

Docs as Part of the Product

Den Delimarsky // Senior PM

docs.microsoft.com UX & core experiences

docs@local ~ % whoami

- Senior PM at Microsoft
- Building docs.microsoft.com
- Based in Vancouver for 2+ years
- Helping drive:
 - Home page
 - User experience
 - Content interaction & controls
 - Anything that is global to docs



```
docs@local ~ % docs --status
```

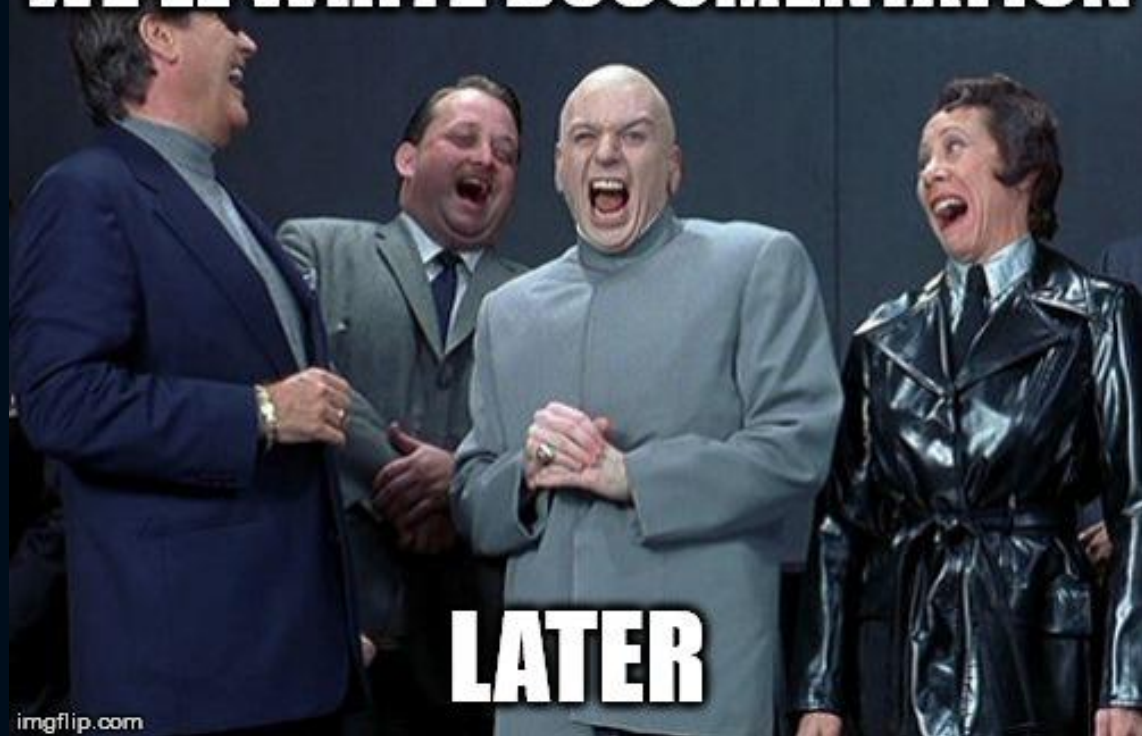
docs@local ~ % docs --status

- **What we expect** from doc experiences
 - Up-to-date and always reflecting the true state of the product.
 - Comprehensive.
 - Easy to edit.
 - Intuitive search and discovery.
 - Connected to the communities inside and outside the company.
 - Rich, interactive presentation.

```
docs@local ~ % docs --status
```

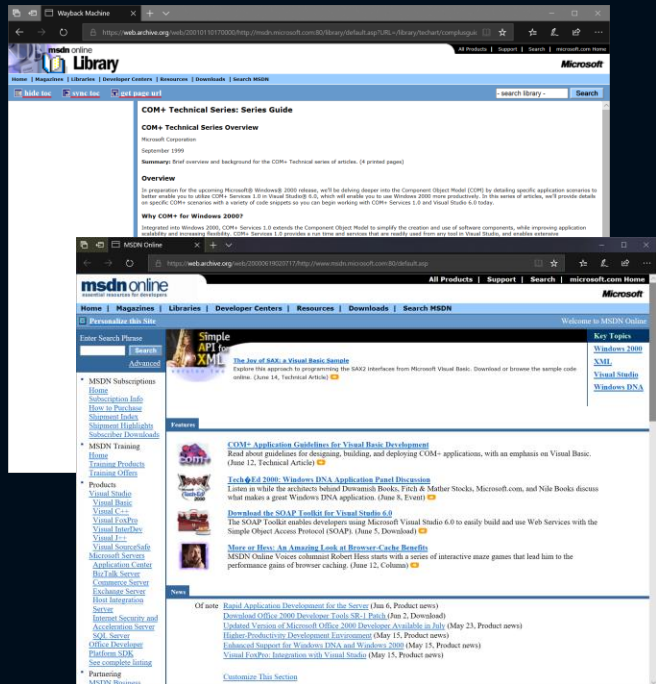
- **What we get** from doc experiences
 - Out-of-date – generated once and forgotten.
 - Inaccurate API docs written by hand.
 - Maintained in content silos.
 - Scattered – every team has their own site with their own format and publishing pipeline.
 - Search is bad due to fragmentation.
 - Text and basic media (images).

