

Developing Webhooks for SharePoint Online



Agenda

- What are WebHooks?
- How to register WebHooks
- Receiving notifications
- Developing and debugging



What are SharePoint Webhooks?

- A Webhook is “a way to be notified of a done change”
- Push model instead of a pull model
 - No synchronous (“-ing”) event support, keep on using RER’s for that
- Follows a universal model widely used in the industry (e.g. WordPress, GitHub, MailChimp, etc.)
- Works with changes to SP List Items (and Libraries)
- Robust – changelog-based with retry logic
- Also available for OneDrive & Outlook



Why Not Use Remote Event Receivers?

- WebHooks have a retry mechanism with an incremental back-off strategy
- WebHooks are lightweight for building service endpoints
 - The payload is small
 - Notifications are batched in the response to the GetChanges() call
- WebHooks are secure, no event information is passed in the notification
- WebHooks are easier to consume by non-SharePoint developers
 - Office Devs have an opportunity to learn a new standard
- No WCF endpoints; regular HTTPS services are sufficient

Webhook-enabled list item event types

- Supported asynchronous list item events in SharePoint:
 - ItemAdded
 - ItemUpdated
 - ItemDeleted
 - ItemCheckedOut
 - ItemCheckedIn
 - ItemUncheckedOut
 - ItemAttachmentAdded
 - ItemAttachmentDeleted
 - ItemFileMoved
 - ItemVersionDeleted
 - ItemFileConverted
- Synchronous events are not supported by WebHooks



Required Permissions To Register Webhooks

- Microsoft Azure Active Directory (AD) applications
 - Set the following Azure AD application permissions:

| Application | Permission |
|-------------------|---|
| SharePoint Online | Read and write items and lists in all site collections. |

- SharePoint add-in
 - Set the following SharePoint permissions

| Scope | Permission Rights |
|-------|-------------------|
| List | Manage |



How To Register Webhooks- Step 1

- Send a subscription request to SharePoint
 - <https://dev.office.com/sharepoint/docs/apis/WebHooks/lists/create-subscription>

POST `/_api/web/lists('list-id')/subscriptions`

| Name | Type | Description |
|--------------------|--------|--|
| resource | string | The URL of the list to receive notifications from. Ex: https://tenancy.sharepoint.com/_api/web/lists({id}) |
| notificationUrl | string | The service URL to send notifications to. Ex: https://your.web.site/your/webhook/service/handlerequest |
| expirationDateTime | date | The date the notification will expire and be deleted. Ex: 2016-10-04T16:30:00+00:00 |
| client-clientState | string | Optional. Opaque string passed back to the client on all notifications. You can use this for validating notifications, or tagging different subscriptions. |

