Deployment Architecture Guide

for ThingWorx Version 8.x

July 2018

Version 1.1

Contents

1

Document Revision History

Revision Date	Version	Description of Change
July 2018	1.1	Added section about deploying ThingWorx Foundation on Azure. Updated deployment architecture images to distinctly show the data and application layers.
May 2018	1.0	Initial version.

2 Introduction

This guide introduces the components and practices that make up ThingWorx deployments. It describes the basic components found in every ThingWorx environment and discusses common deployment concerns and challenges. The guide provides common ThingWorx reference architectures to consider when planning a ThingWorx deployment.

Challenges

Platforms for IoT (Internet of Things) applications must meet the following challenging requirements:

- Manage the throughput of content from thousands of devices at a rapid pace.
- Provide real-time visibility into the performance of the assets, which can be anything from production lines to smart-connected products.
- Provide remote monitoring and diagnostic services to end users, including remote troubleshooting and automatic creation of alerts and trouble tickets.
- Predict failures before they occur, integrate with knowledge-based systems, and integrate augmented-reality technology into the field service processes.

A poor architecture results in a difficult deployment, costly integration between components with each upgrade, and mixed technologies that limit future flexibility and introduce multiple points of failure. Instead, a connected and distributed architecture is required.

- Connected — A good connection with assets enables the platform to coordinate data monitoring, health monitoring, proactive maintenance, software management, and remote service.
- Distributed — A distributed architecture allows for global management where consolidated statistics and KPIs across all locations are desired and provides local managers with focused and real-time information.

Deploying ThingWorx to Your Specifications

When you choose ThingWorx to support the IoT business processes of your company, the deployment architecture must meet your company's requirements for security, scalability, performance, interoperability, flexibility, and maintainability.

You might ask the following questions:

- What are the deployment options that can meet those requirements?
- How can you combine the necessary elements that address your company's requirements into a coherent system?
- How can you be sure that the architecture you deploy can be expanded to support future needs?

The first critical step toward developing a ThingWorx deployment architecture is to gain an understanding of the following points:

- The structural organization, the elements and their roles, and the interfaces of the ThingWorx architecture.
- The configuration options that combine the necessary elements to support your company’s requirements.
- Some ThingWorx configurations that have been proven to work.

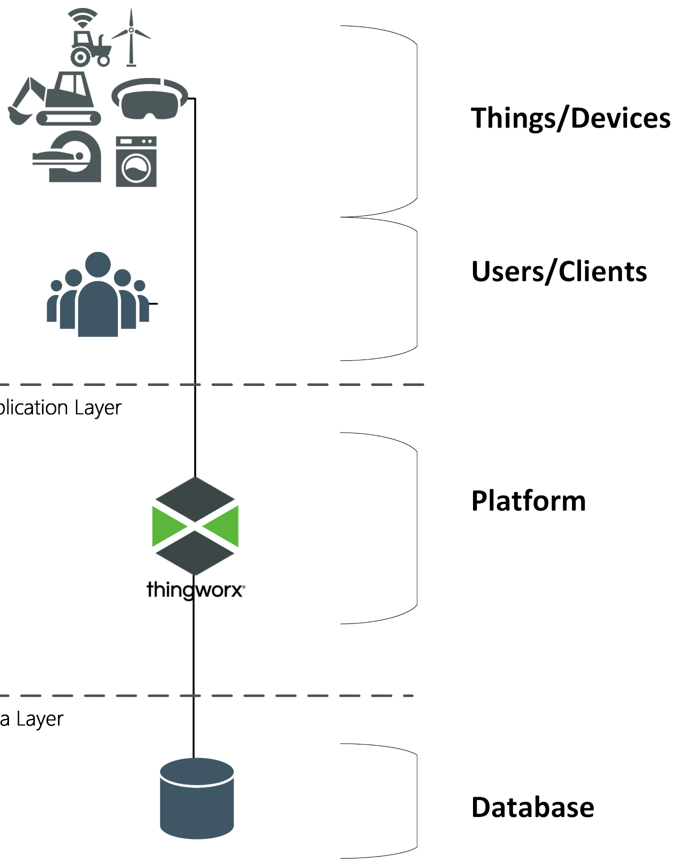
3

ThingWorx Foundation Deployment Components

ThingWorx components can be found in each of the following layers:

- Client layer
- Application layer
- Data layer

The following images show a basic solution or starting point for any ThingWorx solution:



- Things/Devices — This layer contains the things, devices, agents, and other assets that connect with, send data to, and receive content from the ThingWorx platform.
- Users/Clients — This layer contains the products (primarily Web browsers) that people use to access the ThingWorx platform.
- Platform — The platform layer (or application tier) is where ThingWorx Foundation resides, which serves as the hub of the ThingWorx environment. This is the layer where content from the things/devices layer is ingested, user requests from the client layer are answered, and content is analyzed to generate alerts.
- Database — The database layer maintains the following forms of data:
  - ThingWorx runtime model definitions and their persisted properties.
  - Tabular-type data that is persisted by the runtime model as rows of content in blogs, wikis, streams, value streams, and data tables.

With the growth of the ThingWorx solution in capability and complexity, the architectural needs within each tier grow. For more information, refer to ThingWorx Deployment Options on page 23.

The following sections introduce each component of a ThingWorx solution within the tier or layer in which the component operates.

User/Client Components

The user or client accessing the ThingWorx platform through ThingWorx Composer or through runtime mashups is required to have a modern browser that supports HTML/HTML5 (Internet Explorer, Firefox, Safari, and Chrome).

Thing/Device Components

ThingWorx WebSocket-based Edge Microserver

The ThingWorx WebSocket-based Edge MicroServer (WS EMS) works with edge devices or data stores that need to connect to the ThingWorx server over the internet. It enables devices and data stores that are behind firewalls to securely communicate with the ThingWorx server and be full participants in the solution landscape. ThingWorx WS EMS is not a simple connector but allows intelligence and pre-processing of data to be moved to the edge.

ThingWorx Edge SDKs

ThingWorx Edge SDKs are collections of classes, objects, functions, methods, and variables that provide a framework for creating applications that can send data securely from edge devices to the ThingWorx platform. ThingWorx Edge SDKs provide tools for developers experienced in C, .NET, Java, and Android programming languages.

ThingWorx WS EMS and ThingWorx Edge SDKs support connections through proxies. The process of managing the proxy configuration and associated change management varies by customer and/or project. ThingWorx Edge SDKs provide ultimate flexibility because the SDK libraries can be included or referenced by any custom edge component—and therefore can be updated based on the design of the solution.

Platform Components

ThingWorx Connection Server

The ThingWorx Connection Server is a server application that facilitates the connection of remote devices and handles all message routing to and from the devices. The ThingWorx Connection Server provides functionality, such as scalable connectivity over WebSockets using the ThingWorx Communication Protocol.

PTC recommends using connection servers when there are more than 25,000 assets connecting to a ThingWorx Foundation server.

PTC also recommends one connection server for every 100,000 simultaneous connections to the ThingWorx Foundation server. This ratio of devices-to-connection server may change depending on many factors, such as the following:

- The number of devices
- The frequency of write submissions from the devices

Tomcat

Apache Tomcat is an open-source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a pure Java HTTP Web server environment for Java code to run.

ThingWorx Foundation Server

ThingWorx Foundation provides a complete design, runtime, and intelligence environment for Machine-to-Machine (M2M) and IoT applications. ThingWorx Foundation is designed to efficiently build, run, and grow applications that control and report data from remote assets, such as connected devices, machines, sensors, and industrial equipment.

ThingWorx Foundation serves as the hub of your ThingWorx environment. It includes tool sets that help you develop applications to define the behavior of remote assets (or devices) deployed in your environment and relationships between the assets.

Once the assets have been modelled, they can register and communicate with ThingWorx Foundation, allowing you to monitor and manage the physical devices and collect data from them.

PTC System Monitor

The PTC System Monitor is a separate, independent, application performance monitoring system that is powered by the Production Edition of Dynatrace. PTC has tailored the Dynatrace Production Edition to meet the needs for monitoring the PTC service life cycle management system (SLM), such as ThingWorx. The PTC System Monitor provides useful dashboards and instrumentation that allows monitoring while maintaining critical performance requirements.

For more information about the PTC System Monitor, refer to the following documents:

- PTC System Monitor Installation and Deployment Guide - ThingWorx
- PTC System Monitor Administration and Usage Guide - ThingWorx

Database Components

The ThingWorx platform offers a pluggable data store model, which allows every customer to choose the database that best suits their requirements—from small implementations for demo and training environments to highly available, high-volume databases that support thousands of transactions per second.

Value streams, streams, data tables, blogs, and wikis are defined as data providers for ThingWorx. Data providers are considered databases that store runtime data. Runtime data is data that is persisted once the Things are composed and are used by connected devices to store their data (such as temperature, humidity, or position). Model providers are used to store metadata about the Things. Persistence Providers can contain a data provider, a model provider, or both.

ThingWorx data providers have two primary functions:

1. To maintain the ThingWorx model. The following database systems are supported to act as model providers:
    - H2 and PostgreSQL are supported in all versions of ThingWorx 7 and 8.
    - SAP HANA is supported in ThingWorx 7.3 and higher.
    - SQL Server is supported in ThingWorx 7.4 and higher.
2. To contain content written to the ThingWorx value stream.
    - Any supported database can also simultaneously contain value stream content.
    - For high rates of data ingestion, the DataStax Enterprise system is supported.

H2

H2 is an open-source relational database with a low-disk footprint. It is a relational database management system written in Java that can be embedded in Java applications or run in the client-server mode. H2 fulfills both model and data provider requirements for ThingWorx and is embedded within the ThingWorx installation.

PostgreSQL

PostgreSQL is an open source object-relational database management system (ORDBMS) with an emphasis on extensibility and compliance to standards. As a database server, its primary function is to store data securely and allow for retrieval at the request of other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

PostgreSQL allows for high-availability capability at the database level. It can be set up with a master and multiple slave nodes in the same or different availability zones.

PostgreSQL, via its Persistence Provider, supports both model and data providers. For more information on ThingWorx and PostgreSQL deployments, refer to the following documents:

- ThingWorx PostgreSQL Administrator's Guide
- ThingWorx 8 High Availability Administrator’s Guide

Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications, which may run on the same computer or on another computer across a network (including the Internet). For more information on ThingWorx and Microsoft SQL Server deployments, refer to Getting Started with MS SQL Server and ThingWorx.

SAP HANA

SAP HANA is an in-memory relational database management system (RDBMS) that is scalable and performs massive parallel processing with data that is stored in-memory. The system processes high-speed transactions and analytics and manages large data volumes using multi-tenant database containers and dynamic tiering across multi-tier storage. The system fulfills ACID (Atomicity, Consistency, Isolation, Durability) requirements. SAP HANA is also a development platform and application server within the same environment. SAP HANA implements larger databases through scaling out (also referred to as a distributed system), which means that the database spans multiple physical servers in a multi-host system.

SAP HANA has the following deployment options (http://go.sap.com/product/technology-platform/hana/implementation/deployment.html):

- On Premise:
  - Appliance offering: The appliance is a bundle of SAP software, preinstalled on certified pieces of hardware from one of SAP HANA’s certified hardware vendors.
  - Tailored Data Center Integration: Adds the ability to reuse existing data center components for on-premise installation of SAP HANA.
- In the Cloud:
  - Private Cloud:

        - Managed Service with SAP HANA Cloud Platform (HCP).

        > **Note**
        >
        > ThingWorx may not be able to be deployed into an HCP environment.

    - Public Cloud:
        - Infrastructure-as-a-Service in Amazon Web Services (AWS) or Microsoft Azure clouds.
        - Pay-as-you-go with SAP HANA One in third-party clouds like Amazon Web Services (AWS) or Alibaba.

For more information about ThingWorx and SAP HANA deployments, refer to Getting Started with SAP HANA and ThingWorx. For more information about SAP HANA, refer to the SAP documentation.

DataStax Enterprise (DSE)

ThingWorx also uses a high-volume Runtime Data Store provider, DataStax Enterprise (DSE), powered by Apache Cassandra. DSE allows the customer to intake a higher availability of data that their assets are generating and in return, allows them to scale seamlessly when additional devices (or other load) are added. Using DSE as the runtime data store delivers a database platform built for the performance and availability demands of IoT, Web, and Mobile applications. While relational databases may struggle under modern and dynamic workloads, high-volume data, and new business requirements, Cassandra’s modern architecture makes it massively scalable.

DSE offers a fully tested and validated Cassandra deployment with advanced administration and monitoring tools, built-in integration with Solr for indexing and search, and a mechanism for support and patches. This coincides directly with the pluggable data store model that allows for customers to have multiple data repositories to store configuration, model, and high-volume data. Customers are able to choose which data repository meets their requirements for any particular function.