WE'LL WRITE DOCUMENTATION



```
docs@local ~ % docs --history
```

docs@local ~ % docs --history



- Started with the goal to be the one true place for all Microsoft developer resources.
- Powered by a closed, proprietary publishing system.
- Content stored in an internally-crafted non-standard XML flavor.

`docs@local ~ % docs --history`

- Brittle code base not designed for the cloud.
- Everything is manually written – almost zero automation.
- Complicated process to update and publish content – sometimes it took days, if not weeks.
- Teams outgrew MSDN, held back by its update velocity – new sites started appearing.

```
docs@local ~ % reboot
```

docs@local ~ % reboot

- One doc site to rule them all – unify documentation for all company.
- Start from zero, for the cloud, from the cloud.
- Automate all the things.
- Open, using standard open-source tools and formats.
- Global by default – 64 locales built-in.
- We don't know the right way – but we can experiment.



docs@local ~ % reboot

- Consistent editing experience – Markdown is the golden standard.
- Integrated in API reference (part of Javadoc comments, Swagger specs and Python docstrings).
- Edit directly in GitHub or favorite editor.
- Easily preview changes.

docs@local ~ % reboot

Getting started | Micros

https://docs.microsoft.com/en-us/python/cognitive-toolkit/gettingstarted/view=cntk-py-2.5.1

Microsoft | Docs Windows Microsoft Azure Visual Studio Office More

All Microsoft Docs Sign in

Docs / Cognitive Toolkit Python API

Feedback Edit Share Dark

Cognitive Toolkit Python API 2.5.1

Filter by title

- Cognitive Toolkit Python (2.5.1)
 - Setup and installation
 - Getting Started
 - Tutorials
 - Examples
 - Working with Sequences
 - Readers, Multi-GPU, Profiling...
 - Extending CNTK
 - Layers Library Overview
- > Reference

Getting started

05/31/2017 • 5 minutes to read • Contributors

You can optionally try the [tutorials](#) with pre-installed CNTK running in Azure Notebook hosted environment (for free) if you have not installed the toolkit in your own machine.

If you are coming from another deep learning toolkit you can start with an [overview for advanced users](#).

If you have installed CNTK on your machine, after going through the [installation steps](#), you can start using CNTK from Python right away (don't forget to [activate](#) your Python environment if you did not install CNTK into your root environment):

```
Python
```

```
>>> import cntk
>>> cntk.__version__
'2.5.1'

>>> cntk.minus([1, 2, 3], [4, 5, 6]).eval()
array([-3., -3., -3.], dtype=float32)
```

The above makes use of the CNTK `minus` node with two array constants. Every operator has an `eval()` method that runs a forward pass for that node using its inputs, and returns the result. A slightly more interesting example that uses input variables (the more common case) is as follows:

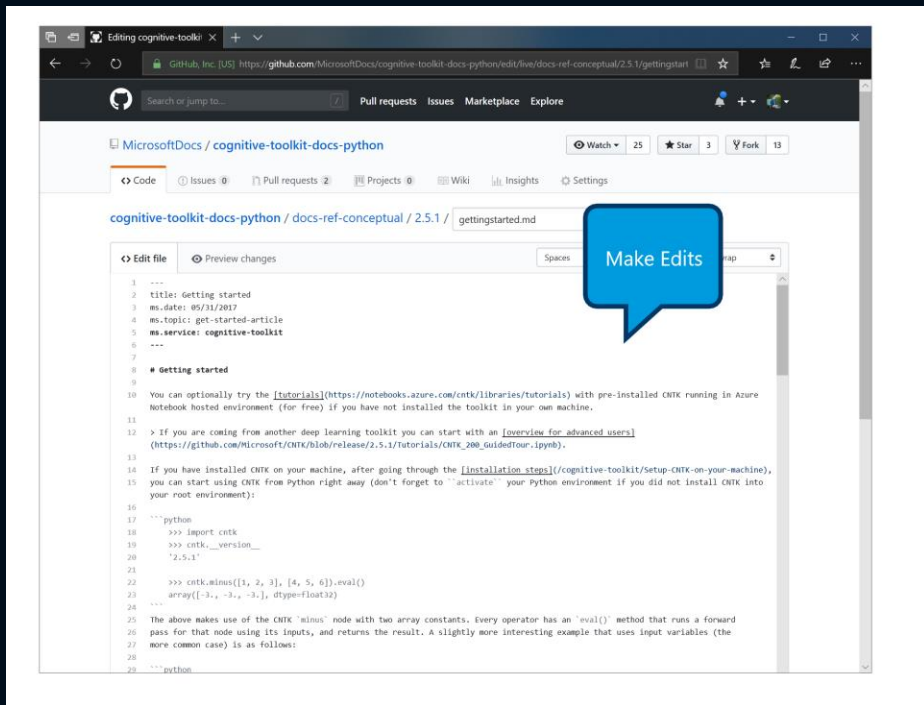
```
Python
```

```
>>> import numpy as np
>>> x = cntk.input_variable(2)
```

