How Bot Service works

Bot Service provides the core components for creating bots, including the Bot Builder SDK for developing bots and the Bot Framework for connecting bots to channels. Bot Service provides five templates you can choose from when creating your bots with support for .NET and Node.js.

Important

You must have a Microsoft Azure subscription before you can use Bot Service. If you do not already have a subscription, you can register for a [free account](https://azure.microsoft.com/en-us/free/).

## **Hosting plans**

Bot Service provides two hosting plans for your bots. You can choose a hosting plan that suits your needs.

### **App Service plan**

A bot that uses an App Service plan is a standard Azure web app you can set to allocate a predefined capacity with predictable costs and scaling. With a bot that uses this hosting plan, you can:

* Edit bot source code online using an advanced in-browser code editor.
* Download, debug, and re-publish your C# bot using Visual Studio.
* Set up continuous deployment easily for Visual Studio Online and Github.
* Use sample code prepared for the Bot Builder SDK.

### **Consumption plan**

A bot that uses a Consumption plan is a serverless bot that runs on [Azure Functions](http://go.microsoft.com/fwlink/?linkID=747839), and uses the pay-per-run Azure Functions pricing. A bot that uses this hosting plan can scale to handle huge traffic spikes. You can edit bot source code online using a basic in-browser code editor. For more information about the runtime environment of a Consumption plan bot, see [Azure Functions Consumption and App Service plans](https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale).

## **Templates**

Bot Service enables you to quickly and easily create a bot in either C# or Node.js by using one of five templates.

| Template | Description |
| --- | --- |
| Basic | Creates a bot that uses dialogs to respond to user input. |
| Form | Creates a bot that collects input from a user via a guided conversation that is created using [FormFlow](https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-formflow) in C# or [waterfalls](https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-prompts) in Node.js. |
| Language understanding | Creates a bot that uses natural language models (LUIS) to understand user intent. |
| QnA Maker | Creates a bot that uses the QnA Maker service to answer user's FAQs. |
| Proactive | Creates a bot that uses Azure Functions to alert users of events. |

[Learn more](https://docs.microsoft.com/en-us/bot-framework/bot-service-concept-templates) about the different templates and how they can help you build bots.

## **Develop and deploy**

By default, Bot Service enables you to develop your bot directly in the browser using the Online Code Editor, without any need for a tool chain.

You can develop and debug your bot locally with the Bot Builder SDK and an IDE, such as Visual Studio 2017. You can publish your bot directly to Azure using Visual Studio 2017 or the Azure CLI. You can also [set up continuous deployment](https://docs.microsoft.com/en-us/bot-framework/bot-service-continuous-deployment) with the source control system of your choice, such as VSTS or GitHub. With continuous deployment configured, you can develop and debug in an IDE on your local computer, and any code changes that you commit to source control automatically deploy to Azure.

Tip

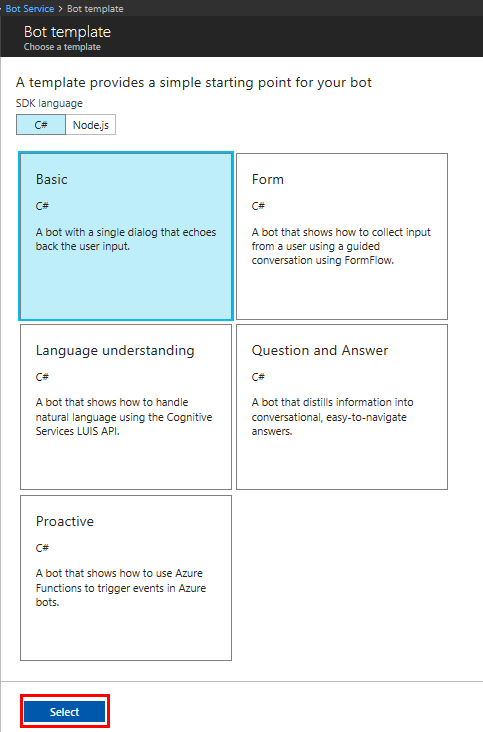
After enabling continuous deployment, be sure to modify your code through continuous deployment only and not through other mechanisms to avoid conflict.

## **Manage your bot**

During the process of creating a bot using Bot Service, you specify a name for your bot, set up its hosting plan, choose a pricing tier, and configure some other settings. After your bot has been created, you can change its settings, configure it to run on one or more channels, and test it with in Web Chat.

# Bot Service templates

Bot Service includes five templates to help you get started with building bots. These templates provide a fully functional bot out of the box to help you get started quickly. When you [create a bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-quickstart), you can choose a template based on C# code or Node.js code.



