Developing Applications - Google App Engine

Google App Engine is a cloud platform for creating and deploying applications quickly and efficiently. Here's a breakdown:

Building Apps with Google App Engine

- **Easy Development**: Write Python code, tweak some HTML, and deploy your app in minutes.
- **Python-Friendly**: Though Python knowledge helps, it's not mandatory. Python is simple and similar to other scripting languages.
- **Java Support**: Java is also supported, but it's more complex and often costs more for hosting.

Key Features of Google App Engine

1. **Simple Applications**:
    - Best for lightweight apps that don't require complex computations.
    - Integrated with Python for basic database operations like storing and retrieving user data.
2. **Limitations**:
    - No support for writing directly to the file system or creating subthreads (to prevent errors by inexperienced developers).
    - Limited database functionality:
        - Basic search and store operations.
        - No support for joins in databases.
        - Python syntax is used instead of SQL, which limits the use of SQL tools for reports or graphs.
3. **AJAX and Web Services**: While mentioned in the documentation, support is minimal.

Payment and Usage Limits

- Google provides free usage within limits (e.g., 200 million megacycles of CPU per day).
- Exceeding limits incurs charges, but costs may rise unexpectedly due to:
    - Inter-server traffic slowing down performance.
    - Simultaneous access by multiple users increasing resource usage.
- **Scaling**: App Engine scales automatically by adding new servers as demand rises.

Ideal Use Cases:

- Best for **small, simple apps** that need scalability.
- Not suitable for apps with heavy computation or complex backend processes.

Google App Engine with Force.com

Google and Salesforce.com partnered to integrate **Google App Engine** with **Force.com**, creating a powerful platform for web and business applications.

Force.com + Google App Engine

1. **What it Offers**:
    - Tools and services for developing web and business apps entirely in the cloud.
    - Combines Google's frontend capabilities with Salesforce's enterprise backend tools.
2. **Benefits**:
    - **Scalability**: Apps can grow easily as traffic and data storage needs increase.
    - **Developer Support**: Force.com provides:
        - Python libraries to read/write to Salesforce.
        - APIs for mobile, analytics, user authentication, and security.
3. **Collaboration**:
    - Apps can use data and features from both Google and Salesforce systems.
    - Examples include integrating consumer-oriented apps on App Engine with enterprise data on Force.com.
4. **Resources for Developers**:
    - Python library documentation.
    - Example Python code to access Force.com.
    - Testing tools and FAQs for best practices.

Google Gears

Google Gears was an open-source browser extension that allowed developers to create **offline web applications**.

Key Features of Google Gears

1. **Offline Functionality**:
    - Enables apps to work without an internet connection or with unreliable connections.
    - Example: Google Reader was enhanced with offline capabilities using Gears.
2. **APIs**:
    - Introduced JavaScript APIs for:
        - Data storage.
        - Application caching.
        - Multithreading.
3. **Cross-Platform Support**:

    - Works on all major browsers (Chrome, Firefox, Opera) and operating systems (Windows, Mac, Linux).
4. **Goal**: Help the industry move toward a single standard for offline capabilities.

Developing Applications - Microsoft Azure

Microsoft Azure is a comprehensive cloud platform for developers to build, deploy, and manage applications.

What is Azure?

- A cloud platform hosted in Microsoft's data centers.
- Provides tools for building applications that run partially or fully in the cloud.
- Supports integration with **on-premises** systems.

Key Features of Azure

1. **Infrastructure**:
    - High availability and dynamic scaling to match usage needs.
    - Pay-as-you-go pricing model.
2. **Developer Tools**:
    - Integrated with Microsoft Visual Studio and .NET Framework.
    - Open architecture supports multiple programming languages (e.g., HTTP, REST, SOAP, XML).
3. **Applications**:
    - Build apps for:
        - Web.
        - Mobile.
        - PCs and servers.
        - Hybrid solutions (cloud + on-premises).
4. **Microsoft Cloud Applications**:
    - Azure also hosts ready-to-use cloud apps like:
        - **Windows Live**.
        - **Microsoft Dynamics**.
        - **SharePoint Online** and **Exchange Online**.

Azure Services

1. Live Services

- A set of tools for managing user data and application resources.
- Enables developers to build **social applications** across devices.

2. Microsoft SQL Services

- Extends SQL Server capabilities to the cloud.
- Supports relational queries, search, and data synchronization for remote users and business partners.

3. Microsoft .NET Services

- Allows developers to create loosely coupled cloud applications.
- Includes:
    - **Access Control**: Secures applications.
    - **Service Bus**: Enables communication between apps and services.
    - **Workflow Execution**: Manages cloud-hosted workflows.

