**XYZ Company Well Architected Framework**

Contents

[Introduction 2](#_Toc70706448)

[Security, Risk, and Compliance 3](#_Toc70706449)

[Security Monitoring and Agents 3](#_Toc70706450)

[Certificates 3](#_Toc70706451)

[Data Separation and Segmentation 3](#_Toc70706452)

[Security Groups and Network Access Control 3](#_Toc70706453)

[Web application Firewall 3](#_Toc70706454)

[Data Encryption 4](#_Toc70706455)

[Public IP Addresses 4](#_Toc70706456)

[Next Generation Firewalls Security 4](#_Toc70706457)

[Azure Policy and AWS Configuration Rules 5](#_Toc70706458)

[Performance and Reliability 6](#_Toc70706459)

[Containerization 6](#_Toc70706460)

[Single points of failure 6](#_Toc70706461)

[Using platform services 6](#_Toc70706462)

[Backup and Recovery 6](#_Toc70706463)

[Operational Excellence 8](#_Toc70706464)

[Monitoring 8](#_Toc70706465)

[Deployment 8](#_Toc70706466)

[ITIL – Incident / Problem / Change 9](#_Toc70706467)

[Tagging 9](#_Toc70706468)

[Automation 10](#_Toc70706469)

[Cloud Infrastructure Automation 10](#_Toc70706470)

[Application / IaaS Automation 10](#_Toc70706471)

[Cost Optimization 11](#_Toc70706472)

[Right Sizing 11](#_Toc70706473)

[Key Cost Contributors 11](#_Toc70706474)

# Introduction

This document is meant to lay a framework for designing cloud solutions for XYZ Company that meet requirements from many parts of the healthcare organization including architecture, security, risk, compliance, privacy, operations, and finance.

There are many things to keep in mind when taking XYZ products to the cloud.

# Security, Risk, and Compliance

## Security Monitoring and Agents

**Splunk, Qualys, and Crowd strike** are required components to install in any IaaS deployment. Any **EC2 instances** and virtual machines that are deployed require these components and **to be tagged appropriately** to ensure any security issues with these virtual machines are identified, directed, and sent to the appropriate teams.

Reference  
<https://gehosting.io/docs/gesos-docs/qcs>

## Certificates

**Certificates should be requested through XYZ Standard Processes** found below. There is not a good way to automate certificates today and we are currently exploring options to automate DNS and Certificate provisioning with CoreTech but don’t have an ETA yet on when it will be enabled

Reference  
<https://gehosting.io/docs/certificates/how-to-upload>

## Data Separation and Segmentation

**Make sure to separate data** as much as possible, storage accounts**/S3 buckets** are very low-cost ways to store data separately **per tenant/customer/environment**

## Security Groups and Network Access Control

Security groups are a first line of defense for your applications to determine what network traffic can and cannot go through to servers and resources. These operate on a per instance basis and can be assigned to subnets, virtual machines, and other resources. **You should aim to have the minimum viable rule set between your application layers and components, the outside world, customer sites, and other resources**. You can only allow MySQL ports to MySQL RDS instances only. SSH should only be allowed from approved admin machines/bastion instances and not from internet resources directly. A zero-trust policy should be adopted to be as explicit as you can with rules.

References:  
Azure Network Security Groups: <https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview>  
AWS Security Groups: <https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html>   
AWS Network Access Control List: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>

## Web application Firewall

**WAF is a technology that uses known databases of attack vectors on web applications and SSL offloading to open a web connection, inspect its content, and reject or forward the request on to the intended server to serve the request up**. These services can help with known bad input, IP Reputation blacklists to prevent data from known attackers, SQL Injection detection, Buffer Overflow protection, and other known OS and Application vulnerabilities.

References:   
AWS WAF Documentation: <https://aws.amazon.com/waf/>   
Azure WAF Documentation: <https://docs.microsoft.com/en-us/azure/web-application-firewall/ag/ag-overview>   
Refer to the XYZ Standards document labeled “Web application firewall minimum security framework” – TODO Get link from Tamas.

## Data Encryption

Both cloud providers have agreements with XYZ around how we are required to store data in rest and in transit. **Encryption should be used at the application/protocol layer wherever possible on all connections that can contain user or patient identifiable information**. **If this type of encryption is not supported by the application investigate services such as AWS Nitro that can transparently encrypt your data in transit**. **Data at rest should always be encrypted**. Most platform services have a base standard of encryption, but you should investigate enabling additional encryption via Key Vault/KMS keys that are unique per environment, application, customer, or other boundary. Enable encryption on platform services such as Blob/S3 wherever possible and **always consume these services over secure connections, preferably via VPC/VNET endpoints to keep traffic local to your environment**.