## **Basic bot**

To create a bot that uses dialogs to respond to user input, choose the Basic template. The **Basic bot** template creates a bot that has the minimum set of files and code to get started. The bot echoes back to the user whatever they type in. You can use this template to get started building conversation flow in your bot.

## **Forms bot**

To create a bot that collects input from a user via a guided conversation, choose the **Form**template. A form bot is designed to collect a specific set of information from the user. For example, a bot that is designed to obtain a user's sandwich order would need to collect information such as type of bread, choice of toppings, size of sandwich, and so on.

Bots created with the Form template in the C# language use [FormFlow](https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-formflow) to manage forms, and bots created with the Form template in the Node.js language use [waterfalls](https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-dialog-waterfall) to manage forms.

## **Language Understanding bot**

To create a bot that uses natural language models to understand user intent, choose the **Language understanding** template. This template leverages [Language Understanding (LUIS)](https://www.luis.ai/) to provide natural language understanding.

If a user sends a message such as "get news about virtual reality companies," your bot can use LUIS to interpret the meaning of the message. Using LUIS, you can quickly deploy an HTTP endpoint that will interpret user input in terms of the intention that it conveys (find news) and the key entities that are present (virtual reality companies). LUIS enables you to specify the set of intents and entities that are relevant to your application, and then guides you through the process of building a language understanding application.

When you create a bot using the Language understanding template, Bot Service creates a corresponding LUIS application that is empty (i.e., that always returns None). To update your LUIS application model so that it is capable of interpreting user input, you must sign-in to [LUIS](https://www.luis.ai/), click **My applications**, select the application that the service created for you, and then create intents, specify entities, and train the application.

## **Question and Answer bot**

To create a bot that distills semi-structured data like question and answer pairs into distinct, helpful answers, choose the **Question and Answer** template. This template leverages the [QnA Maker](https://qnamaker.ai/) service to parse questions and provide answers.

When you create a bot with the Question and Answer template, you must sign-in to [QnA Maker](https://qnamaker.ai/)and create a new QnA service. This QnA service will give you the knowledge base ID and subscription key that you can use to update your [App Settings](https://docs.microsoft.com/en-us/bot-framework/bot-service-manage-settings) values for **QnAKnowldegebasedId** and **QnASubscriptionKey**. Once those values are set, your bot can answer any questions that the QnA service has in its knowledge base.

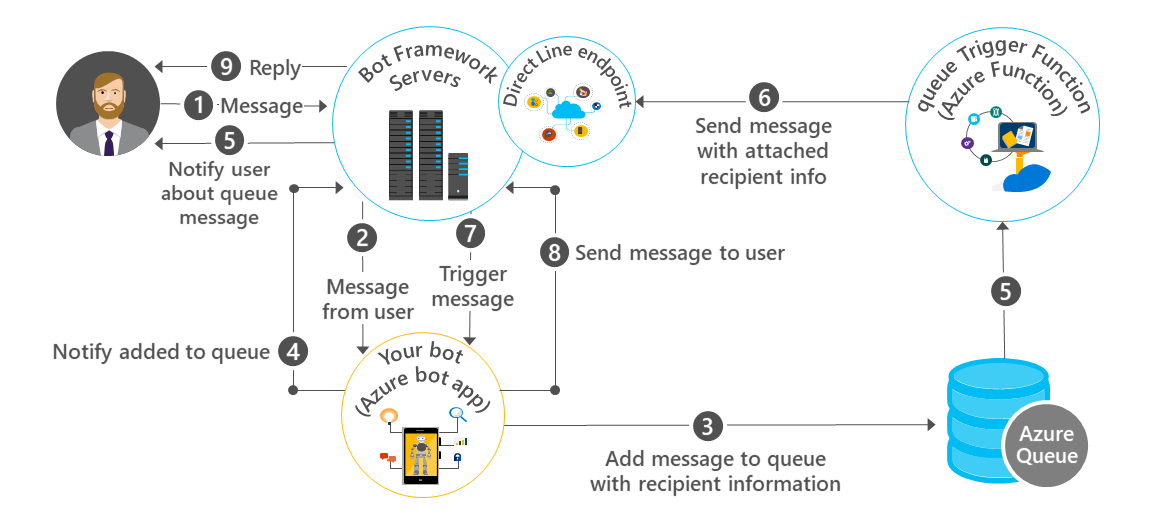
## **Proactive bot**

To create a bot that can send proactive messages to the user, choose the **Proactive template**. Typically, each message that a bot sends to the user directly relates to the user's prior input. In some cases though, a bot may need to send the user information that is not directly related to the user's most recent message. These types of messages are called **proactive messages**. Proactive messages can be useful in a variety of scenarios. For example, if a bot sets a timer or reminder, it may need to notify the user when the time arrives. Or, if a bot receives a notification about an external event, it may need to communicate that information to the user.