4. SharePoint and Dynamics CRM Services

- Tools for collaboration and customer relationship management.
- Developers can use Visual Studio to integrate these services into apps.

Azure Architecture

Azure has a layered architecture:

Layer Zero (Global Foundational Services - GFS):

- Acts like a hardware abstraction layer.
- Interfaces directly with servers.

Layer One (RedDog):

- The Azure operating system, managing:
    - **Storage** (file systems).
    - **Fabric Controller**: Deploys and provisions resources.
    - **Virtualized Computing**.
    - **Development Environment**: Lets developers emulate Azure on their desktops.

Layer Two (Building Blocks):

- Services like LiveMesh that developers can use to build on top of Layer One.

Layer Three (Applications):

- Hosted apps like SharePoint Online, Dynamics CRM, and Exchange Online.
- Third-party apps can also run at this layer.

Scaling with Azure

Azure supports two types of scaling:

1. Vertical Scaling (Scale-Up)

- Add more resources to a single system (e.g., more CPU, RAM).
- Suitable for processor- or memory-intensive apps.

2. Horizontal Scaling (Scale-Out)

- Add more individual systems (e.g., servers).
- Suitable for distributed applications like web servers.
- Helps handle I/O bottlenecks.

Trade-Offs:

- Vertical scaling is limited by hardware constraints.
- Horizontal scaling adds management complexity but is more cost-effective for large-scale apps.

Intuit QuickBase

Intuit QuickBase is a **low-code/no-code platform** designed for creating tailored business applications without requiring extensive technical skills.

Key Features

1. **Ease of Use**:
    - Consultants and businesses can build or customize applications without coding expertise.
    - Over **200 templates** are available for quick customization.
2. **Efficiency**:
    - Applications can be developed in **one-fourth the time**, at **half the cost**, and with **double the profit margins** compared to traditional methods.

    - SaaS (Software as a Service) implementation is faster and less costly.
3. **Business Consultant Program**:
    - Provides training, lead-generation tools, and partner relationship management to consultants.
    - Includes a **free version of QuickBooks Online** to help consultants manage their own businesses.
4. **Target Audience**:
    - Focuses on entrepreneurs, industry experts, and business process consultants.
    - Shifts software customization from IT experts to domain specialists.

Cast Iron Cloud

The **Cast Iron Cloud** is an integration platform that connects SaaS applications with other enterprise systems, offering **Integration as a Service (IaaS)**.

Key Features

1. **Integration Flexibility**:
    - Connects cloud-based, on-premise, and legacy systems efficiently.
    - Provides options for **cloud-based integration** or **on-premise appliances**.
2. **Prebuilt Templates**:
    - Includes a library of **Template Integration Processes (TIPs)** for common SaaS integration scenarios.
    - TIPs allow users to deploy integrations in minutes without custom coding.
3. **Benefits**:
    - No need for investing in infrastructure or middleware expertise.
    - Simplifies data integration, process workflows, and real-time monitoring.
4. **Applications**:
    - Frequently used with platforms like **Salesforce CRM** and **NetSuite ERP** for integrating vertical and legacy systems.
5. **Self-Guided Wizard**:
    - Enables non-technical users to customize TIPs based on specific requirements, similar to tools like Intuit TurboTax.

Bungee Connect

Bungee Connect is a platform for **developing, deploying, and hosting web applications** on a multitenant grid infrastructure.

Key Features

1. **All-in-One Development**:
    - Combines development, testing, deployment, and hosting in one platform.
    - Reduces time-to-market by up to **80%**.
2. **Cloud-Based Functionality**:
    - Accessible entirely through a browser with no downloads or plug-ins required.
    - Supports high interactivity and robust security.
3. **Automated Integration**:
    - Seamlessly integrates **SOAP/REST web services** and databases like **MySQL** and **PostgreSQL**.
4. **Utility Pricing Model**:
    - Businesses pay based on actual app usage.
    - Costs range from **$2 to $5 per user per month** for business applications.
5. **Collaboration Tools**:
    - Built-in team collaboration features.
    - Deep instrumentation for analyzing app performance and usage patterns.

Developing Applications on Google App Engine

Google App Engine is a **Platform as a Service (PaaS)** that simplifies app development and deployment using **Python**.

Getting Started

1. **Prerequisites**:
    - Install Python 2.5 and the Google App Engine SDK.
    - The SDK includes tools for local testing (development server) and uploading apps to the cloud.
2. **Developing an App**:
    - Apps use **CGI standards** to handle web requests and responses.
    - Example:
        - Create a directory (e.g., heresjohnny).
        - Write a Python script (heresjohnny.py) to respond with an HTTP header and a message.
3. **Configuration File**:

Use app.yaml to configure the app. Example:
yaml
CopyEdit
application: heresjohnny
version: 1

```
runtime: python
api_version: 1
handlers:
- url: /.*
  script: heresjohnny.py
```

    -
    - Specifies application ID, version, runtime, and request handling.
4. **Testing and Deployment**:
    - Test locally using the SDK's development server.
    - Upload the app to Google App Engine using appcfg.py.
    - Apps are accessed via URLs like http://application-id.appspot.com.

