HealthCare – Cloud Runbook

Revision and Signoff Sheet

Abstract

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

Delivered by:

Professional Services

## Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
|  |  |  |  |
|  |  |  |  |

## Reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Version Approved | Position | Date |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1.1 Change Record i](#_Toc453873186)

[1.2 Reviewers i](#_Toc453873187)

[2 Purpose 4](#_Toc453873188)

[3 Related Documents 4](#_Toc453873189)

[4 Operations Overview 5](#_Toc453873190)

[5 Platform Operations 6](#_Toc453873191)

[5.1 Tagging 6](#_Toc453873192)

[5.1.1 Description 6](#_Toc453873193)

[5.1.2 Technical Approach 6](#_Toc453873194)

[5.2 Application Service Level Classification 6](#_Toc453873195)

[5.2.1 Description 6](#_Toc453873196)

[5.3 Application Recovery Classification 7](#_Toc453873197)

[5.3.1 Description 7](#_Toc453873198)

[5.4 Patch Management 7](#_Toc453873199)

[5.4.1 Description 8](#_Toc453873200)

[5.4.2 Technical Approach 8](#_Toc453873201)

[5.5 Platform Logging 9](#_Toc453873202)

[5.5.1 Description 9](#_Toc453873203)

[5.5.2 Technical Approach 9](#_Toc453873204)

[5.6 Configuration Management 10](#_Toc453873205)

[5.7 Availability Management 10](#_Toc453873206)

[5.7.1 Description 10](#_Toc453873207)

[5.7.2 Technical Approach 10](#_Toc453873208)

[6 Provisioning & Service Catalog 11](#_Toc453873209)

[6.1 Provisioning 11](#_Toc453873210)

[6.1.1 Description 11](#_Toc453873211)

[6.1.2 Technical Approach 11](#_Toc453873212)

[6.2 Service Catalog 12](#_Toc453873213)

[6.2.1 Description 12](#_Toc453873214)

[6.2.2 Technical Approach 12](#_Toc453873215)

[7 Availability Management 12](#_Toc453873216)

[7.1 Backup & Restore 13](#_Toc453873217)

[7.1.1 Description 13](#_Toc453873218)

[7.1.2 Technical Approach 13](#_Toc453873219)

[7.2 Pilot Light 13](#_Toc453873220)

[7.2.1 Description 13](#_Toc453873221)

[7.2.2 Technical Approach 14](#_Toc453873222)

[7.3 Warm Standby 14](#_Toc453873223)

[7.3.1 Description 14](#_Toc453873224)

[7.3.2 Technical Approach 14](#_Toc453873225)

[7.4 Multi-site (AZ) 15](#_Toc453873226)

[7.4.1 Description 15](#_Toc453873227)

[7.4.2 Technical Approach 15](#_Toc453873228)

[7.5 Multi-site (Region) 15](#_Toc453873229)

[7.5.1 Description 15](#_Toc453873230)

[7.5.2 Technical Approach 15](#_Toc453873231)

[7.6 Other Availability Considerations 16](#_Toc453873232)

[8 System Lifecycle Management 16](#_Toc453873233)

[8.1 CAH Application Architecture Patterns: 16](#_Toc453873234)

[8.1.1 Public (DMZ) Patterns 16](#_Toc453873235)

[8.1.2 Internal Patterns 17](#_Toc453873236)

[8.2 Security Architecture Patterns 20](#_Toc453873237)

[8.2.1 Security Group Architecture 21](#_Toc453873238)

[9 Event Monitoring 22](#_Toc453873239)

[9.1.1 Installing the Cloudwatch Agent: 22](#_Toc453873240)

[9.1.2 Dashboarding with AWS Cloudwatch 23](#_Toc453873241)

[10 License Management 23](#_Toc453873242)

[11 Network Architecture 24](#_Toc453873243)

[11.1 CAH’s VPC and Subnet patterns 24](#_Toc453873244)

[11.2 Cloudformation patterns 24](#_Toc453873245)

[12 Security 25](#_Toc453873246)

[13 Financial Management 25](#_Toc453873247)

[13.1 Trusted Advisor 25](#_Toc453873248)

[13.2 Reserved Instances 25](#_Toc453873249)

