appviewx

SaaS Architecture Guide

Version: 2022.1.0

Copyright AppViewX, Inc.

Trademarks

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface

Revision History

Revision	Description	Date
2.0	Updated release of document for release 2022.1.0 FP2 Beta	November 2022
1.0	Initial release of document for Release 2022.1.0	June 2022

About the Documentation

This guide will walk you through the architecture used by AppViewX to implement SaaS. It covers the multi-tenant architecture, network architecture and the cluster architecture. Information with respect to scaling of clusters, DB isolation for each tenant and high availability of AppViewX with the help of this architecture has been touched upon in this guide.

Audience

The guide is intended for SRE (Site Reliability Engineers) and MSP (Managed Service Providers) to enable them to manage SaaS accounts and infrastructure across multiple regions.

Documentation Conventions

This section defines the Notice icon and text convention used in this guide.

Text Conventions

The following text conventions are used in this document:

Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for whichyou supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text thatappears on the screen, or text that you enter.

Chapter 1: AppViewX SaaS: Overview

Key Highlights of AppViewX Software as a Service

The AppViewX Security Automation and Orchestration Platform is a centralized control plane to automate tasks, orchestrate workflows and gain visibility to manage identities at scale, reduce security and compliance risk and ensure secure application availability

The AppViewX SaaS platform offers the following three products:

- CERT+ SaaS, which lets you:
  - Discover, monitor, analyze, orchestrate and fully automate certificate lifecycle management and key management solutions.
  - Make a shift from reactive mode and be more proactive as you get a complete view of your entire certificate infrastructure.
  - Manage certificates as a service with pre-built integrations and extensible APIs that plugin to your enterprise applications, web servers, microservices, and multi-cloud environments.
  - Analyze certificates for crypto standards like key size, cipher strength, and allowed protocol versions.
  - Setup policies for enforcing high crypto standards.
  - Update certificates as per new policies.
  - Provision certificates for devices and applications.
  - Save resources, time, and effort of installation and maintenance.

  For details, refer the CERT+ SaaS User Guide
- ADC+ SaaS, which lets you
  - Efficiently distribute network load or client requests across servers.
  - Send requests to the available servers, ensuring high application availability.
  - Scale the number of servers (up or down) based on the traffic.

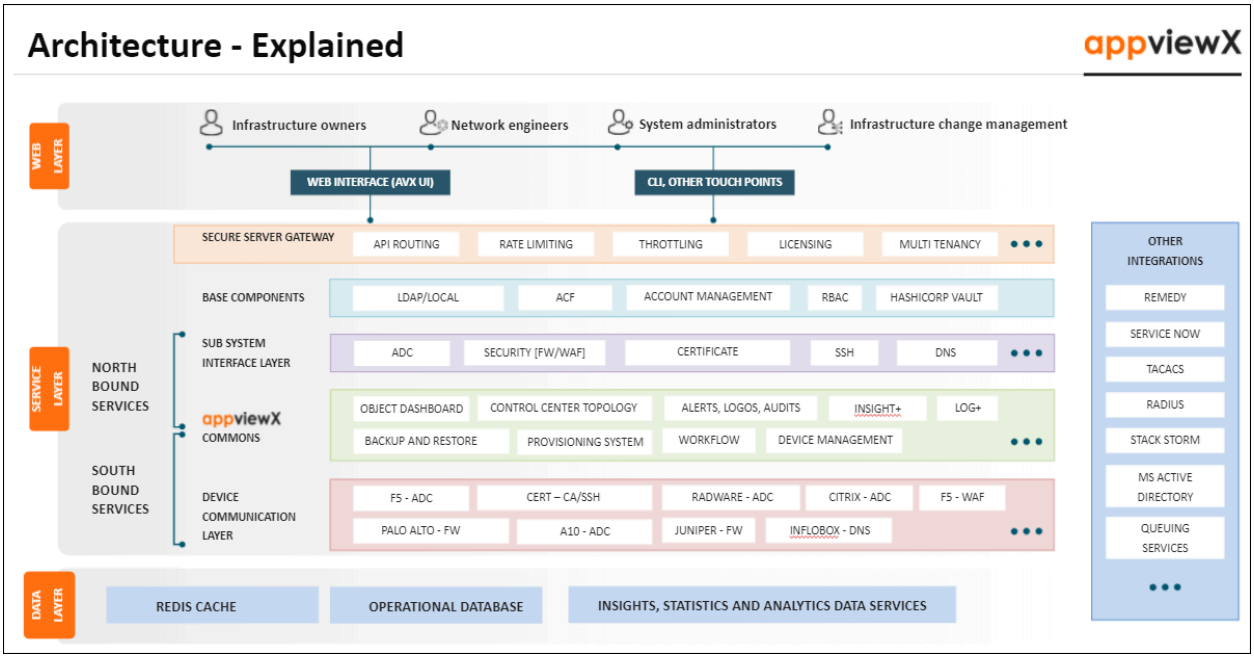
  For details, refer the ADC+ SaaS User Guide
- PKI+, which lets you:

