Microsoft

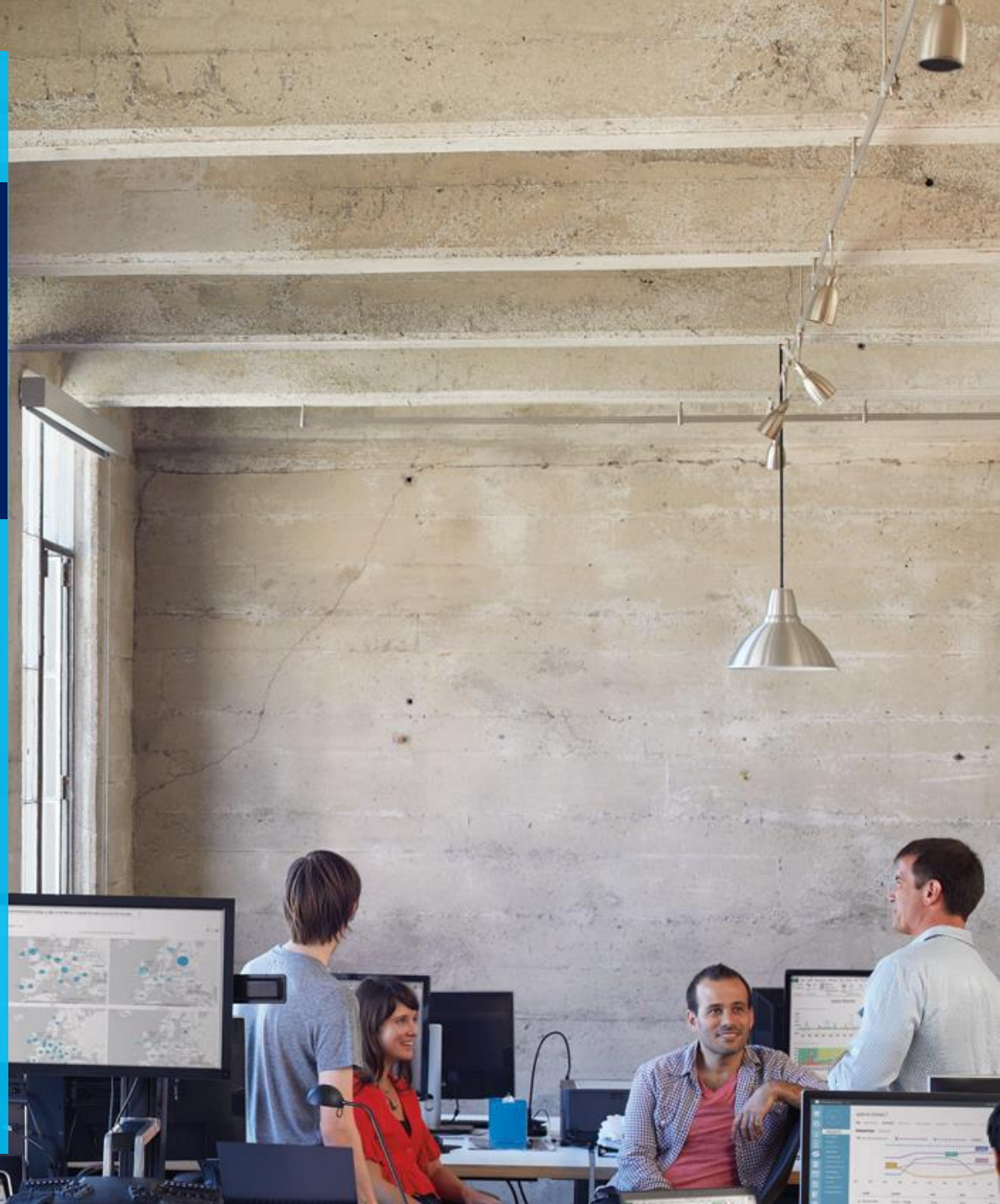
Date: 28th Aug 2026

Microsoft Azure Power Lunch

Today's Topic:
Azure Web PubSub Chat (Public Preview)

