FNZChain Cloud Infrastructure Security Specifications

11/2018

Table of Contents

Document Revisions

Revision Date	Version	Description	Sections Affected	Revised By	Approved By
06/11/2018	v0.1	Draft	All	Bin Li	Bin Li

Introduction

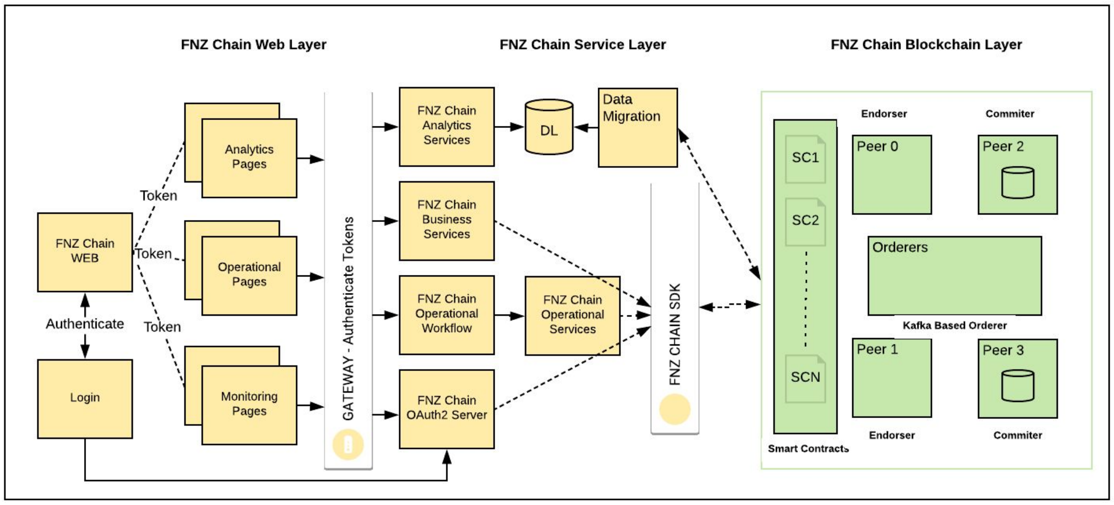
Scope

This security specification is intended to provide details of the security measures applied/implemented as part of the FNZChain project, which has followed the current AWS best security practice and principles on system architecture design in a public Cloud environment. The security measures and implementation is an ongoing task that must be continuously monitored and maintained, therefore this document highlights only the current implementation state with a some suggestions for future improvements. As AWS introduces new features and services almost daily, this document must be reviewed frequently and updated per any new security features implemented.

FNZChain Project Overview

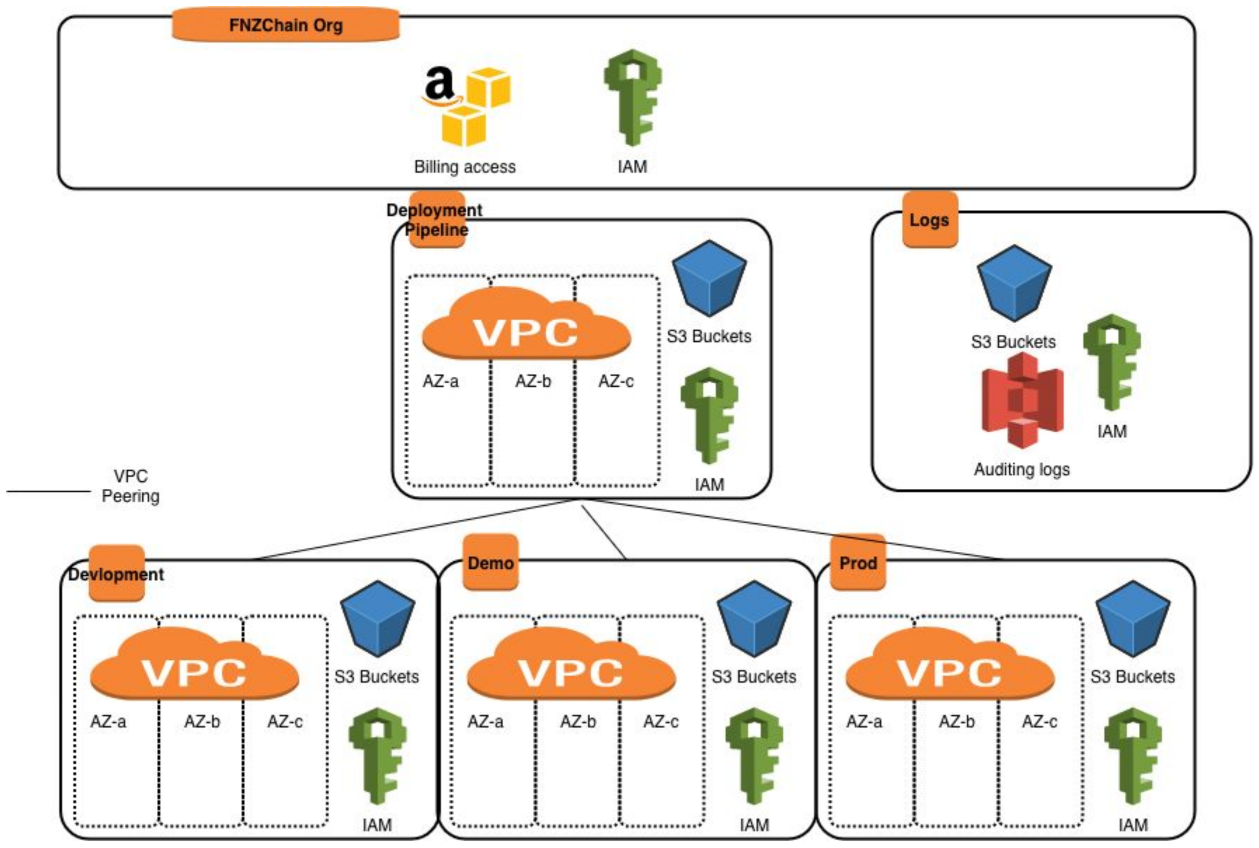
Application Architecture

The FNZ Chain Solution is designed to support transfer agency business across the industry using blockchain technology to reduce the cost and risk related to servicing investors, whilst providing high availability and transaction immutability. The functional scope covers orders, pricing, settlement, distribution, corporate actions and reporting services. The following diagram specifies the logical layers of FNZChain project: Web, Service and Blockchain layer, which also details the logical components that are involved in the current delivery scope.