## Public IP Addresses

**Public IPs** are required to expose your application to the world. It could be on a **VPN, Load Balancer, Management Endpoint, or other cloud configuration item. No server directly should have a Public IP** unless that server is serving as a security appliance such as VPN or Firewall. **Any web applications should live behind a load balancer with WAF protection as well, no web port should be directly accessible to the internet**. If you are using **platform services** and they don’t need to be exposed publicly (such as **Azure SQL / RDS Instances) use VPC/VNET Addressing via private link** or other methods to prevent them from getting a Public IP. If these services require a public IP, ensure very tight security groups are around them and do not expose these services to any more than they need to.

## Next Generation Firewalls Security

The Secure Virtual private network can be processed in several ways

1. AWS and Azure VPN Connection – a gateway subnet to be designed with security and routing rules and a VPN connection to be setup between the AWS / Azure VPC/VNet Gateway and Customer Gateway at hospitals. The connectivity will ensure a secure point-to-point encrypted connection over internet and is followed in small applications. OpenVPN can also be setup for small installations.

References  
AWS VPN Gateway : <https://docs.aws.amazon.com/vpc/latest/userguide/vpn-connections.html>

Azure VPN Gateway : <https://docs.microsoft.com/en-us/azure/vpn-gateway/tutorial-site-to-site-portal>

1. XYZ recommends using VPN connection with Clinics and Hospitals network using ***Palo Alto Firewalls***. This ensures traffic to be encrypted and processed over secure tunnel while passing from the untrusted zone to trusted zone and vice versa.

The Palo Alto setup require a management interface of Palo Alto VM series, which will allow specific IP broadcasted with assigned elastic IP, and to be configured with the customer gateway at hospital network. The private route tables in Deployment VPC will be associated to Application, Database and Palo Alto trust subnets. The public route table in Deployment VPC will be associated to the Management and Untrust subnet of Palo Alto.

The routing rules at Palo Alto firewall allow connections only though the untrust zone (public) of Palo Alto for any inbound internet or hospital traffic to through trusted zone (private) using the private IP of the AWS instances.

*PAUntrustSG*: Untrust security group is attached to public interface of Palo alto interface which filters traffic from hospital network CIDR.

*PATrustSG*: Trust security group is placed just after Palo alto firewall. After filtering requests from hospital CIDR certain traffic will undergo NAT and will be routed to trust interface of palo alto server. From Trust interface traffic will be routed to bastion, app, web, db.

*PAMGMTSG*: This security group is attached to management interface of palo alto server which is solely for managing firewall like adding firewall rules. Ip range of the team managing palo alto firewall should be whitelisted here.

## Azure Policy and AWS Configuration Rules

All accounts and subscriptions had default policies and configuration rules applied to them. Production accounts have a lot tighter sets of controls and rules applied to ensure data security. There is a compliance view dashboard which can show you all your accounts and their compliance status to the controls listed in the reference section. Reach out to the contacts below to get access to this dashboard for your accounts.

Reference

Azure HIPPA / HiTrust Policy Pack: <https://docs.microsoft.com/en-us/azure/governance/blueprints/samples/hipaa-hitrust-9-2>   
AWS HIPPA Pack: <https://docs.aws.amazon.com/config/latest/developerguide/operational-best-practices-for-hipaa_security.html>   
Contacts: Sri Erra [sridhar.erra@ge.com](mailto:sridhar.erra@ge.com) and Tamas Szaniszlo [tamas.szaniszlo@ge.com](mailto:tamas.szaniszlo@ge.com)

# Performance and Reliability

## Containerization

It is recommended looking at your **application** to see if it’s **components can be put into a container or containers**. When deploying containers, they are all self-contained and can be deployed in a cloud managed Kubernetes cluster or native container technologies like AWS ECS or Azure App/Web Services. Containers can be moved around and restarted easier and do not require the same type of intense ongoing maintenance that virtual machines do. When you need to update a container, you simply push and provide a new image to run a new software version versus having to have custom scripts and actions to update a VM.

References  
<https://azure.microsoft.com/en-us/services/container-instances/>  
<https://azure.microsoft.com/en-us/services/kubernetes-service/>   
<https://aws.amazon.com/containers/>

## Single points of failure

Cloud providers SLAs assume that all application components provide multiple instances to achieve reasonably high uptimes. With AWSS, single instances they will only give ~90% uptime on instances, in azure you can achieve 99.9% uptime with Premium VM types and Premium storage at additional costs. In either case both **cloud vendors recommend using multiple instances and load balancing to achieve a higher uptime of your products**

References  
<https://azure.microsoft.com/en-us/support/legal/sla/>  
<https://aws.amazon.com/compute/sla/>

## Using platform services

**Using cloud native services first will reduce management overhead, complexity, and risk to your application** and provide well supported components for many services, including caching, database services, load balancing, Kubernetes cluster management, and application messaging. Most of these solutions use industry standard APIs and tools and are based around technologies we already use today.