DSE is an integrated always-on multi-model database system with real-time and batch analytics using Apache Spark, in-memory technology, continuously available search, and graph database computing. It offers advanced tools for development and production system operations, flexible features such as tiered storage for short-term and long-term data access, multitenancy to run several database clusters within the same system, and advanced security to meet enterprise requirements.

ThingWorx platforms that require a data store for higher volumes and velocity data than what is currently available with H2 or PostgreSQL can have the following benefits with DSE:

- Higher rate of ingestion of data
- More than one data repository for runtime data (maintaining the model data in H2 or PostgreSQL and using DSE for high-volume stream data)
- Elastic scaling properties. You can add more nodes to a DSE ring for higher transaction rates.
- Ability to separate data processes from platform processes
- Cloud-friendly architecture

For more information, refer to Getting Started with DataStax Enterprise and ThingWorx.

High-Availability Components

High-availability solutions are important considerations for business continuity. High-availability components need to be applied at the application layer and database layer to be complete. For ThingWorx high availability, Apache ZooKeeper is an additional required component. For the database layer, the need for additional components depends on the need of the data provider.

ZooKeeper

Apache ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. It is a coordination service for distributed application that enables synchronization across a cluster. Specific to ThingWorx, ZooKeeper is used to coordinate the shift from the active ThingWorx platform to the passive platform. It monitors the active server availability and elects a new ThingWorx Foundation leader during a system failure.

Database HA features

- PostgreSQL: ThingWorx provides the option to use PostgreSQL high availability as part of the data solution. High availability offers the option to set up separate servers to capture reads and writes for data if a failure occurs on the primary server. For more information, refer to ThingWorx High Availability Administrator's Guide.
- SQL Server: Generally, the SQL Standard Edition is suitable for production environments, as it supports most of the features required. For production environments that require high-availability features such as AlwaysOn and other features such as In-Memory OLTP, and table and index partitioning,

SQL Enterprise Edition is recommended. For more information, refer to Getting Started with MS SQL Server and ThingWorx.

4

ThingWorx Deployment Architectures

Deployment Options

On-Premise Deployment

Using an on-premise deployment means hosting and managing the ThingWorx software on your servers. You are responsible for maintaining infrastructure and application support and performance.

With an on-premise deployment, you can perform the deployment yourself or engage PTC Global Services (or a PTC-certified partner) to manage the deployment on company servers. This option is suitable for companies with a robust IT organization and a strong desire to maintain in-house control.

PTC Cloud Services

In a managed-services deployment, the ThingWorx application is deployed, hosted, and managed on a third-party server, usually in a private cloud. An outside organization is responsible for managing the necessary infrastructure and ensuring application performance.

For companies concerned with the IT burden and expertise required to manage ThingWorx, PTC provides a managed-services deployment option. With PTC Cloud Services, companies purchasing ThingWorx can accelerate deployment, minimize IT cost and requirements, and ensure ongoing performance. PTC Cloud Services hosts your ThingWorx solution in a secure environment within commercial cloud services that has ongoing application management, performance tuning, and updates. For additional information about PTC Cloud Services, see www.ptc.com/services/cloud.

ThingWorx Foundation Reference Architectures

The following sections include diagrams that depict typical ThingWorx deployments. The architectures range from simple development systems to global federated production systems.

For information about ThingWorx components, see ThingWorx Foundation Deployment Components on page 13.

Note

To deploy ThingWorx in hybrid and multi-site deployments, refer to Distributed ThingWorx Deployment on page 77.

References

Refer to the following documents on the PTC Support Portal:

ThingWorx Foundation

- Installing ThingWorx 8.3
- ThingWorx 8.3 System Requirements
- ThingWorx Foundation Docker Installation Guide

ThingWorx Persistence Providers

- ThingWorx PostgreSQL Administrator's Guide
- Getting Started with MS SQL Server and ThingWorx
- Getting Started with SAP HANA and ThingWorx
- Getting Started with DataStax Enterprise and ThingWorx

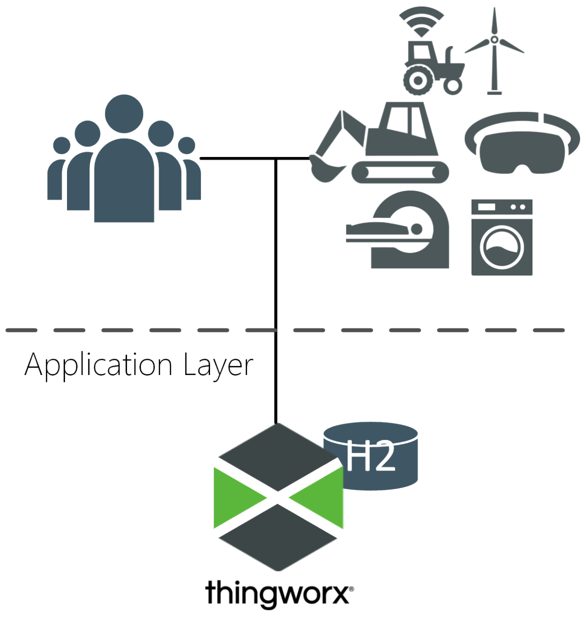
ThingWorx High Availability

- ThingWorx 8 High Availability Administrator's Guide

ThingWorx Foundation Basic System

The following diagram shows a typical local system for development, training, proof-of-concept, functional QA testing, or sandbox purposes. All components are deployed on a single server.

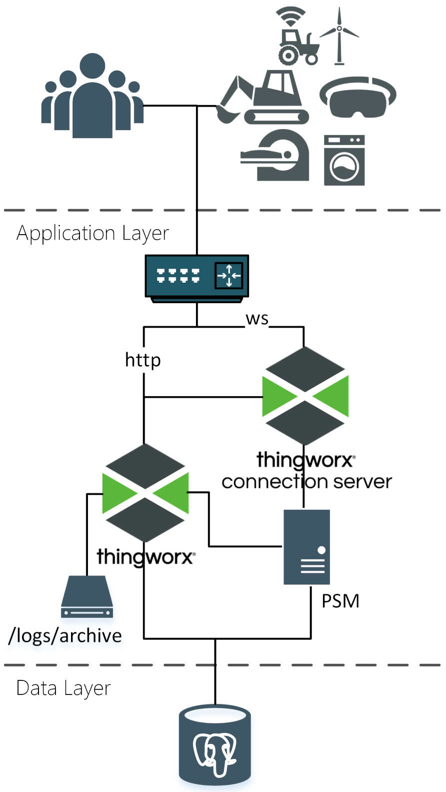
While this deployment option is acceptable for simple scenarios, it must not be considered for most production systems. The provided H2 database does not scale well beyond small systems, and the system is at risk of performance issues when all components compete for the same server resources.



List of Components	Number of Components
ThingWorx Foundation Server	1 (using H2 embedded database)

ThingWorx Foundation Basic Production System

For a basic production system, it is recommended to operate the database on a separate server. This is a good small-to-medium-sized enterprise system, or a medium-to-large manufacturing system.



List of Components	Number of Components
Load Balancer	1
ThingWorx Connection Server	1
ThingWorx Foundation Server	1
PSM	1
Attached or NAS File Storage	1
Database	1

ThingWorx Foundation Large Production System

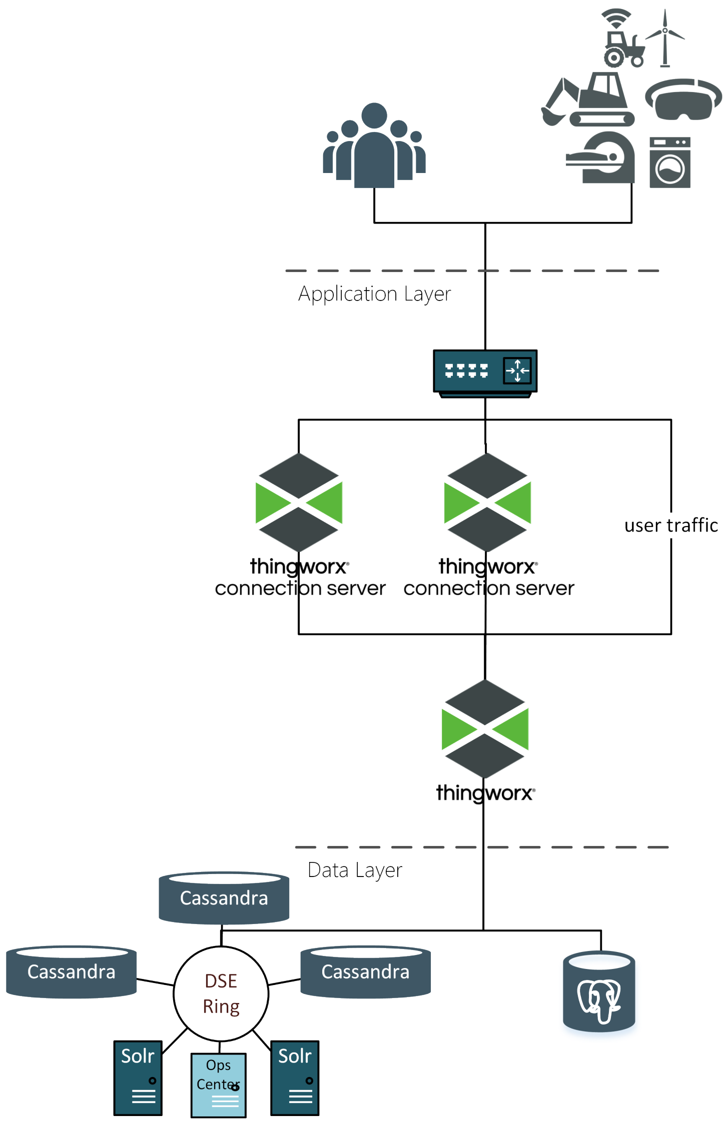
A large production system, also referred to as a ThingWorx Enterprise system, incorporates additional components to support a large number of connected devices and high data ingestion rates.

In addition to the production components, a large production system may include ThingWorx Connection Servers and a DataStax Enterprise ring.

The DataStax system intakes the data coming from the assets into its time-series type database. For ThingWorx, that data is managed as logged value stream content. The PostgreSQL server is still required to maintain the ThingWorx model.

Consider the following points while designing an Enterprise system for use by ThingWorx using DataStax:

- The minimum number of Cassandra nodes within a DataStax ring is three. More Cassandra nodes would be required to meet any disaster recovery requirements.
- The replication strategy `NetworkTopologyStrategy` is recommended, as it supports operation across multiple data centers.
- Replication factor should be set to three.
- Consistency Level should be set to `LOCAL_QUORUM`.
- Solr nodes are only required if data tables are used. In theory, this could mean that the use of Solr nodes is optional.



List of Components	Number of Components
Load Balancer	1 (route device traffic to connection servers.)
ThingWorx Connection Server	2 (depends on number of devices)
ThingWorx Foundation Server	1
Database	1
DataStax Enterprise	Depends on ingestion requirements. Minimum of 5 nodes: 3 Cassandra and 2 Solr nodes. Ops Center node is optional.

ThingWorx Foundation High-Availability System

For a high-availability (HA) deployment, additional components are added to remove single points of failure for the ThingWorx Foundation and its database systems.