- Create root CAs and subordinate CAs and enroll them to the AppViewX PKIaaS certificate authority.
- Onboard custodians to add root CAs and subordinate CAs to the PKI+ system.
- Manage custodians for approving PKI+-related actions.

For details, refer the PKI+ User Guide.

Chapter 2: AppViewX Architecture

AppViewX is designed based on microservice architecture and its deployed on Kubernetes, an open-source platform for deploying and managing containers. The microservice architecture of AppViewX makes it easier to move to containerized workloads and the containers being orchestrated using Kuberneters. Kubernetes provides container runtime, orchestration, self-healing mechanisms, service discovery and load balancing and its used for the deployment, scaling, management, and composition of application containers across clusters.



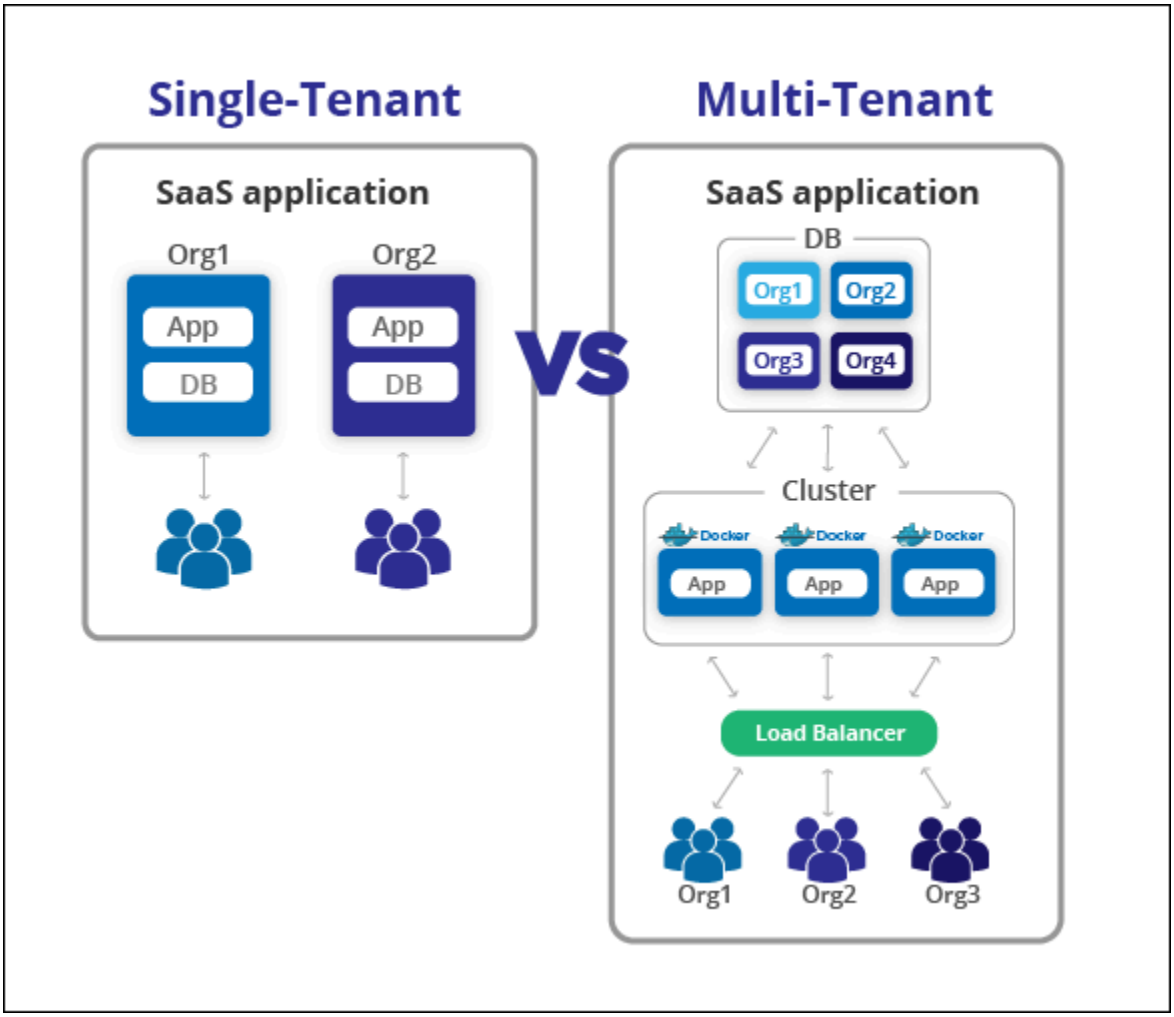
AppViewX Architecture

Chapter 3: Multi-Tenancy Architecture

In a multi-tenancy architecture, a single instance of the software serves multiple accounts/customers. In this setup, the same resources -compute, networking and storage - are shared on the cloud among tenants.

In this ecosystem, a single environment can serve multiple tenants utilising a scalable, available, and resilient architecture. The underlying infrastructure is completely shared, logically isolated, and with fully centralised services.

AppViewX multi-tenant architecture is enabled by a shared compute cluster or a workload cluster where the workloads run and a database cluster where the actual tenant isolation happens by allocating a dedicated schema for each and every tenant. The diagram below depicts the multi-tenant implementation.



