

# NetApp: Swagger UI Alternatives

Group Members: Sophie Sfeir, Anthony McGaw, Jarod Miller,  
Madison Haugh

# Sophie Sfeir

## About me:

- ❑ University of Pittsburgh Student, BS in Computer Science and Spanish, Graduating in Fall 2021

## Why I chose this project:

- ❑ I am interested in learning new ideas/concepts and wanted to learn more about APIs and how they work. I also wanted to gain more professional experience in software development.

# Anthony McGaw

## About me:

- ❑ University of Pittsburgh Student, BS in Computer Science, Graduating in May 2021

## Why I chose this project:

- ❑ I have recently been interested in UX and this felt like a interesting project to get experience working on a project which is important to end-users. Also valuable experience with Agile, React, and API's.

# Jarod Miller

## About me:

- ❑ University of Pittsburgh Student, BS in Computer Science, Graduating in May 2021

## Why I chose this project:

- ❑ I'm interested in gaining experience in full-stack website development and learning new technologies in the tech industry.

# Madison Haugh

About me:

- ❑ University of Pittsburgh Student, BS in Computer Science, Minor in Applied Statistics, Graduating in May 2021

Why I chose this project:

- ❑ I've had previous experience with full-stack web development and was interested in developing and expanding my skill set



# Outline

- ❑ Project Development Process
- ❑ Project Experience
- ❑ Project Goals
- ❑ Choosing a Visualizer
- ❑ Challenges
- ❑ Features/Functionalities
- ❑ Demo
- ❑ Testing
- ❑ Future Goals

# Development Process

## Student Team:

- ❑ Daily collaboration through our team Discord server
- ❑ Weekly meetings on Tuesday nights and Friday afternoons via Zoom
  - ❑ (which evolved into us adding Thursday nights as well)
- ❑ Split up certain feature development into sub-teams within our group

## + Netapp:

- ❑ Second collaborative Discord including our NetApp sponsors to ask questions and discuss challenges
- ❑ Weekly meetings Friday morning at 9am 🌞

# Project Experience

- ❑ Meet with team members frequently and work well together
- ❑ Research and project development process has been a great learning experience
- ❑ Communication with NetApp is positive and helpful
- ❑ Great industry learning experience



# Project Goals

- ❑ Investigate current and new visualizers to determine the best option based on functionality and user experience
- ❑ Research OpenAPI Specification and understand how and what the visualizer is displaying
- ❑ Implement or inherit a deep-search ability for the documentation
- ❑ Implement version tracking
- ❑ Stretch Goal: Enhance styling and determine other potential features

# Choosing an API Visualizer

## Swagger

- ❑ **Pros:** open source, currently implemented
- ❑ **Cons:** one-panel design which leads to long scrolling

## Redoc

- ❑ **Pros:** built-in search function, three-panel design, easy navigation
- ❑ **Cons:** search is not deep, large codebase

## Carte

- ❑ **Pros:** Based off of Swagger UI
- ❑ **Cons:** Built in Ruby, limited documentation

# Choosing an API Visualizer

## Slate

- ❑ **Pros:** built in search, three panel design
- ❑ **Cons:** yaml must be converted to markdown with third-party tool, written in ruby (unfamiliar)

## DapperDox

- ❑ **Pros:** side panel, easy customization,
- ❑ **Cons:** yaml must be converted to json, no search functionality

## LucyBot

- ❑ **Pros:** Search functionality, three panel design
- ❑ **Cons:** Simplistic, paid version

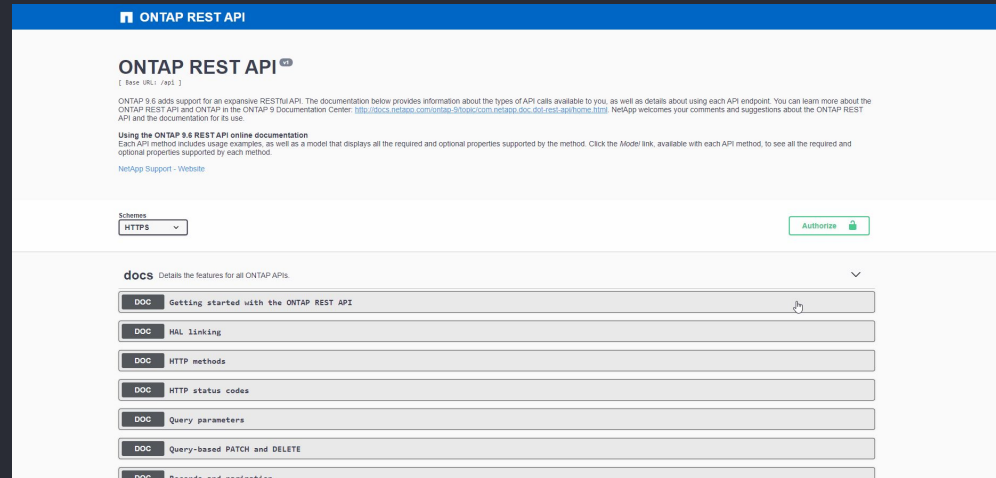
# Swagger UI

## Benefits:

- ❑ NetApp's current implementation
  - Has custom components
- ❑ Try-It-Out feature
- ❑ Built-in Model rendering