When you create a bot by using the Proactive template, several Azure resources are automatically created and added to your resource group. By default, these Azure resources are already configured to enable a very simple proactive messaging scenario.

| Resource | Description |
| --- | --- |
| Azure Storage | Used to create the queue. |
| Azure Function App | A queueTrigger Azure Function that is triggered whenever there is a message in the queue. It communicates to Bot Service by using [Direct Line](https://docs.microsoft.com/bot-framework/rest-api/bot-framework-rest-direct-line-3-0-concepts). This function uses bot binding to send the message as part of the trigger’s payload. Our example function forwards the user’s message as-is from the queue. |
| Bot Service | Your bot. Contains the logic that receives the message from user, adds the message to the Azure queue, receives triggers from Azure Function, and sends back the message it received via trigger's payload. |

The following diagram shows how triggered events work when you create a bot using the Proactive template.



The process begins when the user sends a message to your bot via Bot Framework servers (step 1).

# Develop bots with Bot Builder

The Bot Builder provides an SDK, libraries, samples, and tools to help you build and debug bots. When you build a bot with Bot Service, your bot is backed by the Bot Builder SDK. You can also use the Bot Builder SDK to create a bot from scratch using C# or Node.js. Bot Builder includes the Bot Framework Emulator for testing your bots and the Channel Inspector for previewing your bot's user experience on different channels.

If you want to build a bot using any programming language you want, you can use the REST API.

## **Bot Builder SDK for .NET**

The Bot Builder SDK for .NET leverages C# to provide a familiar way for .NET developers to write bots. It is a powerful framework for constructing bots that can handle both free-form interactions and more guided conversations where the user selects from possible values.

The [.NET Quickstart](https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-quickstart) will guide you through creating a bot with the Bot Builder SDK for .NET.

The Bot Builder SDK for .NET is available as a [NuGet package](https://www.nuget.org/packages/Microsoft.Bot.Builder/).

Important

The Bot Builder SDK for .NET requires .NET Framework 4.6 or newer. If you are adding the SDK to an existing project targeting a lower version of the .NET Framework, you will need to update your project to target .NET Framework 4.6 first.

To install the SDK in an existing Visual Studio project, complete the following steps:

1. In **Solution Explorer**, right-click the project name and select **Manage NuGet Packages...**.
2. On the **Browse** tab, type "Microsoft.Bot.Builder" into the search box.
3. Select **Microsoft.Bot.Builder** in the list of results, click **Install**, and accept the changes.

You can also download the Bot Builder SDK for .NET [source code](https://github.com/Microsoft/BotBuilder/tree/master/CSharp) from GitHub.

### **Visual Studio project templates**

Download Visual Studio project templates to accelerate bot development.

* [Bot template for Visual Studio](http://aka.ms/bf-bc-vstemplate) for developing bots with C#
* [Cortana skill template for Visual Studio](https://aka.ms/bf-cortanaskill-template) for developing Cortana skills with C#

Tip

[Learn more](https://docs.microsoft.com/en-us/visualstudio/ide/how-to-locate-and-organize-project-and-item-templates) about how to install Visual Studio 2017 project templates.

## **Bot Builder SDK for Node.js**

The Bot Builder SDK for Node.js provides a familiar way for Node.js developers to write bots. You can use it to build a wide variety of conversational user interfaces, from simple prompts to free-form conversations. The Bot Builder SDK for Node.js uses restify, a popular framework for building web services, to create the bot's web server. The SDK is also compatible with Express.

The [Node.js Quickstart](https://docs.microsoft.com/en-us/bot-framework/nodejs/bot-builder-nodejs-quickstart) will guide you through creating a bot with the Bot Builder SDK for Node.js.

The Bot Builder SDK for Node.js is available as an npm package. To install the Bot Builder for Node.js SDK and its dependencies, first create a folder for your bot, navigate to it, and run the following **npm** command:

NodeJSCopy

npm init

Next, install the Bot Builder SDK for Node.js and [Restify](http://restify.com/" \t "_blank) modules by running the following **npm**commands:

NodeJSCopy

npm install --save botbuilder

npm install --save restify

You can also download the Bot Builder SDK for Node.js [source code](https://github.com/Microsoft/BotBuilder/tree/master/Node) from GitHub.

## **REST API**

You can create a bot with any programming language by using the Bot Framework REST API. The [REST Quickstart](https://docs.microsoft.com/en-us/bot-framework/rest-api/bot-framework-rest-connector-quickstart) will guide you through creating a bot with REST.