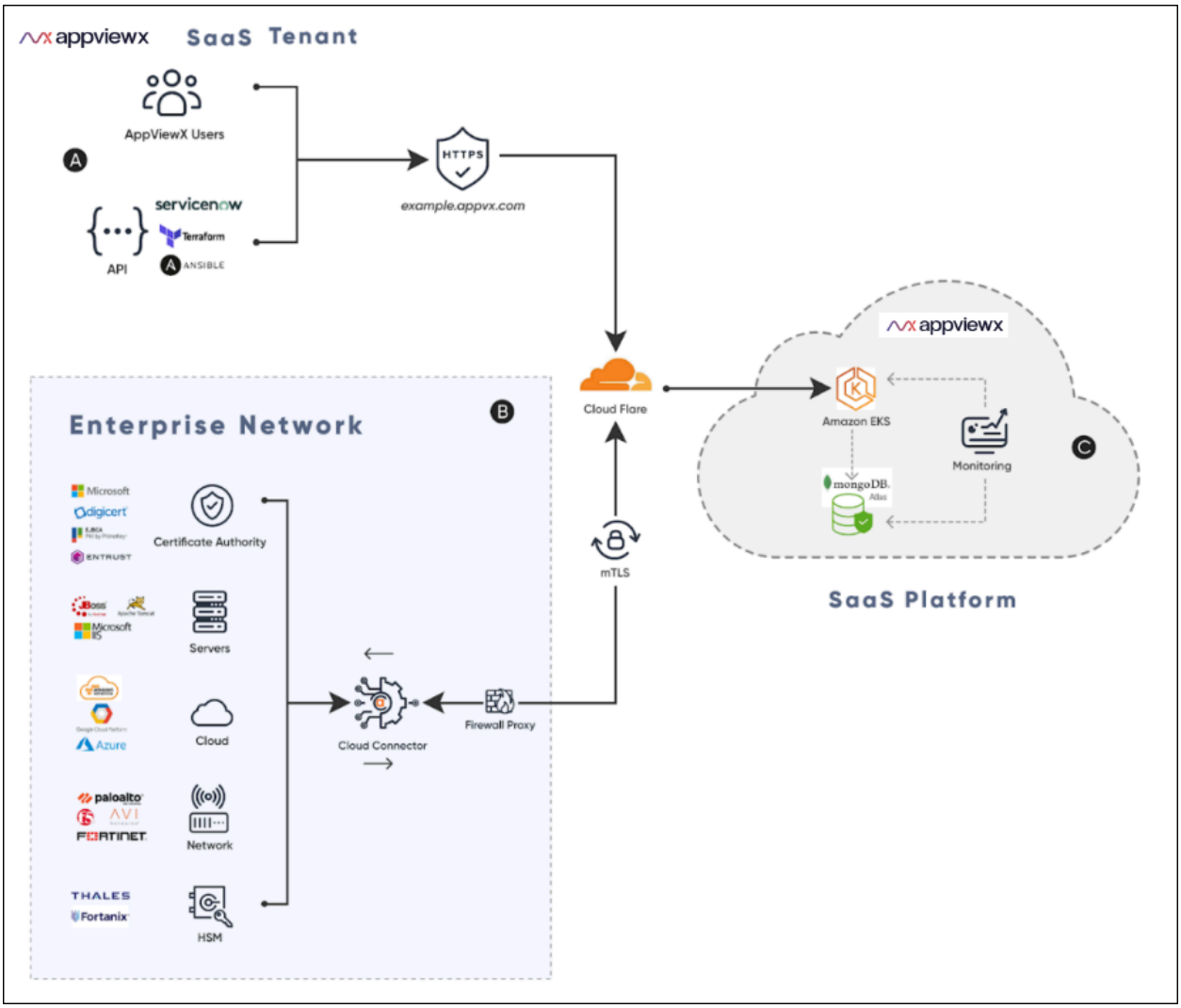
Multi-Tenant Architecture

Chapter 4: SaaS Deployment Architecture

SaaS Deployment Architecture

The AppViewX SaaS deployment architecture is a cloud-based deployment with the following benefits.

- Lower cost of ownership (TCO), significantly reduced maintenance.
- Guaranteed availability (SLA), and enhanced data security
- Faster release cycles and upgrades to access new offerings.
- Avoid installation of the entire AppViewX infrastructure in the tenant network.



AppViewX Multi-Tenancy Architecture

At a high level, the SaaS deployment architecture consists of:

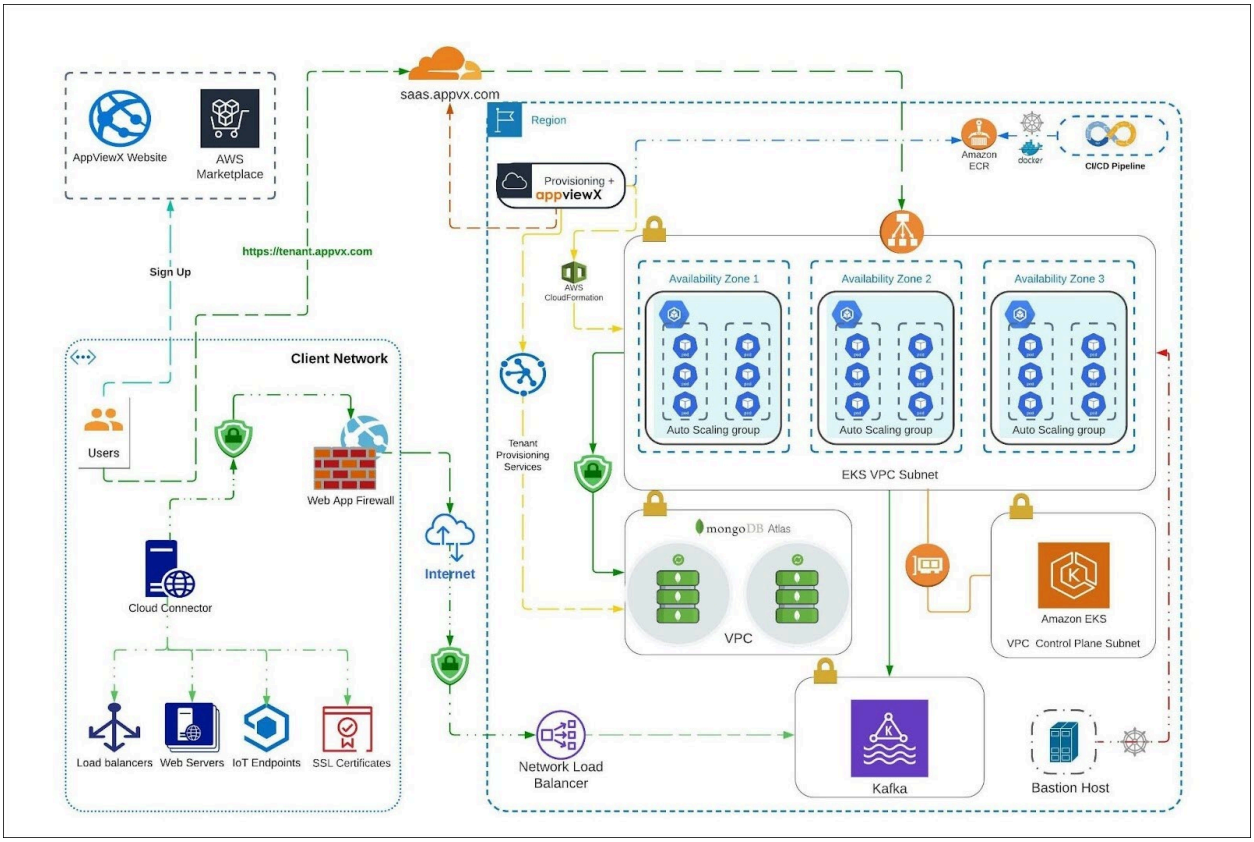
- A public access to the AppViewX SaaS products' tenant secured via https.
- AppViewX Cloud Connector (a lightweight proxy) deployment that enables connectivity between the SaaS platform to the Enterprise network thereby ensuring faster value realisation of critical Certificate Lifecycle Management (CLM) functions such as Discovery, visibility, Automation and self-servicing of SSL / TLS certificates from the tenants infrastructure.
- AppViewX SaaS platform that enables the server-side components such as database, compute, monitoring and tenant provisioning (which includes install and upgrades).