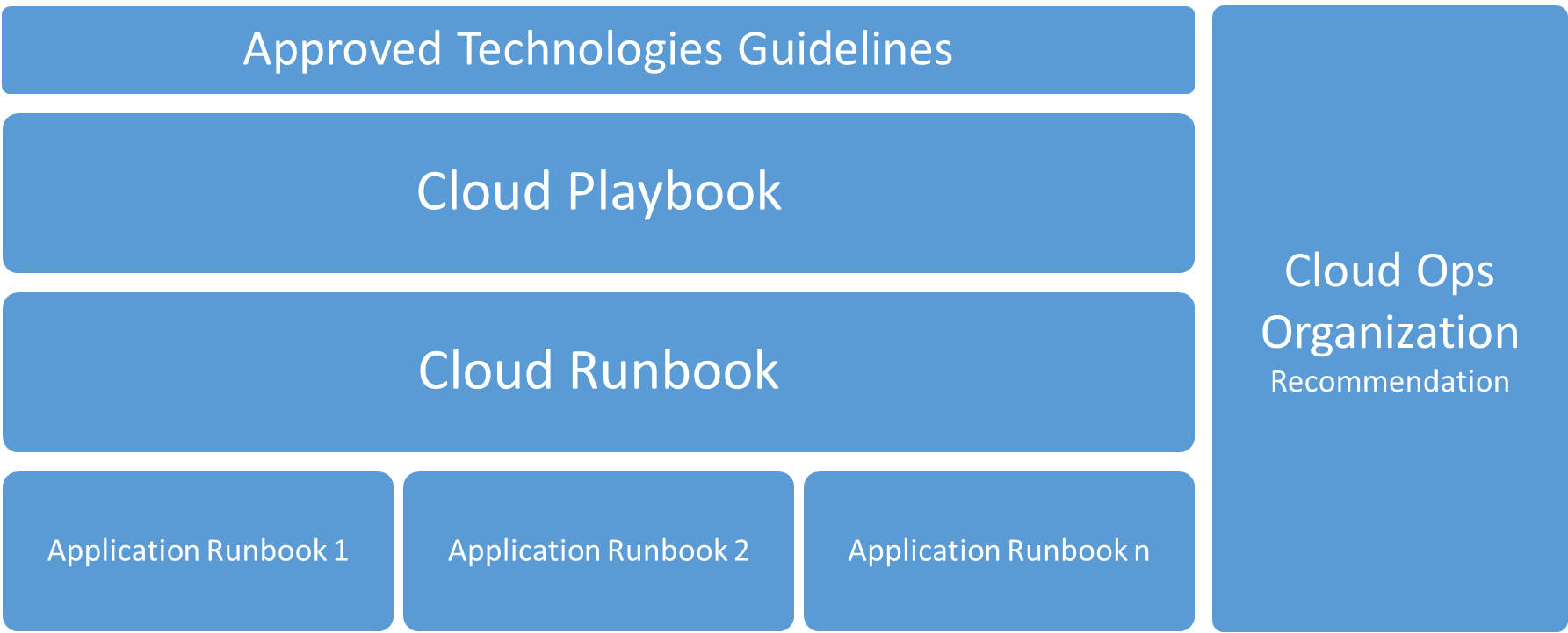
[13.3 Further Considerations 25](#_Toc453873250)

# Purpose

This document should be used as a navigation point for future choices and efforts in the operation of a cloud environment. It is not limited to functions on AWS, or your onsite systems. 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 from technology selections, strategy formation, and cloud execution, all the way through Application-specific Runbook details. The level of detail included in these documents in illustrated below, starting with the highest level strategy information at top, pushing down through the most specific and detailed at the bottom.



* Approved 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 CAH’s approach to deploying applications on Amazon Web Services infrastructure. Documents CAH’s current approach to architecture, provisioning, and operations on AWS; and also provides the next steps to increase cloud adoption maturity.
* Cloud Runbook is an execution level document that provides the technical detail of how to operate AWS infrastructure.
* Application Runbooks provide a view into application specific exceptions from the standard cloud guidelines as outlined in the Cloud Runbook.
* Cloud Ops Organization outlines the resources, skills and teams that are involved in driving the cloud adoption (COE) as well as those who manage the day to day provisioning, usage, monitoring, and cost of cloud resources (CloudOps).

# Operations Overview

As organizations adopt cloud services, the operations 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 who and how they are offered. This document is to be a living document that outlines the IT service management features like proactive monitoring, patch, 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.

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 |
| Application Lifecycle Management | Application Architecture Patterns  Security Group Patterns  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 |

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.

# 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 same across the instances. This allows for searching, scripting and alerts. These function will make up the back bone of your automation and allow a small number of administrators to manage all of the systems.

### Technical Approach

The tag categories (Key Name) below will be used. The Key Values (the actual names of tags) are located in the corresponding Runbook categories found in the following sections.

|  |
| --- |
| Key Name |
| ServiceLevel |
| RecoveryTier |
| ApplicationID |
| EnvironmentLevel |
| CostCenter |
| MedPharma |
| BackupPolicy |
| RecoveryTier |
| License |

Enforcement of tags will be handled through the following approaches:

Using 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.

Using the AWS Console-

* IAM can control CloudFormation access, but user will always be able to create single machine via the EC2 control panel. We can enforce tagging on these device and validate that are in monitoring compliance via Lambda scripting and scheduled procedures.
* Lambda Scripting: Git:// (to be placed here )

## Application Service Level Classification

### Description

This tag expresses the Service Level of the specific application or portions the application. This value combined with Recovery tier, 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: ServiceLevel**

|  |
| --- |
| Key Value |
| 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: RecoveryTier**

|  |
| --- |
| Key Value |
| 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 patching process in your current deployment. Making backups and snapshots of systems before a patch is applied is far simpler and more convenient.

### Technical Approach

System Prerequisites:

* All systems must have maintenance window
* All patches delivered within 31 days of availability
* All patching tasks 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 app will be patch 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: EnvironmentLevel**

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

**Key Name: ApplicationID**

|  |
| --- |
| Key Value |
| [custom] - (Found in CAH 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.

#### Additional technical approaches (AWS specific)

* **Amazon EC2** (e.g. Service Class 1, 2 and 3)
  + Develop a script to patch Windows and Linux instances using Simple Systems Manager (SSM)
    - Once 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 Class 3 and 4. Potential any other system that can not 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 will be used to build a QA and Staging environment.
    - On Testing 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 Service Class 3,4 and 5)
  + Develop a script to patch and create a new AMI to by 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 in platform logging of 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.
* With AWS CloudTrail, you can get 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

CAH’s current plan is to use their 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 automated 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.
* Config will be used to hold the system to specific in specific known configurations.
* SNS will send notification to the Administration team
* Config will valid 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

AWS Config

## Availability Management

### Description

There are two primary services available from AWS to in platform logging of 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.
* With AWS CloudTrail, you can get 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

CAH’s current plan is to use their 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 automated 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.
* Config will be used to hold the system to specific in specific known configurations.
* SNS will send notification to the Administration team
* Config will valid 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).