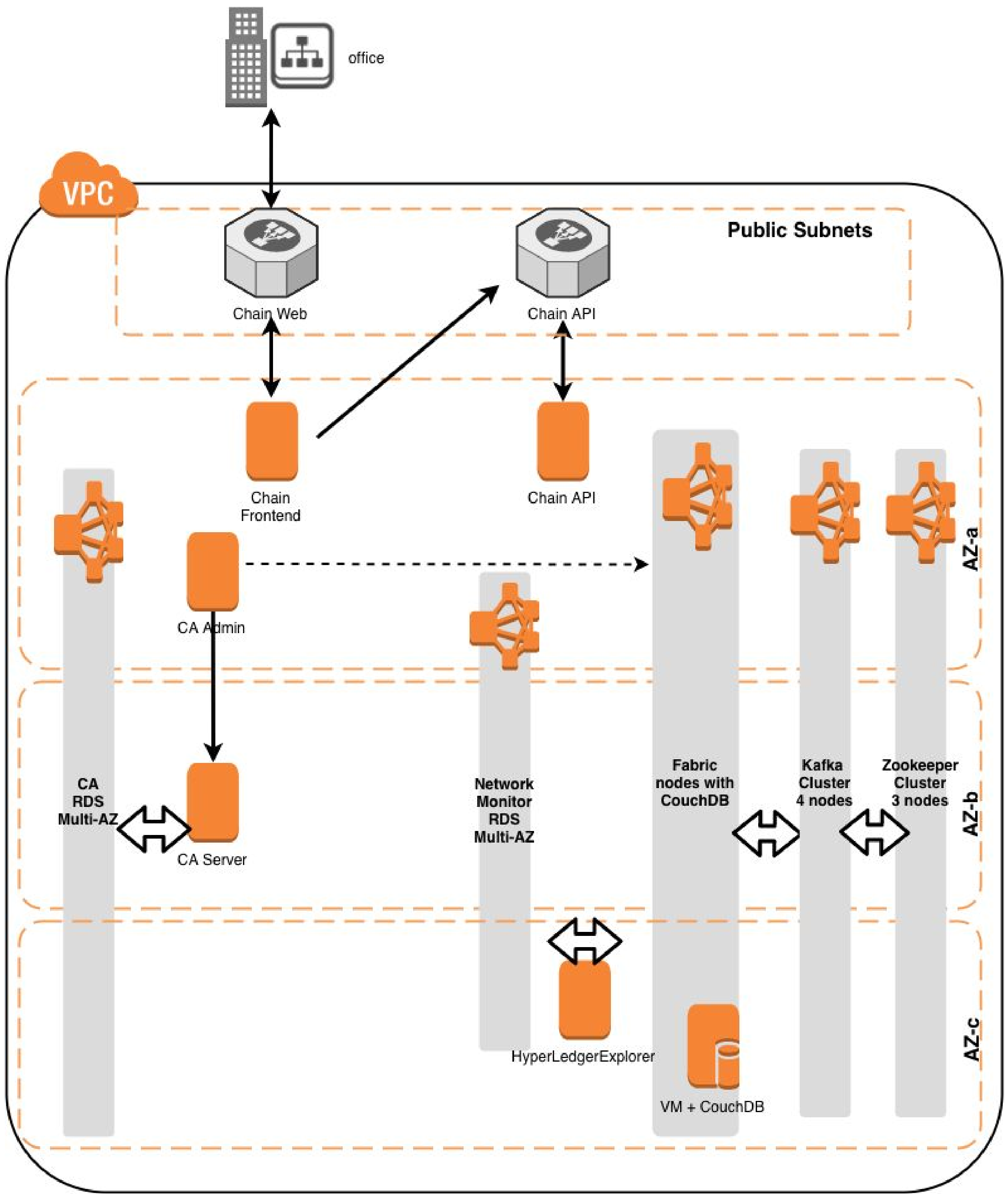
Application AWS Accounts Setup

An FNZchain project deployment is isolated in an individual VPC for networking communication control and secure software development management. Due to the physical limitations of a single VPC, multiple AWS accounts are used in the AWS infrastructure for further logical and physical network separation (see the picture below). The VPC ENVs are not routable to each other, with the exception of the deployment pipeline VPC which peers with other VPCs in order to share and schedule the deployment processes. The logs (Cloudtrail, VPC flow, ELB logs) are stored in a separate AWS account for auditing purpose and more restricted access control.



Application Implementation

FNZChain project is based on Hyperledger fabric, which is an open source blockchain framework implementation and hosted in The Linux Foundation. Depended on Hyperledger fabric, the FNZchain project implements all essential components with added customised chain code of required business logic. The break down components of the project can be found in the following system architecture diagram. (A FNZChain project setup are deployed in a single VPC)



Each component is implemented with several security features that are described in the rest of this document, but highlighted here for clarification in the production environment. All component hosts are running in a VPC behind a firewall; replicable and with faster recoverable via IaC:

- **Zookeeper cluster**
  3 nodes cluster across all AZs in London region, built-in redundancy and fault tolerance, survival with one node failure. The associated security group only allows the Kafka cluster to communicate with the Zookeeper API port.

- **Kafka cluster**
  4 nodes cluster across all AZs in London region, built-in redundancy and fault tolerance, survival with three nodes failure. The associated security group only allows the fabric nodes to communicate with the Kafka API port.

- **Hyperledger fabric node network**
  Network nodes across all AZs in London region, data replication among all nodes stored in KMS encrypted EBS volume. Frequent EBS snapshots for backups and faster recovery. The associated security group only allows the Chain API, frontend to communicate with the peer/orderer ports.

- **RDS**
  Multiple RDS with multi-AZ deployment across all AZs in London region for backups and recovery. Disk is encrypted with KMS. The associated security group only allows match service host to connect. Not publicly available from the internet.

- **Chain API**
  EBS encrypted with KMS, single instance with internet ELB only for debugging and only allows office access.

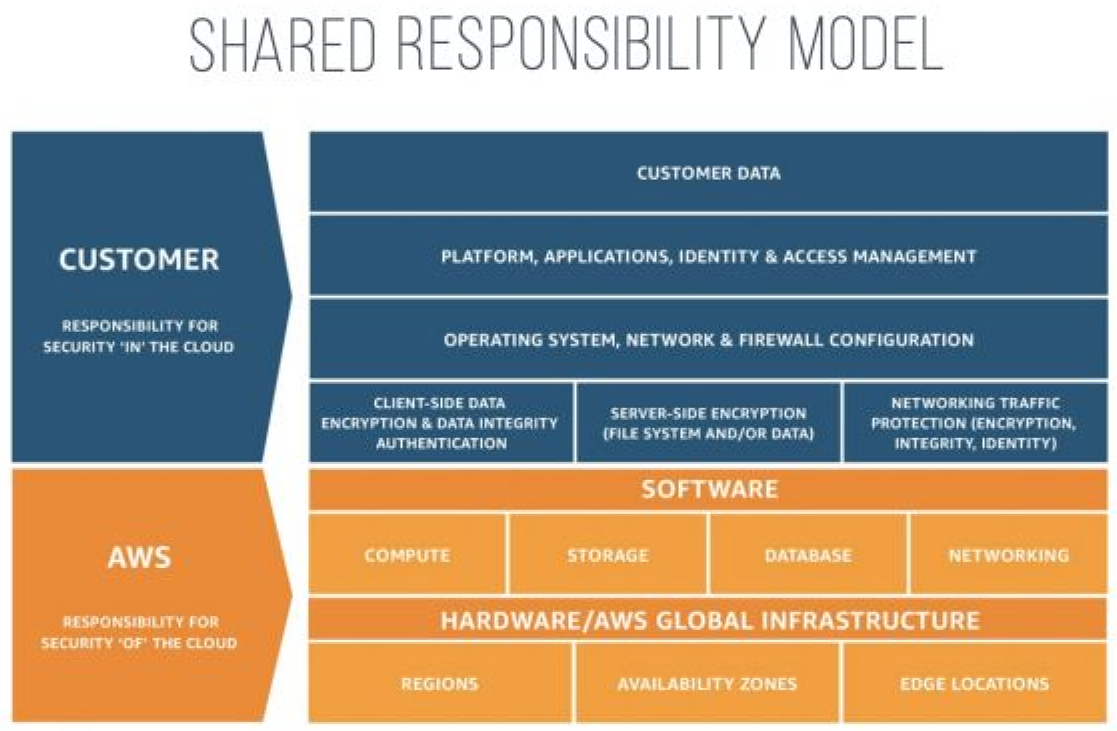
- **Chain Frontend**
  EBS encrypted with KMS, single instance with public facing ELB with TLS for office access.