Google Gears

Google Gears is an **open-source browser extension** for creating offline web applications.

Key Features

1. **Offline Capability**:
    - Stores data locally to ensure app functionality during unreliable or no internet connectivity.
2. **APIs**:
    - Includes JavaScript APIs for:
        - Data storage.
        - Application caching.
        - Multithreading.
3. **Compatibility**:
    - Works on all major browsers (Chrome, Firefox, Opera) and operating systems (Windows, Mac, Linux).
4. **Example**:
    - Google Reader was enhanced with offline functionality using Gears.

Microsoft Azure

Microsoft Azure is a **cloud services platform** that supports app development and deployment with tools like **Visual Studio** and **.NET Framework**.

Key Features

1. **Infrastructure Management**:
    - Provides dynamic scaling and automated infrastructure management.
    - Pay-as-you-go pricing model.
2. **Core Services**:
    - **Live Services**: Handles user data and resources for social applications.
    - **SQL Services**: Web-based relational database services.
    - **.NET Services**: Enables loosely coupled cloud applications with workflow management and secure access control.
3. **Layered Architecture**:
    - **Layer Zero**: Manages hardware abstraction.
    - **Layer One**: Azure operating system (formerly "RedDog").
    - **Layer Two**: Building blocks for developers (e.g., LiveMesh).
    - **Layer Three**: Hosted applications like SharePoint Online.
4. **Use Cases**:
    - Build hybrid applications that integrate cloud and on-premise systems.
    - Develop apps for web, mobile, and enterprise environments.

Summary

- **QuickBase** is ideal for non-technical consultants who need to create tailored business applications quickly and affordably.
- **Cast Iron Cloud** simplifies SaaS and enterprise system integration with prebuilt templates and wizard-based customization.
- **Bungee Connect** provides an all-in-one platform for developing and hosting interactive, scalable applications with a pay-per-use model.
- **Google App Engine** enables fast app development using Python, focusing on scalability and simplicity.
- **Google Gears** adds offline capabilities to web apps, enhancing user experience during connectivity issues.
- **Microsoft Azure** supports comprehensive cloud solutions, integrating seamlessly with Microsoft's ecosystem for enterprise and web applications.

Salesforce.com development

1. Create an Account

- Go to developer.force.com and sign up for a **Developer Edition account**.
- Complete the setup process by providing your name, username, password, etc.
- Log in at login.salesforce.com.

2. Create a Custom Object

- Custom objects store data specific to your application.

1. Go to **Setup** (top-right corner).
2. Navigate to **Create → Objects** and click **New Custom Object**.
3. Fill in the details:
    - **Label:** Lunch
    - **Plural Label:** Lunch
    - **Object Name:** Lunch
    - **Description:** An object that holds lunch expense information.
4. Check **Allow Activities** and **Allow Reports**.
5. Save the object.

3. Add Fields

You'll add fields to the custom object to capture specific information (e.g., date, cost, contact).

Step 1: Add a Date Field

- Go to the **Custom Fields & Relationships** section of the custom object.
- Click **New** to start the field wizard.
- Choose **Data Type: Date** and fill in:
    - **Field Label:** Date
    - **Field Name:** Date
    - **Description:** Date of lunch
    - **Default Value:** Today()
- Mark the field as **Required** and save.

Step 2: Add a Cost Field

- Choose **Data Type: Number** and fill in:
    - **Field Label:** Cost
    - **Length:** 4
    - **Decimal Places:** 2
    - **Field Name:** Cost
    - **Description:** Cost of lunch
- Mark the field as **Required** and save.

Step 3: Add a Contact Field

- Choose **Data Type: Lookup Relationship**.

- In the **Related To** dropdown, select **Contact**.
- Fill in:
    - **Field Label:** Contact
    - **Field Name:** Contact
    - **Description:** Person I had lunch with
- Save the field.

4. Create a Tab

Tabs make your application visible to users.

1. Go to **Setup → Create → Tabs**.
2. Click **New** under Custom Object Tabs.
3. Select the **Lunch** object and choose a tab style (e.g., Apple icon).
4. Accept the defaults and save.

5. Build the App

The app ties everything together.