# Provisioning & Service Catalog

## Provisioning

### Description

To increase the identification, creation, and re-use of architecture patterns, all deployments (when possible and feasible) on AWS will be done through the use of CloudFormation scripting. AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you leverage AWS products such as Amazon EC2, Amazon Elastic Block Store, Amazon SNS, Elastic Load Balancing, and Auto Scaling to build highly reliable, highly scalable, cost-effective applications in the cloud without worrying about creating and configuring the underlying AWS infrastructure. AWS CloudFormation enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).

### Technical Approach

CloudFormation scripts will used to deploy through Development, QA, Staging, and Production. In the future, ServiceNow could be used by the requester to fill out the required tags / parameters that would then be dynamically populated at launch time into CloudFormation.

* Cloudformation templates are stored in AWS CodeCommit. Contact the CloudOps team for access.
* Naming convention for templates - [To be determined by CAH during future iterations]
* Best practice for versioning or forking a template in CodeCommit [TBD]

Cloudformation templates are handled in two files, the template file and the parameter file. The first, Template, file is used to define the resources and flow of creation

* + S3 Bucket name with the templates that are going to be used.
  + Under this bucket, a folder named “StackConfigs”

Template:

https://s3.amazonaws.com/YourBucket/YourFolder/YourTemplate.template

Parameter:

<https://s3.amazonaws.com/YourBucket/YourFolder/StackConfigs/YourParameters.json>

This will enforce your tagging policies and will hold the information for deployments.

Links to AWS CloudFormation documentation:

* AWS CloudFormation Concepts - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-concepts.html>
* How Does AWS CloudFormation Work - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-howdoesitwork.html>
* Working with Stacks - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacks.html>
* Working with Templates - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-guide.html>
* Sample Templates - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-sample-templates.html>

#### Code Repository (AWS CodeCommit)

* + Each Project in the Code Repository should limited to one Project
  + Each BU should have multiple project in the repo
  + Each project should include both the application code and the infrastructure code for that project and should be able to be deployed in parallel or by themselves.
    - Each piece of a deployment should be broken up by the team responsible for that piece. i.e. Each layer should be deployable without impact to other layer of the application.

## Service Catalog

### Description

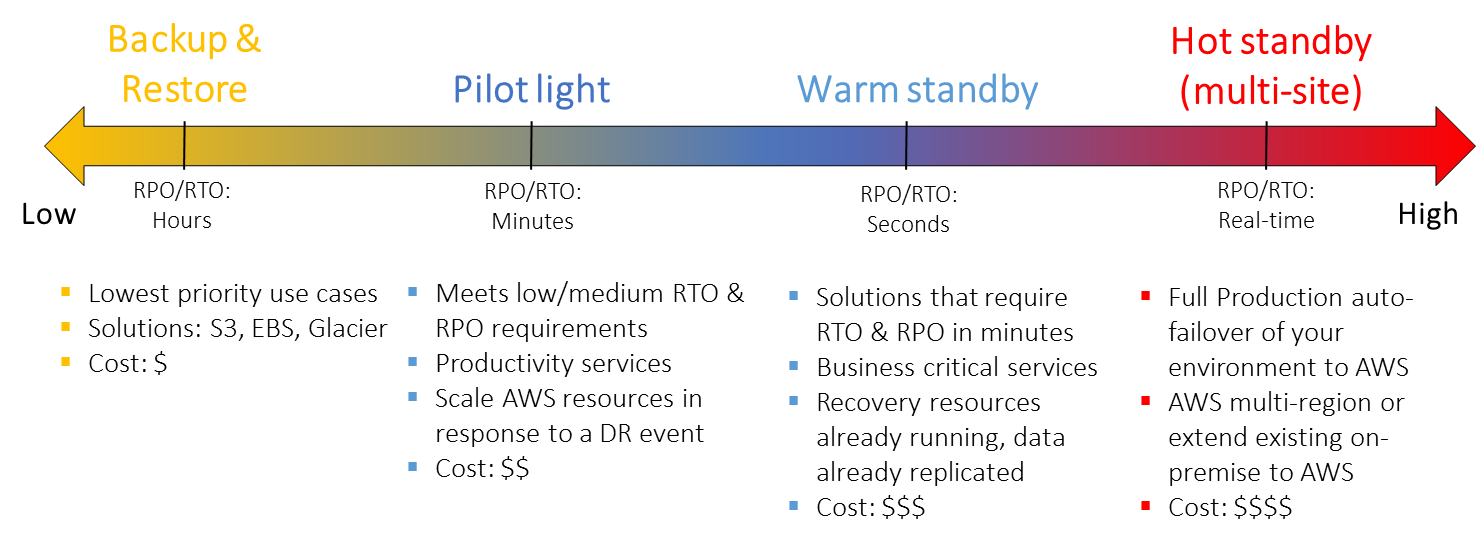
In the future, a service catalog (e.g. Service Catalog) could be used by the requester to fill out the required tags / parameters that would then be dynamically populated at launch time into CloudFormation.

