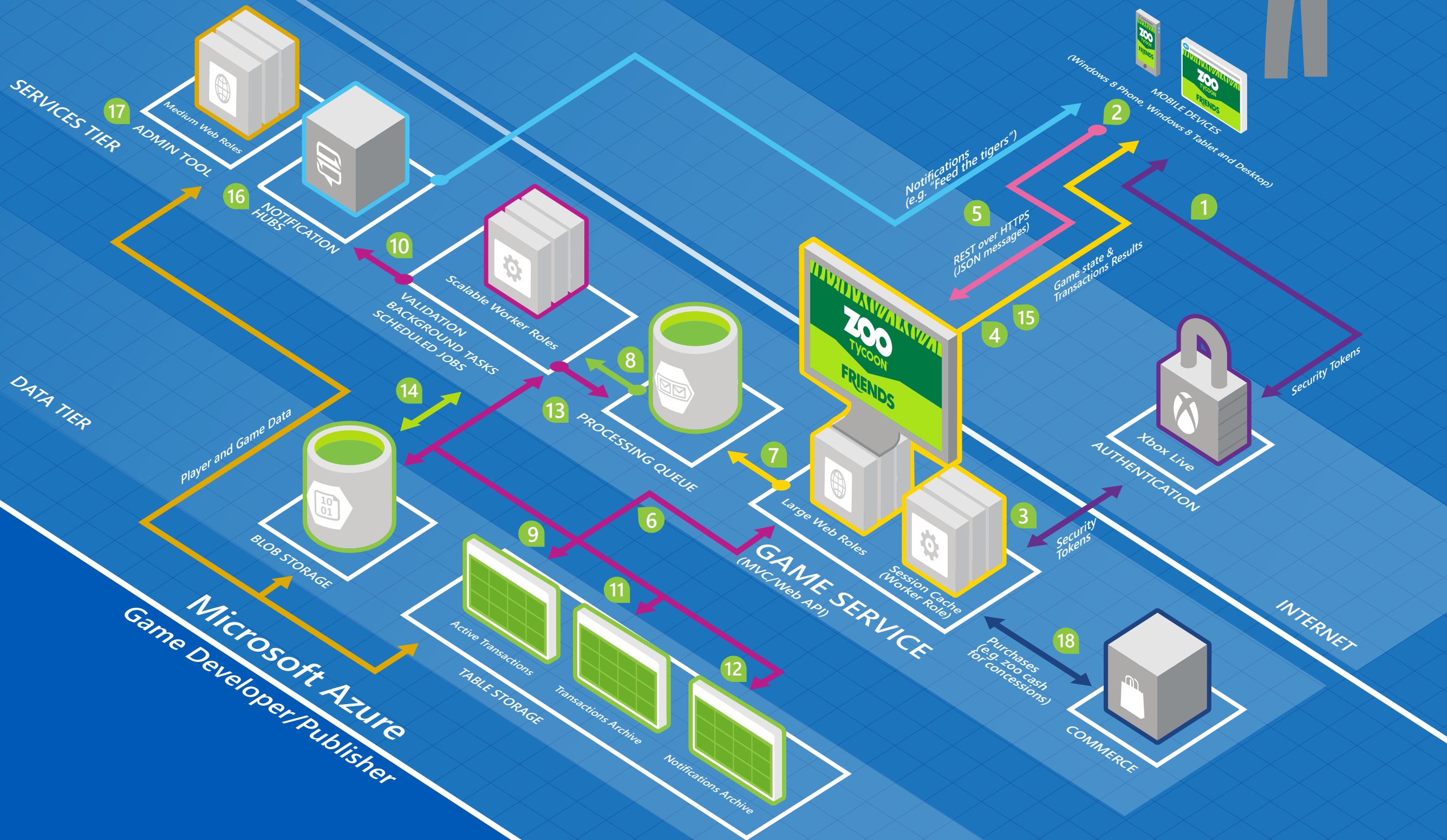# Zoo Tycoon Friends Architecture

The game **Zoo Tycoon Friends** lets players adopt virtual animals and build habitats and exhibits, make in-game purchases of items, and invite friends to visit their zoo. The Game service validates every action taken by the player since many involve purchases with "zoo cash." To do this, the Game service replays the players' actions to test the actions and the results.

SERVICES TIER

DATA TIER

17 ADMIN TOOL

Medium Web Roles

16 NOTIFICATION HUBS

10

VALIDATION BACKGROUND TASKS SCHEDULED JOBS

Scalable Worker Roles

Player and Game Data

14

BLOB STORAGE

13 PROCESSING QUEUE

8

Notifications (e.g. "Feed the tigers")

5 REST over HTTPS (JSON messages)

2

(Windows 8 Phone, Windows 8 Tablet and Desktop)

MOBILE DEVICES

15 Game state & Transactions Results

4

1 Security Tokens

Xbox Live

AUTHENTICATION

GAME SERVICE (MVC/Web API)

Large Web Roles

7

3 Security Tokens

Session Cache (Worker Role)

9

6

11

Active Transactions

Transactions Archive

12

Notifications Archive

TABLE STORAGE

18 Purchases (e.g. zoo cash for concessions)

COMMERCE

INTERNET

Microsoft Azure

Game Developer/Publisher

1. Player authenticates to Xbox Live using a Microsoft account and receives a token.

2. The client app contacts the Zoo Tycoon Friends Game service with the token. All messages are sent via REST APIs over HTTPS.

3. The service validates the token with Xbox and generates a Session ID that is returned to the client, and is also stored in the session cache.

4. The Game service sends the current game state (from blob storage) to the client app so the player can begin playing.

5. The player's actions (via REST APIs) are sent to the Game service. Each action is now a transaction that must be validated by the service.

6. Transactions are stored in the active transactions table.

7. A request is placed in the processing queue.

8. A worker role pulls the request from the queue.

9. The worker role retrieves all transactions for the same user from the active transactions table.

10. The worker role validates all batched transactions by "replaying" the action and testing the result.

11. After validation, the processed transactions are stored in the transactions archive then deleted from the active transactions table.

12. A success/failure notification is stored in the notifications storage table. The Game service checks the table when pinged by the client.

13. The worker role deletes the message from the processing queue. A new game state is generated. Any actions that require notifications are handled by a background task.

14. The new game state for the player is stored in the blob storage.

15. The Game service sends the validated results to the player. If an action fails, the player is notified, and the failed transaction is rolled back on the client app.

16. A background task triggers notifications sent to the user.

17. The Admin service performs actions as required on the system using data stored in blobs and table storage.

18. When a player buys virtual currency from the Windows Store, the Game service web role validates the receipt through the Commerce service and credits the player with virtual currency.