- **Hyperledger network explorer**
  EBS encrypted with KMS.

AWS Well-Architected Framework

The AWS Well-Architected Framework includes the very best practice and years of experience architecting solutions which operate successfully on AWS across businesses over more than 10 years. The Well-architected framework provides a consistent set of best practices for businesses using the AWS platform to evaluate architectures, and provides a set of checklists that can be used to evaluate how well an architecture is aligned to AWS best practices. Together with a comprehensive security compliance list, a properly designed project running in AWS platform is equipped with essential security requirements.

However, AWS as a platform adopts a model of shared responsibility (see picture below). This shared model reduces the operational burden because AWS operate, manage, and control the layers of IT components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. AWS is responsible for the security of the cloud, and as a customer you are responsible for security in the cloud.


SHARED RESPONSIBILITY MODEL
CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

(pic source: AWS)

Since the shared responsibility, how secure the project is also depends on how well you prepare and implement the required best practice. The AWS Well-Architected Framework is based on five pillars — operational excellence, security, reliability, performance efficiency, and cost optimization. The remainder of this document will focus on the security and reliability pillars, which are more relevant and in the scope of this document.

Definitions

Pillar Name	Description
Security	The ability to protect information, systems,

	and assets while delivering business value through risk assessments and mitigation strategies.
Reliability	The ability of a system to recover from infrastructure or service disruptions, dynamically acquire computing resources to meet demand, and mitigate disruptions such as misconfigurations or transient network issues.

Security Specifications

General Design Principles & Best Practice

- **Strong identity foundation**
  The principle of least privilege and enforced separation of duties with appropriate authorization for each interaction is employed from the very beginning with AWS accounts and resources. The least privilege is applied to the AWS IAM user as well as roles on a request basis.

- **Use Infrastructure as code (IaC) for automation as much as possible, wherever possible.**
  The general deployment pipeline is: Terraform launches and configures required resources, and Chef further bootstraps the resource with dependencies, applications and handles VM level deployment. If containerising is possible, Docker is employed to simplify and control the deployment.
  Automation allows the creation and replication of systems at low cost and avoid the expense of manual effort. It also allows tracking of changes in code, audit of the impact, and reversion to previous parameters when necessary.
  Security best practices also automate using IaC so the same standards are applied in all environments, not only in Production. When testing, the most common networking issues can be identified before rolling changes into production.

- **Test systems at production scale.**
  Multiple production-like environments are replicated in different AWS accounts and isolated from each other, hence testing can be done comprehensively. Such processes are possible and rapid due to the use of IaC and version control.

- **Security at all layers.**

The same security measures and standards are applied to all layers from the bottom up, including edge network, VPC, subnet, load balancer, every instance, operating system, and application.

- **Prepare for security events.**
  Disaster backup recovery and fault tolerance strategy is already implemented when choosing certain components, and built with HA options. The IaC practice also allows rapid rebuild and replication which helps to meet RTO and RPO requirements.

- **Apply Traceability**
  Network infrastructure logs are enabled from the bottom layer, AWS CloudTrail, VPC flow log and elb log are enabled, saved and uploaded into a S3 bucket in the London region. The application logs are sent to AWS cloudwatch log and/or Datadog/papertail log to analyse, to avoid accessing the VM.