### Technical Approach

[To be determined by CAH during future iterations]

# Availability Management

Availability Management encompasses multiple aspects of designing for and operating to insure the availability of applications and services to support the business service level agreement. This includes restoring applications based on Recovery Point and Time objectives. Recovery Point Objectives (RPO) defines how much data loss can be tolerated. While Recovery Time Objective (RTO) defines how quickly the service needs to be restored. Each application should be evaluated based on the business criticality to determine the RTO/RPO requirements. Most organizations classify applications using tiers to convey which recovery option is appropriate for the given application.



Through the use of High Availability (keep your solution running), Backup (make sure your data is safe), and Disaster Recovery (get your data back after a major disaster) the cloud provides comprehensive strategies to meet the desired user experience, SLO/SLA, risk, and cost objectives for each of your applications.

## Backup & Restore

### Description

The Backup and Restore approach to availability management gives low priority applications a reasonably easy to set-up and cost-effective method to restore an application’s availability to its users and to protect from data loss.

Key tenets of backup and restore:

1. Select an appropriate tool or method to backup your data into AWS.

2. Ensure that you have an appropriate retention policy for this data.

3. Ensure that appropriate security measures are in place for this data, including encryption and access policies.

4. Regularly test the recovery of this data and the restoration of your system

### Technical Approach

**Key name: BackupPolicy**

|  |
| --- |
| Key Value [pattern] |
| [schedule]:[retention] |

This Key Value pattern above means:

* [schedule]: A schedule Key Value of 2330|30|7 means the application is snapshotted at 11:30pm UTC, every day of the month, every day of the week
* [retention]: A retention Key Value of 1|7|4|6|2|2 means the snapshots will be retained on the following schedule: 1 backup a day stored (important to set if you do more than one backup a day), 7 daily backups stored, 4 weekly backups stored, 6 monthly backups stored, 2 semi-annual (every 6 months) backups stored, 2 yearly backups stored

## Pilot Light

### Description

The term pilot light is often used to describe a DR scenario in which a minimal version of an environment is always running in the cloud. The idea of the pilot light is an analogy that comes from the gas heater. In a gas heater, a small flame that’s always on can quickly ignite the entire furnace to heat up a house. This scenario is similar to a backup-and-restore scenario. For example, with AWS you can maintain a pilot light by configuring and running the most critical core elements of your system in AWS. When the time comes for recovery, you can rapidly provision a full-scale production environment around the critical core.

Key tenets of Pilot Light:

1. Select an appropriate tool or method to backup your data into AWS.

2. Ensure that you have an appropriate retention policy for this data.

3. Ensure that appropriate security measures are in place for this data, including encryption and access policies.

4. Regularly test the recovery and the restoration of your system

### Technical Approach

A copy of the production environment, on site or in the cloud is keep in an stopped state in the cloud. While the DB is synchronized to the Stopped environment at regular interval to meet the RPO requirements.

* + Example
    - Building an environment that mirrors your current environment, with automation scripting for the system turn up. This allows for a quick shift either through Route53 or another network change.
  + This means:
    - This reduces total outage time and allows for a low cost alternative to building an On Site DR environment.
* Suitable to Service Level 3 or Tier 3 and below

## Warm Standby

### Description

The term warm standby is used to describe a DR scenario in which a scaled-down version of a fully functional environment is always running in the cloud. A warm standby solution extends the pilot light elements and preparation. It further decreases the recovery time because some services are always running. By identifying your business-critical systems, you can fully duplicate these systems on AWS and have them always on. These servers can be running on a minimum-sized fleet of Amazon EC2 instances on the smallest sizes possible. This solution is not scaled to take a full-production load, but it is fully functional. It can be used for non-production work, such as testing, quality assurance, and internal use.

Key tenets of Warm Standby:

1. Select an appropriate tool or method to backup your data into AWS.

2. Ensure that you have an appropriate retention policy for this data.

3. Ensure that appropriate security measures are in place for this data, including encryption and access policies.

4. Regularly test the recovery and the restoration of your system

### Technical Approach

Using one primary AZ and region. All systems are duplicated to another AZ and left in a running state.

* + Example:
    - The complete system is left in a running state, in many cases a smaller implementation and is activated as the primary production site in the event of a primary site failure.
  + This means:
    - All nodes always exist, the cost for this type of DR is more expensive, however this can be mitigated by using reserved instances. The benefit of this mode is that a failover will take less time than other options, and it is still very functional for most older applications.
* Suitable to Service Level 3 or 4 and Tier 2 or 3

## Multi-site (AZ)

### Description

A multi-site solution runs in AWS as well as on your existing on-site infrastructure, in an active-active configuration. The data replication method that you employ will be determined by the recovery point that you choose. For more information about recovery point options. In addition to recovery point options, there are various replication methods, such as synchronous and asynchronous methods.

Key tenets of Warm Standby:

1. Select an appropriate tool or method to backup your data into AWS.

2. Ensure that you have an appropriate retention policy for this data.

3. Ensure that appropriate security measures are in place for this data, including encryption and access policies.

4. Regularly test the recovery and the restoration of your system

### Technical Approach

**Use multiple primary AZ’s. All systems are duplicated to another AZ and taking traffic at all times.**

* + Example
* A site with regional needs. This application is built in two AZ’s which can answer for each other and are both active at all times.
  + This means:
    - All nodes always exist, the cost for this type of DR/HA is the most expensive, however this can be mitigated by using reserved instances. The benefit of this mode is that a fail over will take the least amount of time (potentially none) than other options.
    - This can be a more complicated implementation. It does require more effort and potentially development work, specifically for application that require sticky sessions.
* Suitable to Service Level 4 or 5 and Tier 1 or 2

## Multi-site (Region)

### Description

The use of at least 2 AZ’s, both in different AWS regions and ensuring all systems are duplicated to another region and are both taking traffic at all times.

