they are offered. This document is to be a living document that outlines the IT service management features like proactive monitoring, patch, business continuity, and financial management, and processes like change and incident management. There are options identified that can be implemented dependent upon the maturity of your organization.

To be optimal, service management and operational functions must be proactive and supported by automation, as opposed to reactive and supported by manual human intervention. This applies both to deploying resources and automating responses to potential issues by designing for failure. Repetitive manual tasks should be reduced through the use of automation to allow operations teams to focus on value-add work. Operational areas that are important to address early in the cloud journey include Financial Management, Platform Operations, Monitoring & Incident Resolution, Provisioning & Service Catalog, Availability Management and Application Lifecycle Management. Similar to the iterative approach of agile development, there will be an iterative approach to cloud operations.

The specific categories are as follows:

|  |  |
| --- | --- |
| Operations Category | Sub-categories |
| Platform Operations | Tagging  Application Service Level Classification  Application Recovery Classification  Patch Management  Platform Logging |
| Provisioning & Service Catalog | Infrastructure Provisioning  Service Catalog |
| Availability Management | Backup & Recovery (DR)  High Availability |
| Platform Lifecycle Management | Application Architecture Patterns  Security Group Patterns  AMI Lifecycle Management  Platform Monitoring  License Management  Network Architecture Patterns  Load Balancers  Firewall Management  Application Logging  Resource Cost Optimization  Configuration Management |
| Financial Management | Chargeback Process  Billing Method  Budgeting and Forecasting  Reporting |

# Platform Operations

## Tagging

Description

Tagging is the most important documentation you can do in cloud and enables the strategies discussed further in this document. While each section discusses specific items the constant theme is tagging. Tagging is the practice of adding information to an instance or virtual machine. Each tag should have a specific function and it is important that they have the same name and the values are the same across the instances. This allows for searching, scripting and alerts. These function will make up the backbone of your automation and allow a small number of administrators to manage all of the systems.

Technical Approach

Tags are key value pairs that consist of Key Names (categories) and Key Values (the actual names of the tags). The Key Names identified below will be used. The Key Values are located in the corresponding Runbook categories found in the following sections.

|  |
| --- |
| ServiceLevel |
| RecoveryTier |
| ApplicationID |
| EnvironmentLevel |
| CostCenter |
| MedPharma |
| BackupPolicy |

Enforcement of tags will be handled through the following approaches:

* **CloudFormation-** Tagging can be enforced as part of the initial parameters and constraints, in each script. These constraints will fail a script if tags are missing.



This style of Coding turns into a cloud formation section looking like this:

"Tags":[  
 {"Key" : "Application", "Value": {"Ref" : "TagsApplicationParam"}},  
 {"Key" : "CostCenter", "Value": {"Ref" : "TagsCostCenterParam"}},  
 {"Key" : "Environment", "Value": {"Ref" : "TagsEnvironmentParam"}},  
 {"Key" : "Backup", "Value": {"Ref" : "TagsBackupParam"}},  
 {"Key" : "ServiceClass", "Value": {"Ref" : "TagsServiceClassParam"}},  
 {"Key" : "RecoveryTier", "Value": {"Ref" : "TagsRecoveryTierParam"}}  
]

* **Using the AWS Console-**
* IAM can control CloudFormation access, but the user will always be able to create a single machine via the EC2 control panel. We can enforce tagging on these devices via Lambda scripting and scheduled procedures.
* Lambda Scripting:
  + Located at: %CodeCommit%/LambdaFunctions/MissingTag.py
* These tags can be placed on most item in the AWS environments:

|  |  |  |
| --- | --- | --- |
| Resource | Tagging support | Tagging restrictions |
| AMI | Yes | None |
| Bundle task | No |  |
| Customer gateway | Yes | None |
| Dedicated Host | No |  |
| DHCP option | Yes | None |
| EBS volume | Yes | None |
| Instance store volume | No |  |
| Elastic IP | No |  |
| Instance | Yes | None |
| Internet gateway | Yes | None |
| Key pair | No |  |
| NAT gateway | No |  |
| Network ACL | Yes | None |
| Network interface | Yes | None |
| Placement group | No |  |
| Reserved Instance | Yes | None |
| Reserved Instance listing | No |  |
| Route table | Yes | None |
| Spot Instance request | Yes | None |
| Security group - EC2-Classic | Yes | None |
| Security group - VPC | Yes | None |
| Snapshot | Yes | None |
| Subnet | Yes | None |
| Virtual private gateway | Yes | None |
| VPC | Yes | None |
| VPC endpoint | No |  |
| VPC flow log | No |  |
| VPC peering connection | Yes | None |
| VPN connection | Yes | None |