There are three REST APIs in the Bot Framework:

* The [Bot Connector REST API](https://docs.botframework.com/en-us/restapi/connector/#navtitle) enables your bot to send and receive messages to channels configured in the [Bot Framework Portal](https://dev.botframework.com/).
* The [Bot State REST API](https://docs.botframework.com/en-us/restapi/state/#navtitle) enables your bot to store and retrieve state associated with the conversations that are conducted through the Bot Connector REST API.
* The [Direct Line REST API](https://docs.botframework.com/en-us/restapi/directline3/#navtitle) enables you to connect your own application, such as a client application, web chat control, or mobile app, directly to a single bot.

## **Bot Framework Emulator**

The Bot Framework Emulator is a desktop application that allows you to test and debug you bots, either locally or remotely. Using the emulator, you can chat with your bot and inspect the messages that your bot sends and receives. [Learn more about the Bot Framework emulator](https://docs.microsoft.com/en-us/bot-framework/bot-service-debug-emulator) and [download the emulator](http://emulator.botframework.com/) for your platform.

## **Bot Framework Channel Inspector**

Quickly see what bot features will look like on different channels with the Bot Framework [Channel Inspector](https://docs.microsoft.com/en-us/bot-framework/bot-service-channel-inspector).

# Bot scenarios

This topic explores key scenarios for powerful and successful bots built using Bot Service.

You can download or clone the source code for all the scenarios bot samples from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Commerce bot scenario**

The [Commerce bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-commerce) scenario describes a bot that replaces the traditional e-mail and phone call interactions that people typically have with a hotel's concierge service. The bot takes advantage of Cognitive Services to better process customer requests via text and voice with context gathered from integration with backend services.

In the Commerce bot scenario, a customer can make a request for concierge services with a hotel. She is authenticated via an Azure Active Directory v2 authentication endpoint. The bot can look up the customer's reservations and and provide different service options. For example, the customer might book a cabana by the pool. The bot uses Language Understanding Intelligent Services (LUIS) to parse the request and then the bot walks the user through the process of booking a cabana for an existing reservation.

## **Cortana Skill bot scenario**

The [Cortana Skill bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-cortana-skill) scenario takes advantage of Cortana. Using the natural interface of your voice and a custom Cortana Skill bot, you can ask Cortana to speak to a organization, such as a mobile auto detailing company, to help you make an appointment based on where you’re at when you make the call. The bot can provide a list of services, times available, and duration.

## **Enterprise Productivity bot scenario**

The [Enterprise Productivity bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-enterprise-productivity) scenario shows you how to integrate a bot with your Office 365 calendar and other services to increase your productivity.

The bot integrates with Office 365 to make it quicker and easier to create a meeting request with another person. In the process of doing so you could access additional services like Dynamics CRM. This sample provides the code necessary to integrate with Office 365 with authentication via Azure Active Directory. It provides mock entry points for external services as an exercise for the reader.

## **Information bot scenario**

This [Information bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-informational) can answer questions defined in a knowledge set or FAQ using Cognitive Services QnA Maker and answer more open-ended questions Azure Search.

Often information is buried in structured data stores like SQL Server that can be easily surfaced via search. Imagine looking up a customer's order status by simple conversational commands. Using Cognitive Services QnA Maker, the user is presented with a set of valid search options like, lookup a customer, review customer's most recent order, etc. With the QnA format defined the user can easily ask questions that are backed by Azure Search which can look up data stored in a SQL Database.

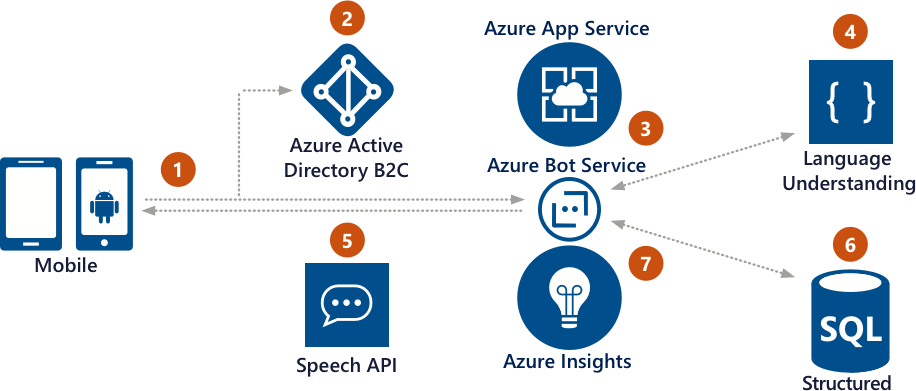
## **IoT bot scenario**

This [Internet of Things (IoT) bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-internet-things) bot makes it easy for you to control devices around your home, such as a Philips Hue light using interactive chat commands.

Using this simple bot, you can control your Philips Hue lights in conjunction with the free If This Then That (IFTTT) service. As an IoT device, the Philips Hue can be controlled locally via their exposed API. However, this API is not exposed for general access from outside the local network. However, IFTTT is a "[Friend of Hue](http://www2.meethue.com/en-us/friends-of-hue/ifttt/)" and thus has exposed a number of control commands that you can issue such as turning lights on and off, changing the light color, or the light intensity.