## Shortcomings:

- ❑ No deep search functionality
- ❑ Single panel design



scrolling....scrolling...scrolling

# Decision: Redoc



Search...

cluster

networking

svm

name-services

SAN

NVMe

application

support

security

storage

DOC /storage/volumes

GET /storage/volumes  
New in v9.9

GET /storage/volumes New in v9.9

Retrieves volumes.

**Expensive properties**

There is an added cost to retrieving values for these properties. They are not included by default in GET results and must be explicitly requested using the `fields` query parameter. See [Requesting specific fields](#) to learn more.

- `is_svm_root`
- `analytics.*`
- `application.*`
- `encryption.*`
- `clone.parent_snapshot.name`
- `clone.parent_snapshot.uuid`
- `clone.parent_svm.name`
- `clone.parent_svm.uuid`
- `clone.parent_volume.name`
- `clone.parent_volume.uuid`
- `clone.split_complete_percent`
- `clone.split_estimate`
- `clone.split_initiated`
- `efficiency.*`
- `error_state.*`
- `files.*`
- `nas.export_policy.id`
- `nas.gid`
- `nas.path`
- `nas.security_style`
- `nas.uid`

GET /storage/volumes

**Response samples**

200 default

Content type  
application/json

Copy Expand all Collapse all

```
{
  "_links": {
    + "self": { - },
    + "next": { - }
  },
  "num_records": 0,
  - "records": [
    + { - }
  ]
}
```

# Why Redoc?

## Reasoning

- ❑ Built-in search
  - Main feature
- ❑ Three-panel display
  - Much more user friendly
- ❑ Language
  - React, Typescript
  - Best suited for our skillset

## Shortcomings

- ❑ Extra work converting from Swagger UI

The image shows a side-by-side comparison of two API documentation tools: NetApp Redoc (left) and Swagger UI (right), both displaying the API endpoint `POST /storage/volumes`.

**NetApp Redoc (Left):**

- Features a sidebar with a search bar and a list of API endpoints categorized by tags (e.g., storage/volumes, /storage/volumes/{volume-uuid}/metrics, /storage/volumes/{volume-uuid}).
- The main content area shows the endpoint details, including a description, required properties (e.g., `svm.uuid`, `svm.name`, `name`, `aggregates.name`, `aggregates.uuid`), default property values (e.g., `state`, `size`, `style`, `type`, `encryption.enabled`, `snapshot.policy.name`, `guarantee.type`), and related ONTAP commands (e.g., `volume create`, `volume clone create`).
- The bottom section shows the request body schema, which is a JSON object with fields like `comment`, `language`, `comment`, and `language`.

**Swagger UI (Right):**

- Features a sidebar with a search bar and a list of API endpoints categorized by tags (e.g., storage/volumes, /storage/volumes/{volume-uuid}/metrics, /storage/volumes/{volume-uuid}).
- The main content area shows the endpoint details, including a description, required properties (e.g., `svm.uuid`, `svm.name`, `name`, `aggregates.name`, `aggregates.uuid`), default property values (e.g., `state`, `size`, `style`, `type`, `encryption.enabled`, `snapshot.policy.name`, `guarantee.type`), and related ONTAP commands (e.g., `volume create`, `volume clone create`).
- The bottom section shows the request body schema, which is a JSON object with fields like `comment`, `language`, `comment`, and `language`.

# Challenges

- 1 Reading through the OpenAPI specification and understanding the items within the documentation
- 2 Installing the necessary dependencies and getting our local development servers running
- 3 Learning and putting into practice the elements of an Agile Development Team



# Challenges cont...

- 4 Understanding and analyzing the large codebase from which we would be developing



- 5 Navigating our busy schedules and meeting as a team

- 6 Preserving features already implemented into SwaggerUI as we transition to Redoc