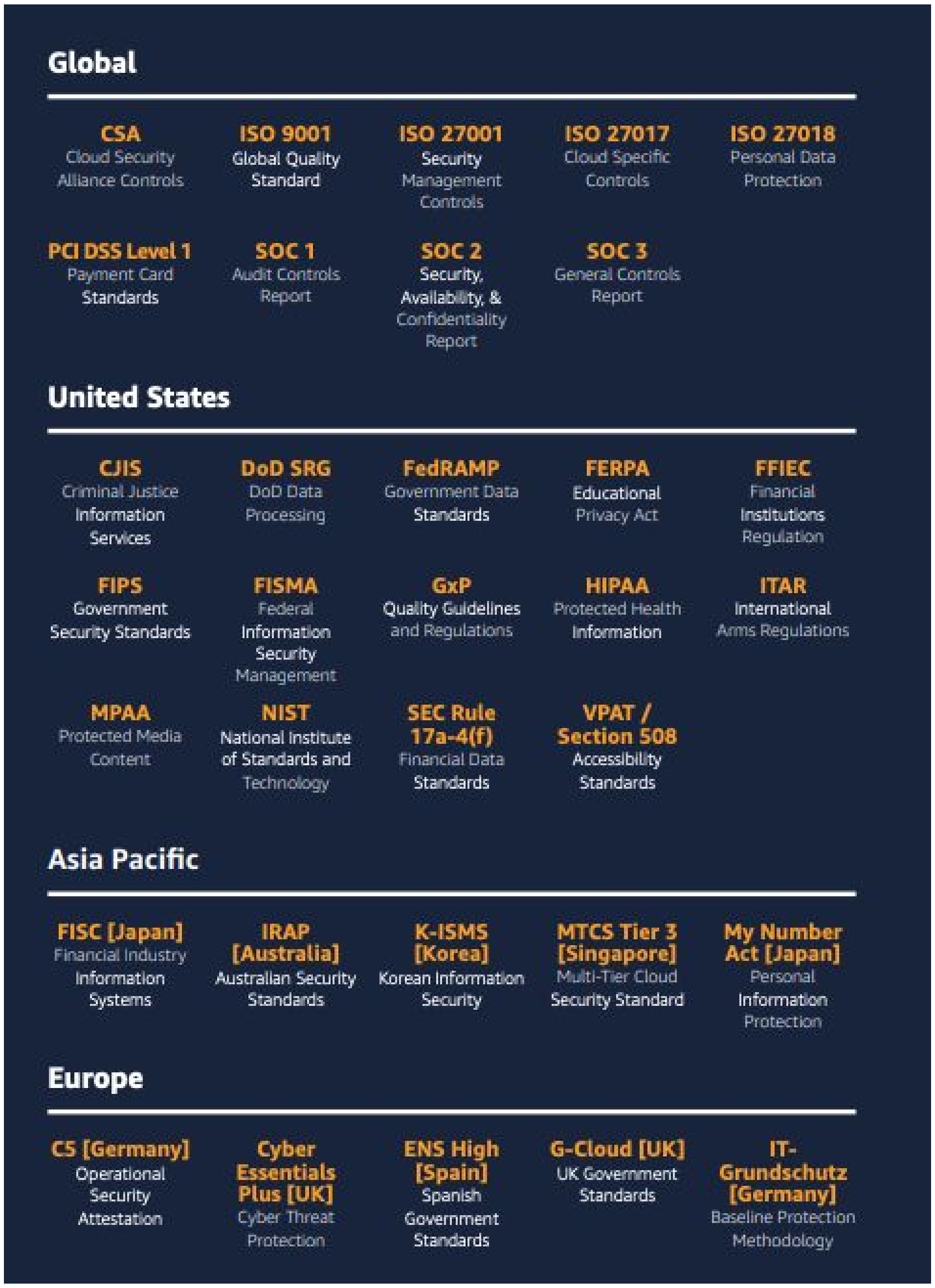
Security Specification

AWS Security Compliance

The AWS physical infrastructure and software systems running above are continuously audited to pass security requirements globally and locally. The subset of security compliances can be found in the following diagram (full and more up-to-date list can be checked in https://aws.amazon.com/compliance/programs/).

The security compliance list is comprehensive and covers most recognised standards in the computer industry, which provide more confidence for anyone building operations on the infrastructure following AWS suggested best practices and design principles. In addition, the AWS platform has been approved for use in UK financial institutions ([02],[03],[07]) following the FCA regulation guidance ([11], [12]).

The FNZ Chain project running on AWS benefits from the many security controls that AWS operates, thus reducing the number of security controls that need to be maintained locally.



Global

CSA Cloud Security Alliance Controls	**ISO 9001** Global Quality Standard	**ISO 27001** Security Management Controls	**ISO 27017** Cloud Specific Controls	**ISO 27018** Personal Data Protection
PCI DSS Level 1 Payment Card Standards	**SOC 1** Audit Controls Report	**SOC 2** Security, Availability, & Confidentiality Report	**SOC 3** General Controls Report	

United States

CJIS Criminal Justice Information Services	**DoD SRG** DoD Data Processing	**FedRAMP** Government Data Standards	**FERPA** Educational Privacy Act	**FFIEC** Financial Institutions Regulation
FIPS Government Security Standards	**FISMA** Federal Information Security Management	**GxP** Quality Guidelines and Regulations	**HIPAA** Protected Health Information	**ITAR** International Arms Regulations
MPAA Protected Media Content	**NIST** National Institute of Standards and Technology	**SEC Rule 17a-4(f)** Financial Data Standards	**VPAT / Section 508** Accessibility Standards	

Asia Pacific

FISC [Japan] Financial Industry Information Systems	**IRAP [Australia]** Australian Security Standards	**K-ISMS [Korea]** Korean Information Security	**MTCS Tier 3 [Singapore]** Multi-Tier Cloud Security Standard	**My Number Act [Japan]** Personal Information Protection

Europe

C5 [Germany] Operational Security Attestation	**Cyber Essentials Plus [UK]** Cyber Threat Protection	**ENS High [Spain]** Spanish Government Standards	**G-Cloud [UK]** UK Government Standards	**IT-Grundschutz [Germany]** Baseline Protection Methodology

(pic source: AWS)

AWS Data Protection

EU Article 29 Working Party has approved AWS Data Processing Agreement AWS DPA which contains “model clauses” – standard provisions approved by the Working Party. This means one can sign the DPA without authorization from data protection authorities and gives additional options regarding which AWS Regions to use to process personal data. AWS is fully compliant with all applicable EU data protection laws ([10]).

Data Classification

Data Encryption

Data at Rest

Data at rest represents any data that persists for any duration. This includes block storage, object storage, databases, archives, and any other storage medium on which data is persisted. Protecting the data at rest reduces the risk of unauthorized access, when encryption and appropriate access controls are implemented.

Currently data at rest encryption in the FNZ Chain project is at device storage level in order to ensure performance:

- **RDS disk and snapshot backups are encrypted using AWS KMS keys**
  Separate individual keys are used for different RDS disk encryption. RDS contains a backup of all the blockchain transactions and further application information (no personal information).

- **EBS disk and snapshot backups are encrypted using AWS KMS keys**
  Fabric nodes are using EBS for local CouchDB storage, thus EBS is also encrypted with KMS key, and differently from organisation to organisation. The storage is the main location for blockchain transactions.

- **S3 buckets are encrypted with AWS KMS keys**
  All S3 buckets are encrypted with AWS KMS, including terraform states, log files and docker images with individual keys separated by content.

Data at Transit

Data in transit is any data that is transmitted from one system to another. It is common best practice to use secure protocols that implement the latest in cryptography standards such as Transport Layer Security (TLS).

AWS services provide HTTPS endpoints using TLS for communication, that is, data is encrypted in transit when communicating with the AWS APIs. Since all the hosts in the FNZ Chain project are running behind firewalls, all public access from the Internet is forbidden by default:

- **ELB + SSL for public access**

When public access is absolutely necessary (Web UI), the ELB is created with the AWS Certificate Manager (ACM) for the SSL certificates. The clients accessing the Web UI through HTTPS protocol for TLS encryption in web-based workloads. SSL is terminated at the ELB.

Data Backup/Replication/Recovery

The data backup, replication, and recovery approach in general helps protect against deletion or destruction of data. It is one of the most important requirements when designing a system. Many AWS services are already designed to smooth data backup/replication/recovery processes in projects. For instance, S3 can create copies of the content that can then be replicated to other locations and AWS accounts for additional protection. Use of RDS to snapshot the database instances then creates copies of the content that can be replicated to other locations and AWS accounts for additional protection; EBS volumes can take snapshots and copied across AWS regions etc.

In FNZChain project, these features are implemented wherever the specific services are used. The components are also selected carefully based on the requirements, and already have replication/recovery built-in features. When solutions exist, these components are also built in the form of high availability clusters to ensure the overall availability.

- **Multi-AZ RDS deployment with snapshot**
  The multi-AZ deployment is for replication and regular snapshots for faster recovery.

- **S3 bucket cross-region replication**
  Enables automatic, asynchronous copying of objects across buckets in different AWS Regions for backups

- **EBS snapshot for faster recovery**

FNZ Chain Data Privacy

FNZ Chain categorises transactional and reference data into three parts: Shared Ownership; Shared between interested parties; Completely own by single party.