Architecture Components Overview

The AppViewX SaaS platform is enabled with the help of Provisioning, Compute, Database, Monitoring clusters and an AppViewX Cloud Connector.

The key tenets of the platform include:

1. Provisioning Cluster
2. Compute Cluster
3. Database Cluster
4. Monitoring Cluster
5. AppViewX Cloud Connector



Deployment Architecture

- Provisioning Cluster (SaaS Management Portal)
- Compute Cluster
- Database Cluster

- Monitoring Cluster
- The AppViewX Cloud Connector

Provisioning Cluster (SaaS Management Portal)

Provisioning Cluster (aka SaaS Management portal) is an SRE cluster used to orchestrate the compute and database instances that powers the AppViewX SaaS. The cluster provides key capabilities like management and visibility into tenants and clusters which includes licensing, upgrading the SaaS tenants and so on, from a Single pane of glass. The granular features and capabilities of the portal are explained below.

- The Provisioning cluster is an internal AppViewX platform for SaaS lifecycle management which can be deployed cross zone or cross region for high availability.

> **Note:** The cluster can be deployed in a dedicated AWS account (not necessary to be deployed in the AWS account where the actual compute cluster is deployed).

- Key features of the provisioning cluster include:
  1. Tenant Life Cycle Management - Onboarding, Offboarding, Licence upgrade.
  2. Cluster Management - Create , Delete , Modify, Upgrade Compute clusters
  3. AppViewX SaaS Life Cycle Management - Install AppViewX on Compute, Upgrade AppViewX (via Canary upgrades), Install Infrastructure components (Istio , ELK etc.,)
- The Provisioning Cluster uses cloudformation templates for creating the Compute cluster and AppViewX automation workflows for mapping the DB clusters with the compute and the other process tied to AppViewX SaaS life cycle management.

AppViewX Provisioning Cluster Architecture

Compute Cluster

AppViewX compute cluster is a managed-compute infrastructure that runs the AppViewX business logic. The compute cluster is powered by Amazon Elastic Kubernetes Service (Amazon EKS) which is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.

The compute cluster is deployed via the Provisioning cluster using the cloud formation template. Compute cluster encompasses the below.

- EKS
- AppViewX workloads
- Infrastructure components
- Bastion Host

EKS Cluster

EKS clusters are composed of the following main components—a control plane and worker nodes. Each cluster runs in its own, fully managed Virtual Private Cloud (VPC).