Series website: http://azurepowerlunch.com/

On-Demand Session Information (Recordings) are available here:
Azure Power Lunch YouTube Channel



Naveed Zaheer – Cloud Solution Architect



- Been with Microsoft for more than 22 years and in software industry for more than 31 years
- Spent 10+ years with Microsoft Consulting Services in areas such as AppDev, Distributed Applications, SOA and Cloud Applications
- Working with Azure since its inception
- Currently in a Cloud Solution Architect role



Azure Web PubSub chat

Build scalable, reliable chat experiences with purpose-built APIs

PUBLIC PREVIEW

Overview, architecture, adoption guidance, and key takeaways

Azure Web PubSub: Choose the development experience that fits your application

Capability	Choose it when you want to	Developer value
Web PubSub (base service)	Design a custom real-time application by using connections, users, groups, and events.	Get flexible messaging primitives without operating WebSocket infrastructure.
Socket.IO on Azure	Use the Socket.IO programming model or move an existing Socket.IO application to Azure.	Keep familiar Socket.IO APIs while Azure manages connection scaling and coordination.
MQTT support	Connect MQTT clients over WebSocket or exchange messages between MQTT and native Web PubSub clients.	Reuse MQTT libraries and add protocol interoperability without building a translation layer.
Web PubSub chat	Build chat with rooms, members, ordered messages, and message history.	Start with chat-specific APIs and managed capabilities instead of assembling a chat backend yourself.

Agenda

A concise overview designed to lead into a technical demo.