ReDoc

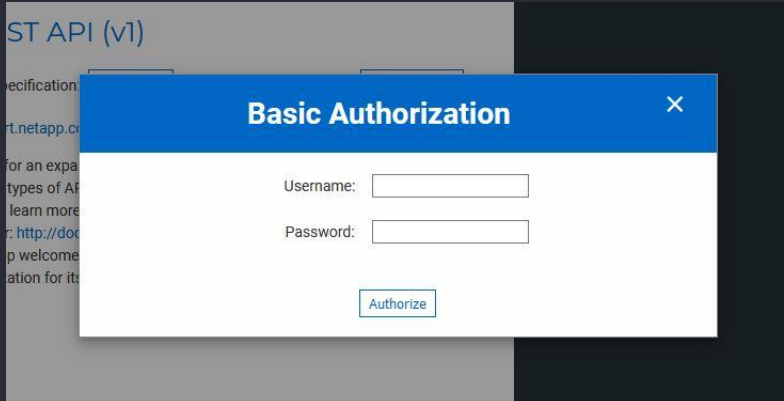




# Features and Functionalities

# Front-End Features

## Authorization Login



## Search Tooltip




## Editing Global Theme

```
main: '#0067C5', // NetApp Blue
```

# Transition Process

## Importing the models

 **NetApp**

Search...

Models

volume

volume

volume\_reference

volume\_response

volume\_metrics

volume\_statistics\_reference

error

error\_arguments

error\_response

error\_responses

job\_link\_response

href

self\_link

related\_link

collection\_links

volume

uuid

comment

create\_time

language

name

size

string

Unique identifier for the volume. This corresponds to the instance-uuid that is exposed in the CLI and ONTAPI. It does not change due to a volume move.

string [ 0 .. 1023 ] characters

A comment for the volume. Valid in POST or PATCH.

string <date-time>

Creation time of the volume. This field is generated when the volume is created.

string

Enum: "ar", "ascii", "utf-8", "u" ... 66 more

Language encoding setting for volume. If no language is specified, the volume inherits its SVM language encoding setting.

string [ 1 .. 203 ] characters

Volume name. The name of volume must start with an alphabetic character (a to z or A to Z) or an underscore (.). The name must be 197 or fewer characters in length for FlexGroups, and 203 or fewer characters in length for all other types of volumes. Volume names must be unique within an SVM. Required on POST.

integer

Physical size of the volume, in bytes. The minimum size for a FlexVol volume is 20MB and the minimum size for a FlexGroup volume is 200MB per constituent. The recommended size for a FlexGroup volume is a minimum of

```
{
  "uuid": "028baa66-41bd-11e9-81d5-00a0986138f7",
  "comment": "string",
  "create_time": "2018-06-04T19:00:00Z",
  "language": "ar",
  "name": "vol_cs_dept",
  "size": 0,
  "state": "error",
  "style": "flexvol",
  "is_svm_root": true,
  "access_time_enabled": true,
  "type": "rw",
  "links": {
    "self": { ... }
  },
  "aggregates": {
    "": { ... }
  },
  "constituent_per_aggregate": {},
  "use_mirrored_aggregates": true,
  "flexcache_endpoint_type": "none",
  "is_object_store": true,
  "application": {
    "name": "string"
  }
}
```

```
- name: models
  x-displayName: Models
  description: |
    ## volume
    <SchemaDefinition schemaRef="#/components/schemas/volume" />

    ## volume_reference
    <SchemaDefinition schemaRef="#/components/schemas/volume_reference" />

    ## volume_response
    <SchemaDefinition schemaRef="#/components/schemas/volume_response" />

    ## volume_metrics
    <SchemaDefinition schemaRef="#/components/schemas/volume_metrics" />

    ## volume_statistics_reference
    <SchemaDefinition schemaRef="#/components/schemas/volume_statistics_reference" />

    ## error
    <SchemaDefinition schemaRef="#/components/schemas/error" />

    ## error_arguments
    <SchemaDefinition schemaRef="#/components/schemas/error_arguments" />

    ## error_response
    <SchemaDefinition schemaRef="#/components/schemas/error_response" />

    ## error_responses
    <SchemaDefinition schemaRef="#/components/schemas/error_responses" />

    ## job_link_response
    <SchemaDefinition schemaRef="#/components/schemas/job_link_response" />
```

# Doc Tags - Backend

```
export interface OpenAPITag {  
  name: string;  
  description?: string;  
  longDescription?: string;  
  externalDocs?: OpenAPIExternalDocumentation;  
  'x-displayName'?: string;  
}
```

openapi.ts



```
export type MenuItemGroupType = 'group' | 'tag' | 'section';  
  
export type MenuItemType = MenuItemGroupType | 'operation';  
/** Generic interface for MenuItems */  
export interface IMenuItem {  
  id: string;  
  absoluteIdx?: number;  
  name: string;  
  description?: string;  
  longDescription?: string;  
}
```

MenuStore.ts



```
export class GroupModel implements IMenuItem {  
  // #region IMenuItem fields  
  id: string;  
  absoluteIdx?: number;  
  name: string;  
  description?: string;  
  longDescription: string;  
  type: MenuItemGroupType;  
  
  items: ContentItemModel[] = [];
```