FNZ Chain implement private databases also called side DB to persist the private data which solves points 2 and 3, below. This solution is GDPR complaint as you can delete the data when required if the data is managed in a private database. More details on how the private data works is explained in the next section.

Data Store

The ledger is the sequenced record of state transitions for the blockchain application. Each transaction results in a set of asset key-value pairs that are committed to the ledger as it is created, updated, or deleted. The immutable source of truth for v1.0 is appended into the file system of the peer, which also has LevelDB embedded. LevelDB has, by default, a key value database and supports keyed queries, composite key queries, and key range queries. If you also need complex, rich queries, CouchDB supports the basic capabilities of LevelDB, and adds full data-rich queries. With optional support of a document database such as CouchDB, the content is JSON and fully queryable, where the data model is compatible with existing key/value programming model. As a result, the application changes are not required when modeling chaincode data as JSON when utilising CouchDB.

The following describes types of data and storage location.

1) Public Data → CouchDB (Every Organisation will have their own CouchDB server which replicates transactions)

2) Historic Data → LevelDB (This is located on the Peer node)

3) Private Data → Transient Store (This is located on the Peer node)

AWS Virtual Private Cloud (VPC)

Infrastructure protection has traditionally been a key part of information security, and also become an important foundation in AWS design principles. It is any digital business operation's first line of protection as well as defense. A well designed AWS infrastructure together with good scaling and recovery strategy from infrastructure level has played a key role for many successful projects globally using AWS platform. The AWS Cloud infrastructure security consists of the following aspects:

- Infrastructure Access Control Management
- Network and Host-Level Boundaries protection
- Infrastructure Security Maintenance

Infrastructure Access Control Management

Access Management ensuring that only authorized and authenticated users are able to access certain resources and only in a predefined and intended manner. Defined principles, such as users, groups and roles etc., which have limited right to access groups of the infrastructure resources, are the fundamental elements that form the core of authentication and authorization.

The control of any access to the AWS environment is essential to the overall security of AWS cloud infrastructure. Centralized access control should be implemented for easy administration and maintenance purpose. The access control determines not only who or what can have access to a specific system resource, but also the type of actions that can be performed on the resource (read, write, etc.). Most common access methods are used in the AWS platform, including passwords, MFA, cryptographic keys, and certificates.

In AWS, Identity Access Management (IAM) is the key service enables users to securely control access to AWS services and resources. Using IAM you can create and manage AWS users and groups and use permissions to allow and deny permissions to AWS resources. In FNZChain project, IAM is setup from very beginning before applying for any other services. The overall implementation of IAM has strictly followed the following best practices and principles:

- **AWS organization to manage multiple AWS accounts.**
  See above "Application AWS Accounts Setup" section.

- **Protect and guard AWS account root user.**
  Once AWS accounts have been setup, the complicated password for the root user are generated and the root user MFA are also enabled; An IAM user with admin and power user privileges are created and used for admin and deployment purpose: Enabled IAM users billing access in AWS account settings.

- **Created IAM Groups to unify similar access behaviours.**
  The IAM users/roles are created with no privileges at all by default. All the privileges associated with user/role later should be granted based on per request basis. Currently is based on emails, plan to use Jira to track the requests and privileges, for sensitive resource requests, the approval of a manager is required.

- **Enforced MFA for IAM users with AWS console.**
  In addition to enforcing strong password combination and expiry policy for individual IAM users, the MFA is also enabled mandatorily for AWS console access.

- **Manage IAM users/roles with the principle of the least privileges.**
  The IAM users/roles are created with no privileges at all by default. All the privileges associated with user/role later should be based on per request basis. i.e. use Jira to track the requests and privileges, for sensitive resource requests, the approval of a manager is needed.

- **Use of Base User Group**
  The base user group is applied to any of newly created IAM users, to simplify and ease administration and maintenance tasks. The group is combined with the privileges of login AWS IAM console the user itself; change its own password based on password combination policy; enable/resync MFA; manage its own access credentials etc.

- **Create remote access policy**
  Enabled AWS CloudTrail for logging service API calls; ensure AWS cloudWatch logs to meet the log requirements; leverage IAM policies, bucket policies and security groups together to prevent the unauthorized access.

- **Use assumed role on IAM roles.**
  In the event that accessing AWS resources in another account, use assumed role. All instances accessing other AWS services use token based IAM role credentials instead of user credentials. It is particularly useful when you have multiple AWS accounts but wish to have access from a centralised point. However, this is currently not configured for IAM users.

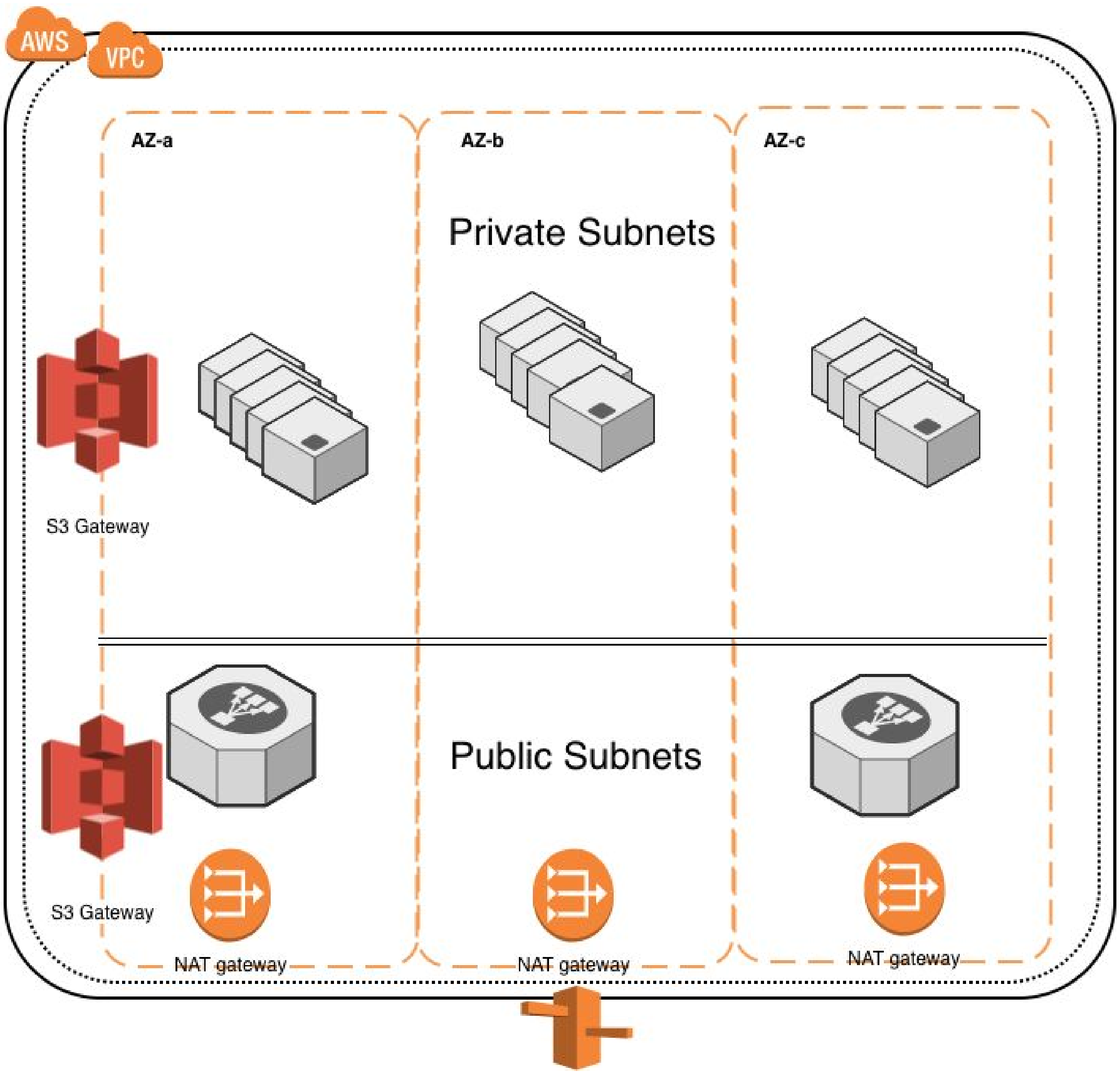
Network and Host-Level Boundaries protection

