REAN Cloud Platform Service

Security and Technical Architecture

Document No #
In response to:
Task Order REAN#
CLI#

Prepared for:
REAN Cloud LLC
ATTN: XXX XXX
2201 Cooperative Way
Herndon, VA 20171

In Support of:
REAN Cloud LLC

Prepared by:
Samal Dimdung
Enterprise Solutions Architect

Approvals

XXX XXX
Manager, Solutions Architecture

Document History

Version	Description	Date
1.0	Initial Release	11/18/2017

1 Introduction

NOTE: The REAN Cloud Platform Service Security and Technical Architecture is a “living” document and will be updated throughout the of RCPS (REAN Cloud Platform Service).

1.1 Owner, Sponsor, Security

	Name/Title	Contact Info
Project Owner	ABC XXX	xyz@reancloud.com
Sponsor		
Security		

1.2 Overview

RCPS is a platform for secure hosting of REAN Cloud’s applications on a public cloud. It allows applications to incrementally progress towards “push-button deployment” through automation and integration of “common services” as they realized the benefits of Development Operations (DevOps) principles and practices. RCPS primarily comprise specific tailoring or design constraints applied on Amazon Web Services (AWS), East Cloud services, and a set of tools and procedures for deploying code changes across the stack. RCPS will leverage the at AWS cloud services to support multiple tenants and multiple applications for each tenant with layered security boundaries. The architecture is predicated on a set of Virtual Private Clouds (VPCs) that isolates the production development, stage, security and management environments. The Management and Security VPCs are shared environments that will be used across all tenants to enforce policies and manage and monitor environments.

1.3 Purpose

The purpose of this document is to provide a description of the high-level architectural elements of the RCPS Platform. This document will be supported by two other document that elaborate the RCPS Platform architecture further. The first supporting document is the RCPS Platform Design document, which provides details on "How" the RCPS Platform works. The second supporting document is the RCPS Platform ConOps document, which address the process and procedures that describe "How" to on-board and incrementally transform RCPS applications to realized DevOps benefits.

1.4 Reference

1. Amazon Web Services (service description on Federal Cloud Computing Information Web Site), December 4, 2014 (on http://cloud.cio.gov/fedram/amazon)
2. RCPS Identity Access Management Plan

2 Architecture Drivers

2.1 Business Driver

REAN Cloud LLC intends to utilize the RCPS Platform as a Service (PaaS) to help individual Mobile/Marketing application team a realized the long-term RCPS mission of savings, efficiency, and service, along with priorities. The RCPS Platform will service the following specific business goals:

- **Cost Saving and Avoidance**
    - Lower Total Cost of Ownership (TCO): Achieve lower application TCO during and after transformation to DevOps practices.
    - Build/reuse Assets: Reuse platform assets to lower the cost of migration and transformation to DevOps.
    - Cost Management: Provide transparency and visibility into platform costs and usage, including chargeback capabilities, forecasting, appropriate use of reserved instances, and shutdown of on-demand instances when not in use.
- **Agility**
    - Frequent deliveries: Achieve higher frequency of deliveries
    - High throughput of deliveries: Achieve higher volume of deliveries per unit, over time.

- Reduce obsolescence: Continuously refresh technology at all levels of the platform from hardware to operating system to middleware and thereby reduce the cost of technology obsolescence.
- Extend to other tenants: Provide multi-tenancy and platform management to allow use of the platform by all XYZ organization, external tangent if any.

- **Security**
  - Higher Availability: Utilize the elasticity of the cloud to achieve higher availability.
  - Continuous Monitoring: reduce risk through new cloud security standards and procedures for continuous monitoring.
  - Platform Homogeneity: Achieve homogeneity in platform and its components to enable quick response to security incidents.
- **Continuous Service Improvement**
  - Real-time visibility: Achieve real-time visibility of trends over time of cost and performance of hosted business services at multiple levels (tenant's, application, and Application Programming Interface (API).
  - Faster platform adoption: Enable faster platform adoption with standards, toolkits, blueprints, reference architectures, and standard operating procedures.
  - Self-servicing: Enable self-service provisioning and provide instant continuous automated feedback for a rendered platform service and performance against service levels.
  - Hide complexity and extend sophisticated features: Extend sophisticated platform service as a standard to even the smallest of applications, such as multi-region/zone availability, adaptive resource usage, and plush-button deliveries.
  - Increase product quality: Embed quality checks and assurance at every step from code to deployment using automation and workflow for reviews.