## Application Service Level Classification

Description

This tag expresses the service level of the specific application or portions of the application. This value combined with the RecoveryTier tag, describes the architecture choices advisable for a new application and can be included in external scripting to verify that systems are running correctly and changes that occur have not pushed an application outside of the designed tolerances.

|  |  |
| --- | --- |
| Key Name | Key Value |
| ServiceLevel | Class5 |
|  | Class4 |
|  | Class3 |
|  | Class2 |
|  | Class1 |

Availability Patterns (detailed in sections below) associated with each service class:

|  |  |
| --- | --- |
| Service Class | Availability Pattern Associated |
| Service Class 5 | Warm-standby (in multi-AZ), Multi-site (in multi-AZ), or Multi-region |
| Service Class 4 | Warm-standby (in multi-AZ) or Multi-site (in multi-AZ) |
| Service Class 3 | Backup & Restore single instances with Snapshots stored in S3 |
| Service Class 2 | Backup & Restore single instances with Snapshots stored in S3 |
| Service Class 1 | Backup & Restore single instances with Snapshots stored in S3 |

## Application Recovery Classification

Description

The RecoveryTier tag identifies the backup strategy as well as the availability pattern for the application. This tag is also expressed in BackupPolicy, discussed later. A tag of RecoveryTier: Tier1 should generate the most aggressive backup and life cycle policy, generating a snapshot at least every 4 hours.

|  |  |
| --- | --- |
| Key Name | Key Value |
| RecoveryTier | Tier1 |
|  | Tier2 |
|  | Tier3 |
|  | Tier4 |
|  | Tier5 |

Availability Patterns (detailed in sections below) associated with each service class:

|  |  |  |
| --- | --- | --- |
| Recovery Tier | RTO/RPO (in hours) | Availability Pattern Associated |
| Recovery Tier 1 | 24/4 | Warm-standby (in multi-AZ), Multi-site (in multi-AZ), or Multi-region |
| Recovery Tier 2 | 48/24 | Warm-standby (in multi-AZ) or Multi-site (in multi-AZ) |
| Recovery Tier 3 | 72/24 | Backup & Restore single instances with Snapshots stored in S3 |
| Recovery Tier 4 | 168/24 | Backup & Restore single instances with Snapshots stored in S3 |
| Recovery Tier 5 | 169+/24 | Backup & Restore single instances with Snapshots stored in S3 |

## Patch Management

Description

Patching in the cloud is similar to the patch process in your current environment. Making backups and snapshots of systems before a patch is applied is far simpler and more convenient.

Current CUSTOMER Policy

* All systems must have an identified maintenance window
* All patches will be delivered within 31 days of availability
* All patching tasks are automated
* Testing should be automated (or app teams automatically notified of completion)
* Non prod environments should be patched prior to prod
* Before a system is patched, a snapshot should be made of it.

Tagging:

* These two tags allow for automation scripts to be used to interact with you instances at the application level and control the implementation of the patches.
* Controlling what environment level and which application will be patched by a command or script. This is executed through SSM and through Lambda scripts.
* These scripts will automate the flow of patching through the environments using the tags described.

Tags for Patch management:

|  |  |
| --- | --- |
| Key Name | Key Value |
| EnvironmentLevel | Production |
|  | Staging |
|  | QA |
|  | Development |

|  |  |
| --- | --- |
| Key Name | Key Value |
| ApplicationID | [custom] - (Found in CUSTOMER ServiceNow APM, custom value per project) |

Technical approach per Operating System

* **Windows Patching** 
  + WSUS repository and reporting
  + SSM command execution, for deployment of groups of systems
  + Scheduled Tasks for timed executions
  + CloudTrail for API logging
* **Linux Patching**
  + Switch to AWS AMI
  + Use AWS Package Repo
  + Updates will be done through the non-prod account and then pushed the Prod account.
  + CloudTrail for API logging

Additional technical approaches (AWS specific)