The following components are required for the ThingWorx platform:

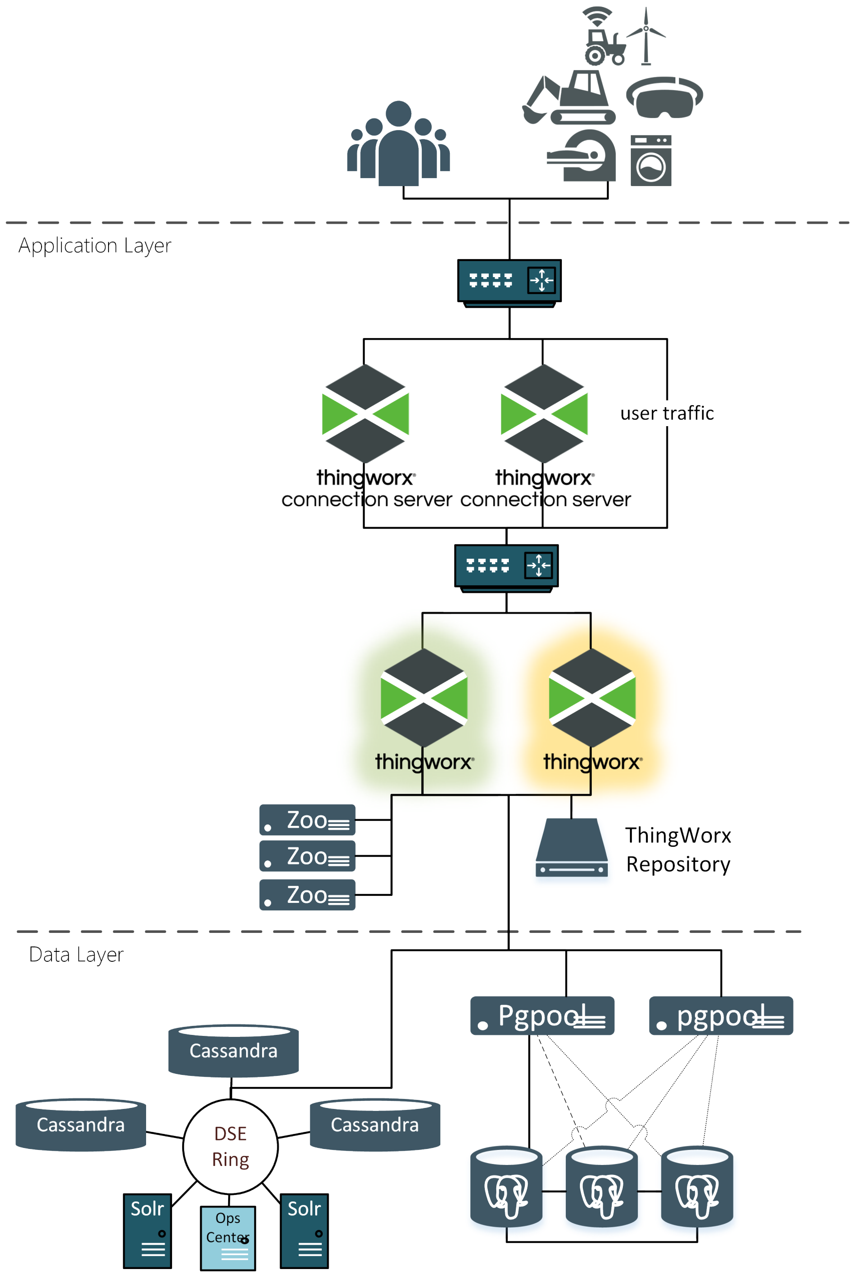
- A high-availability load balancer to direct traffic to the active ThingWorx platform.
- A lead ThingWorx platform. This platform is the primary server, mostly active, and would only be offline for maintenance windows and unplanned downtime.
- A standby ThingWorx platform. This platform is the secondary server, ready to take over the ThingWorx-based load when the lead platform is offline.
- ThingWorxStorage is on-disk storage that is equally accessible by lead and standby platforms.
- Three Apache ZooKeeper nodes. These nodes monitor the active ThingWorx platform and constantly check and decide whether it is responsive and functioning as expected. The ZooKeeper nodes form a quorum and decide when the active ThingWorx platform is offline and then activate the switch to the standby system. If the active ThingWorx platform goes offline, the ZooKeeper nodes performs two tasks:
  - Activate the standby ThingWorx server to become the active instance.
  - Reconfigure the ThingWorx load balancer to direct traffic to the now-active platform.

The following components are required for the PostgreSQL database:

- Two or three PostgreSQL server nodes. A minimum of two nodes, ideally three nodes, are used. All three nodes are online and able to perform queries and replication tasks.
- Two pgpool-II nodes. A minimum of two, ideally three nodes, that perform middleware tasks between the PostgreSQL servers and its clients (in this case, the ThingWorx platform). These nodes maintain connections between client and servers, manage replication of content among PostgreSQL server nodes, and load balance queries to the nodes.

The following components are required for the DataStax Enterprise system:

- Cassandra nodes to contain value stream content. To avoid a single point of failure and always maintain a majority vote, an odd number of Cassandra servers should be deployed. A minimum of three Cassandra nodes is recommended.
- Solr nodes provide an indexing service to the content stored within the Cassandra nodes.



List of Components	Number of Components
ThingWorx Connection Server	2 (depends on number of devices)
Load Balancer	2. One to route device traffic to connection servers. One to route to active ThingWorx Foundation server.
ThingWorx Foundation Server	Minimum of 2: one active (in green), one passive (in yellow). There can be additional passive Foundation servers.

List of Components	Number of Components
Networked/Enterprise Storage	Disk space for ThingWorx storage repositories shared with all ThingWorx Foundation servers.
ZooKeeper	Minimum of 3. Should be in odd-numbered allotments.
Database	Depends on database: • PostgreSQL: 3 nodes. • MS SQL Server (not pictured): minimum of 2 as part of an MS failover configuration.
pgpool-II	2 (for PostgreSQL only).
DataStax Enterprise	Depends on ingestion requirements. Minimum of 5 nodes: 3 Cassandra and 2 Solr nodes. Ops Center node is optional.

For details about deploying ThingWorx in a high-availability environment, see ThingWorx High Availability Administrator's Guide.

5

Standard Deployment: ThingWorx Foundation on Azure

ThingWorx can be deployed in cloud platforms, such as Microsoft Azure. Many Azure services are available to help with deploying ThingWorx and managing it over time.

Azure Components and Services

Regions

Regions are geographic areas where Azure resources are physically located.

Availability Zones

Availability Zones are isolated locations within a region. Each region contains multiple availability zones to support high availability deployments.

Availability Sets

Availability Sets are separated (but not isolated) resources within an availability zone.

Virtual Network

Virtual Networks are used for configuring logical network topology, defining sub networks, configuring routing tables, and assigning private IP ranges.

VM instances

VM instances are the virtual machines used within Azure. They host key software components of the ThingWorx platform, such as ThingWorx Connection Server (if needed), ThingWorx platform (main application), and ZooKeeper.

Application Gateways

Application Gateways distribute incoming application traffic across multiple VM instances. It enables you to achieve fault tolerance in your applications, providing the required amount of load balancing capacity needed to route application traffic.

Azure databases

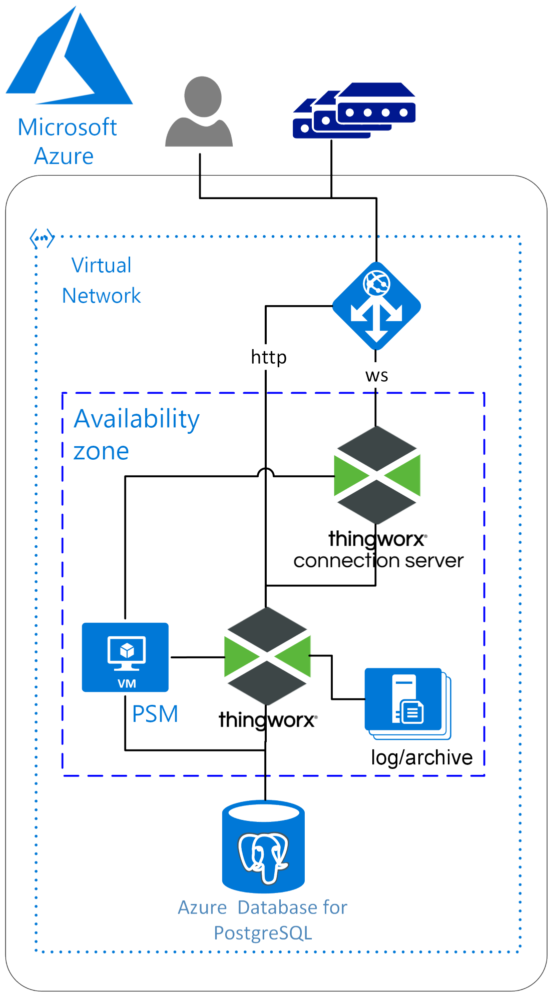
Azure Databases is the PaaS database offering within Azure. It offers single instances as well as highly available and fault-tolerant deployments. Azure Database for PostgreSQL is the supported option for ThingWorx.

Azure Files

Azure Files provides file storage systems that can be shared and accessed by multiple virtual machines.

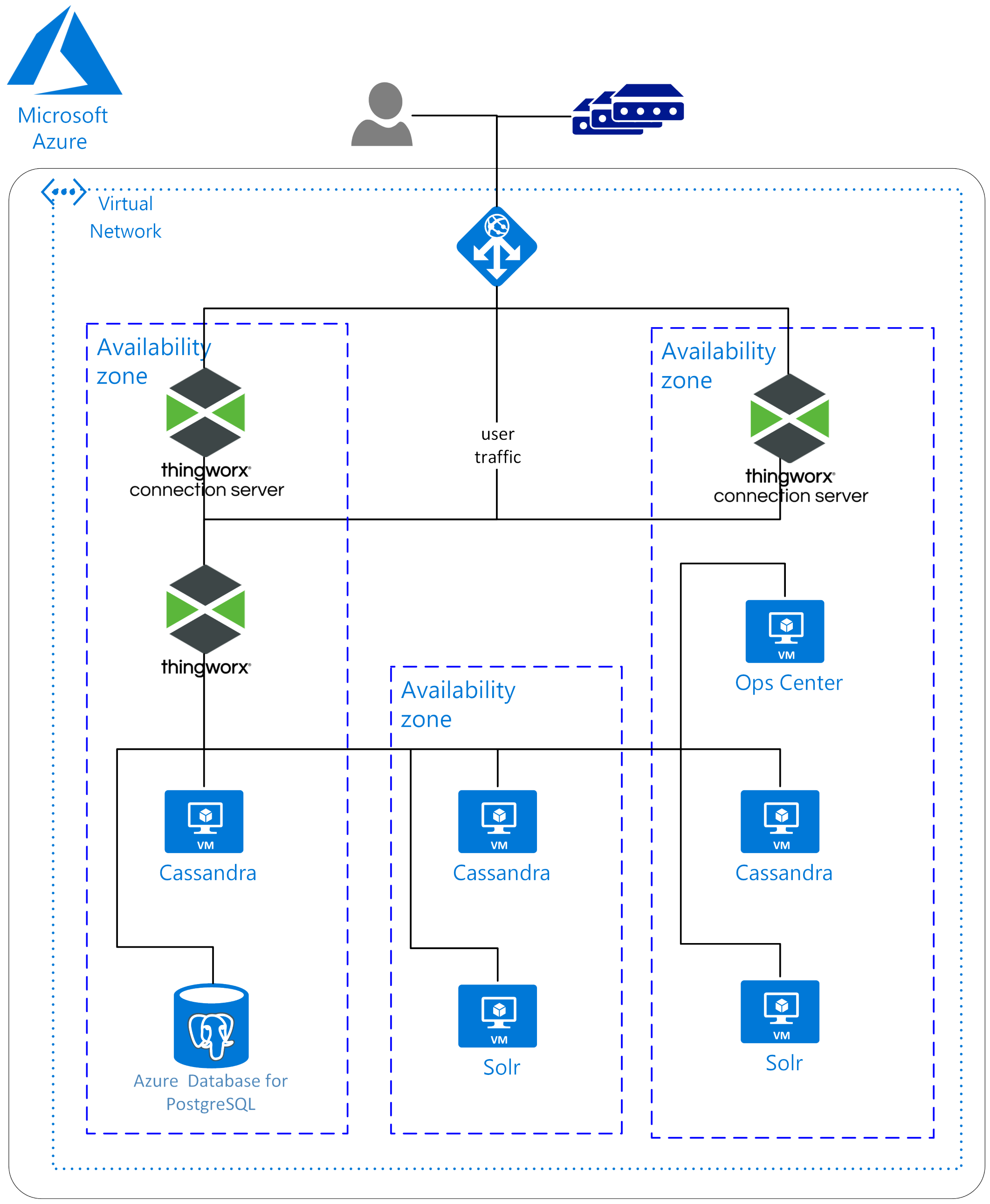
Reference Architectures

Production Deployment



List of Components	Number of Components
Azure Region	1
Azure Virtual Network	1
Azure Application Gateway	1
ThingWorx Connection Server	1
ThingWorx Foundation Server	1
PSM	1
Azure File Storage	1
Azure Database for PostgreSQL	1