01 Why managed chat now

02 What Web PubSub chat provides

03 How the architecture works

04 Chat hub vs. standard hub

05 Security, reliability, and data ownership

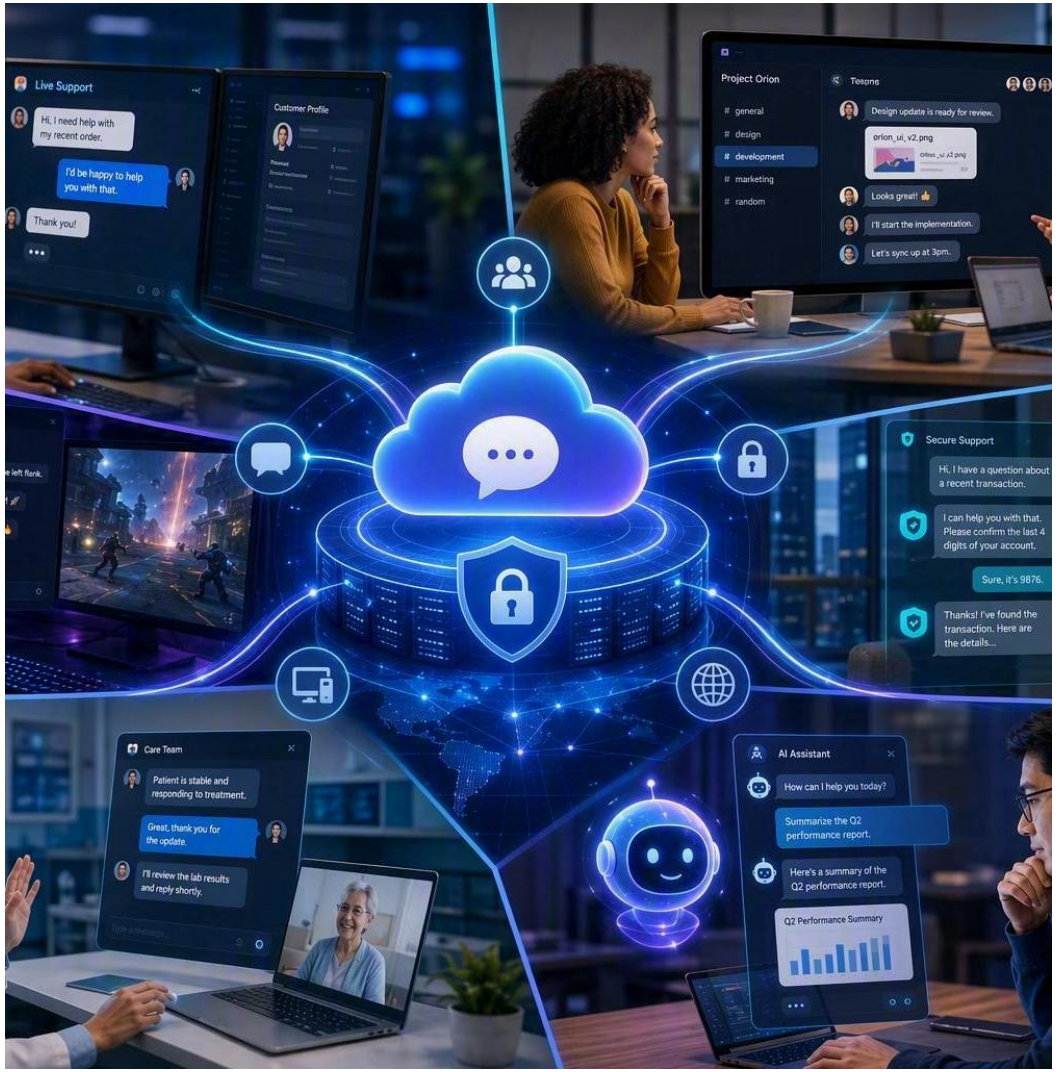
06 Getting started and key takeaways

Why managed chat?

Production chat requires more than a WebSocket connection.

- **Real-time transport**
- **Fan-out and ordering**
- **Connection lifecycle**
- **Message persistence**
- **Cross-platform client experience**

Web PubSub chat provides a higher-level, chat-specific abstraction so teams can focus on the experience instead of assembling the messaging plumbing.



Chat-native capabilities

Purpose-built client and server APIs use familiar chat concepts.

Rooms

One-to-one and group spaces with membership

Messages

Ordered delivery plus persistent room history

Members & users

Identity and membership modeled directly

Roles & permissions

Built-in and custom access control

Reliability

Reconnect and recover after interruptions

Scale

Azure Web PubSub scaling and geo-replication foundations

How it works

The service separates responsive client chat from trusted backend control.



Client experience

JavaScript client SDK connects to a chat hub, exchanges messages, loads history, and manages room interactions.

Managed chat layer

Chat hub handles rooms, messages, members, delivery, fan-out, and retrieval.

Trusted backend

Chat REST API supports administrative and server workflows such as roles, permissions, and governance.

Persistent data

Selected Azure Storage account retains chat data through managed identity access.

Choose the right abstraction

Both options run on Azure Web PubSub, but optimize for different workloads.

Chat hub

Conversation-centered applications

- ✓ Built-in rooms and membership
- ✓ Built-in message persistence
- ✓ Chat-specific APIs
- ✓ Chat-focused client experience

Standard hub

Custom or non-chat real-time workloads

- ✓ Full protocol control
- ✓ Custom event models
- ✓ Telemetry and signaling
- ✓ Notifications and live dashboards

Getting started

A practical path from resource configuration to the first room message.

1. **Create or open**
   Azure Web PubSub resource
2. **Link storage**
   Persistent Storages
3. **Add a Chat Hub**
   Associate it with storage
4. **Generate access**
   Client URL for testing
5. **Install SDK**
   @azure/web-pubsub-chat-client
6. **Connect and chat**
   Create a room and send

Production guidance

Issue client access URLs from an authenticated backend, derive the chat user ID from the signed-in identity, and use managed identity or Microsoft Entra ID for keyless server authentication.

Demo

Key takeaways

1 **Faster delivery**

Use chat-native primitives instead of designing custom protocols and persistence.

2 **Managed reliability**

Ordered delivery, reconnection, fan-out, and history are built into the chat experience.

3 **Control and ownership**

Backend APIs govern permissions while persisted chat data stays in the selected Azure Storage account.

4 **Flexible architecture**

Choose a chat hub for conversations, a standard hub for custom real-time events, or use both.

Recommended next step: run the quickstart, then validate identity, permissions, storage, and operational requirements for your scenario.

Resources

- Announcing Azure Web PubSub chat in public preview
- What is Web PubSub chat?

Q & A