# Commerce bot scenario

The [Commerce bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-scenario-commerce) scenario describes a bot that replaces the traditional e-mail and phone call interactions that people typically have with a hotel's concierge service. The bot takes advantage of Cognitive Services to better process customer requests via text and voice with context gathered from integration with backend services.



Here is the logic flow of a Commerce bot that functions as a concierge for a hotel:

1. The customer uses the hotel mobile app.
2. Using Azure AD B2C, the user authenticates.
3. Using the custom Application Bot, user requests information.
4. Cognitive Services helps process the natural language request.
5. Response is reviewed by customer who can refine the question using natural conversation.
6. After the user is happy with the results, the Application Bot updates the customer’s reservation.
7. Application insights gathers runtime telemetery to help development with bot performance and usage.

## **Sample bot**

The sample Commerce bot is designed around a fictitious hotel concierge service. Written in C#, customers access the Bot once they've authenticated Azure AD B2C with a hotel via the chain's member services mobile app. The chain stores reservations in a SQL Database. A customer can use natural phrase questions like "How much to rent a pool cabana for my stay". The Bot in turn has context about what hotel and the duration of the guest's stay. In addition, Language Understanding (LUIS) Service makes it easy for the bot to get context from even a simple phrase like "pool cabana". The Bot provides the answer and then can offer to book a cabana for the guest, providing choices around the number of days and type of cabana. Once the Bot has all the necessary data, it books the request. The guest can also use their voice to make the same request.

You can download or clone the source code for this sample bot from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Components you'll use**

The Commerce bot uses the following components:

* Azure AD for Authentication
* Cognitive Services: LUIS
* Application Insights

### **Azure Active Directory (Azure AD)**

Azure Active Directory (Azure AD) is Microsoft’s multi-tenant cloud based directory and identity management service. As a Bot developer, Azure AD lets you focus on building your Bot by making it fast and simple to integrate with a world class identity management solution used by millions of organizations around the world. Azure AD supports a B2C connector allowing you to identify individuals using external IDs such as Google, Facebook, or a Microsoft Account. Azure AD removes the responsibility from you having to manage the user's credentials and instead focus your Bot's solution knowing you can correlate the user of the Bot with the correct data exposed by your application.

### **Cognitive Services: LUIS**

As a member of the Cognitive Services family of technologies, Language Understanding (LUIS) brings the power of machine learning to your apps. Currently, LUIS supports several languages that enables your Bot to understand what a person wants. When integrating with LUIS, you express intent and define the entities your Bot understands. You then teach your Bot to understand those intents and entities by training it with example utterances. You have the ability to tweak your integration using phrase lists and regex features so that your Bot is as fluid as possible for your particular conversation needs.

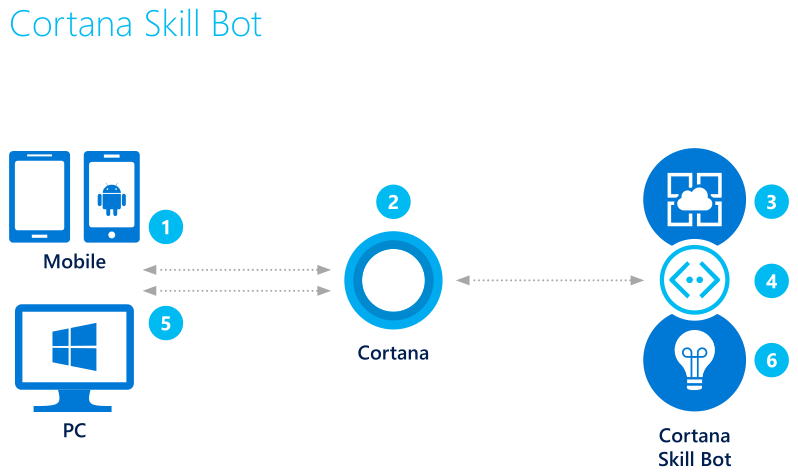
### **Application Insights**

Application Insights helps you get actionable insights through application performance management (APM) and instant analytics. Out of the box you get rich performance monitoring, powerful alerting, and easy-to-consume dashboards to help ensure your Bot is available and performing as you expect. You can quickly see if you have a problem, then perform a root cause analysis to find and fix the issue.

# Cortana Skills Bot Scenario

The Cortana Skills Bot extends Cortana to make it easy to book a mobile auto maintenance appointment using voice with context from your calendar.

Cortana is your personal assistant. Using the natural interface of your voice and a custom Cortana Skill Bot, you can ask Cortana to speak to an organization, such as an auto shop, to help you make an appointment. The service can provide a list of services, times available, and duration. Cortana can look at your calendar to see if you have something at a conflicting time and if not, create the appointment and add it to your calendar.