The control plane is composed of three master nodes, each running in a different AZ to ensure AWS high availability. Incoming traffic directed to the Kubernetes API passes through the AWS network load balancer (NLB).

Worker nodes run on Amazon EC2 instances located in a VPC. EKS provides managed node groups with automated lifecycle management. This lets users automatically create, update, or shut down nodes with one operation. EKS uses Amazon's latest Linux AMIs optimised for use with EKS. When nodes are terminated, EKS gracefully drains them to make sure there is no interruption of service.

High Availability

Amazon EKS runs and scales the Kubernetes control plane across multiple AWS Availability Zones to ensure high availability. Amazon EKS automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane.

The EKS cluster consists of EC2 instances deployed in multiple availability zones within the region. Each instance has replicas of the services and nodes which exist across all the EC2 instances.

Each zone or instance has an active pod listening to other instances. In case of a failure of any instance, the active pod ensures seamless functioning of the application by activating the nodes from any other working cluster.

Note: EKS clusters are deployed within specific regions and each region has multiple availability zones. Example - Region : us-east-1 and the respective zones : us-east-1a, us-east-1b, us-east-1c.

AppViewX Workloads & Infrastructure Components

AppViewX workloads are containerized workloads running as microservices and these containers are orchestrated using Amazon Elastic Kubernetes Service (Amazon EKS).

The workloads are a mix of AppViewX Business logics that enable communication from User Interface to AppViewX core services and AppViewX SaaS services which is used for enabling the SaaS communication from the AppViewX's SaaS compute to the customer network.

The infrastructure components encompasses third party components that are used for the purpose of service mesh, log aggregation and monitoring the utilisation of the application workloads and so on.

All these workloads, infrastructure components are deployed from the provisioning cluster.

Bastion Host

A bastion host is another EC2 instance based on Linux OS which is created on the same VPC of the wokernodes and this is used for cluster admin operations and troubleshooting the application if required.

The bastion host is accessed via SSH keys generated during the EKS cluster creation and each and every cluster have their own SSH key and the key is downloaded only from the AppViewX SaaS provisioning cluster.

Database Cluster

AppViewX Database cluster is a managed database infrastructure of AppViewX SaaS which holds the customer data. The database cluster is powered by MongoDB Atlas which brings together capabilities that are critical to a modern, cloud-native, microservice-aligned database architecture, including scalability, availability, and uptime.

The AppViewX Database Cluster is enabled via MongoDB Atlas which is a global cloud document database service. The Atlas service MongoDB ensures availability, scalability, and security compliance. The granular features of this cluster are:

- A single database with multiple schemas.
- Individual schemas are generated for each Licensed tenant.
- Snapshots are created for the licensed tenant in the DB cluster and each of the snapshots contains mandatory schemas such as :
  - appSession
  - appviewx
  - appviewxCA

- These schemas are created before the tenants are onboarded.
- Apart from the three mandatory schemas, Snapshot Ids are created for the following schemas:
  - connectedPlatform
  - imageDetails
  - templateDB
  - workFlowDB
  - workFlowDBEn..
- The tenant data is secured and isolated due to this segregation of schemas.
- It also ensures the singularity of data for each licensed tenant.

The AppViewX Database Cluster is made highly available by enabling the cluster deployment on multiple zones or even more resilient by enabling the cluster deployment on multiple regions. Each of these Clusters have a unique URL and credential associated with it.

Monitoring Cluster

Monitoring Cluster is a managed infrastructure of AppViewX SaaS which caters to monitoring, and understanding the performance of the application and machine critical services which is a condensed form of metadata, metrics, and events about the application and its underlying services. This is enabled with a monitoring stack comprising Prometheus, Grafana, Loki, Promtail, and AlertManager.

The monitoring cluster has a Status dashboard and is deployed in a separate cluster which is again a subset of AppViewX powered with AppViewX monitoring capability enabled via Prometheus, Grafana, Loki, Promtail, and AlertManager and it is deployed on AWS (like an onprem AppViewX deployment) with its own database, compute etc which can be deployed cross zone or cross region for high availability.