AWS RDS / Azure SQL and Postgres services are a great example of this. They provide a database layer that is compatible with your applications except for some Microsoft SQL tables to watch out for. You don’t need to have someone to manage the instance, upgrade the versions, update the servers, manage space, or other administrative functions when using these services. They can also scale on demand where you can simply choose a bigger tier and have zero or minimal outage to get much more capacity. There is replication and replica support built-in to have reporting instances as well if that would provide useful functionality for your application

Elasticsearch, Redis, Blob Storage, are all other example services available that you can leverage from both AWS and Azure versus running your own clusters of these applications.

References  
<https://docs.microsoft.com/en-us/azure/architecture/framework/scalability/overview>   
<https://docs.aws.amazon.com/wellarchitected/latest/performance-efficiency-pillar/definition.html>

## Backup and Recovery

**Application or product can do backup of the database or storage systems**. AWS and Azure both provides capability and solutions for backup and recovery, during design finalization it is important to pick right backup strategy as moving from local to multiple region or global solution is more costly.

References  
AWS backup and recovery options:

<https://docs.aws.amazon.com/prescriptive-guidance/latest/backup-recovery/backup-recovery.pdf>

<https://d1.awsstatic.com/BackupRecovery/BackupRestoreInfographic.pdf>

Azure backup and Recovery solution:

<https://azure.microsoft.com/en-in/services/backup/>

<https://azure.microsoft.com/en-us/services/site-recovery/>

<https://azure.microsoft.com/en-us/services/storage/archive/>

# Operational Excellence

## Monitoring

The deployment on Cloud should be well integrated with monitoring framework.

In case of AWS deployments, **the installation and integration of CloudWatch agent as part of bootstrapping of EC2 instances and defining right set of server log to process and put on Cloud watch logging is important.** This will ensure with appropriate log. Audit of actions and events are captured in CloudTrail

References

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs>

[Working with CloudTrail Log Files - AWS CloudTrail (amazon.com)](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-working-with-log-files.html)

On Azure, it is important to integrate with Azure Monitor for global monitoring. At cases, when it is important to investigate on application response , throughput or page load time , the SDK integration with Azure App Insight is extremely important. This are light weight plugins integrated with base code to give flexibility on real time performances of web applications and throughputs.

References  
[Azure Monitor Logs - Azure Monitor | Microsoft Docs](https://docs.microsoft.com/en-us/azure/azure-monitor/logs/data-platform-logs)

[Azure Application Insights Developers Guide](https://dev.applicationinsights.io/)

The monitoring solution can even be extended to integrate with third party solutions like New Relic, Splunk or Datadog. These agents will ensure right level of insights on application on real time and reporting based on the key KPI

## Deployment

GEHealthCloud enables integration of DevSecOps platform governed and deployed by the central solutions. The integration and deployment should be automated and compliant to security and compliance standards.

This includes a) Code quality and coverage b) Code security Scan c) Securing the platform and the container d) Dynamic Security Test and management

Tools like Qualys, Cucumber, Splunk is integrated with deployable artifacts. A standard build and deployment strategy should also cover the following

1. Correct source control (GitLab/Github/Azure Repo)
2. Branching strategy for releases
3. Configuration control and Management
4. Artifact management and maintain of build releases in artifactory or container registry (for image bundle)
5. Dynamic container-based build platform (Jenkins or CodeBuild or Azure Pipeline can be used | on-demand or on-commit)
6. Deployment Pipeline automation (on VM | App service | Container platform or Kubernetes PODS)
7. Automated testing and failover (Sonar etc. )
8. Workflow based automated code promotion on demand from DEV > STAGE > PROD as needed

References  
<https://github.build.ge.com/GEHC-DevOps-Infra>

[Azure DevOps Services | Microsoft Azure](https://azure.microsoft.com/en-in/services/devops/)

[DevOps - Amazon Web Services (AWS)](https://aws.amazon.com/devops/)

## ITIL – Incident / Problem / Change

ITIL items

## Tagging

**Everything in your cloud instance should be tagged according to XYZ Standards** linked below. **Include additional tags as well to help identify resources, the customer or consumer of the resource, what its for, and application information**. This helps business owners drill into costs based on tags to find out how much each customer is costing, which applications have higher costs than anticipated, and which resources belong to each application and organization.