Here is the logic flow of a Cortana Skills bot for an auto shop:

1. The user accesses Cortana from their PC or mobile device.
2. Using either text or voice commands, the user asks for an automobile maintenance appointment.
3. Because the bot is integrated with Cortana, it has access to the user's calendar and applies logic to the request.
4. With that information, the bot can query the auto service for valid appointments.
5. Presented with contextual options, the user can book the appointment.
6. Application insights gathers runtime telemetery to help development with bot performance and usage.

## **Sample bot**

With a Cortana Skills Bot, it's all about personal context. Using Cortana you could use your voice to ask for "Bob's Mobile Maintenance" to come work on your car based on your location. Using personal information exposed via Cortana your bot can confirm the location based on where the user is at when they're talking to the bot.

You can download or clone the source code for this sample bot from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Components you'll use**

The Cortana Bot uses the following components:

* Cortana
* Application Insights

### **Cortana**

Now you can add support to your Bot by creating a Cortana Skill. You use the the Cortana skills kit to build new features (called skills) for Cortana. A skill is a construct that allows Cortana to do more. You build skills to integrate with your Bots allowing Cortana to complete tasks and get things done. As part of the invocation process, Cortana can (with the user's consent) pass information about the user to a skill at runtime, so that the skill can customize its experience accordingly. Cortana's contextual knowledge allows your Bot to be useful and possibly even clever for them. Once invoked, certain types of skills can manipulate Cortana's interface to have a conversation between the skill and the end user. Once published, users can see and use your skill on Cortana for Windows 10 Anniversary Update+ (Desktop and Mobile), iOS, and Android.

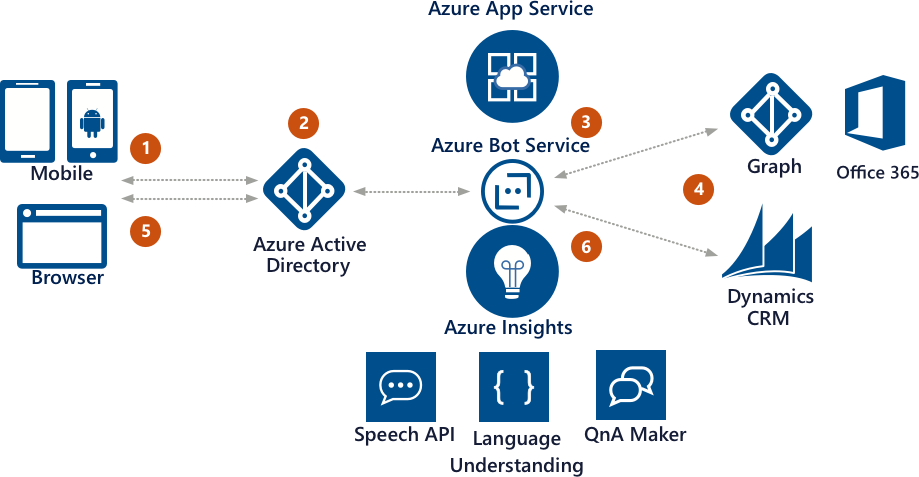
### **Application Insights**

Application Insights helps you get actionable insights through application performance management (APM) and instant analytics. Out of the box you get rich performance monitoring, powerful alerting, and easy-to-consume dashboards to help ensure your Bot is available and performing as you expect. You can quickly see if you have a problem, then perform a root cause analysis to find and fix the issue.

# Enterprise Productivity Bot Scenario

The Enterprise Bot shows how you can increase your productivity by integrating a bot with your Office 365 calendar and other services.

Quickly accessing customer information without having to have a bunch of windows open is what the Enterprise Productivity Bot is all about. Using simple chat commands, a sales rep can look up a customer and check their next appointment via the Graph API and Office 365. From there they can access customer specific information stored in Dynamics CRM such as get a case or create a new one.



Here is the logic flow of an Enterprise Productivity bot:

1. The employee accesses the Enterprise Productivity bot.
2. Azure Active Directory validates the employee's identity.
3. The Enterprise Productivity bot is able to query the employee's Office 365 calendar via the Azure Graph.
4. Using data gathered from the calendar, the bot accesses case information in Dynamics CRM.
5. The information is returned to the employee who can filter down the data without leaving the bot.
6. Application insights gathers runtime telemetery to help development with bot performance and usage.