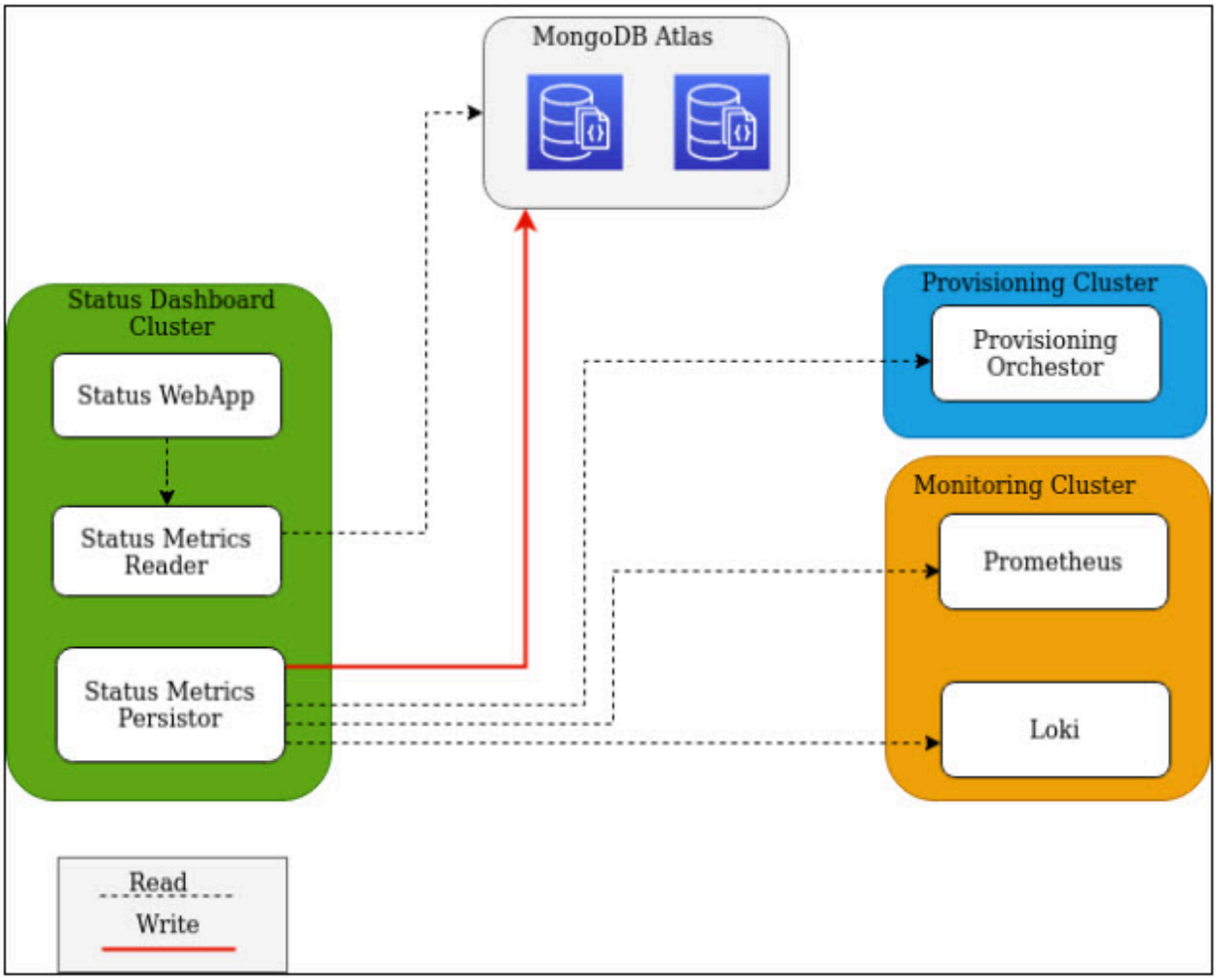
The status dashboard application will have three microservices

- Status dashboard Webapp
- Status Metrics Reader
- Status Metrics Persistor

The Webapp talks to Status metrics Reader and displays AppViewX services uptime details.

The Status Metrics Reader reads data from a dedicated instance residing in MongoDB Atlas.

The Status Metrics Persitor aggregates data from Monitoring and Provisioning clusters and save them in the dedicated instance which resides in MongoDB Atlas.