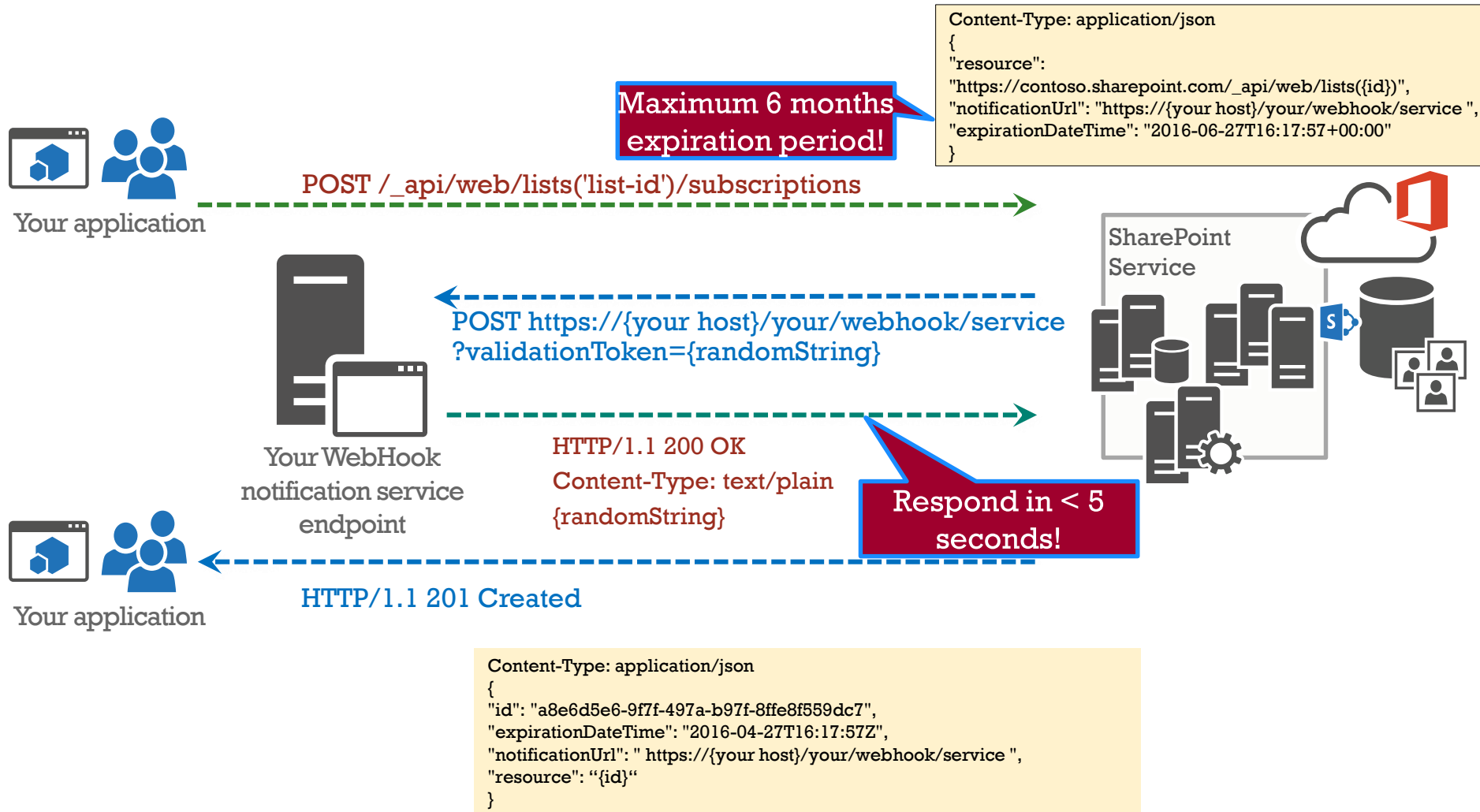


How To Register Webhooks – Step 2

- Receive the response from SharePoint
- The response contains a validation token
- Respond to validation request & return the validation token
 - Must be done in < 5 seconds
- Receive the subscription ID
 - Store the subscription ID



Subscribe to a Webhook



Receiving Notifications

- SharePoint detects changes in lists subscribed to
- SharePoint calls the subscription's service endpoint and sends the following information:

| Property | Description |
|-----------------------|--|
| subscriptionId | The ID of the webhook subscription. If you want to update the webhook subscription, for example you prolong the webhook expiration, then you need this ID. |
| resource | The ID of the list for which the change happened. |
| siteUrl | The server relative URL of the site holding the resource for which the change happened. |

- SharePoint only sends notifications that changes happened
 - Your service endpoint must reply in less than 5 seconds



Event notification in action



Getting Changes

- SharePoint only sends notifications that changes happened
 - You must use an asynchronous approach to determine what changed and still reply in less than 5 seconds
- Use the GetChanges() CSOM API to return what changed
 - Pass a changeToken to SharePoint to avoid getting the same changes more than once
 - changeTokens tell SharePoint at what point you want to receive changes from
 - Your service endpoint should store and persist changeTokens