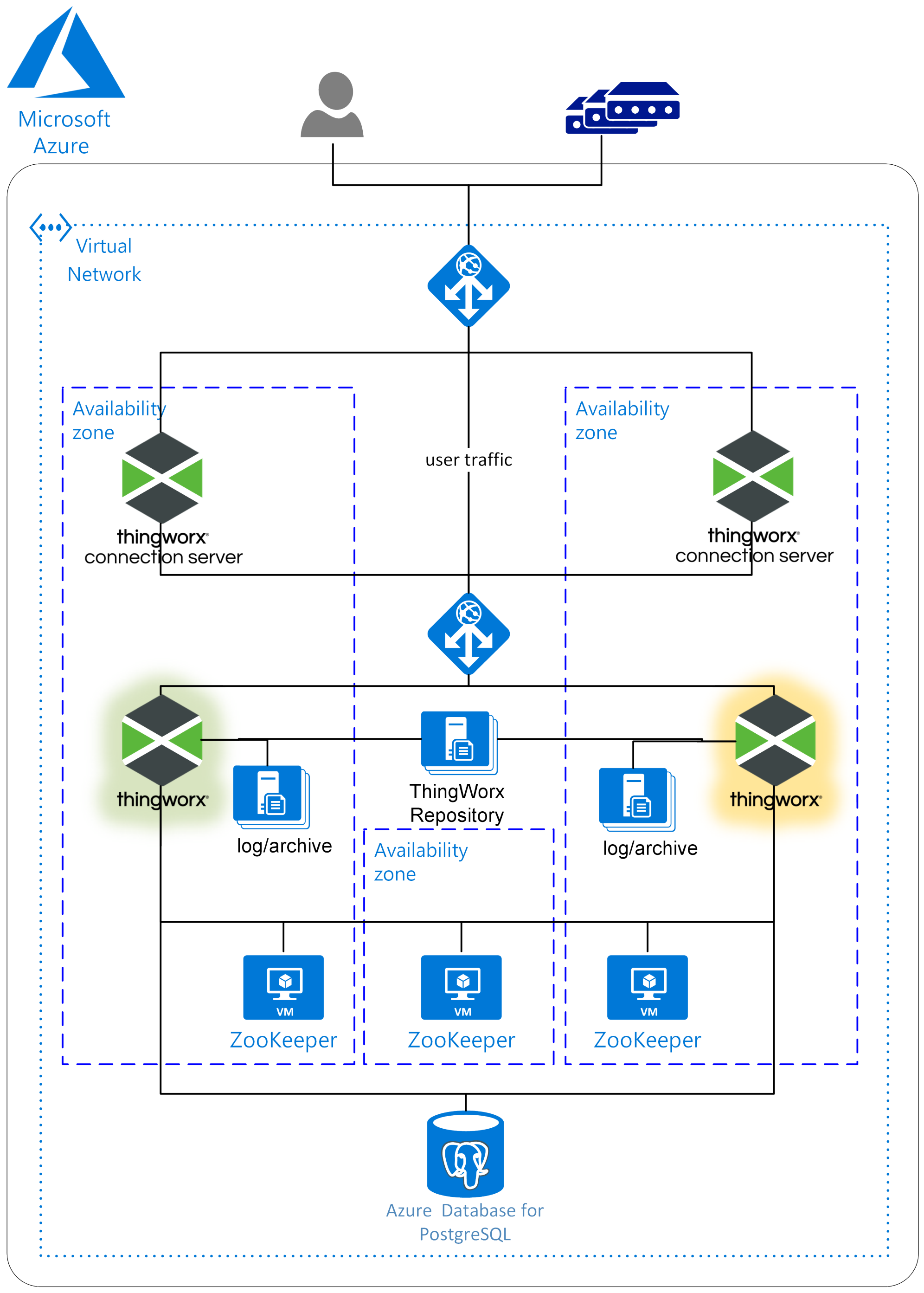
Enterprise Deployment



List of Components	Number of Components
Azure Region	1
Azure Virtual Network	1
Axure Availability Zones	3
Application Gateway	1
ThingWorx Connection Server	2

List of Components	Number of Components
ThingWorx Foundation Server	1
Azure Database for PostgreSQL	1
DataStax Enterprise	Not required. Minimum of 5 nodes: 3 Cassandra and 2 Solr nodes spread across multiple availability zones. Ops Center node is optional.

High-Availability Deployment



List of Components	Number of Components
Azure Region	1
Azure Virtual Network	1
Azure Availability Zones	3

List of Components	Number of Components
Azure Application Gateway	2 • One to distribute device loads to Connection Servers and direct user traffic to ThingWorx Foundation Server. • One to direct Connection Server traffic to active ThingWorx Foundation Server.
ThingWorx Connection Server	2
ThingWorx Foundation Server	Minimum of 2: one active (in green), one passive (in yellow). There can be additional passive Foundation servers.
Azure Files	3 One for each Foundation server to store and archive logs. One to share ThingWorx Storage repositories to all Foundation servers.
ZooKeeper	Minimum of 3. Should be in odd-numbered allotments spread across 3 availability zones.
Azure Database for PostgreSQL	1

ThingWorx Azure IoT Connector Deployment

The ThingWorx Azure IoT Hub Connector consists of a connection server, the Azure IoT Hub Adapter, and an Azure IoT Hub Adapter Extension. This package enables remote devices that are running an application developed using a Microsoft Azure SDK to connect to the ThingWorx platform. The Azure IoT Hub Connector handles message routing for the devices that communicate through the Azure system. It also handles message routing from the ThingWorx platform to devices.

Components

Azure IoT Components

- Azure IoT Hub — A fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a back end solution, such as ThingWorx.

ThingWorx Azure IoT Hub Connector

The ThingWorx Azure IoT Hub Connector integrates devices that use a Microsoft Azure IoT SDK with ThingWorx Foundation. An installation of the ThingWorx Azure IoT Hub Connector consists of the following components:

- The ThingWorx Connection Server with its Azure IoT Hub Adapter, which handles translation of messages and communication with an Azure IoT Hub and the ThingWorx Foundation.
- An Azure IoT Hub Adapter Extension that supports Azure entities in ThingWorx.
- The ThingWorx Connection Services Extension, which provides the Connection Services Hub.

Once you have installed, configured, and started the Azure IoT Hub Connector, you can add Azure IoT entities using ThingWorx Composer to represent your devices. Alternatively, you can import device data into ThingWorx from an Azure IoT Hub. Once you configure devices to connect with the Azure IoT Hub, the Azure IoT Hub Connector detects the data and pushes it to the ThingWorx Foundation. By pushing data to ThingWorx, you can take advantage of its features for developing applications quickly and easily.

Deployment Considerations

An Azure IoT Hub and Azure Blob Storage account and a Blob Storage Container should be established before configuring the ThingWorx Azure IoT Connector.

Production deployments should have a minimum of two ThingWorx Azure IoT Hub Connectors to support high availability. More Connectors may be required, depending on the device load and partition count within your Azure IoT hub account.

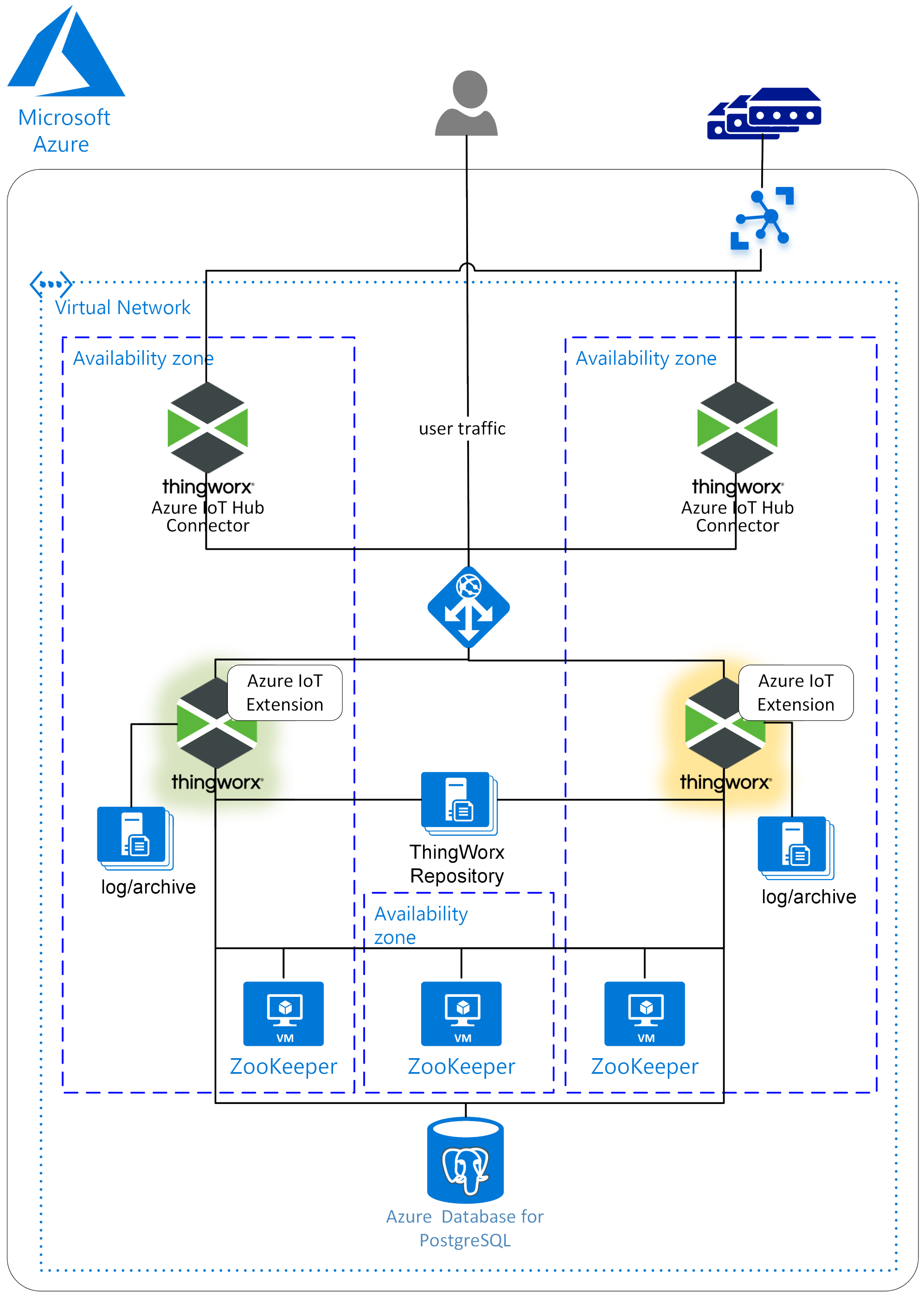
References

- Connect your device to your IoT hub using Java
- ThingWorx Azure IoT Hub Connector Installation and Operations Guide
- ThingWorx Connection Services Help Center

Reference Architecture

The following diagram shows a high-availability deployment of ThingWorx Foundation that utilizes ThingWorx Azure IoT Connectors to access an Azure IoT Hub. This deployment has the following features:

- Devices are sending content to Azure IoT Hub.
- The ThingWorx Azure IoT Connectors are communicating with the Azure IoT Hub to read messages from the devices and store the message content in ThingWorx.
- The ThingWorx Azure IoT Connectors submit the content to the active ThingWorx Foundation server through the Azure Application Gateway.
- User interactions bypass Azure IoT Hub, accessing ThingWorx Foundation through the Azure Application Gateway.
- The ThingWorx Azure IoT Connectors are established in different Availability Zones as part of the high-availability deployment.



List of Components	Number of Components
Azure Region	1
Azure IoT Hub	1
Azure Virtual Network	1
ThingWorx Azure IoT Connectors	Minimum of 2

List of Components	Number of Components
Azure Application Gateway	1 Route user traffic and ThingWorx IoT Connectors traffic to the active ThingWorx Foundation server.
ThingWorx Foundation Server (with Azure IoT Extension)	Minimum of 2: one active and one standby. There may be additional ThingWorx Foundation servers on standby.
File Storage	For each ThingWorx Foundation server, to store and archive log files.
Azure Files	3 One for each Foundation server to store and archive logs. One to share ThingWorx Storage repositories to all Foundation servers.
ZooKeeper	Minimum of 3, spread among 3 Availability Zones
Azure Database for PostgreSQL	1

6

Other Deployments: ThingWorx Foundation on AWS

ThingWorx can be deployed in cloud platforms, such as Amazon Web Services (AWS). Many AWS services are available to help with deploying ThingWorx and managing it over time.

AWS Components and Services

Regions

Regions are geographic areas where AWS resources are physically located.

Availability Zones

Availability Zones (AZ) are isolated locations within a region. Each region contains multiple availability zones to support high availability deployments.

Virtual Private Cloud

Virtual Private Cloud (VPC) is used for configuring logical network topology, defining sub networks, configuring routing tables, and assigning private IP ranges. Additionally, VPC allows for defining hardware VPNs between AWS VPC and on-premise IT infrastructure.

EC2 Computing instances

EC2 computing instances are used to host key software components of the ThingWorx platform: ThingWorx Connection Server (if needed) and ThingWorx platform (main application).

Application Load Balancing

Application Load Balancing (ALB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve fault tolerance in your applications, providing the required amount of load balancing capacity needed to route application traffic.