Key tenets of Multi-site (region):

1. Select an appropriate tool or method to backup your data into AWS.

2. Ensure that you have an appropriate retention policy for this data.

3. Ensure that appropriate security measures are in place for this data, including encryption and access policies.

4. Regularly test the recovery and the restoration of your system

### Technical Approach

* Use multiple primary AZ’s and regions. All systems are duplicated to another AZ/region and taking traffic at all times. This means that the System is geographically separated and is able to handle the intrinsic issues that come from a higher latency then 50 milliseconds.
  + Example
    - A site with multiple Regional needs, in the US and APAC. These sites can answer for each other and are both active at all times. Mostly they serve users in their region.
  + This means:
    - All nodes always exist, the cost for this type of DR is the most expensive, however this can be mitigated by using reserved instances. The benefit of this mode is that a fail over will take the least amount of time (potentially none) than other options.
    - A multi Region architecture is an Advanced Architecture, it may not be suitable to all applications
* Suitable to Service Level 5 and Tier 1

## Other Availability Considerations

Regardless of the above it is good development and planning to create and adjust application to support multiple AZ in one region. This deployment pattern does not need to cost extra and can be cost effective.

**Deploying a system that is spread between two different AZ’s. Using ELB’s to split the traffic being sent to the System allows for a complete failure of one AZ.**

* + This does require the Application to be able to be split into two AZ’s. At very least this can be used to create a two active sites that are coupled and a client will be aware of a very short outage during a failure. Ideally these application are less coupled and able to support applications that shift from one AZ to another.
* Suitable to Service Level 3,4,5 and Tier 1 and 2

# System Lifecycle Management

CAH has chosen to create, use and enforce standard architecture practices wherever possible to decrease one-off solutions. This increases the simplicity of training CloudOps staff. The details, including graphical representations of these architectures are included in the Cloud Runbook.

## CAH Application Architecture Patterns:

### Public (DMZ) Patterns

Application patterns, associated with the common infrastructure patterns. That are available to the public over the internet or through client vpn connections. The patterns show an ELB, Elastic Load Balancer, at the Edge and at the Private layer. This is the best practice and a suggested method where and when possible.

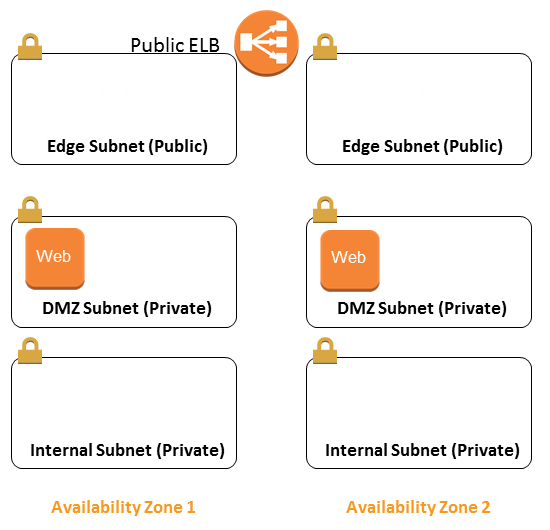
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Public(DMZ) Patterns** | **Characteristics** | **Template Name** | **Parameters** | **Launcher** |
| 1-Tier Application | 1 instance per AZ in a public tier. | DMZTier1.template | DMZTier1.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |
| 2-Tier Application | 1 server per AZ and an RDS solution in the Private tier. | DMZTier2.template | DMZTier2.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |
| 3-Tier Application | 1 server per Private and Public tier and AZ and an RDS solution in the Private tier | DMZTier3.template | DMZTier3.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |

### Internal Patterns

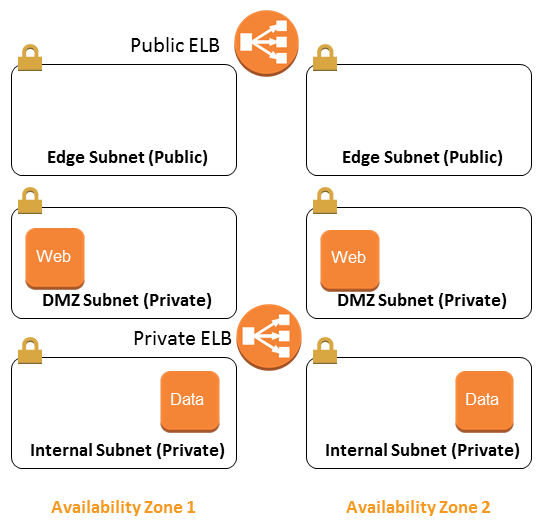
Application patterns for internal or high security application. These application have no direct access to the internet. These patterns also have ELB’s, even in an internal only, it is a suggested patter for a full stack system to use ELB’s.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Internal Patterns** | **Characteristics** | **Template Name** | **Parameters** | **Launcher** |
| Internal 1-Tier | 1 server per AZ | Internal1TIER.template | Internal1TIER.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |
| Internal 2-Tier | 1 server per AZ and an RDS solution in the Private tier | Internal2TIER.template | Internal2TIER.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |
| Internal 3-Tier | 2 server per AZ and an RDS solution in the Private tier | Internal3TIER.template | Internal3TIER.json | AWSLaunch.sh  (./AWSLauncher.sh test10 us-east-1 s3.amazon.com/Bucketname) |