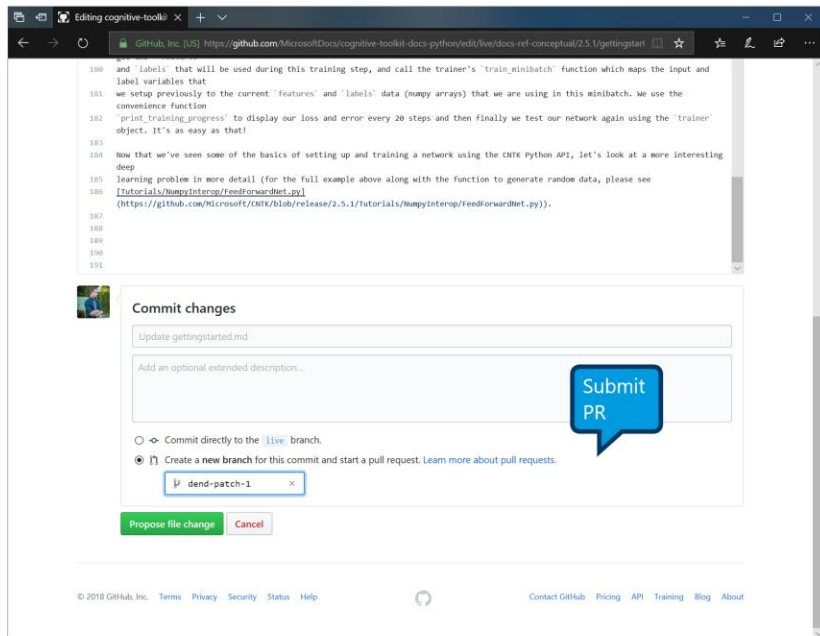
Is this page helpful?

Yes No

docs@local ~ % reboot



docs@local ~ % reboot



docs@local ~ % reboot

- Automation at the heart of the publishing process
 - **API Doc Tooling** (Node, Java, Python, .NET, REST, PowerShell, CLI)
 - **Content Build and Validation**
 - **Content Testing Suite** (404s, orphaned pages, SEO compliance)
 - **GitHub Bots** (automatically merge PRs, channel external feedback to internal bug tracker)
 - **Sample Code Indexing** (powered by GitHub & Azure infrastructure)

docs@local ~ % reboot

Making URLs readable

[https://msdn.microsoft.com/en-us/library/8ehhxeaf\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/8ehhxeaf(v=vs.110).aspx)



<https://docs.microsoft.com/dotnet/api/system.collections.generic.icomparer-1>

docs@local ~ % reboot

- Convention over configuration – we infer content structure from folders in GitHub.
- **/content/test.md** becomes **docs.microsoft.com/cloud/content/test**
- Easy to set up redirects when things change, directly from the repo – broken links are much easier to fix.