Mandatory Tags for Azure environment:

uai - This tag is used for tying usage to a registered application in ServiceNow. Example - uai1234567

env - Tag captures the environmental state for the tagged object, (i.e. dev, stage, prod, stage, etc.) Example – prod

Mandatory Tags for AWS environment:

name - This is used for the common name for anything tagged. Example – my-app-1

uai - This tag is used for tying usage to a registered application in ServiceNow. Example - uai1234567

env - Tag captures the environmental state for the tagged object, (i.e. dev, stage, prod, stage, etc.) Example – prod

Resource Tagging Best Practice:

name – This is used for the common name for anything tagged. Example – my-app-1

uai – This tag is used for tying usage to a registered application in ServiceNow. Example - uai1234567

AccountID - AccountID of individual accounts the resources are deployed into. Example – 752686809366. With the accountID tag, management of logs from various accounts into centralized location is made easier.

env - Env tag defines the environmental stage for a tagged resource. (i.e., dev, test, prdge, prd). This can be used to create and assign IAM roles based on the type of environment and restrict access to objects if the resource belongs to production environments.

Reference  
<https://gehosting.io/docs/compliance/azure-tagging>  
<https://gehosting.io/docs/compliance/tagging>  
<https://devcloud.swcoe.ge.com/devspace/display/WIUCK/Resource+Tagging+Best+Practices>

<https://devcloud.swcoe.ge.com/devspace/display/WIUCK/HC+Tagging+Standards>

# Automation

References  
<https://docs.microsoft.com/en-us/azure/architecture/framework/devops/monitoring>  
<https://docs.aws.amazon.com/wellarchitected/latest/performance-efficiency-pillar/monitoring.html>

## Cloud Infrastructure Automation

Cloud Items – **CloudFormation / terraform / arm**

## Application / IaaS Automation

**IaaS / Chef / Ansible**

Use automated technology to deploy your resources.  If it’s cloud formation, terraform, arm templates, just have documentation as to what is stood up and make it re-producible.  This helps keep dev/engineering/production/stage environments very consistent and improves your ability to deploy new environments.

If you are using IaaS (Infra as a Service) Try to develop chef/ansible scripts to do your VM deployments as well, deploying your application code/etc.

References  
Detail guide on how to automate infrastructure services using Terraform:

<https://ge.box.com/s/rokovpmq8d5myhdcv3rtqojea0odu8rw>

# Cost Optimization

Cloud costs money, every byte of data received, stored, and processed generates charges in the fractions of a penny but they add up very fast. Every minute you are consuming storage, running a VM, have a load balancer online or a container running adds cost. You should be sure to always keep an eye on what your running and the spend associated with it.

## Right Sizing

**Right size your environment, do performance testing and tuning to find the right size of your resources**. If you provision a 4vcpu resource and only are consuming 1vcpu of capacity, you are still charged the full price of your instance. Use burstable (T series in AWS / B Series in Azure) to give you capacity to burst when most of your workload is idle or small but you do have bursts of activity like a nightly job).

Know what’s in your environment, frequently someone will try something out but forget to remove/deprovision it and before you know it you are charged 740 hours for a resource that was only intended to be used for 1-2 hours for a test. Ensure you deprovision everything associated with a resource as well. When you remove a virtual machine, make sure the network adapters, disks, IP Addresses, snapshots, backups, and any other static resources created with that VM are removed, don’t assume they are removed because you deleted a resource

## Key Cost Contributors

**If your work is highly scheduled or part of a sandbox/dev/engineering environment that is only used for a few hours a day auto start/stop can be configured so the resources are available when people need them but are deprovisioned when they are not being used.**

**Use Linux wherever possible, Windows and SQL Licensing adds up very quickly** and should only be used in applications where it is a hard requirement. Look at the AWS / Azure price list for VMs and they about double in price if you use a windows virtual machine. Use the correct SQL License and verify you are using features that require that license type. SQL Enterprise is extremely costly and gets picked frequently without justification.

**Reserved instances are a large concern and seen as an easy point to save money**. **They may not give you the ROI you think you see because GE has a steep discount on both clouds already**. **RIs also lock you into instance sizing and SKUs so if you need to grow and shrink down the road**, they may limit your ability to do so. You should ensure you have a stable and cost-efficient environment before starting a reserved instance conversation.

Reference  
Auto Start/Stop Azure: <https://docs.microsoft.com/en-us/azure/automation/automation-solution-vm-management>  
Auto Start/Stop AWS: <https://aws.amazon.com/premiumsupport/knowledge-center/start-stop-lambda-cloudwatch/>   
AWS Price Calculator: <https://calculator.s3.amazonaws.com/index.html>   
AWS Cost Management: <https://aws.amazon.com/aws-cost-management/>   
Azure Price Calculator: <https://azure.microsoft.com/en-us/pricing/calculator/>   
Azure Cost Management: <https://azure.microsoft.com/en-us/services/cost-management/#features>  
CoreTech RI Guidelines: <https://gehosting.io/docs/billing/understanding-reserved-instances-and-savings-plans>   
CoreTech Guardrails Pricing: <https://gehosting.io/guardrails/features>