* **Amazon EC2** (e.g. Service Level 1, 2 and 3)
  + Develop a script to patch Windows and Linux instances using Simple Systems Manager (SSM)
    - One script to target each OS, or Specialized Tagged Application
    - Exclude EC2 instances that were launched from an auto scaling group
* **Amazon AMI Environment Roll** (This is Service Level 3 and 4. Potential any other system that cannot be auto scaled.)
  + Develop a script to patch Windows and Linux instances using Simple Systems Manager (SSM)
    - Each script will create an AMI of the current production system.
    - These AMI’s will be used to build a QA and Staging environment.
    - Once testing is complete, a new AMI will be created or the ELB will be switched between the production and staging environment.
* **Amazon EC2 - Auto Scaling** (This is for Service Level 3,4 and 5)
  + Develop a script to patch and create a new AMI to be assigned to a new launch configuration
    - Launch new EC2 based on latest launch configuration AMI
    - Patch using SSM
    - Clone current launch configuration and assign newly patched AMI
    - Assign new launch configuration to the existing auto scaling group
    - Force the launch of the newly patched EC2 instance

## Platform Logging

Description

There are two primary services available from AWS to aid in platform logging for your systems. These are Amazon CloudWatch and AWS CloudTrail.

* Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources. You can also use CloudWatch logs to monitor and troubleshoot your systems and applications using your existing system, application, and custom log files. In addition, your existing system, application, and custom log files can be sent to CloudWatch Logs and monitored in near real-time.
* AWS CloudTrail provides a history of AWS API calls for your account, including API calls made via the AWS Management Console, AWS SDKs, command line tools, and higher-level AWS services (such as AWS CloudFormation). The AWS API call history produced by CloudTrail enables security analysis, resource change tracking, and compliance auditing.

Technical Approach

CUSTOMER’s current plan is to use the existing Splunk instance for historic log review and near real-time CloudTrail for event review; however, this will be evaluated on an application by application basis because using CloudWatch logs, fed by CloudTrail, you can automate responses to known issues very easily. This automated response to known issues may also be available with Splunk, but further investigation and testing are needed.

* VPC Flow logs will be used to review specific network traffic for security related tasks, system health and load review.
* Using the tagging described in this document, scripts will be developed that control the resiliency of the systems that are in the monitoring/logging systems
* Logging systems will notify, via SNS to the associated teams based on the tagging systems.
* AWS Config will be used to hold the system to specific in specific known configurations.
* SNS will send notification to the Administration team
* AWS Config will validate that patches and updates are in place and remediated correctly.
* (Optional) You can enable log file encryption, which provides an extra layer of security for your log files. For more information, see [Encrypting CloudTrail Log Files with AWS KMS–Managed Keys (SSE-KMS)](http://docs.aws.amazon.com/awscloudtrail/latest/userguide/encrypting-cloudtrail-log-files-with-aws-kms.html).

## Configuration Management

Using AWS Config

AWS Config is used to evaluate the configuration settings of your AWS resources. You do this by creating AWS Config rules, which represent your ideal configuration settings. AWS Config provides customizable, predefined rules to help you get started. You can also create your own custom rules from scratch. While AWS Config continuously tracks the configuration changes that occur among your resources, it checks whether these changes violate any of the conditions in your rules. If a resource does violate a rule, AWS Config flags the resource and the rule as noncompliant.

The AWS Config console shows you the compliance status of your rules and resources. You can use the console to assess how your AWS resources comply overall with your desired configurations, and you can learn which specific resources are noncompliant and which configuration attributes are the cause. You can also use the AWS CLI, the AWS Config API, and AWS SDKs to make requests to the AWS Config service for compliance information.

By using AWS Config to evaluate your resource configurations, you can more easily assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.

This tool will allow CUSTOMER to validate tagging policies, resources, and security groups. This also allows for the coding of custom Lambda scripts to check for more specific items. In the case of the CUSTOMER Cloud Operations team, this will most likely include application of security groups, OS patches and other non-application specific items.

Example AWS Lambda Functions for AWS Config Rules (Node.js)

AWS Lambda executes functions in response to events that are published by AWS services. The function for a custom Config rule receives an event that is published by AWS Config, and the function then evaluates whether the configuration data provided by the event complies with the rule. The operations in a function for a Config rule differ depending on whether it performs an evaluation that is triggered by configuration changes or triggered periodically.

For information about common patterns within AWS Lambda functions, see [Programming Model](http://docs.aws.amazon.com/lambda/latest/dg/programming-model-v2.html) in the AWS Lambda Developer Guide. AWS Config will invoke a function like the following example when it detects a configuration change for a resource that is within a custom rule's scope.

If you use the AWS Config console to create a rule that is associated with a function like this example, choose Configuration changes as the trigger type. If you use the AWS Config API or AWS CLI to create the rule, set the MessageType attribute to ConfigurationItemChangeNotification. This example evaluates an Amazon EC2 instance and checks whether its instance type matches a specified value (for example, t2.micro).