docs@local ~ % reboot

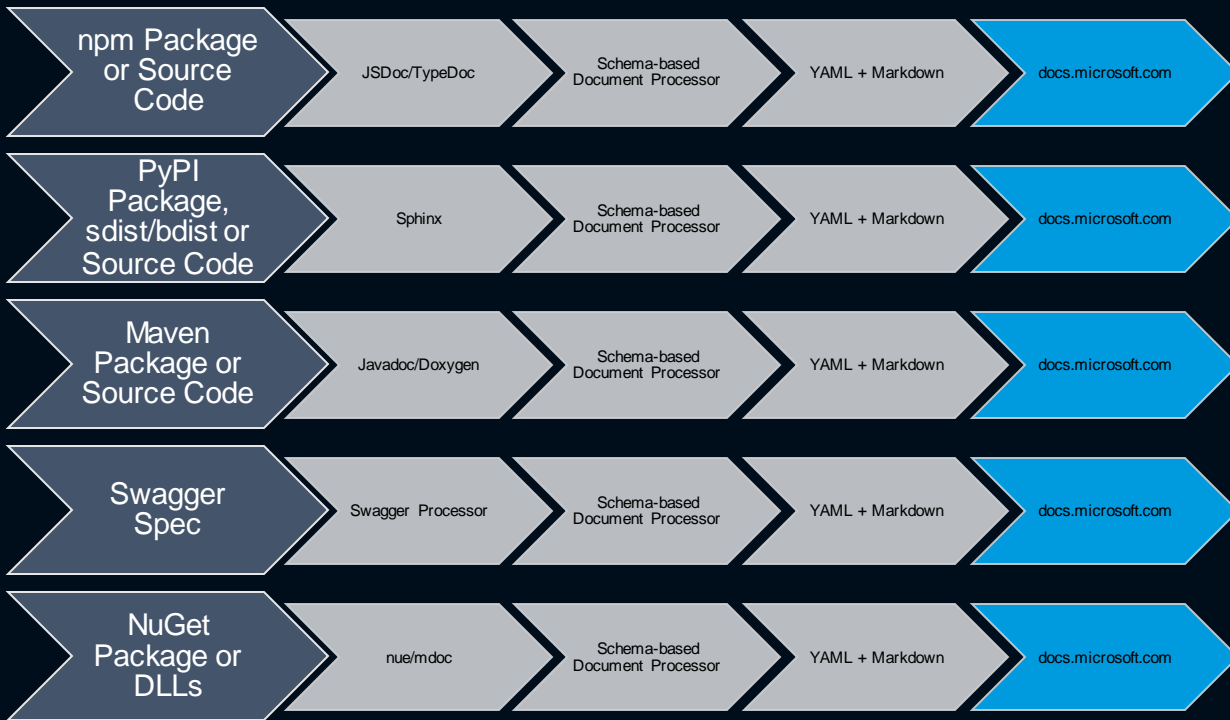
- Content Versioning

- No “burning in” into the URL.
- Ensures URL consistency even when new versions are released.
- Easily discoverable.
- Reduces friction and broken links.
- Using query param - ?view={moniker}

docs@local ~ % reboot

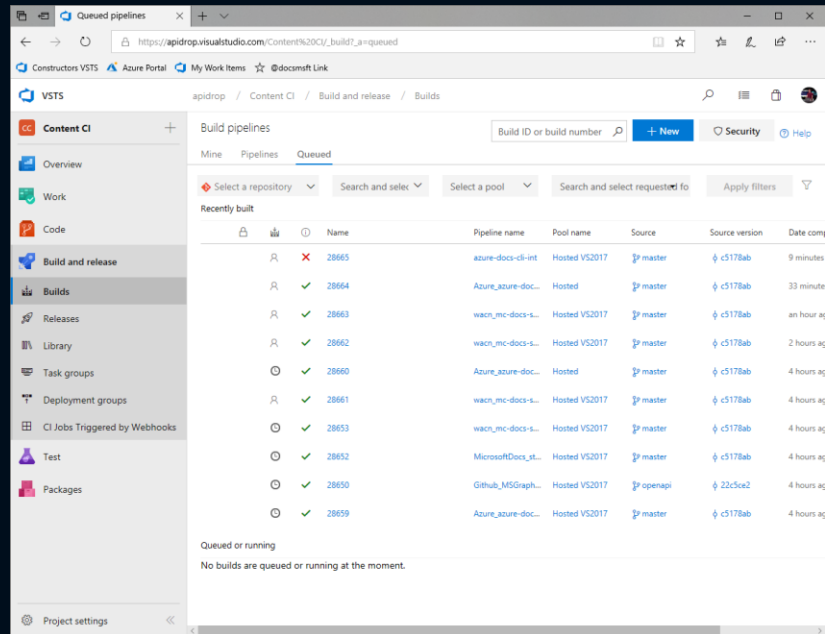
- API documentation discoverable from one place – the API Browser.
- No need to hop between N+1 sites to find the API.
- Semantic understanding of the APIs.
- Reduce discovery and documentation friction.
- Provide the artifacts (npm, pypi, source) and the docs are staged automatically.
- Intertwined with human-edited content.

docs@local ~ % reboot



docs@local ~ % reboot

- **100K+** API documentation CI executed in the past year.
- **10MM+** lines of auto-generated docs dropped into GitHub.