2.2 Technical Drivers

2.2.1 Oppuritinities

- New automation tools for Continuous Integration and Continuous Delivery
- Software defined environments through cloud and infrastructure as code (IaaS)
- Ability to define lightweight (micro) ephemeral applications containers
- Rapidly maturing orchestration and management platform across the development lifecycle.
- Maturing Cybersecurity framework standards, services, and tools for the cloud.

2.2.2 Constraints

- Legacy technologies have not equivalence in the cloud
- Application code may have design assumptions that need to be refactored to run on the cloud.
- Many existing tools and platforms overlap
- Heavy application coupling and data dependencies have occurred over time
- Existing implementation and deployment model may require re-factoring to meet new promotion processes and policies.

2.2.3 Architectural Drivers

By combining the technical opportunities and constraints with business drivers, we have arrived at the following quality attributes stories or on-functional requirements. The quality attribute stories are basically statement that have a context for a stimulus and its source; the affected environment and their artifacts; and the platform's response and its measure. Architectural driver were selected from quality attributes stories, representing high important to the success of the platform and a high degree of difficulty to achieve them. All architectural decision documented in the rest of the document were heavily influenced by those architecture drivers over others providing the basis for rationale.

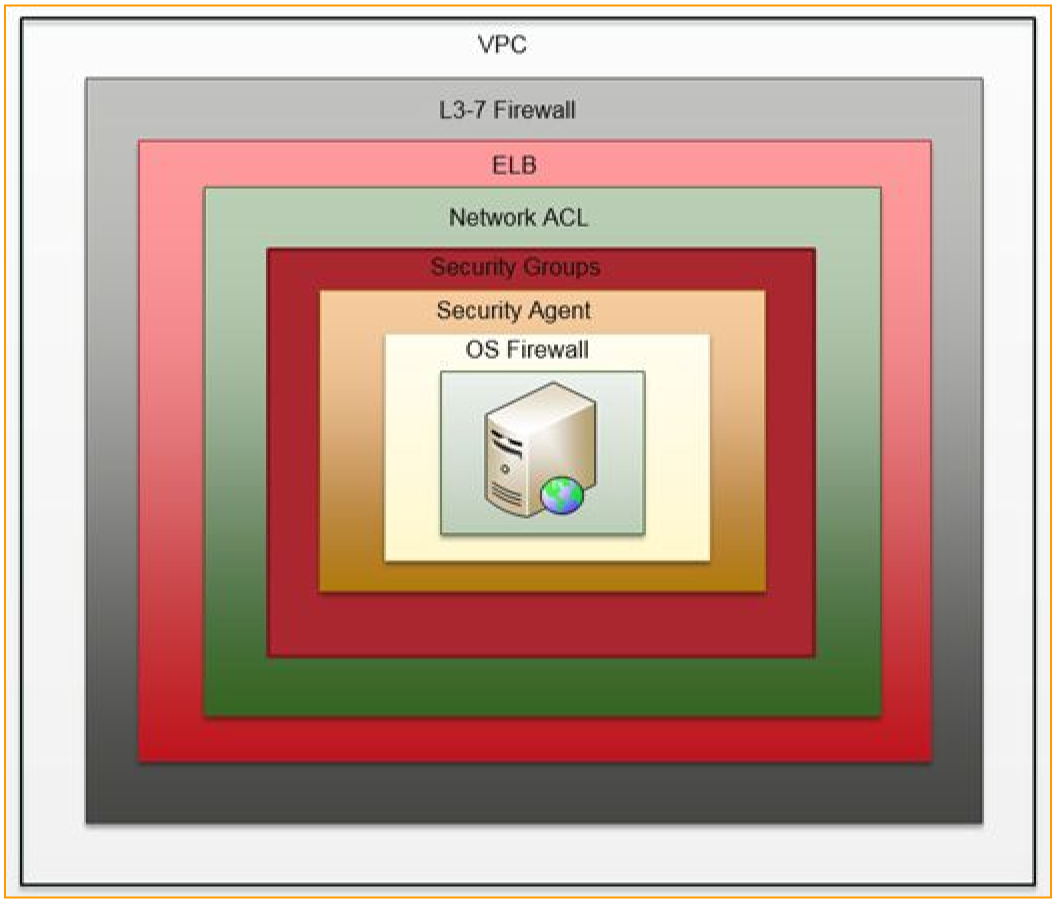
2.2.4 Utility Tree View

2.2.5 Design Principles

3 Platform Concept

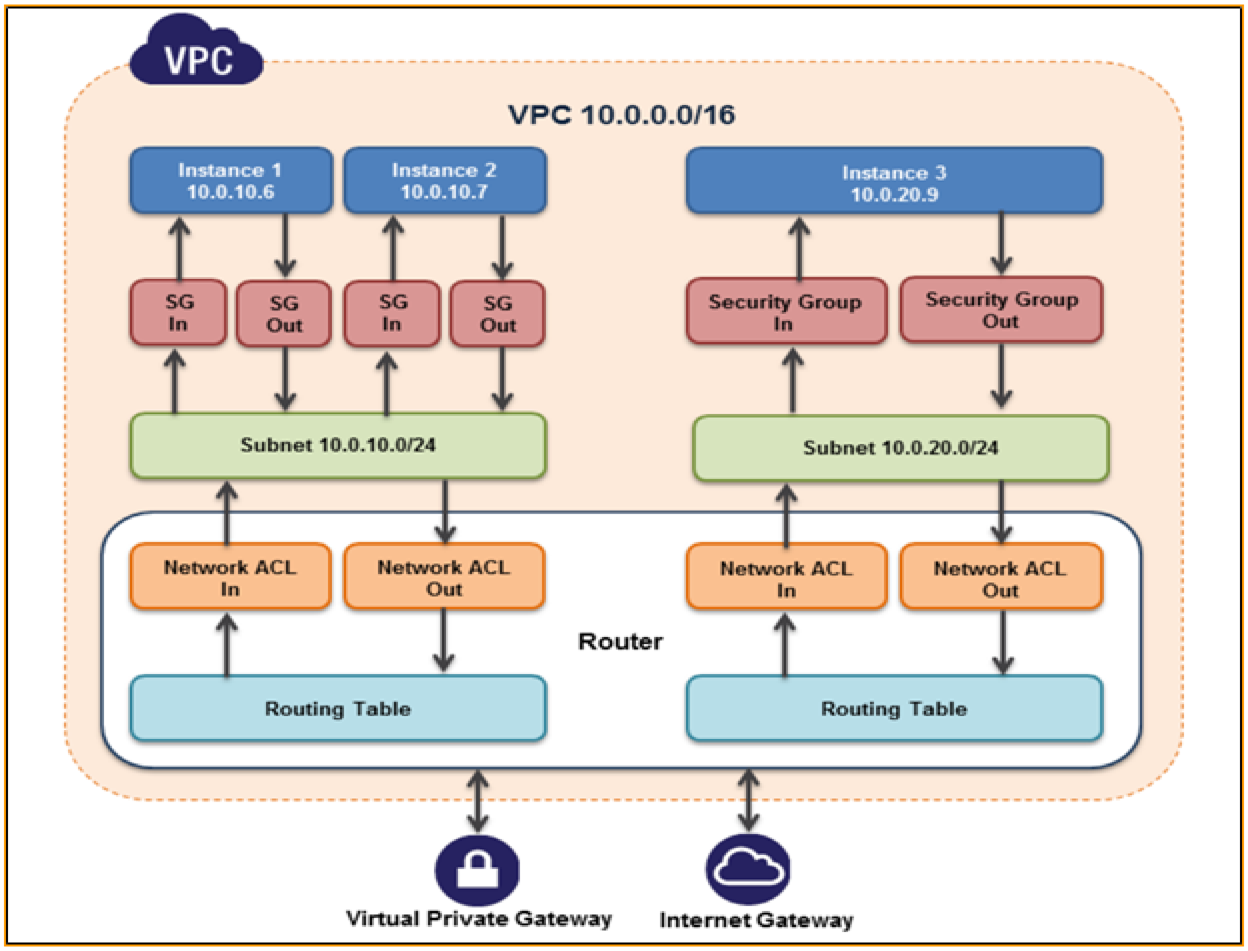