RDS databases

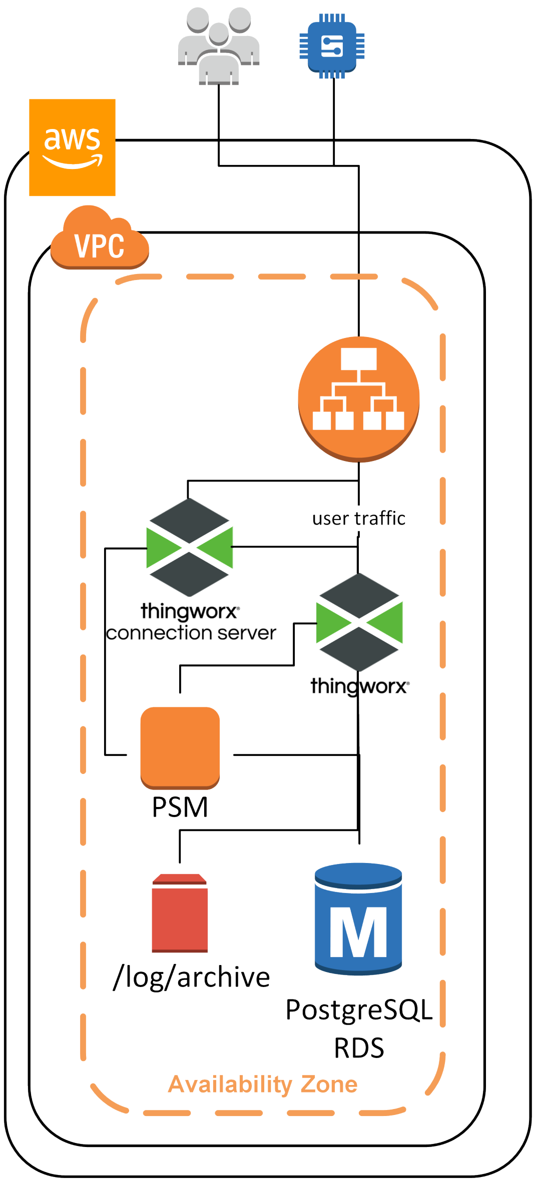
AWS RDS is a database service that offers single instances as well as highly available and fault-tolerant deployments. AWS PostgreSQL RDS and AWS SQL Server RDS are the supported options for ThingWorx.

Elastic File System

Elastic File System (EFS) provides file storage systems that can be shared and accessed by multiple EC2 instances.

Reference Architectures

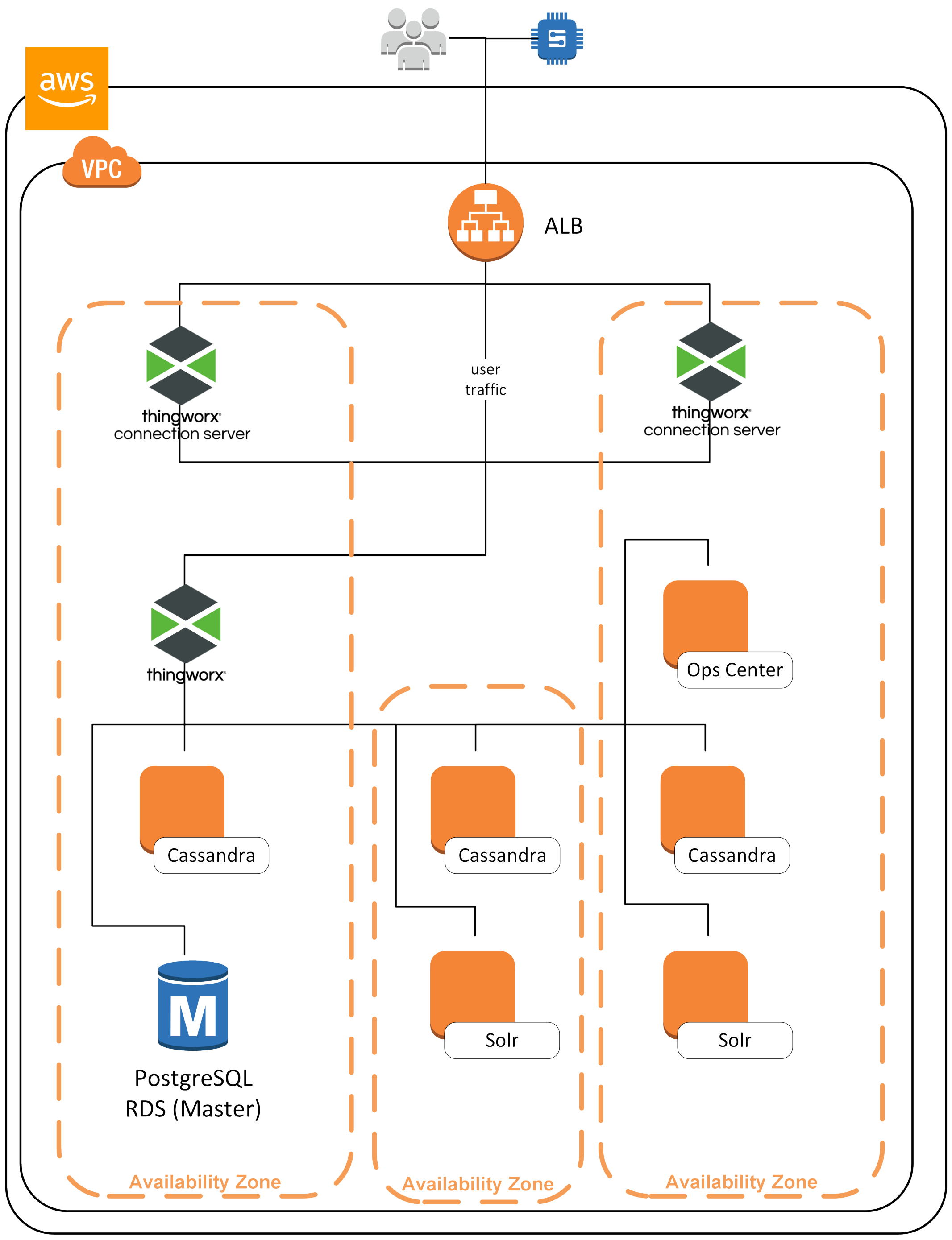
Production Deployment



List of Components	Number of Components
AWS region	1
AWS VPC	1
AWS Application Load Balancer	1 (if you are using a ThingWorx Connection Server)
ThingWorx Connection Server	1 (optional)
ThingWorx Foundation Server	1

List of Components	Number of Components
PSM	1
AWS EFS	1 (to maintain ThingWorx logs)
PostgreSQL RDS	1

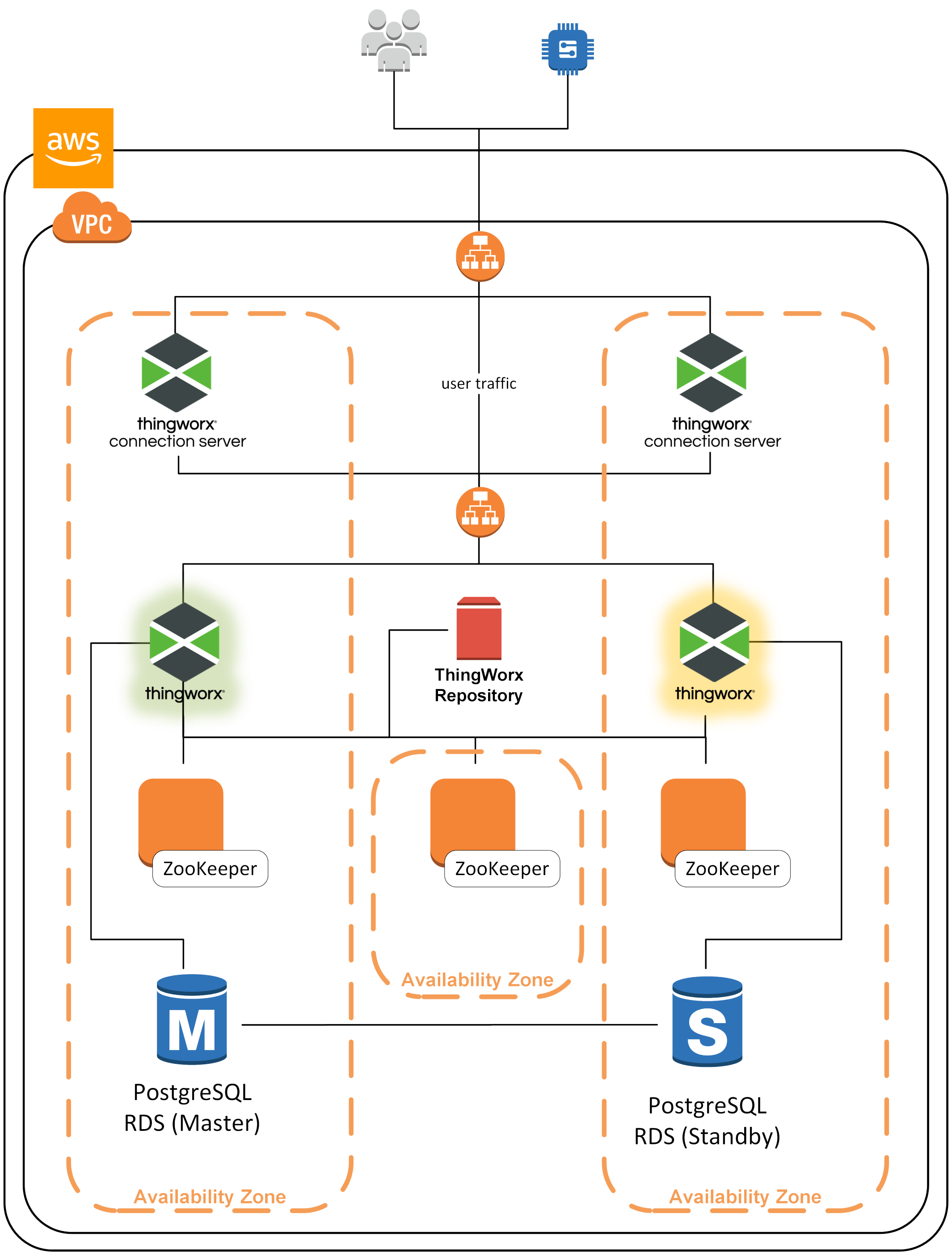
Enterprise Deployment



List of Components	Number of Components
AWS Region	1
AWS VPC	1
AWS Availability Zones	3

List of Components	Number of Components
Application Load Balancer	Needed when Connection Servers are part of deployment. Distributes device loads to Connection Servers and directs user traffic to ThingWorx Foundation Server.
ThingWorx Connection Server	2
ThingWorx Foundation Server	1
PostgreSQL RDS Instance	1
DataStax Enterprise	Required for high ingestion rates. Minimum of 5 nodes: 3 Cassandra and 2 Solr nodes spread across multiple availability zones. Ops Center node is optional.

High-Availability Deployment



List of Components	Number of Components
AWS Region	1
AWS VPC	1
AWS Availability Zones	3

List of Components	Number of Components
Application Load Balancer (ALB)	2 When Connection Servers are part of the deployment, one ALB is needed to distribute device loads to Connection Servers and direct user traffic to ThingWorx Foundation Server. The second ALB is required to direct Connection Server traffic to the active ThingWorx Foundation Server.
ThingWorx Connection Servers	2
ThingWorx Foundation Server	Minimum of 2: one active (in green), one passive (in yellow). There can be additional passive Foundation servers.
AWS EFS	Disk space to contain ThingWorx Storage repositories, accessed by all ThingWorx Foundation servers.
ZooKeeper	Minimum of 3. Should be in odd-numbered allotments spread across 3 availability zones.
Database	PostgreSQL RDS Master and Standby spread across multiple availability zones.

ThingWorx AWS IoT Connector Deployment

The ThingWorx AWS IoT Connector offers connectivity between ThingWorx Foundation and remote devices that communicate through the AWS device cloud. Once device data is available on ThingWorx Foundation, developers can use the ThingWorx Composer and Mashup Builder tools to build applications that monitor their devices, collect data through the connector, and use that data for business purposes.

Components

AWS IoT Components

AWS IoT provides secure, bidirectional communication between Internet-connected things (such as sensors, actuators, embedded devices, or smart appliances) and the AWS cloud. This allows you to collect, store, and analyze telemetry data from multiple devices.

- IoT MQTT Broker — Provides a secure mechanism for things and AWS IoT applications to publish and receive messages from each other. You can use either the MQTT protocol directly or MQTT over WebSocket to publish and subscribe. You can use the HTTP REST interface to publish.
- IoT Rule Engine — Provides message processing and integration with other AWS services. You can use a SQL-based language to select data from message payloads, process, and send the data to other services, such as ThingWorx.
- Amazon Kinesis — Allows you to collect, process, and analyze real-time, streaming data to get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit application requirements. Amazon Kinesis can ingest real-time data such as application logs, Website clickstreams, IoT telemetry data, and more into databases, data lakes, and data warehouses.
- Device Shadow — A JSON document used to store and retrieve current state information for a thing (device, application, and so on).