docs@local ~ % reboot

- This powers:
 - **9.5K+** JavaScript API documentation pages
 - **55K+** Java API documentation pages
 - **16K+** Python API documentation pages
 - **15K+** REST API documentation pages
 - **499K+** .NET API documentation pages

docs@local ~ % reboot

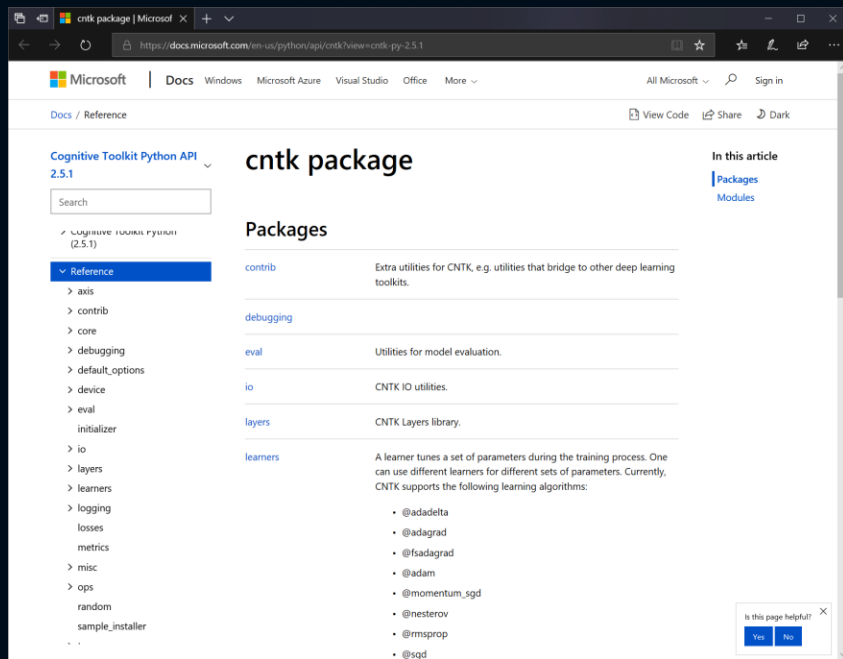


Builds run multiple times a day, always documenting latest public versions of APIs in addition to secondary (supported) versions.

docs@local ~ % reboot

- All API docs have standard URL patterns
 - /python/api/{package-name}/{entity}
 - /java/api/{entity-qualified-name}
 - /javascript/api/{package-name}/{entity}
 - /rest/api/{product}/{op-group}/{operation}
 - /cli/{product}/{command}

docs@local ~ % reboot



- Documentation linked to source code.
- Switch between versions on the fly.
- Logically grouped API entities in the table of contents.
- Grouping generated automatically – no human ever does that.
- Allows us to scale to 10K+ APIs in minutes.

docs@local ~ % reboot

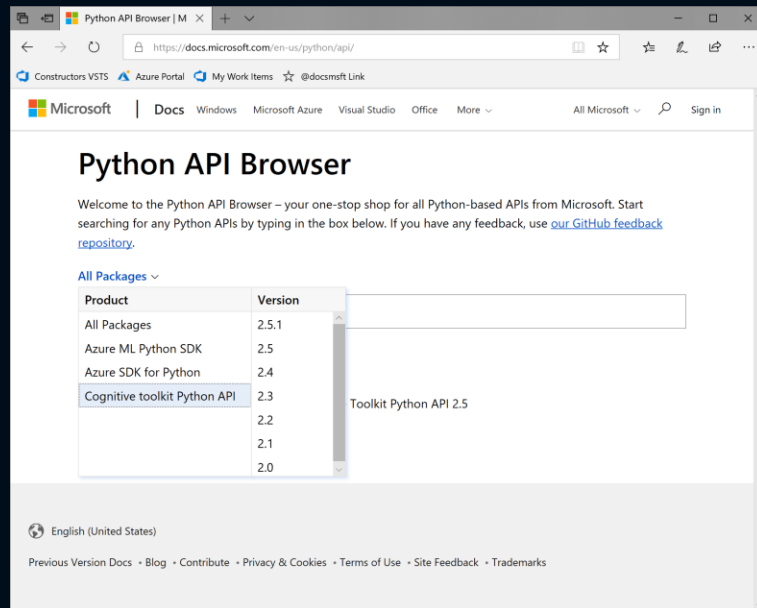
- Contracts over hand-crafted documents.
- Schema defines entities and overall hierarchy.
- Template globally applied.
- Driving consistency in presentation.
- Updates don't break existing documentation.

docs@local ~ % reboot

- Generate any post-processing artifacts after build – IntelliSense and cross-reference files.
- Artifacts can be used by product teams (Javadoc to be shipped with product).

docs@local ~ % reboot

- Structured documentation enables us to power rich API discovery experiences.
- Find the necessary API in seconds.
- Search across all products in a platform.
- IDE “auto-suggest” – in a search experience.



```
docs@local ~ % docs --run
```

docs@local ~ % docs --run

NEWS

Nation Shudders At Large Block Of Uninterrupted Text

3/09/10 5:00pm • SEE MORE: SCIENCE & TECHNOLOGY ▾



The giant mass of prose was devoid of so much as a large pulled quote for readers to glance at before moving on.