3.1 AWS Boundaries - Defense in Depth

The VPC, depicted in Figure 3 – 1 represents a logically isolated network. It is one in several security control layers that every packages must pass through before reaching the actual application or its data. First, there is the VPC layer, which require all traffic, inbound and outbound, to pass through the Internet Gateway (IGW). All traffic entering through the IGW is routed through an L3/7 firewall, which is managed by XYZ Security. Network Access Control Lists (NACLs) and Security Groups are used to further filter traffic that passes through the L3 firewall layer. Traffic then must pass through the Security Agent automatically installed on every instances. Finally, an OS level firewall is an available for each application. All outbound response traffic must pass through the same security layers before reaching the internet. That's way, all the infrastructure will be secure and safe.



3.2 AWS Communication Pathways

All traffic between Instances will also go through similar controls, as mentioned in section 3.1. All illustration of the communication pathways between instances and with the internet is depicted in Figure below.



As referenced in Amazon's Security Whitepaper
http://awsmedia.s3.amazonaws.com/pdf/aws_security_whitepaper.pdf
For example, when Instance 1 needs to communicate with Instances 3, it must first pass through Instance 1's Outbound Security Group rules, and then its subnet outbound Network Access Control List (NACL). It will then go through the subnet routing table into Instance 3's subnets routing table, Inbound NACL, before passing through Instance's Security Group rules and finally passing into Instance 3's network interface. This means that even within a single VPC, communication between two instances can go through fours or more layers of security to communicate with each other.