ThingWorx AWS IoT Components

- ThingWorx AWS IoT Connector — The ThingWorx AWS IoT Connector consists of the ThingWorx Foundation connection server, the AWS IoT Adapter, and an AWS IoT Adapter Extension. This package enables remote devices that are running an AWS IoT application to connect to ThingWorx Foundation. The AWS IoT Connector handles message routing for the devices that communicate through the AWS IoT system. It also handles message routing from the ThingWorx Foundation to devices via the device shadow and MQTT Broker of the AWS IoT system.

Deployment Considerations

The number of ThingWorx AWS IoT Connectors should be at most equal to the number of Kinesis shards. If there is only one Kinesis shard, then there should only be one EC2 instance running the AWS IoT Connector. For high-availability configurations, it should use two ThingWorx AWS IoT Connectors. For more details about deployment, refer to the “Deploying an AWS IoT Connector” chapter of ThingWorx AWS IoT Connector Installation and Operations Guide.

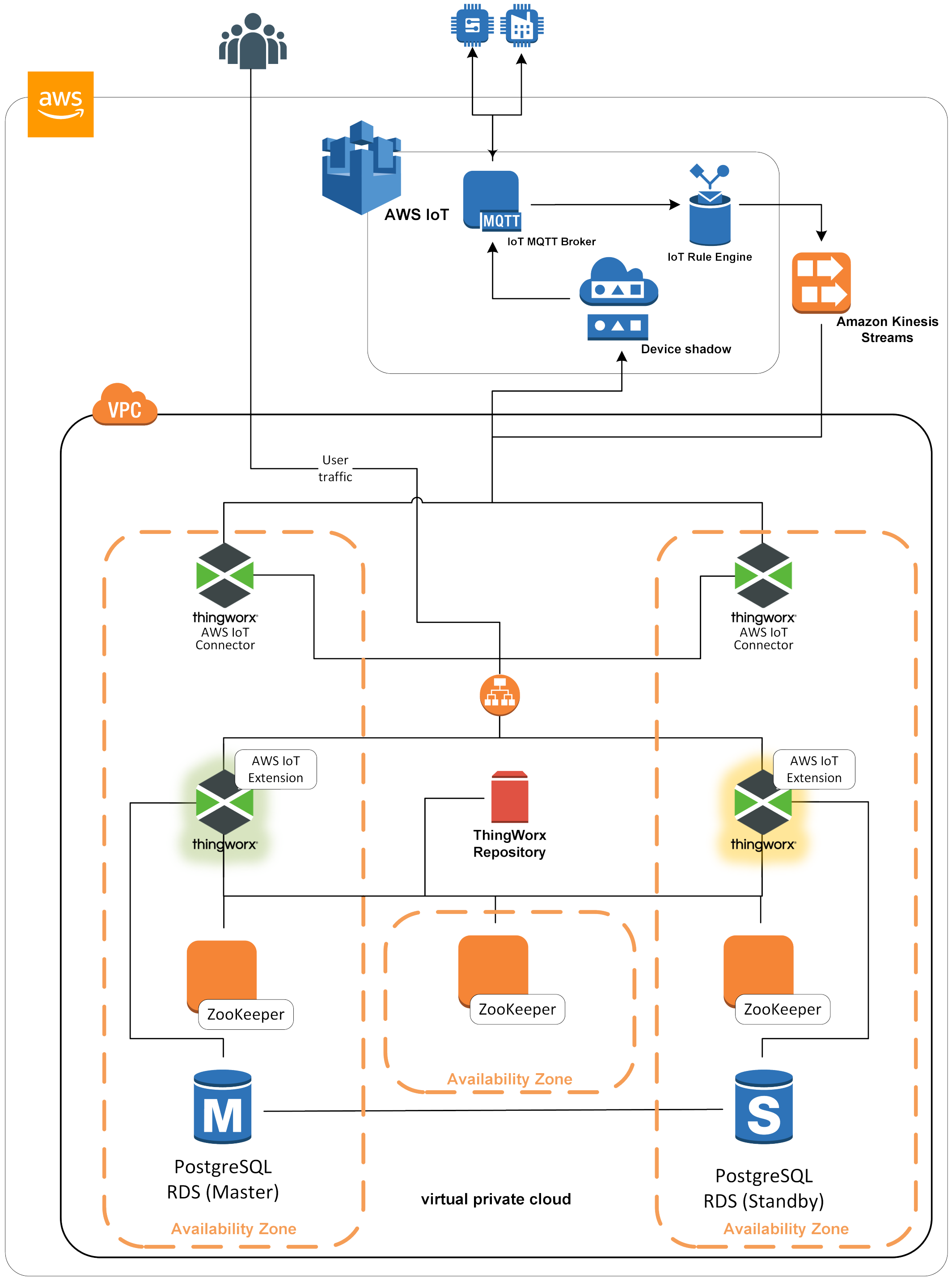
References

- ThingWorx AWS IoT Connector Installation and Operations Guide
- ThingWorx Connection Services Help Center

Reference Architecture

The following diagram is a high-availability deployment of ThingWorx Foundation that uses ThingWorx AWS IoT Connectors to access AWS IoT.

- Devices are sending content to AWS IoT.
- The ThingWorx AWS IoT Connectors are connected with the Kinesis shard(s) within AWS IoT to read messages from the devices and store the content.
- The ThingWorx AWS IoT Connectors submit the content to the active ThingWorx Foundation server through the Elastic Load Balancer (ELB).
- User interactions bypass AWS IoT, accessing ThingWorx Foundation through the ELB.
- The ThingWorx AWS IoT Connectors are established in different Availability Zones as part of the high-availability deployment.



List of Components	Number of Components
AWS Region	1
AWS IoT implementation	Consists of the following:

List of Components	Number of Components
	• IoT MQTT Broker • IoT Rules Engine • IoT Shadow
AWS Services	AWS Kinesis Streams (with 1 or more Kinesis shards)
AWS VPC	1
AWS Availability Zones	3
ThingWorx AWS IoT Connectors	• Minimum of 1 connector for operation • Minimum of 2 connectors for high availability
AWS Application Load Balancer	1 ALB to direct traffic to the active ThingWorx Foundation server.
ThingWorx Foundation Server (with AWS IoT Extension)	Minimum of 2: one active, one passive. There can be additional passive ThingWorx Foundation servers.
AWS File Storage	File server space to maintain ThingWorx Storage repositories, to be accessed by all Foundation servers.
ZooKeeper	Minimum of 3, deployed in odd number allotments and spread among 3 availability zones
PostgreSQL RDS	2 nodes: 1 master and 1 standby

7 ThingWorx Authentication Deployment

In this section, we review authentication solutions available to ThingWorx implementations. The solutions include out-of-the-box authentication, authentication through corporate LDAP (Lightweight Directory Access Protocol) or Active Directory, and authentication through an SSO (Single Sign-On) configuration.

Components

Directory Service

A directory service maintains a company's list of users and their authentication and authorization credentials. Microsoft's Active Directory, OpenLDAP, and Apache Directory Server are common implementations of directory services.

Central Authentication Server (CAS)

A CAS is a third-party tool that manages the authentication of users across a federation to allow users to access data from multiple applications by signing in only once. PTC supports a configuration of PingFederate in this role.

PingFederate is a third-party product provided by PTC as part of its Single Sign-On (SSO) solution. PingFederate acts as an authorization server that facilitates the exchange of SAML assertions and OAuth access tokens.

Identity Provider (IdP)

An IdP is a third-party tool that manages user identity data and supplies user information. An IdP can be a user-management system or an active directory that stores user names, passwords, and other credentials. The CAS references the IdP when authenticating a user.

Service Provider

A Web server from which you want to access information.

Resource Provider

An application within the SSO federation that contains protected data.

References

PTC Documents

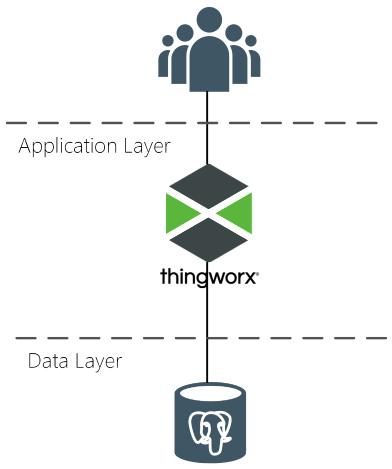
- PTC Single Sign-On Architecture and Configuration Overview

ThingWorx Help Center Links

- Authenticators — Authentication mechanisms within ThingWorx
- Directory Services Authentication — Configure ThingWorx to authenticate through a directory service
- Single Sign-On Authentication — Configure ThingWorx to use SAML authentication and OAuth delegated authorization.

Basic Authentication Architecture

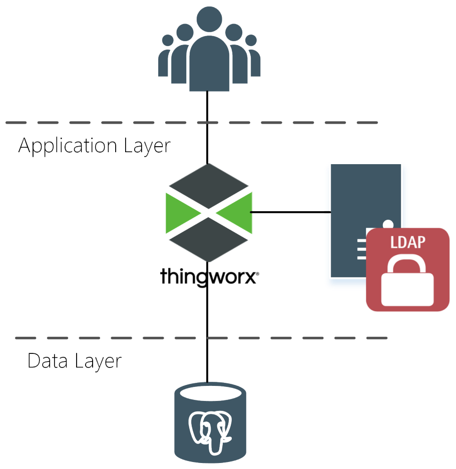
Basic Authentication for ThingWorx uses the authentication scheme within the Tomcat Application Server. From a deployment perspective, there are no additional requirements for hardware or software components.



List of Components	Number of Components
ThingWorx Foundation Server	1
Database	1

Architecture for Authentication Through LDAP

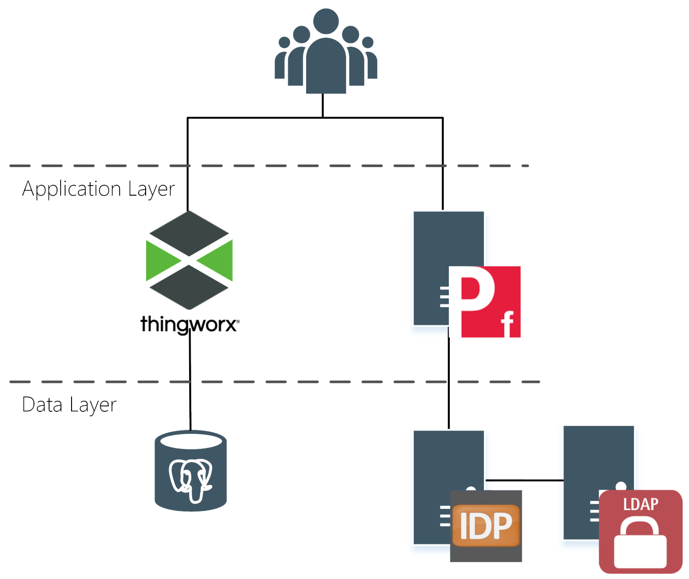
Another common authentication deployment is to use a company's corporate LDAP (Lightweight Directory Access Protocol) as the authentication source. In this case, ThingWorx is configured to connect to the corporate LDAP for authentication and authorization tasks.



List of Components	Number of Components
ThingWorx Foundation Server	1
Database	1
LDAP server	1

Architecture for SSO Authentication using PingFederate

ThingWorx 8 provides an integration with PingFederate, which can then be configured to operate with one or more Single Sign-On authentication systems. The following diagram describes a PingFederate system in a high-availability configuration:



List of Components	Number of Components
ThingWorx Foundation Server	1
ThingWorx Database	1
PingFederate server	1
Identity Provider service (IdP)	1
Directory Server (LDAP)	1

8 ThingWorx Analytics Deployment

As a part of the ThingWorx IoT technology platform, ThingWorx Analytics provides tools for embedding advanced analytics capabilities into smart, connected solutions. These tools can help analyze and understand the data collected from connected devices. They can analyze what is happening right now and why, what might happen next, and what can be done to change the outcomes.

Components

ThingWorx Analytics Server

The ThingWorx Analytics Server is an application that provides the analytic capabilities to understand the collected data and offer predictions.

ThingWorx Analytics Extension

The ThingWorx Analytics Extension contains a zipped archive of software, extension, and documentation files. This ZIP file must be downloaded and extracted before the extension can be imported into ThingWorx Foundation.

Deployment Considerations

ThingWorx Analytics 8.0 is offered as a standalone server through a set of Docker images that contain all necessary ThingWorx Analytics Server libraries and settings. This offering provides a simple installation process, low maintenance requirements, and is easy to scale by adding CPU and memory as needed. However, this configuration cannot be scaled by adding additional servers.