You can download or clone the source code for this sample bot from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Sample bot**

Because Bots are accessible from a variety of channels, you could use it at your desk from a corporate portal or from Skype while on the go--you just need to be authenticated. With Azure AD integration your Enterprise Productivity Bot knows that that if you're able to access it, you've been authenticated by Azure AD. From there you can ask the bot to check when your next appointment is with a particular customer. The Bot gets this information by querying Office 365 via the Graph API. Then, if there's an appointment in the next seven days, the Bot queries CRM looking for any recent cases for the customer. The Bot responds with either no cases found or the number of open and closed cases. From there you can ask the Bot to list out the cases by type and drill into individual cases.

## **Components you'll use**

The Enterprise Productivity Bot uses the following components:

* Azure AD for Authentication
* Graph API to Office 365
* Dynamics CRM
* Application Insights

### **Azure Active Directory (Azure AD)**

Azure Active Directory (Azure AD) is Microsoft’s multi-tenant cloud based directory and identity management service. As a Bot developer, Azure AD lets you focus on building your Bot by making it fast and simple to integrate with a world class identity management solution used by millions of organizations around the world. By defining an Azure AD app, you can control who has access to your Bot and the data it exposes, without implementing your own complex authentication and authorization system.

### **Graph API to Office 365**

The Microsoft Graph exposes multiple APIs from Office 365 and other Microsoft cloud services through a single endpoint at [https://graph.microsoft.com](https://graph.microsoft.com/). Microsoft Graph makes it easier for you and Bot to to execute queries. The API exposes data from multiple Microsoft cloud services, including Exchange Online as part of Office 365, Azure Active Directory, SharePoint, and more. You can use the API to navigate between entities and relationships. You can use the API from your Bots using the SDK or REST endpoints as well as from your other apps with native support Android, iOS, Ruby, UWP, Xamarin and more.

### **Dynamics CRM**

Dynamics CRM is a customer engagement platform. Using Bots and CRM's APIs, you can build rich interactive Bots that can access the rich data stored in CRM. The power of Dynamics CRM is available to your Bot to create cases, check on status, knowledge management searches and more.

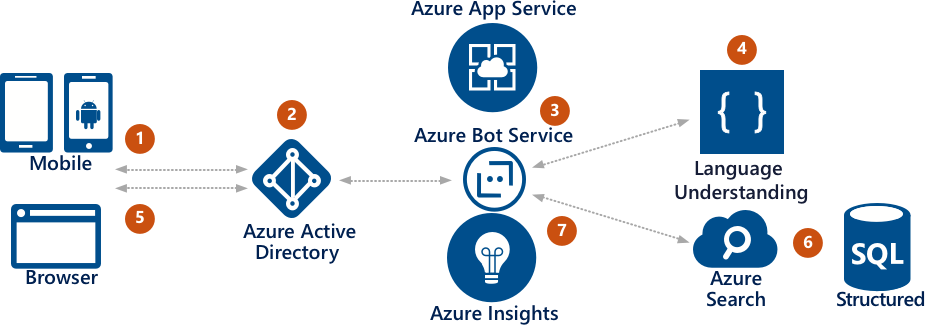
### **Application Insights**

Application Insights helps you get actionable insights through application performance management (APM) and instant analytics. Out of the box you get rich performance monitoring, powerful alerting, and easy-to-consume dashboards to help ensure your Bot is available and performing as you expect. You can quickly see if you have a problem, then perform a root cause analysis to find and fix the issue.

# Information Bot Scenario

This Information Bot could answer questions defined in a knowledge set or FAQ using Cognitive Services QnA Maker and answer more open-ended questions by using Azure Search.

Often information is buried in structured data stores like SQL Server that can be easily surfaced via search. Imagine looking up a customer's order status by simple conversational commands. Using Cognitive Services QnA Maker, the user is presented with a set of valid search options like, lookup a customer, review a customer's most recent order, etc. With the QnA format defined the user can easily ask questions that are backed by Azure Search which can look up data stored in a SQL Database.



Here is the logic flow of an Information bot:

1. The employee starts the Information bot.
2. Azure Active Directory validates the employee's identity.
3. The employee can ask the bot what type of queries are supported.
4. Cognitive Services returns a FAQ bot built with the QnA Maker.
5. The employee defines a valid query.
6. The bot submits the query to Azure Search which returns information about the application data.
7. Application insights gathers runtime telemetery to help development with bot performance and usage.

## **Sample bot**

The sample Bot, written in C#, runs in Microsoft Azure working with data indexed by Azure Search from a SQL Database instance. The Bot exposes a list of questions that can be asked with information on how to phrase the question (the answer) using Cognitive Services: QnA Maker. The user of the Bot can then type a query that looks up data via Azure Search in a broad or specific area of the database that is indexed. The sample provides a simple database with customers and order information. Application Insights tracks Bot usage and helps you monitor the Bot for exceptions. The Bot is published under as an Azure AD app so that you can restrict who has access to the information.