3.3 AWS Elasticity and Security

3.4 AWS Networking VPC

The NetOps/Networking team will provide blocks of IP address that can be used within AWS VPCs to make sure there is no overlaps for on-premise to Cloud or Cloud one environment to another environment like SecOps, Management, Development, Test/Stage and Production.

In typical AWS Cloud Environment, there will be two categories of VPCs:

- Platform
- Tenant

Platform VPCs hosts system that are used to provide platform services across multiple tenants. There are four platform VPCs:

- Management,
- Security,
- Identity Management and
- Network Services.

Tenant VPCs are used to host system that are specific to each tenant. For each tenant, there will be up to six VPCs:

- Lab,
- Dev,
- Test,
- Prod, and
- Tenant management.

VPCs are optional depending on the requirements of the tenant.
Typically Enterprise level IP (CIDR) range allocation will be as below:

- 10.152.0.0/13
    - 10.152.128.0/17 - Platform Environment
        - 10.152.128.0/20 - Management VPC/Env
        - 10.152.144.0/20 - Security VPC/Env
        - 10.152.160.0/20 - Identity Management VPC/Env
        - 10.152.176.0/20 - Network Service VPC/Env
        - 10.152.192.0/20
        - 10.152.208.0/20
        - 10.152.224.0/20
        - 10.152.240.0/20

10.153.0.0/16	- Tenant Environment (Lab, Dev, Test, Prod, and mgmt)
10.154.0.0/16	- Future Tenant
10.155.0.0/16	- Future Tenant
10.156.0.0/16	- Future Tenant
10.157.0.0/16	- Future Tenant
10.158.0.0/16	- Future Tenant

4 Platform Topology

4.1 AMI's

4.2 Tier-Level Topology

4.3 Multi-Tier Topology

4.4 Tenant-Level Topology

4.5 Multi-Tenant Topology