AWS Virtual Private Cloud (VPC) is essentially a private datacenter belonging to a business which is located on the Internet. It can be accessed remotely but not physically. VPC allows you to define your network topology that spans across an AWS Region with multiple Availability Zones (AZ) using a private IP address range you determine.

Such topology is extremely flexible to configure and manage like a local data centre. For instance, for public traffic, a defined subnet with an associated public route table goes to an internet gateway attached to the VPC. The absence of a route to the internet gateway prevents instances from being directly reachable. A publicly routable subnet by having a route that A subnet can also have a NACL attached to it (stateless firewall). The NACL can also be configured to narrow the scope of traffic allowed. Within a VPC, one can create multiple subnets which are individually located in AZs, such that a higher fault tolerance can be achieved compared to only a single data center's employees.

When a host is launched within a VPC, it has its own security group (stateful firewall). Such firewall is outside of the operating system layer and can be used to define rules for allowed traffic. Furthermore, the relationships between security groups can then be configured to establish the connection between hosts, which allows finer-grained traffic flow control within the VPC.

In FNZ Chain, all the related projects and components are built upon the same VPC standard, which ensures that the same network configurations are expected in production as well as non-production environments. The VPC standards that are implemented in the FNZ Chain project are summarized in the below diagram and the details which follow:



- **VPC setup is created using Terraform with modules across AWS accounts for consistency and rapid duplications.**
  Terraform is used as an Infrastructure as Code tool to persistent the setup and version control and the terraform state of projects/components are uploaded and stored in AWS S3 with encryption.

- **VPC CIDR range are not overlapping.**
  The VPC CIDR range is approved by FNZ infrastructure team to ensure no IP overlapping between AWS VPCs and private data centers and therefore ensure future connectivity.

- **Use private subnets wherever possible and suitable.**

All business operation hosts are located in private subnets not public subnets, so everything is running behind a controlled firewall. When access is necessary, an ELB in public subnets is used to bridge the traffic with https, i.e. a web application. The Security group (SG) of ELB and SG of related VMs are also separate to ensure finer-grained access control.

- **Subnets are crossing all AZs.**
  Both Private and Public subnets span across ALL Availability Zones in an AWS Region (Datacenters) to ensure highest availability and fault tolerance.

- **Make good use of available endpoint and gateways services.**
  S3 endpoint is used for enabling the traffic between EC2 and S3. There are two S3 endpoints for production route tables, one for public subnets, one for private subnets. NAT gateway is used for enabling the traffic from private subnets service to the outside world (i.e. system updates). There are the same number of NAT gateways as private subnets for production private route tables. One for each individual private subnets.

- **VPC peering for deployment.**
  VPC peering is only allowed in terms of deployment pipeline needs, strictly no traffic between production VPC ENV and non-production ENV.

- **Route53 for localised host domain as well as public where needed.**
  Not only used in public host domains, R53 is also established internally within VPC for communication between project components using identical host names;

- **Terraform state (for each VPC with all above features) is created and stored in a server-side encrypted S3 bucket.**

Future work improvements:

- Standardise IAM user management for each VPC using terraform for access control management.

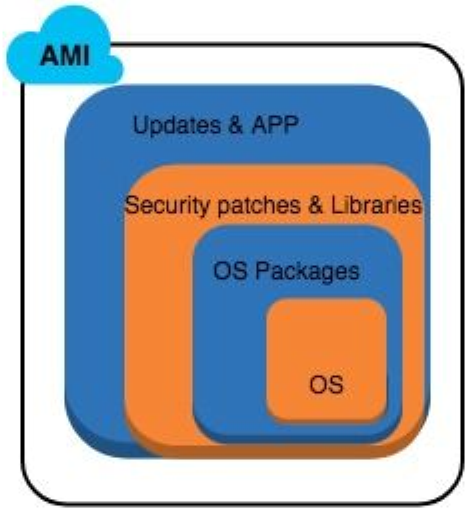
- Enable many more AWS service endpoints when available to relieve the network traffic pressure.

AWS Machine Image (AMI)

An Amazon Machine Image (AMI) is a master image for the creation of virtual servers (known as EC2 instances) in the AWS environment. An AMI is essentially a master image containing the information required to launch ‘instances’, or virtual servers in the AWS.

AMI provide a template for the root volume required to launch a particular instance with particular OS installed. It typically include the operating systems as well as an application server and applications when needed. It also includes launch permissions that restrict the ability to launch instances from, which enables any permitted AWS accounts launch a same copy of instance as the AMI owner. Finally, a block device mapping specifies the volumes to attach to the instance once it is launched.