You can download or clone the source code for this sample bot from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Components you'll use**

The Information Bot uses the following components:

* Azure AD for Authentication
* Cognitive Services: QnA Maker
* Azure Search
* Application Insights

### **Azure Active Directory (Azure AD)**

Azure Active Directory (Azure AD) is Microsoft’s multi-tenant cloud based directory and identity management service. As a Bot developer, Azure AD lets you focus on building your Bot by making it fast and simple to integrate with a world class identity management solution used by millions of organizations around the world. By defining an Azure AD app, you can control who has access to your Bot and the data it exposes, without implementing your own complex authentication and authorization system.

### **Cognitive Services: QnA Maker**

Cognitive Services QnA Maker helps you provide an FAQ data source which your users can query from your Bot. When approaching vast amounts of information stored in different systems, it can be useful to help users filter down the information source and set. A single SQL database can have enormous amounts of information that when a free form search is applied brings back too much information. By first using QnA Maker, you can define a road map for your Bot users so they now how to ask intelligent questions that can then be retrieved via Azure Search.

### **Azure Search**

Azure Search is a cloud search service for apps that let you get your search indices up and running quickly. Running on top of Microsoft Azure, you can easily scale up and down as your usage demands. You can connect search results to business goals with great control over search ranking and surface data hidden in your databases.

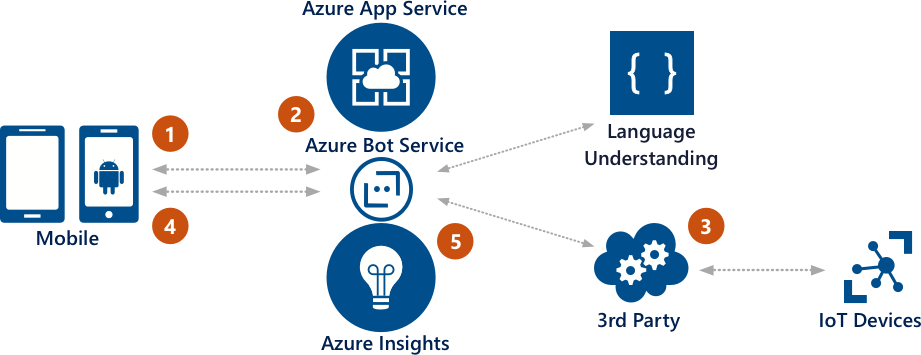
### **Application Insights**

Application Insights helps you get actionable insights through application performance management (APM) and instant analytics. Out of the box you get rich performance monitoring, powerful alerting, and easy-to-consume dashboards to help ensure your Bot is available and performing as you expect. You can quickly see if you have a problem, then perform a root cause analysis to find and fix the issue.

# Internet of Things (IoT) Bot Scenario

This Internet of Things (IoT) Bot makes it easy for you to control devices around your home, such as a Philips Hue light using voice or interactive chat commands.

People love to talk to their things. Since the days of the first TV remote, people have loved not having to move to affect their environment. This IoT bot allows a person to manage a Philips Hue by simple chat commands or voice. In addition, when using chat, a person can be given visual choices related to colors to pick.



Here is the logic flow of an IoT bot:

1. The user logs into Skype and accesses the IoT bot.
2. Using voice, the user asks the bot to turn on the lights via the IoT device.
3. The request is relayed to a 3rd party service that has access to the IoT device network.
4. The results of the command are returned to the user.
5. Application insights gathers runtime telemetery to help development with bot performance and usage.

## **Sample bot**

The IoT bot will allow you to quickly use chat commands from channels like Skype or Slack to control your Hue. To facilitate remote access, you'll call IFTTT applets predefined to work with Hue.

You can download or clone the source code for this sample bot from [Samples for Common Bot Framework Scenarios](https://aka.ms/bot/scenarios).

## **Components you'll use**

The Internet of Things (IoT) Bot uses the following components:

* Philips Hue
* If This Then That (IFTTT)
* Application Insights

### **Philips Hue**

Philips Hue connected bulbs and bridge let you to take full control of your lighting. Whatever you want to do with your lighting, Hue can. Hue has an API you can use from your local network. However, you want to be able to access your Hue controlled devices and lights from anywhere using a friendly Bot interface. Thus you'll access Hue via IFTTT.

### **IFTTT**

IFTTT is a free web-based service that people use to create chains of simple conditional statements, called applets. You can trigger an applet from your Bot to have it do something on your behalf. There are a number of predefined Hue applets available to turn lights on and off, change the scene, and more.

### **Application Insights**

Application Insights helps you get actionable insights through application performance management (APM) and instant analytics. Out of the box you get rich performance monitoring, powerful alerting, and easy-to-consume dashboards to help ensure your Bot is available and performing as you expect. You can quickly see if you have a problem, then perform a root cause analysis to find and fix the issue.