WASHINGTON—Unable to rest their eyes on a colorful photograph or boldface heading that could be easily skimmed and forgotten about, Americans collectively recoiled Monday when confronted with a solid block of uninterrupted text.

```
docs@local ~ % docs --run
```

- Good documentation is not a wall of text.
- Reducing friction from reading to trying – how can we allow you to see how things work in seconds?
- Structured content allows us to understand *where* we can enable interactivity.

docs@local ~ % docs --run

The screenshot shows a web browser window with the URL `https://docs.microsoft.com/en-us/rest/api/aks/managedclusters/createorupdate`. The page title is "Managed Clusters - Create Or Update". Below the title, it says "Service: AKS" and "API Version: 2018-03-31". The description states: "Creates or updates a managed cluster. Creates or updates a managed cluster with the specified configuration for agents and Kubernetes version." There is a "Try It" button. Below this, the "Request URL" is shown as `PUT https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/managedClusters/{clusterName}`. The "Parameters" section lists:

- `subscriptionId`: Pay-As-You-Go
- `resourceGroupName`: [input field]
- `resourceName`: [input field]
- `api-version`: 2018-03-31
- `name`: [input field]

 The "Headers" section shows:

- `Content-Type`: application/json
- `name`: [input field]

 A table titled "URI Parameters" is also present, listing the parameters and their descriptions.

Name	In	Required	Type	Description
<code>subscriptionId</code>	path	True	string	Subscription credentials which uniquely identify Microsoft Azure subscription. The subscription ID forms part of the URI for every service call.
<code>resourceGroupName</code>	path	True	string	The name of the resource group.
<code>clusterName</code>	path	True	string	The name of the cluster.

- REST “Try It”
- Powered by Swagger specs.
- Run REST calls from a documentation page.
- Instantly see output, with no apps involved.

docs@local ~ % docs --run

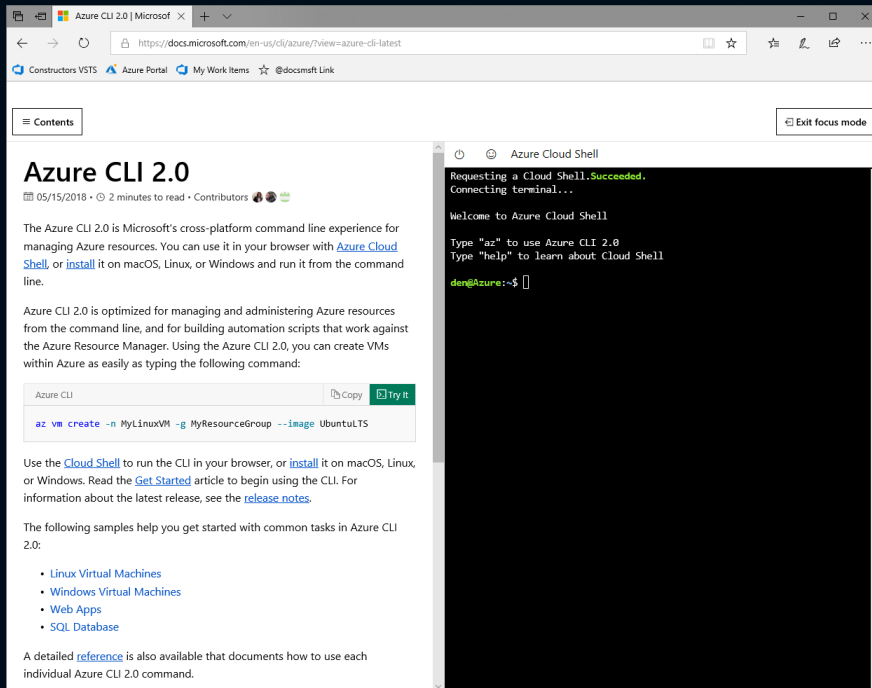
The screenshot shows the docs@local application interface. On the left, the documentation for the `String.Format Method` is displayed, including its namespace (`System`), assemblies, and a description of its function. Below this, the 'Overloads' section lists four different signatures of the `Format` method. On the right, the '.NET Editor' is active, showing a C# code snippet that uses `String.Format` to format a string. The code is as follows:

```
1 Decimal pricePerOunce = 17.36m;  
2 String s = String.Format("The current price is {0} per ounce.",  
3     pricePerOunce);  
4 Console.WriteLine(s);  
5 // Result: The current price is 17.36 per ounce.
```

Below the code editor, the 'Output' section shows the result of running the code: "The current price is 17.36 per ounce."