FNZ Chain projects and components are all based on the self-built AMI, the contents of which are illustrated in the following picture and the detail explained below:



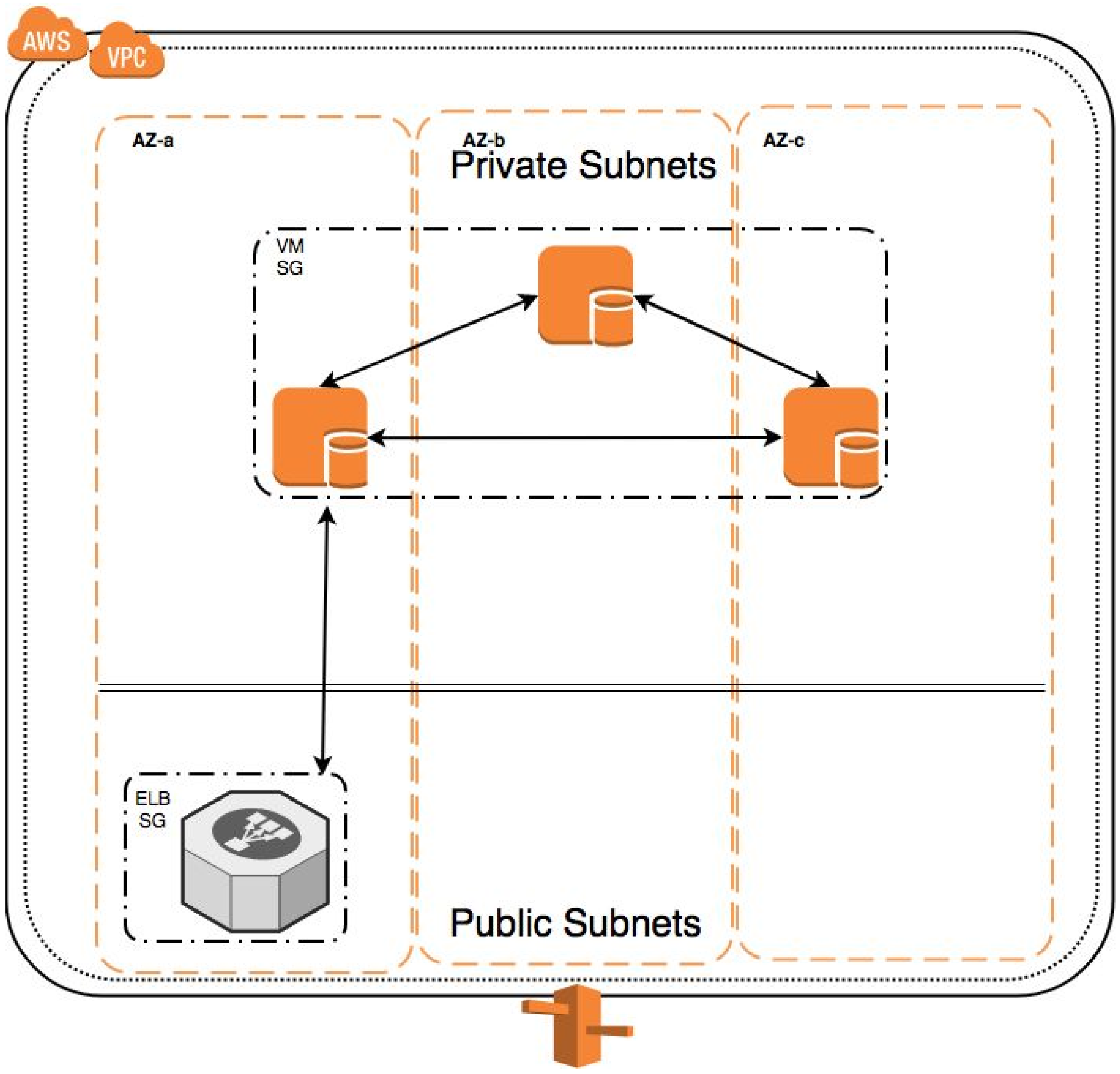
- **CentOS 7 + SELinux**
  The FNZ Chain AMI is originated for official centos 7 AMI (https://wiki.centos.org/Cloud/AWS) includes all security patches at the time of building, and the SELinux module (https://en.wikipedia.org/wiki/Security-Enhanced_Linux) enforced.

- **Keep instance updated.**
  At the time of launching a new instance based on the FNZ Chain AMI, the instance is applied with security updates again to make sure the up-to-date state.

- **Rebuild FNZ Chain AMI periodically.**
  The FNZ Chain AMI is rebuilt periodically to capture relevant security updates and apply them to the latest projects and components. It reduces the overall time in automating the instances and enables faster recovery and deployments.

FNZ Chain Application in AWS

All the projects and components of FNZ Chain follow the same design principles which are also kept in-line with overall AWS Well-Architected Framework suggested design patterns. Those principles and design patterns are describe in the following:



- **Projects components are created using Terraform with modules among VPC ENV for consistency and rapid duplications.**
  Terraform is used as an Infrastructure as Code tool to persistent the setup and version control and the terraform state of projects/components are uploaded and stored in AWS S3 with encryption. For individual components/project, the terraform state is initialized and stored in server-side encrypted S3 bucket.

- **All business operation are located in private subnets not public subnets, so everything is running behind a controlled firewall.**

- **All applications are build based on a base Centos AMI, which is updated and patched at the time of customising and SELinux module are enforced.**

All components and projects are then based on the base AMI, updated and patched again at the time of launch. All components are organised individually in a seperate folder, unless it makes sense to put project components together, and treat them as a whole (for one button deployment).

- **Project attribute options in terraform (can be turned on/0ff on-demand):**
    - is chef bootstrapping is needed? (default is true)
    - is VM spot or on-demand instance type ? (default to on-demand)
    - is EBS volume needed to be attached to VM? new or exist? size? encryption KMS needed? (default EBS is not needed)
    - is component needs new SG or IAM role? (default is true)
    - is EIP is needed is the VM need to be accessible from internet? (default is false)
    - is ELB/ALB is needed for the component? single or individual ELB/ALB? (default is false**)**

- **Wherever public access is necessary, an ELB in public subnets are used to bridge the traffic with https, i.e. a web application.**
  The SG of ELB and SG of related VMs are also separate to ensure more access control. The security group of single VM or the cluster has all TCP open for itself, the VMs within the same cluster/security group are able to communicate with each other. The ingress of the ELB/ALB security group only has specific port opened towards trusted IP (i.e. London office wireless IP), unless it is 80 or 443 open to the world. The egress of ELB SG is strictly limited to the specific port towards the VM SG;

- **Only basic cloudwatch monitoring and VM health check in place, DataDog monitoring for specific applications.**

Future work improvements

- VPN access for remote work.
  To reduce the whitelisted IPs from controlled bastion host.

- Comprehensive monitoring solution with prometheus together with alarming strategies to reach the appropriate line of support.

FNZ Chain Infrastructure Security Maintenance

The infrastructure security is not a one man job nor a one-off job, it depends on a whole team's efforts (both developers and operations) and demands continuous commitment, regular maintenance and even more frequent governance. This is particularly true in a fast and constantly evolving AWS platform. There are many new services and features in the AWS platform being added everyday, requiring participants keep abreast of all changes, apply and implement new services and features on existing infrastructure to improve architecture patterns and security.

In the FNZ Chain project, all the deployment and maintenance including setups and configuration are automated using Infrastructure as Code (IaC) practice and version controlled in bitbucket. The journey to the cloud includes securing each and every resource that is migrated and built over time within a governance model that envelops people, process and technology. Whatever the security practices are today, it can be re-created or improved with the IaC base.

- **User group access control on bitbucket.**
  Users are separated in groups to have access control down to individual level. The users are also authenticated using fnzchain.com google account, where MFA is enforced.