Monitoring Cluster Architecture

The AppViewX Cloud Connector

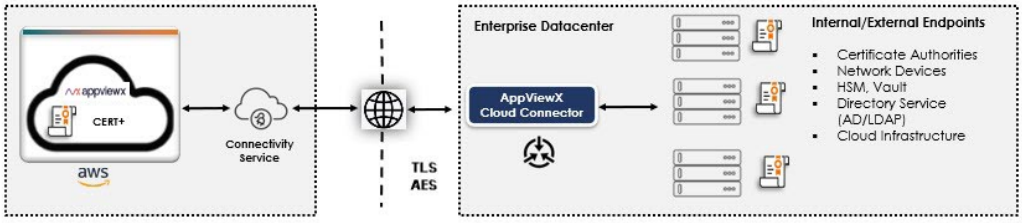
AppViewX Cloud Connector is a lightweight plug-in that establishes connectivity between AppViewX Cloud and the Enterprise Network. The cloud connector serves as a secure channel for communication between AppViewX SaaS and your enterprise network without requiring any complex network or infrastructure configuration

Services that require the AppViewX Cloud Connector for using the AppViewX products (examples):

- **CERT+**
  - Discovering certificates from an endpoint within the enterprise network via Smart Network Scan and Managed Device Scan.
  - Discovering certificates from Certificate Authorities (CAs) that are internal to the enterprise. For example : EJBCA.
  - Discovering certificates from public Certificate Authorities (CAs)

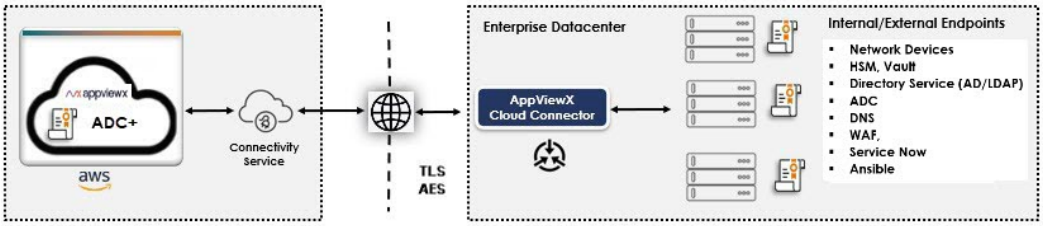
  In this case, AppViewX provides a default instance of the Cloud Connector called **cloud-dc**.

  To enable this, at the time of adding a new AppViewX Cloud Connector instance, from the **Data Center** dropdown list, select **DC Routing**.



- **ADC+**:
  - Communicating with ADC devices and discover the Application Services from the ADC infrastructure
  - Gain Visibility and to fetch the real time state/status of the Applications discovered
  - Self Service the Applications to allow/deny traffic
  - Backup the configuration of the ADC devices
  - Restore the configuration of the ADC devices
  - Automate and Orchestrate the ADC configuration within and across devices

  To enable this, at the time of adding a new AppViewX Cloud Connector instance, from the **Data Center** dropdown list, select **DC Routing**.



Key features of the AppViewX Cloud Connector:

- A self-serviceable, Linux-based lightweight setup
- Secure communication between the AppViewX SaaS and the AppViewX Cloud Connector using TLS and AES encryption
- Connectivity from the AppViewX SaaS to the enterprises' network endpoints
- No complex network setup (Inbound Firewall Whitelisting, VPN setup, and so on)
- Features of the AppViewX Cloud Connector

Features of the AppViewX Cloud Connector

Chapter 5: Disaster Recovery

For details on AppViewX's disaster recovery implementation, refer to the Disaster Recovery Guide.

Chapter 6: Security

For details on the security services offered by AppViewX, refer to the Security Monitoring User Guide.

Chapter 7: Compliance

Type	Status	Vendor	Notes
ISO 27001	Completed	BSI	Certificate Received
TRUSTe	In progress	TrustArc	Currently in Progress

For more details, refer to the Audit and Compliance Guide.

Chapter 8: Glossary

An explanation of the terms used in this guide:

Term	Description
SaaS	Software as a Service
EKS	Elastic Kubernetes Service
TLS	Transport Layer Security
AES	Advanced Encryption Standard
AZ	Availability Zone
VPN	Virtual Private Network
VPC	Virtual Private Cloud
EC2	Elastic compute
AWS	Amazon Web services
HA	High Availability
DR	Disaster Recovery
mTLS	Mutual TLS Authentication