- .NET REPL
- Run C# code in a stateless container.
- Zero friction to get started — no auth required.
- Any C# snippet can integrate it.

docs@local ~ % docs --run



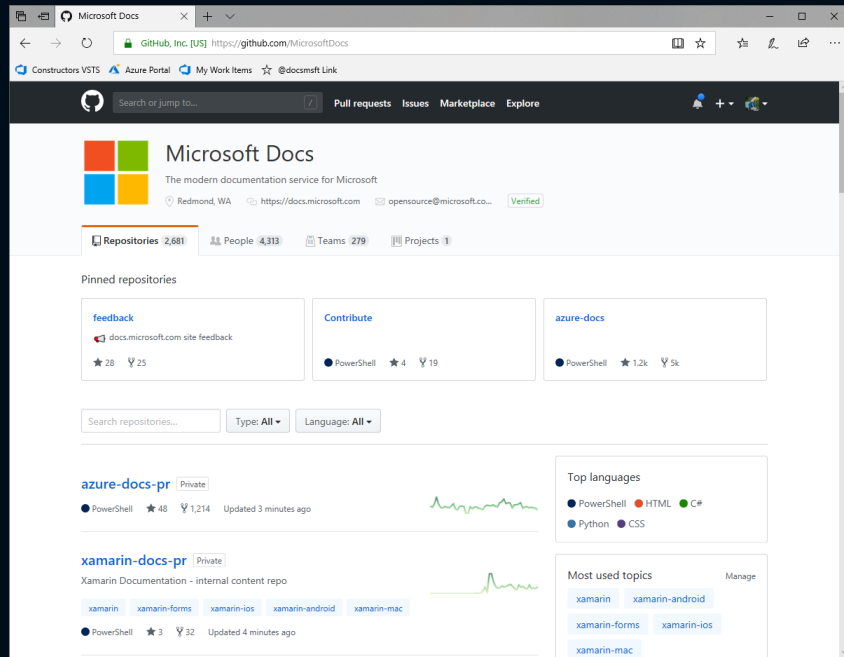
The screenshot shows a web browser window displaying the Azure CLI 2.0 documentation page. The page title is "Azure CLI 2.0" and it includes a "Contents" button. The main content area describes the Azure CLI 2.0 as Microsoft's cross-platform command line experience for managing Azure resources. It mentions that the CLI can be used in a browser with the Azure Cloud Shell, or installed on macOS, Linux, or Windows. A code block shows the command to create a Linux VM: `az vm create -n MyLinuxVM -g MyResourceGroup --image Ubuntu1604`. Below the code block, there are links to "Get Started" and "release notes". A list of sample tasks is provided: Linux Virtual Machines, Windows Virtual Machines, Web Apps, and SQL Database. A "Try it" button is also visible. On the right side of the screenshot, the Azure Cloud Shell terminal is open, showing the prompt `den@Azure: ~$` and the output of the `az vm create` command: "Requesting a Cloud Shell. Succeeded. Connecting terminal... Welcome to Azure Cloud Shell. Type 'az' to use Azure CLI 2.0. Type 'help' to learn about Cloud Shell."

- Azure Cloud Shell
- Linux in the browser.
- Works with Bash and PowerShell Core.
- Stateful container connected to Azure subscription.

```
docs@local ~ % docs --community
```

docs@local ~ % docs --community

- **2.5K+** repositories
 - **1.1K+** public
- **4.3K+** internal members
- A huge shift in how the entire company sees documentation and contributions to open source.