- **FNZChain AMI is updated every 3 months.**
  The FNZCHain AMI base CentOS image is updated and refreshed with latest security patches for any new/replacement deployments.

- **Terraform**
  All the projects and components are created using terraform for faster deployment replication and.

- **Amazon Inspector**
  Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. It contains a knowledge base of hundreds of rules that are mapped to common security best practices and vulnerability definitions and updated by AWS frequently. Keeping abreast of such automated services frequently helps those maintaining the system to spot and identify a volubility in the system very quickly.

Future work improvements

- AWS Systems Manager Patch Manager.
  This will ensure running instances are properly patched and updated with the most recent security measures constantly, if there are any.

- Define the proper maintenance and auditing procedures for ongoing security checks.

- Use AWS Config to maintain an AWS resource inventory for auditing purpose.

Well-Architected Framework Pillar Checklist

Security Pillar

Well-Architected Framework	FNZChain	Implementation Description
Identity and access management		
Protecting AWS credentials	Yes (partially)	AWS IAM users with MFA enabled for access, passwords policy, IAM role; TBD: access key rotation, define processes for VPN and VM access.
Fine-grained authorization	yes	AWS IAM users and roles with least privileges down to individual resources level; TBD: IAM user assume role
Detective controls		
Capture and analyze logs	Yes (to discuss with security)	AWS Cloudwatch logs and datadog log for all services (project components), both in application level, AWS Cloudwatch for VM health monitoring; TBD: build own ELK solutions hosted in AWS
Integrate Auditing Controls with Notification and Workflow	yes	AWS VPC flow logs, cloudtrail and ELB logs are pushed and stored in S3, AWS Cloudwatch alarms sent to slack, email; TBD: integrate Cloudwatch alert with pagerduty; the application alert notification integrate with slack, email and pagerduty.
Infrastructure protection		
Protecting network and host-level boundaries	Yes (needs clarification with info sec)	AWS VPC built with predefined AWS security best practices and policies; TBD: continuous improvements when new security features apply.
System security configuration and maintenance	yes	Infrastructure as Code practice in version control, which can rebuild and replicate quickly; TBD: continuous improvements on IaC.

Enforcing service-level protection	yes	Employed AWS VPC endpoints and gateways services; TBD, applying new endpoints and gateway whenever needed and possible. IAM policies for service -> service interaction.
Data protection		
Data classification	no	
Encryption/tokenization	no?	
Protecting data at rest	yes	Local data stored in CouchDB is in a separate non-root EBS volume and encrypted using KMS (KMS keys are also different on individual components basis); TBD: applying Hashicorp vault or CloudHSM with customised keys.
Protecting data in transit	yes	ELB+SSL (ACM)
Data backup/replication/recovery	yes	AWS RDS multi-AZ deployment for data backups, the fabric nodes is a backup for each other, a new nodes will sync among existing CouchDB data; TBD: enable a fabric node in different region in another AWS account/VPC
Incident response		
Clean Room	no	TBD: a clearly defined escalation path for the support in the event of alarm.

Reliability Pillar

Well-Architected Framework	FNZChain	Implementation Description
Limit Management		
	yes	Understand and be aware of various physical limitations within AWS, apply limitation changes on demand

Networking		
VPCs	yes	Multiple VPCs in various AWS accounts using non ip-overlapping CIDR blocks (non-conflicting with current structure)
AZs	yes	Use all AZs in one region and architect the project faulty tolerant across all AZs
Redundant VPNs	no	Plan to implement VPN in place to allow remote access
Prevent DDos attacks	no	Not needed as only internal use.
Implementations for Availability Goals		
Adapt to changes in demand	no	TBD: Autoscaling not enabled for shadow register. ELB for public facing web tier; resizing Multi-AZ RDS.
Monitoring	yes	Cloudwatch health checks at all VM hosts on CPU and MEM, alerts sent when configured alarms are tripped; alerting on all VM failures.
Deploying changes	yes	Automated deploy using terraform and chef for fast replication and deployment, TBD: blue/green deployment for automated rollback.
Backups	yes	Automated backups via multi-AZ RDS and fabric distributed ledger.
Implementing resiliency	yes	Implemented fault isolation zones for the application, RDS is Multi-AZ.
Testing resiliency	no	TBD: automated testing deployment pipeline
Disaster recovery	yes	Encrypted backups via RDS to same AWS Region; fabric nodes are replicated and the backup for each other.
Availability		99.95% (chain web is single instance in one AZ),

(https://news.ycombinator.com/item?id=2470298) (https://www.reddit.com/r/aws/comments/8h9j92/what_is_the_availability_of_availability_zones/)		TBD, redesign the web with multiple instances ideally with elb and autoscaling

Reference

[01] AWS Cloud Security: https://aws.amazon.com/security/
[02] AWS Security Compliance: https://aws.amazon.com/compliance/programs/
[03] AWS Compliance Center for financial services: https://www.atlas.aws/
(https://aws.amazon.com/blogs/security/aws-compliance-center-financial-services/
https://www.atlas.aws/asset/download/14189)
[04] AWS Well Architected Review Overview:
https://d0.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf
[05] AWS Well Architected Review Security Pillar:
https://d0.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf
[06] AWS Well Architected Review Reliability Pillar:
https://d0.awsstatic.com/whitepapers/architecture/AWS-Reliability-Pillar.pdf
[07] Capital One in AWS: https://aws.amazon.com/solutions/case-studies/capital-one/
[08] AWS overview of security processes:
https://aws.amazon.com/whitepapers/overview-of-security-processes/
[09] Centos7 AWS AMI: https://wiki.centos.org/Cloud/AWS
[10] AWS data protection: https://aws.amazon.com/compliance/eu-data-protection/
[11] FCA FG 16/5 Guidance for firms outsourcing to the ‘cloud’ and other third-party IT services: https://www.fca.org.uk/publication/finalised-guidance/fg16-5.pdf
[12] FCA European Banking Authority Recommendations on outsourcing to cloud service providers:
https://www.eba.europa.eu/documents/10180/2170121/Final%2Bdraft%2BRecommendations%2Bon%2BCloud%2BOutsourcing%2B(EBA-Rec-2017-03).pdf

Appendix

AWS Services Availability (not reliability)

AWS Service	**Component**	**Availability Design Goal**
Amazon API Gateway	Control Plane	99.950%
	Data Plane	99.990%
Amazon EC2	Control Plane	99.950%
	Single AZ Data Plane	99.950%
	Multi AZ Data Plane	99.990%
Amazon RDS	Control Plane	99.950%
	Single AZ Data Plane	99.950%
	Multi AZ Data Plane	99.990%
Amazon Route53	Control Plane	99.950%
	Data Plane (query resolution)	100.000%
AWS Identity & Access Management	Control Plane	99.900%
	Data Plane (authentication)	99.995%
Key Management System (KMS)	Control Plane	99.990%
	Data Plane	99.995%
Amazon S3	Service (Standard)	99.990%