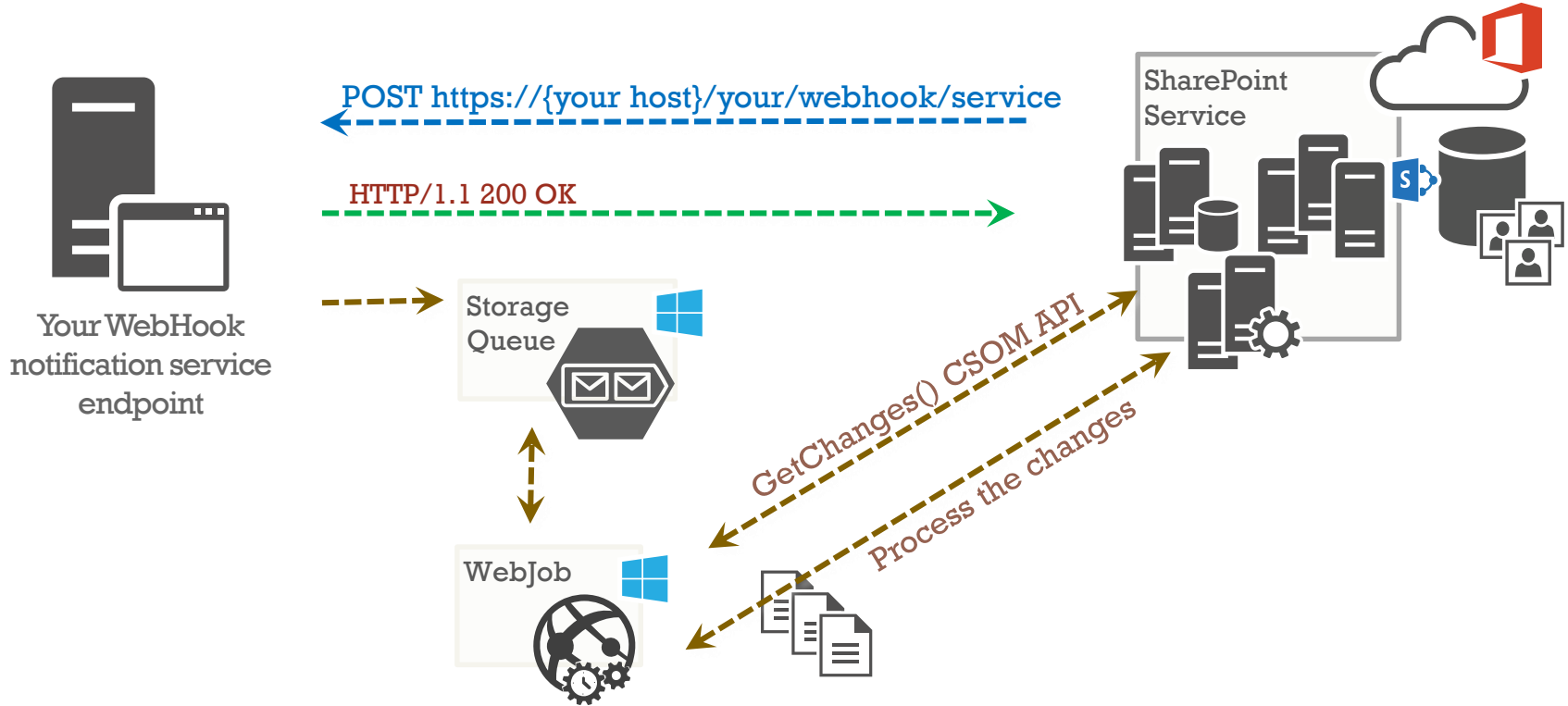


How SharePoint processes change notifications

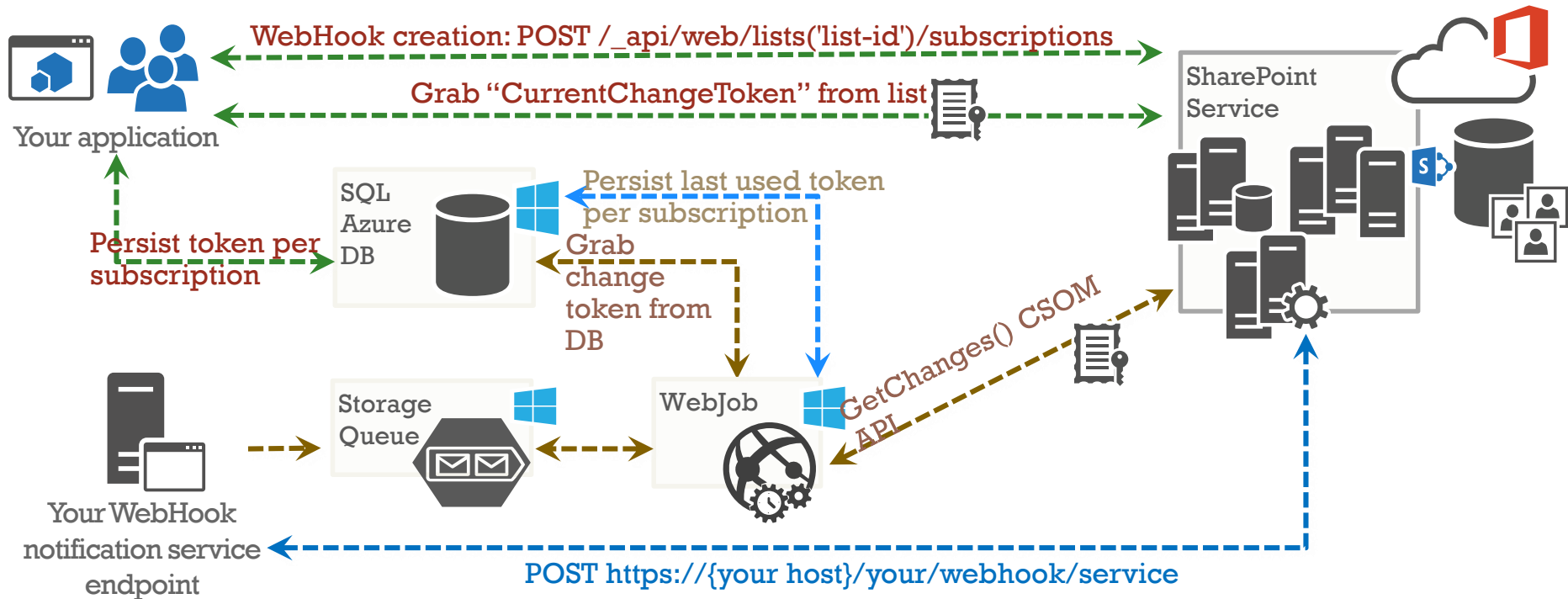
- SharePoint does not call subscribed notification services in real-time
- When changes occur in SharePoint lists, SharePoint queues the WebHook callout
 - The queue is read once each minute
- SharePoint batches WebHook callout requests for each subscription
 - If 5000 changes occur in a batch operation only 1 WebHook callout is made
 - When you call GetChanges SharePoint will return 5000 change events



Processing a notification event



GetChanges() pattern



Handling Expired Webhook Subscriptions

- Webhook subscriptions expire in 6 months
- If there are no changes to a SharePoint list for 6 months
 - No webhook notifications are fired
 - Webhook subscriptions are deleted automatically by SharePoint
- Basic model (fragile)
 - When notifications are received, check the subscription lifetime
 - If the subscription is about to expire, extend the lifetime of the subscription
- Complex model (reliable)
 - Create a job that runs at least once every 6 months to read subscription IDs from a persisted storage location, then extend each subscription using the IDs



Required Permissions to Update WebHooks

- Microsoft Azure Active Directory (AD) applications
 - Set the following Azure AD application permissions:

| Application | Permission |
|------------------------------|---|
| Office 365 SharePoint Online | Read and write items and lists in all site collections. |

- SharePoint add-in
 - Subscription can only be updated by entity that created it.
 - Set the following SharePoint Add-in permissions (or higher):

| Scope | Permission Rights |
|-------|-------------------|
| List | Manage |



Renewing a webhook subscription

- Send an update request to SharePoint
 - <https://dev.office.com/sharepoint/docs/apis/WebHooks/lists/update-subscription>

PATCH `/_api/web/lists('list-id')/subscriptions('id')/`

| Name | Type | Description |
|--------------------|--------|--|
| notificationUrl | string | The service URL to send notifications to. Ex: https://your.web.site/your/webhook/service/handlerequest |
| expirationDateTime | date | The date the notification will expire and be deleted. Ex: 2016-10-04T16:30:00+00:00 |
| client-clientState | string | Optional. Opaque string passed back to the client on all notifications. You can use this for validating notifications, or tagging different subscriptions. |