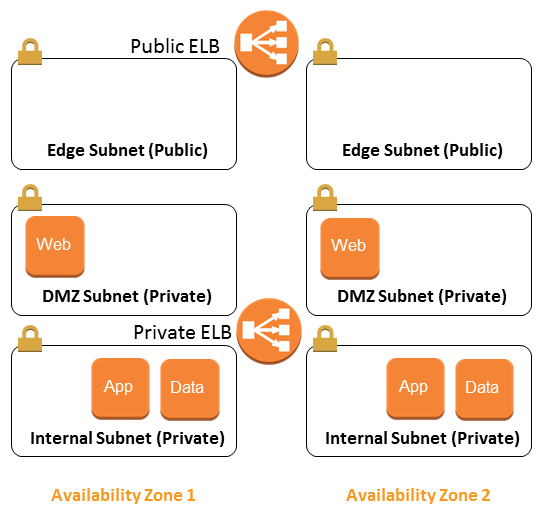
**DMZ 1-Tier Application Architecture**



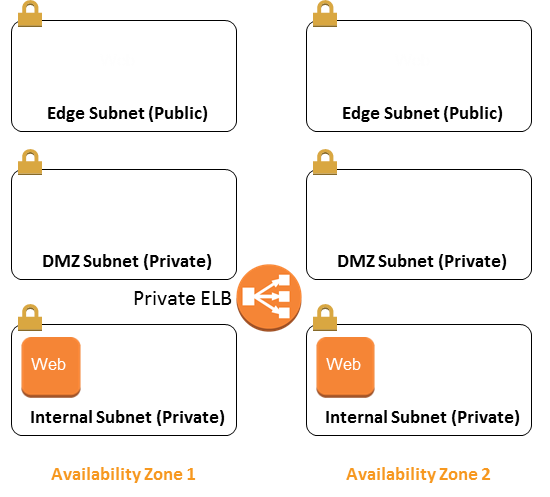
**DMZ 2-Tier Application Architecture**



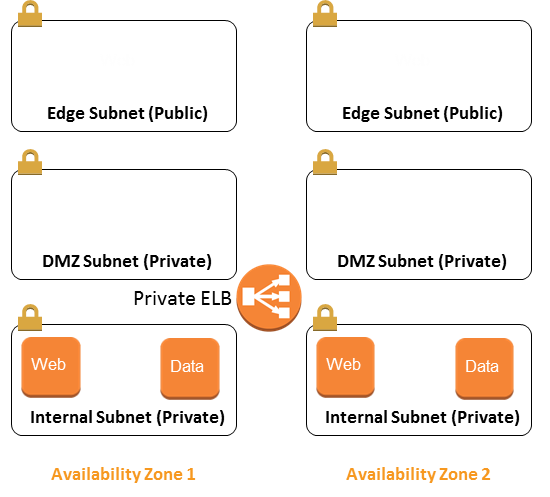
**DMZ 3-Tier Application Architecture**



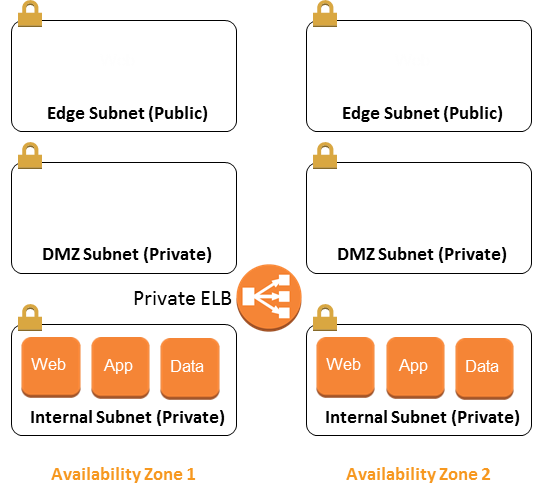
**Internal 1-Tier Application Architecture**