ThingWorx Analytics 8.0 can also be deployed in a distributed environment. This deployment model (with a minimum of six servers) provides greater capacity but is more complex to configure and maintain.

For either deployment choice, a separate PostgreSQL database should be used to avoid performance lags for both ThingWorx Foundation and ThingWorx Analytics.

Hard drive space for Analytics data should be maintained on external volumes for ease of backup and recovery as well as ease of upgrading to a new version of ThingWorx Analytics.

Once the ThingWorx Analytics server is established, the ThingWorx Analytics Extension should be deployed onto the ThingWorx Foundation server to complete the configuration.

References

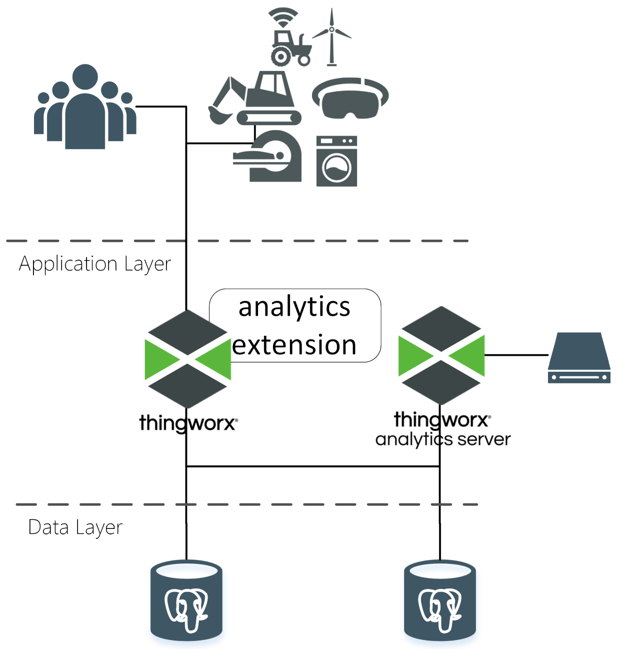
ThingWorx Analytics Deployment Mode Decision

ThingWorx Analytics Docker Installation Guide

ThingWorx Analytics Extension Installation Guide

Reference Architecture

The following diagram depicts a basic ThingWorx Analytics production deployment:



You can use any ThingWorx Foundation reference architecture in this guide for ThingWorx Analytics Server deployment.

List of Components	Number of Components
ThingWorx Foundation Server	1 (with Analytics Extension)
ThingWorx Foundation Database	1
ThingWorx Analytics Server	1
ThingWorx Analytics Database	1

9

Vuforia Studio Deployment

The deployments discussed here are focused on the server-side aspects of Vuforia Experience Service, the components that store and distribute Vuforia Studio experiences.

Components

Vuforia View

Vuforia View delivers experiences rich with 2D and 3D graphics, augmented reality (AR), and real-time product data. Vuforia View uses specialized markers called ThingMarks, that once scanned, immediately deliver relevant 2D, 3D, and AR Experiences. Experiences augment the view of your immediate surroundings with context-sensitive information and graphics, enabling you to interact directly with the things around you.

Vuforia Studio

Vuforia Studio is a web-native, easy-to-use tool for authoring domain and task-specific experiences that provide an integrated view of digital and physical product data, dashboards, and alerts with 2D, 3D, and augmented reality.

Vuforia Experience Service

Vuforia Experience Service is an enterprise-class, secure, and scalable server that publishes, updates, and deletes experience content generated through Vuforia Studio. It also identifies and delivers Experience content requested through Vuforia View.

Vuforia Global Experience Index (GXI)

The GXI allows for multiple Vuforia Experience Services to be queried by the Vuforia View application. The GXI creates and maintains an index of all Experiences that are maintained within the associated Vuforia Experience Service. When the GXI receives a ThingMark query, it identifies the Vuforia Experience Service that hosts the desired Experience and redirects the query to that specific Vuforia Experience Service. By default, Vuforia View uses the GXI, but it can also be manually configured to reference a different Vuforia Experience Service.

Deployment Considerations

- ThingWorx Foundation must be installed and established before installing the Vuforia Experience Service.
- For production systems, it is recommended to run ThingWorx Foundation and Vuforia Experience Services on separate servers.
- For production systems, it is recommended to use PostgreSQL as the database associated to the Vuforia Experience Service.
- To avoid performance lag, it is recommended to have separate PostgreSQL servers for ThingWorx Foundation and Vuforia Experience Service.

References

- Getting Started with Vuforia Studio
- Installing and Deploying Vuforia Studio Experience Service

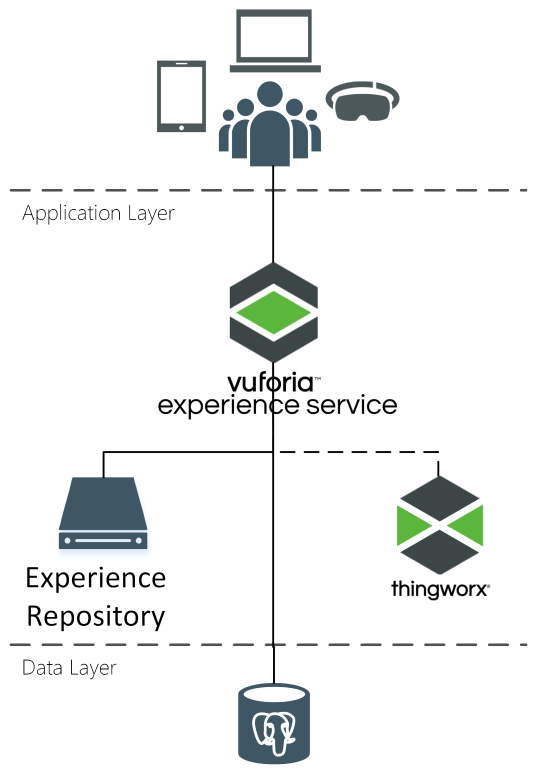
Reference Architectures

The following diagrams focus on Vuforia Experience Service, and only depict a simplified reference to ThingWorx Foundation. However, any ThingWorx Foundation architecture shown in this guide may be used instead.

Production Deployment

The following diagram depicts a production deployment of the Vuforia Experience Service. It includes the following features:

- One Vuforia Experience Service
- Hard disk space to store Experiences
- PostgreSQL database to maintain Experience metadata
- Access to ThingWorx Foundation Server to manage authentication and authorization

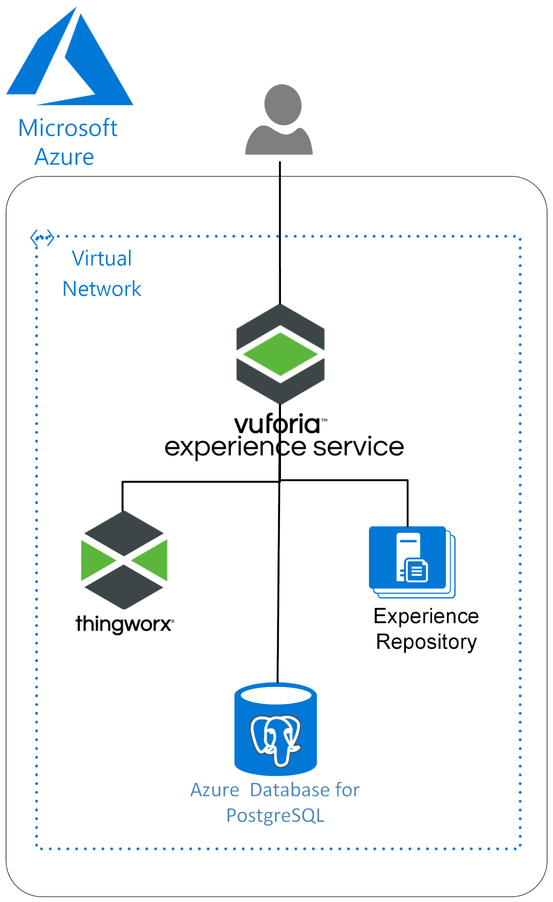


List of Components	Number of Components
Vuforia Studio Experience Service	1
ThingWorx Foundation Server	1
Enterprise NFS Repository	1
Vuforia Experience Service Database	1

Cloud Deployment

Vuforia Experience Service can be deployed in cloud services such as Azure and AWS.

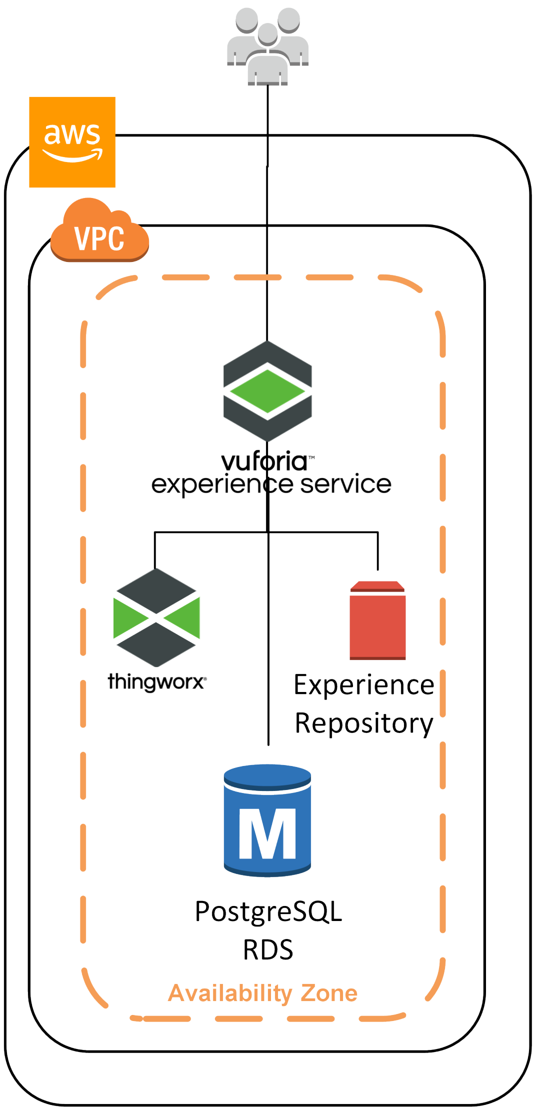
Azure Deployment



List of Components	Number of Components
Azure Region	1
Azure Virtual Network	1
Vuforia Experience Service	1 Virtual Machine
Azure Files	1 storage repository attached to the

List of Components	Number of Components
	Vuforia Experience Service virtual machine
ThingWorx Foundation Server	1 Virtual Machine
Azure Database for PostgreSQL	1

AWS Deployment



List of Components	Number of Components
AWS Region	1
AWS VPC	1
Vuforia Experience Service	1 EC2 instance

List of Components	Number of Components
AWS EFS	1 storage repository attached to the Vuforia Experience Service EC2
ThingWorx Foundation Server	1
PostgreSQL RDS	1

10

ThingWorx Navigate Deployment

The ThingWorx Navigate product family combines a ThingWorx Platform server with one or more resource providers to provide simplified, role-based applications. Within this product family, the ThingWorx Navigate View PLM App Extension offers several applications for viewing PLM (Product Lifecycle Management) data from Windchill. The applications are simpler to use than Windchill alone and are expected to be adopted by a larger number of users with less training required.

Components

ThingWorx Navigate View PLM App Extension

The ThingWorx Navigate View PLM App Extension uses a resource provider (Windchill) for PLM information and an optional external identity provider (IdP) for authentication. The identity provider can also be Windchill to consolidate user maintenance.

Windchill Product Family

- Windchill: PTC's Product Lifecycle Management application.
- Windchill Directory Server: A directory server that is a required piece of the Windchill deployment.
- Creo CAD Worker: An optional service for Windchill, which produces Creo View content from recently-added CAD content. The ability to utilize viewables is included in most of the Navigate View PLM App mashups.

Deployment Considerations

The ThingWorx Navigate View PLM App Extension uses the ThingWorx platform as a server. It can be deployed in the following ways:

- Monolithic deployment — Combines the Windchill application, the ThingWorx platform, and the Navigate extension on one host. Databases for the applications may still be on separate hosts. This deployment is recommended for non-production environments or small deployments.
- Distributed deployment — Uses separate hosts for the ThingWorx platform (ThingWorx Navigate server) and the Windchill application, allowing horizontal scaling of Windchill. Communication between the applications uses either a certificate-based SSL connection or a trusted non-SSL connection (non-production). This deployment is recommended for production systems.