docs@local ~ % docs --community

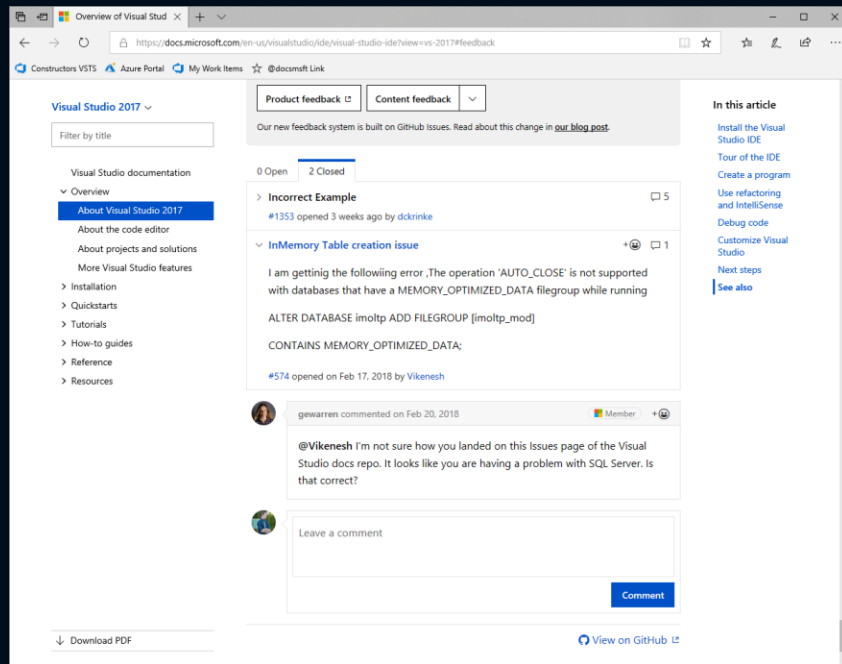


([stats courtesy of GitHub](#))

- A lot of our projects were moved over to GitHub (VSCode, TypeScript, .NET, Monaco Editor).
- Natural place to have documentation, with a huge community of passionate developers.

docs@local ~ % docs --community

- Shifting feedback from silo-ed platforms to be open.
- GitHub Issues – for content and site feedback.
- Documentation is treated like a product – doc issue = bug.



docs@local ~ % docs --community

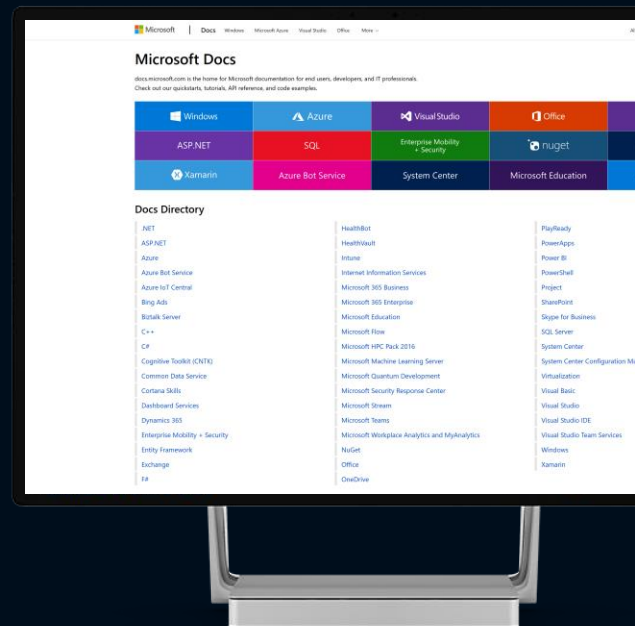
- Key learning – transparency matters.
- Your customers know their needs better than you do – talk to them. All the time.
- Working with your community is not the same as asking them to do the work for you.
- Fostering the community and building trust takes time – coaching them on best practices and approaches is important.

docs@local ~ % docs --community

- Automation is your friend (again)
 - Contribution License Agreements (CLAs)
 - PR reviews (*“Is my PR changing the right things?”*)
 - Content build validation (*“Is what I added causing issues?”*)
 - Test any inserted code (*“Does it build?”*)

docs@local ~ % docs --summary

	Before			After
Open Source Docs	✗	No	+	Yes
Localization	✗	Poor	+	64 Languages
Mobile Support	✗	None	+	Major platforms
Accessibility	—	Varied	+	Built-in
Content Location	—	Fragmented	+	Unified
Sample Testing	—	Sparse	+	Automated CI
API Docs	✗	Manual	+	Automatic
Feedback	—	Varied, closed	+	GitHub
Analytics	—	Fragmented	+	Unified
Engineering	✗	Duplicated	+	Shared



```
docs@local ~ % docs --legacy
```

docs@local ~ % docs --legacy



Expectation



Reality

`docs@local ~ % docs --legacy`

- Mo' sites, mo' problems.
- Not as simple as simple as shutting the old site down in favor of the new one.
- Content migration takes time — you will discover problems. A lot of problems.
- Redirection is important — customers don't like broken links. Neither do search engines.
- Links are “baked into” products over years — you don't want to break those.

docs@local ~ % docs --legacy

- You will inevitably get feedback that “old was better” – that’s not a cue to rebuild the old experience on the new site.
- Communication is important – set expectations.
- Habits die hard – it will take time for people to rely on new workflows.

docs@local ~ % docs --contact



@denniscode



@docsmsft



<https://docs.microsoft.com>



<https://aka.ms/docfx>



dendeli@microsoft.com