**Internal 2-Tier Application Architecture**



**Internal 3-Tier Application Architecture**

****

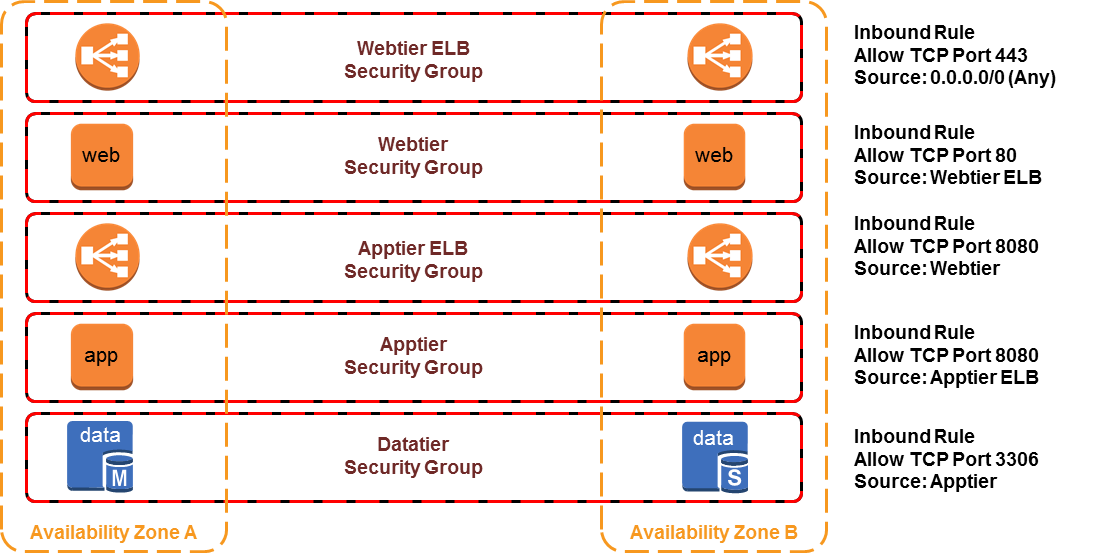
## Security Architecture Patterns

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

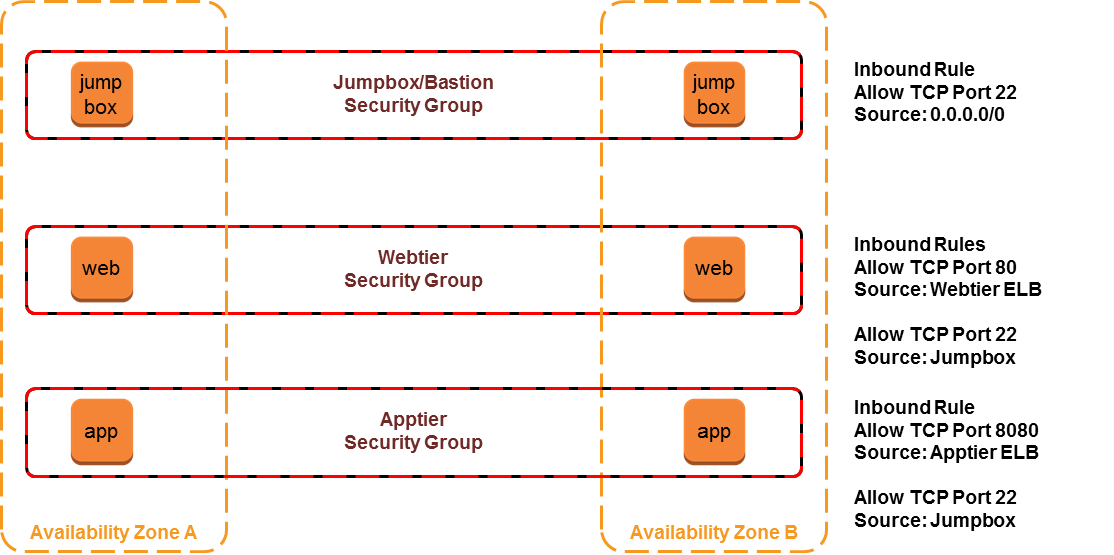
For each security group, you add rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. This section describes the basics things you need to know about security groups for your VPC and their rules.

### Security Group Architecture

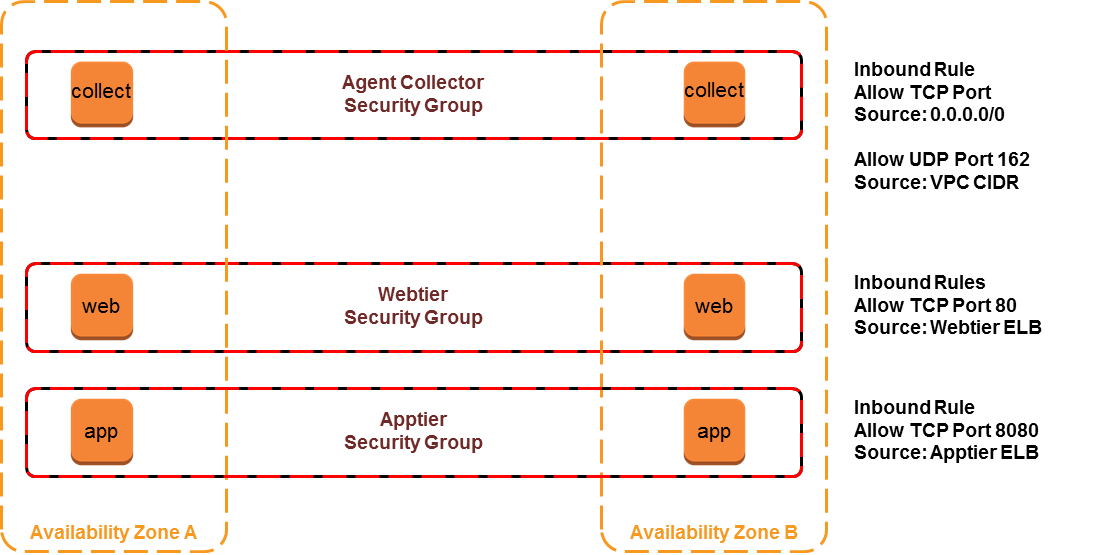
#### Per Application Tier



#### One-to-Many



#### Many-to-One

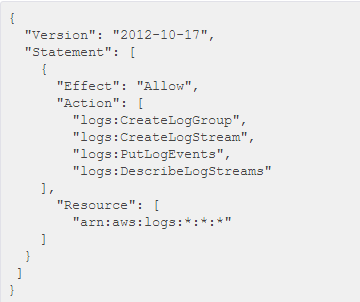


# Event Monitoring

Use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate. You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health.