References

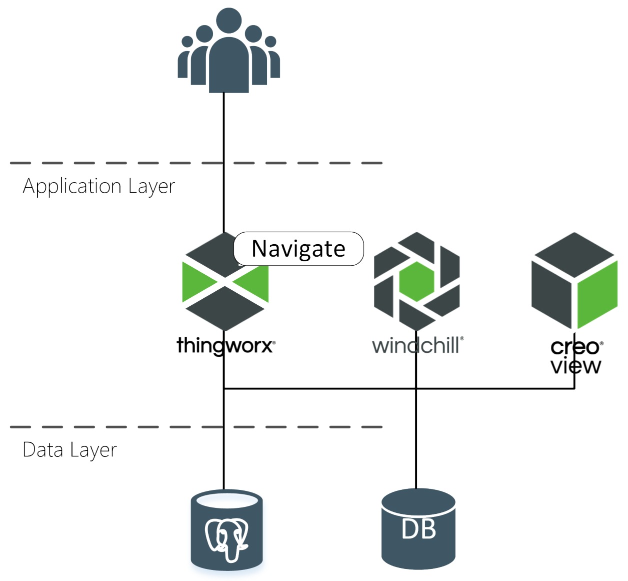
- ThingWorx Navigate Platform Sizing Guide
- Windchill Architecture Overview

Reference Architectures

ThingWorx Navigate Production Deployment

The following diagram depicts a straightforward deployment of ThingWorx Navigate, which includes the following features:

- The user accesses Windchill content through ThingWorx Navigate. In this design, Windchill is not accessible outside the firewall.
- Windchill has access to additional tools it may need — Windchill Directory Server (required) and a Creo View CAD worker (optional).

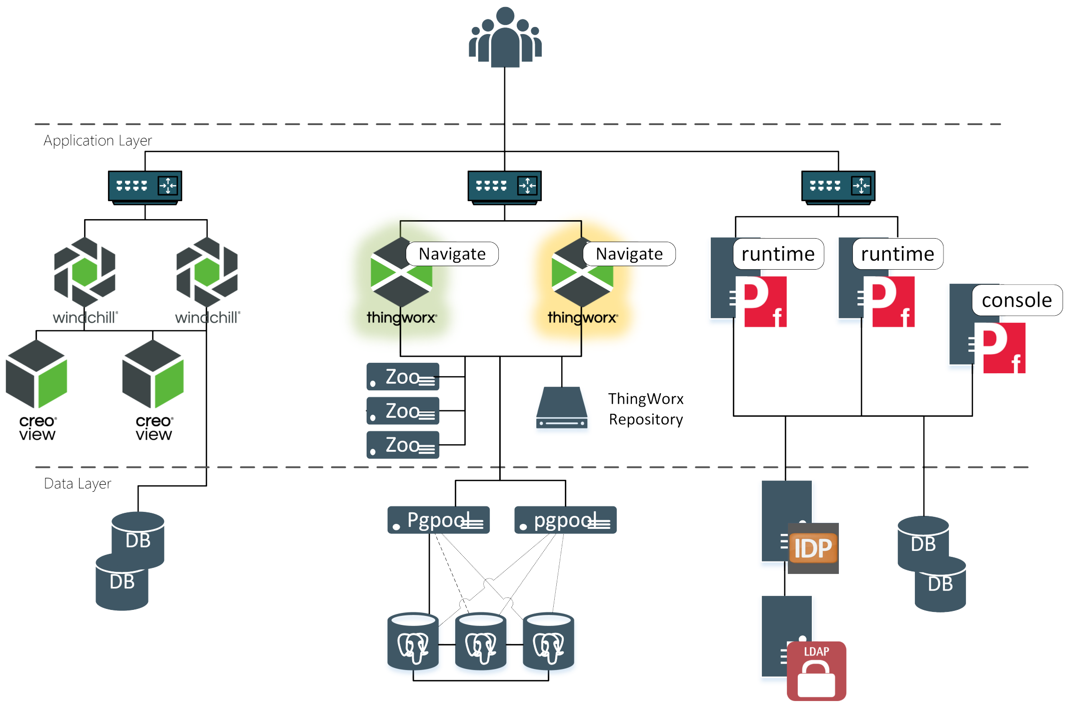


List of Components	Number of Components
ThingWorx Foundation Server (with Navigate Extension)	1
ThingWorx PostgreSQL Database	1
Windchill CAD Worker	1
Windchill Database	1
Windchill Directory Server	1 (installed on Windchill Server)
Windchill Server	1

ThingWorx Navigate High-Availability Deployment

The following diagram depicts a high-availability deployment for four combined products: Windchill, ThingWorx Navigate, PostgreSQL (for the ThingWorx Navigate solution), and PingFederate. It includes the following features:

- Load balancers to direct traffic to either the Windchill cluster, the ThingWorx Navigate cluster, or the PingFederate cluster
- Multiple Windchill nodes for a Windchill cluster configuration
- Multiple CAD worker nodes
- Two ThingWorx Navigate servers—one as the primary server and the other in standby
- Three ZooKeeper nodes to monitor the ThingWorx Navigate servers and select a new active server as needed
- Two pgpool II nodes to distribute database requests to the available PostgreSQL servers
- Three PostgreSQL servers — one actively writing new content, one serving read requests, and one on standby. All three remain in sync.
- Two PingFederate runtime nodes to receive and respond to SSO requests
- PingFederate manages Windchill and ThingWorx Navigate's access to Identity Providers



List of Components	Number of Components
Load Balancer	3 (1 for Windchill, 1 for ThingWorx Navigate, 1 for PingFederate)
ThingWorx Foundation Server	2 (1 active, 1 passive)
Enterprise NFS Repository	1 (for ThingWorx Storage repositories)
ZooKeeper Nodes	Minimum of 3

List of Components	Number of Components
pgpool II	2 nodes
ThingWorx PostgreSQL Database	3 nodes
Windchill Server	Minimum of 2
Windchill Database	2 (HA for database not accurately depicted)
Windchill CAD Worker	2
PingFederate Runtime	2
PingFederate Console	1
PingFederate Database	2 (HA for database not accurately depicted)
IdP	1 + n (HA not depicted, depends on IdP product and customer deployment)
LDAP	1 + n (HA not depicted, depends on LDAP product and customer deployment)

11 Distributed ThingWorx Deployment

ThingWorx is designed for federated deployment where the components of an enterprise application can run where it is the most appropriate for performance and autonomy. This design feature makes it easy to provide a distributed and tiered data storage and analysis capability.

For example, a central ThingWorx server can connect to each of the plant-level ThingWorx servers to pull information together and aggregate it for display of regional or corporate-level views. Then, as users drill down into the data, the plant-level server can propagate the details to the central server.

There are multiple deployment scenarios supported by ThingWorx, including cloud (PTC or third party) and on premise (on-site or in a corporate data center). If the customer solution is deployed globally, PTC recommends that servers be geolocated for optimal performance.

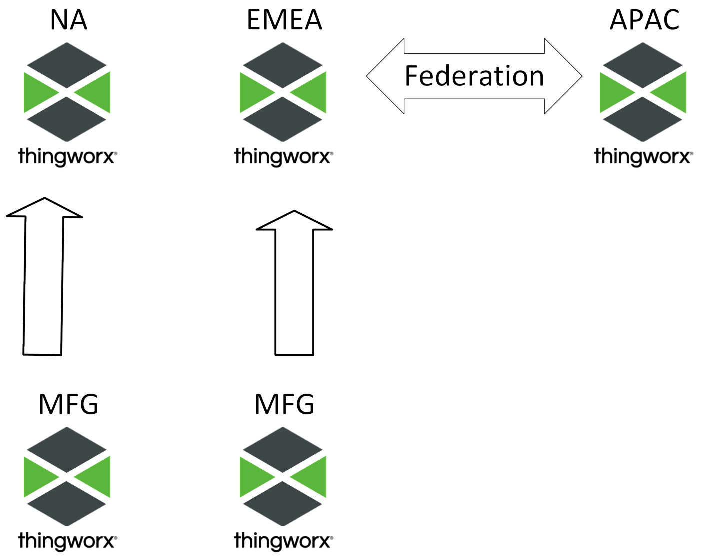
Federation is composed of multiple elements:

- ThingWorx request servers — where all incoming requests are routed. A request may be initiated by a user accessing a Mashup or devices communicating with the Things. These servers scale based on number of connections and volume of data requests.
- Thing servers — where the Things are running in memory and communicating with the request servers. These are memory-intensive servers since the actual logic is running on them. They can also be scaled horizontally based on memory and CPU limitations.
- Data servers — where the actual application data is stored. These servers can also be scaled based on the amount of storage access that is required.

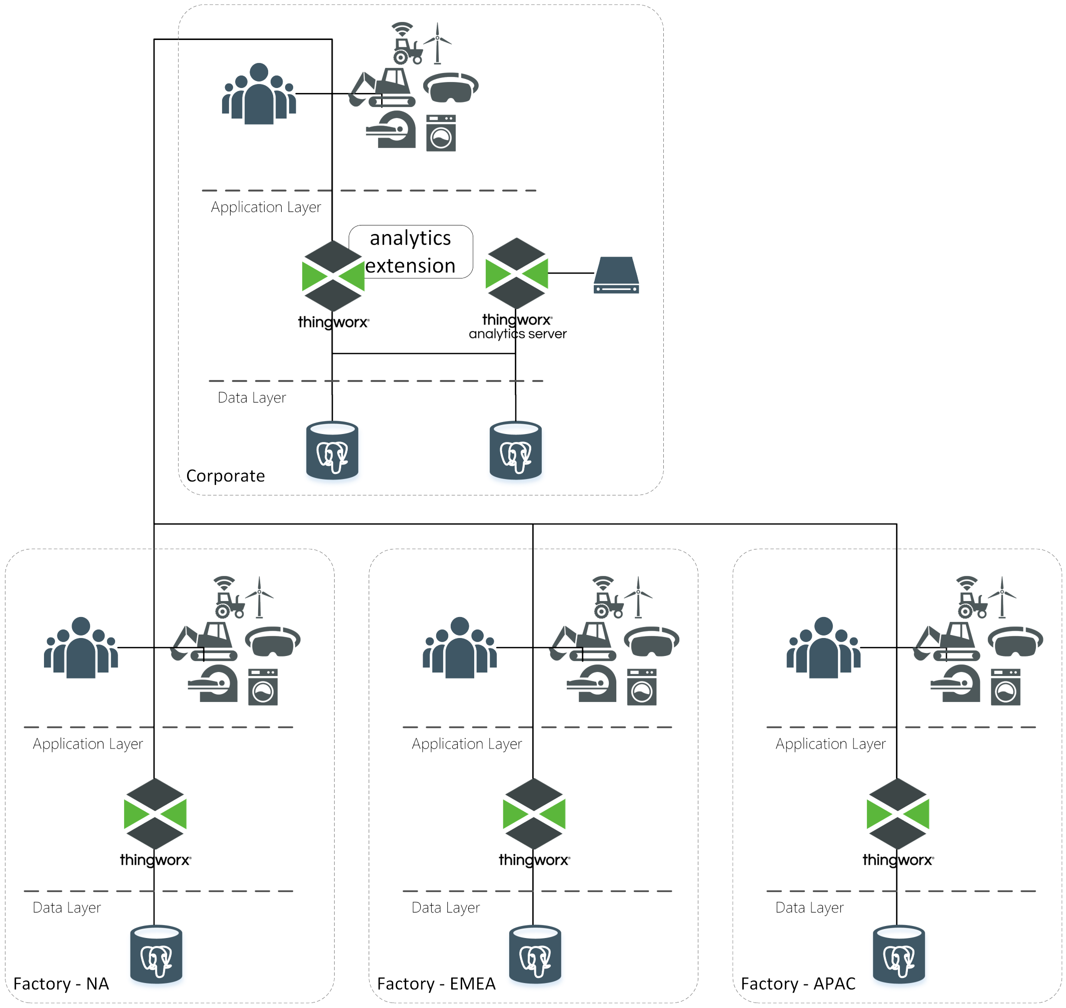
These different capabilities can be rolled up into one server or delegated across many servers to achieve the desired performance with the existing number of devices.

Federation can be used in many scenarios, such as handling large workloads (for example, high frequency stream writes) or connecting independent regional servers or server clusters. Federation is used to aggregate data to another system to make all relevant data accessible through one system.

In the following diagram, federation is used to aggregate data from local manufacturing sites as well as geographically separated Enterprise platforms.



ThingWorx Federated Example: Connected Factories



List of Components (Per Factory)	Number of Components
ThingWorx Foundation Server	1
ThingWorx Database	1

List of Components (Corporate)	Number of Components
ThingWorx Analytics Database	1
ThingWorx Analytics Server	1
ThingWorx Foundation Server (with Analytics Extension)	1
ThingWorx Database	1