Webhook retry mechanism

- Built-in retry mechanism
- If a response is not received in less than 5 seconds..
 - SharePoint retries a few times
 - Exponentially backs off
 - Ultimately discards the message



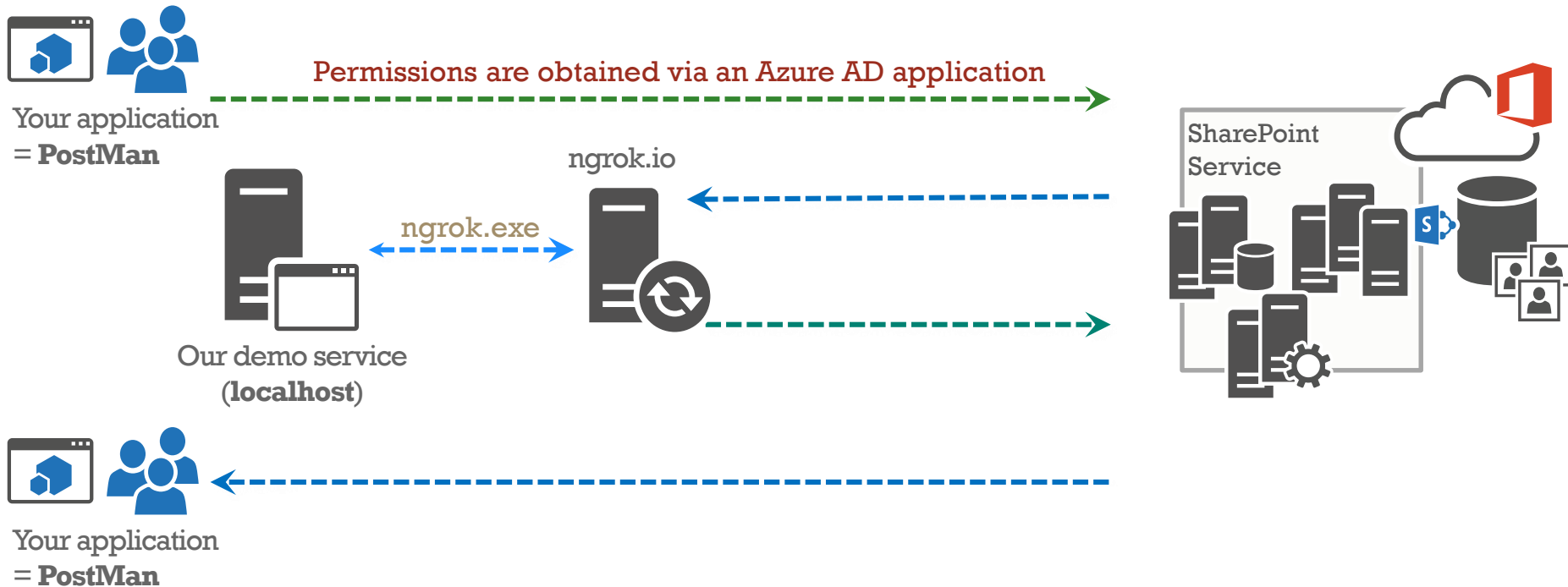
Developing and debugging WebHooks

- Connect remote debugger to your service and web job running in Azure
- Use ngrok (<https://ngrok.com/>) as an alternative to create a tunnel to your service running on localhost
 - Specify ngrok as your service endpoint and notifications will be routed to your local debug environment



Demo setup

Postman: <https://www.getpostman.com/>
Ngrok: <https://ngrok.com/download>



Summary

- What are WebHooks?
- How to register WebHooks
- Receiving notifications
- Developing and debugging



Reference

- WebHooks on Office Dev Center
 - <https://dev.office.com/sharepoint/docs/apis/webhooks/overview-sharepoint-webhooks>
- Reference Implementation Deployment Guide
 - <https://github.com/SharePoint/sp-dev-samples/blob/master/Samples/WebHooks.List.AzureAD/Deployment%20guide.md>
- Video Walkthrough
 - https://www.youtube.com/watch?v=P4a1_EwokwM
- Community walkthrough (by Prian)
 - <http://www.c-sharpcorner.com/article/sharepoint-webhooks-as-event-receivers-for-sharepoint-online/>