### Installing the Cloudwatch Agent:

* **To configure your IAM role or user for CloudWatch Logs**
* Open the Identity and Access Management (IAM) console at <https://console.aws.amazon.com/iam/>.
* In the navigation pane, click **Roles**, and then in the **Role Name** column, click an IAM role.
* On the **Permissions** tab, under **Inline Policies**, click **Create Role Policy**.
* On the **Set Permissions** page, click **Custom Policy**, and then click **Select**.
* For more information about creating custom policies, see [IAM Policies for Amazon EC2](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-policies-for-amazon-ec2.html)in the *Amazon EC2 User Guide for Linux Instances*.
* On the **Review Policy** page, in the **Policy Name** field, type a name for the policy.
* In the **Policy Document** field, paste in the following policy:



### Dashboarding with AWS Cloudwatch

Dashboards offer direct views into instance health and resource usage.

The usage of the described tags:

* ServiceLevel
* RecoveryTier
* EnvironmentLevel
* ApplicationID
* CostCenter

# License Management

Use bring your own license (BYOL) to AWS in situations where an Enterprise Agreement (EA) with the license provider is present and that provider allows for license portability to AWS. In this scenario, use tags on any and all resources with those licenses deployed in AWS, so license count/auditing can be performed periodically.

**Key name: License**

|  |
| --- |
| Key Value |
| [To be determined by CAH during future iterations] |
|  |
|  |
|  |
|  |

# Network Architecture

## CAH’s VPC and Subnet patterns

* One account for Production
* One account for Non-production
* One account for POC

This separates the code and network functionality so that accidental pushes to production are reduced.

During the AWS Platform Jumpstart engagement with AWS ProServe, CAH decided on the following Network, VPC, AZ, and Connectivity decisions:

* + Approved Regions
    - US-EAST- 1 (Virginia)
    - US-WEST-2 (Oregon)
    - EU-WEST (Ireland)
    - AP-NORTHEAST-1 (Tokyo)
  + Number of Availability Zones
    - Minimum and maximum of 2 Availability Zones per Region
  + Number of VPC’s
    - 3 Amazon VPCs in Prod AWS account (Prod, Management and Workspaces)
    - 3 Amazon VPC in Non-Prod AWS account (Prod, Management and Workspaces)
    - 1 Amazon VPC in Lab/POC AWS account
    - Any number of Amazon VPCs in Fuse AWS account
    - Any number of Amazon VPCs for Pivotal Cloud Foundry
    - 4 Amazon VPCs for Dev, Stage, QA, and Prod in Cardinal Health @ Home AWS account
  + VPC Connectivity
    - Management VPC will be peered to all VPCs in the same region
    - VPN connections initially, being the process of instantiating Direct Connect lines
      * Establish 4 AWS Direct Connect lines in 4 AWS Regions
        + Dublin Ohio Data Center

1 - 10Gbs line and port speed to US-EAST-1

1 - 10Gbs line and port speed to US-WEST-2

1 - 30Mbs line and 1 Gbs port speed to EU-WEST

1 - 30Mbs line and 1 Gbs port speed to AP-NORTHEAST-1

* + - * + McGaw Illinois Data Center

1 - 10Gbs line and port speed to US-EAST-1

1 - 10Gbs line and port speed to US-WEST-2

1 - 30Mbs line and 1 Gbs port speed to EU-WEST

1 - 30Mbs line and 1 Gbs port speed to AP-NORTHEAST-1

## Cloudformation patterns

* All resources should be scripted using AWS CloudFormation
* Each layer of resources is built as an isolated stack
  + One Subnet for Edge per AZ which his used for a url filter, logging system and other HIPAA compliance needs.
  + One Public Subnet net per AZ for public facing systems. These system will be default run through the edge subnet and through the edge security systems if possible
  + One Private Subnet per AZ that is directly connect to the VPN and is only accessible from the core and the public subnet.
    - There will be exception to this security rule, but they should be documented.
  + These choices are driven by the Service Level that the application or portion of the Application has been assigned.

# Security

Refer to the Security Playbook / Runbook located at: [link]

(This engagement is currently underway with CAH and is scheduled for completion in July 2016. Once complete, this section should be completed either by integrating the Security Playbook into this document or referring to it by link to its location.)

# Financial Management

## Trusted Advisor

Review AWS Trusted Advisor at least weekly to receive best practices (or checks) in four categories:

* cost optimization (server utilization, recommend reserved instances, etc.)
* security
* fault tolerance
* performance improvement

The status of the check is shown by using color coding on the dashboard page: Red (take action), Yellow (review), Green (no action necessary).

## Reserved Instances

* Amazon EC2 Reserved Instances allow you to reserve Amazon EC2 computing capacity for 1 or 3 years, in exchange for a significant discount (up to 75%) compared to On-Demand instance pricing.
* Reserved Instances can significantly lower your computing costs for your workloads and provide a capacity reservation so that you can have confidence in your ability to launch the number of instances you have reserved when you need them.

## Further Considerations

* Turn Off Environments, when you are not using them
* Turn off Dev, QA and Staging environments when not needed
* Shutdown POC’s once finished
* Use data lifecycle policies to delete unneeded snapshots and data