1. Go to **Setup → Create → Apps** and click **New**.
2. Fill in the app details:
    - **App Label:** Lunch Tracker
    - **App Name:** Lunch_Tracker
    - **Description:** This application tracks your lunch expenses.
3. Add the **Lunch** tab to the app and set the **Home** tab as the default landing page.
4. Mark the app as **Visible to all users** and save.

6. Test the App

- From the **App Menu** (top-right), select **Lunch Tracker**.
- Click the **Lunch** tab and create a new lunch expense entry.
- Fill in:
    - **Date:** The date of the lunch.
    - **Cost:** The expense amount.
    - **Contact:** Lookup or create a new contact (e.g., Bruce Dickinson).
- Save the record.

Summary

Using Salesforce's point-and-click tools:

- You created a custom object (Lunch).
- Added fields for date, cost, and contact.
- Built a custom tab and app (Lunch Tracker).
- Tested the app without writing any code.

What is Microsoft Azure?

- **Azure** is a **cloud computing platform** by Microsoft that allows you to **create, deploy, and manage applications** in the cloud.
- It includes **on-demand compute and storage services** and provides tools for **scaling** and **managing** web applications.

Getting Started with Azure Development

1. Download the Azure SDK

- To develop applications, download the **Azure Software Development Kit (SDK)**:
    - Visit Microsoft's Azure SDK download page.
    - Ensure your machine meets the **system requirements**:
        - Supported operating systems include:
            - Windows Server 2008, Windows Vista (Business/Ultimate), etc.
        - Required software:
            - **.NET Framework 3.5 SP1**
            - **IIS 7.0** with ASP.NET and WCF HTTP Activation.
            - **Microsoft SQL Server Express 2005 or 2008**.
            - **Windows PowerShell (optional)**.

2. Install the SDK

- Uninstall any old SDK versions before installing the new one.
- Install the SDK, which includes tools like:
    - **Development Fabric**: Simulates Azure runtime locally.
    - **Development Storage**: Emulates cloud storage (Blob, Queue, Table).

Enabling IIS 7.0

Azure development requires **IIS 7.0** (Internet Information Services) for running applications. Follow these steps:

For Windows Server 2008:

1. Open **Server Manager** from Administrative Tools.
2. Add **.NET Framework 3.0** under Features.
3. Enable **WCF HTTP Activation**.
4. Install **Web Server (IIS)** with required features:
    - **Static Content**
    - **ASP.NET**

For Windows Vista:

1. Open **Control Panel → Programs → Programs and Features**.
2. Click **Turn Windows Features On or Off**.
3. Enable:
    - **WCF HTTP Activation**
    - **ASP.NET**
    - **Static Content** under Internet Information Services.

Testing the SDK

1. Open the **Azure SDK Command Prompt** from the Start Menu.
2. Navigate to the **sample directory** included with the SDK.
3. Run the following commands:
    - `RunDevStore.cmd`: Sets up development storage.
    - `runme.cmd`: Launches a sample app (e.g., "Hello World").
4. The app runs in a **local environment**, and the development fabric icon appears in the system tray.

Creating an Azure Application

Using Visual Studio 2008:

1. **Run Visual Studio as Administrator**.
2. Create a new project:
    - **File → New → Project**.
    - Choose **Cloud Services** from the Visual C# templates.
    - Select **Web Cloud Service** to create a web role.

3. Add custom code:
    - Modify the **Default.aspx** page to display custom text.

Testing Locally:

- Press **F5** in Visual Studio to run the app locally.
- Use the **Development Fabric** to view the app and check logs.
- Press **SHIFT + F5** to stop debugging.

Publishing to the Azure Cloud

1. Once the app is tested, deploy it to Azure:
    - Use the **Azure Management Portal**.
    - Upload your app files and configurations.

Application Management

- Once deployed, applications can be managed using tools like **Kaavo IMOD** for:
    - **Backup and security**: Schedule automatic backups and encrypt data.
    - **Performance monitoring**: Set up alerts for resource usage.
    - **Scaling**: Easily increase capacity during high demand.

Troubleshooting Cloud Apps

1. **Challenges**:
    - Limited visibility compared to traditional on-premises IT.
    - Difficult to differentiate SaaS-related issues from network problems.
2. **Solutions**:
    - Use monitoring tools (e.g., packet inspection) to analyze traffic and detect issues.
    - Identify bottlenecks like bandwidth contention or connectivity problems.

Key Features of Azure Development

- **Development Fabric**: Test apps locally before deployment.
- **Seamless Integration**: Use Visual Studio for development.
- **Scalability**: Apps can scale horizontally (adding more nodes) or vertically (adding more resources to a node).

- **Management Tools**: Monitor performance, troubleshoot, and scale using Azure-native tools or third-party solutions.