Group.model.ts

```
this.description = tagOrGroup.description || '';  
this.longDescription = tagOrGroup['x-ntap-long-description'] || '';
```

# Doc Tags - Frontend

```
// longDescription added to this list of constant variables being destructured  
const { name, description, longDescription, externalDocs, level } = this.props.item as GroupModel;
```

```
return (  
  <>  
    <Row>  
      <MiddlePanel compact={false}>  
        <Header>  
          <ShareLink to={this.props.item.id} />  
          {name}  
        </Header>  
      </MiddlePanel>  
    </Row>  
    <AdvancedMarkdown source={description || ''} htmlWrap={middlePanelWrap} />  
    <AdvancedMarkdown source={longDescription || ''} htmlWrap={middlePanelWrap} />  
    {externalDocs && (  
      <Row>  
        <MiddlePanel>  
          <ExternalDocumentation externalDocs={externalDocs} />  
        </MiddlePanel>  
      </Row>  
    )}  
  </>  
);
```

ContentItems.tsx

# Search Functionality



- ❑ Built with lunr.js
  - ❑ Small, full-text search library

## Example

A very simple search index can be created using the following:

```
var idx = lunr(function () {  
  this.field('title')  
  this.field('body')  
  
  this.add({  
    "title": "Twelfth-Night",  
    "body": "If music be the food of love, play on: Give me excess of it...",  
    "author": "William Shakespeare",  
    "id": "1"  
  })  
})
```

Searchable index is built with passed in fields and items

## SearchWorker.worker.ts

```
function initEmpty() {
  builder = new lunr.Builder();
  builder.field('title');
  builder.field('description');
  builder.ref('ref');

  builder.pipeline.add(lunr.trimmer, lunr.stopWordFilter, lunr.stemmer);

  index = new Promise(resolve => {
    resolveIndex = resolve;
  });
}

initEmpty();

const expandTerm = term => '*' + lunr.stemmer(new lunr.Token(term, {})) + '*';

export function add<T>(title: string, description: string, meta?: T) {
  const ref = store.push(meta) - 1;
  const item = { title: title.toLowerCase(), description: description.toLowerCase(), ref };
  builder.add(item);
}
```



## SearchWorker.worker.ts (modified)

```
function initEmpty() {
  builder = new lunr.Builder();
  builder.field('title');
  builder.field('description');
  builder.field('longDescription');
  builder.ref('ref');

  builder.pipeline.add(lunr.trimmer, lunr.stopWordFilter, lunr.stemmer);

  index = new Promise(resolve => {
    resolveIndex = resolve;
  });
}

initEmpty();

const expandTerm = term => '*' + lunr.stemmer(new lunr.Token(term, {})) + '*';

export function add<T>(title: string, description: string, longDescription: string, meta?: T) {
  const ref = store.push(meta) - 1;
  const item = { title: title.toLowerCase(), description: description.toLowerCase(), longDescription: longDescription.toLowerCase(), ref };
  builder.add(item);
}
```

```
import Worker from './SearchWorker.worker';

function getWorker() {
  let worker: new () => Worker;
  if (IS_BROWSER) {
    try {
      // tslint:disable-next-line
      worker = require('workerize-loader?inline&fallback=false!./SearchWorker.worker');
    } catch (e) {
      worker = require('./SearchWorker.worker').default;
    }
  } else {
    worker = require('./SearchWorker.worker').default;
  }
  return new worker();
}

export class SearchStore<T> {
  searchWorker = getWorker();

  indexItems(groups: Array<IMenuItem | OperationModel>) {
    const recurse = items => {
      items.forEach(group => {
        if (group.type !== 'group') {
          this.add(group.name, group.description, group.longDescription || '', group.id);
        }
        recurse(group.items);
      });
    };
    recurse(groups);
    this.searchWorker.done();
  }

  add(title: string, body: string, longDescription: string, meta?: T) {
    this.searchWorker.add(title, body, longDescription, meta);
  }
}
```

## SearchStore.ts

Initial code changes to add x-ntap-longDescription to search index

Base search in Redoc only goes through section titles and descriptions.

# Deep Search Index Builder

```
const recurse = items => {
  items.forEach(group => {
    if(group.type !== 'group') {
      // only operation types have the parameters/responses/etc.
      if(group.type === 'operation') {
        const params: string[][] = this.addParams(group.parameters);
        const req: string[][][] = this.addRequestBody(group.requestBody);
        const resp: string[][][] = this.addResponses(group.responses);
        const objects = req[0].concat(resp[0]);
        const properties = req[1].concat(resp[1]);
        this.add(group.name, group.description || '', group.longDescription || '',
          params[0], params[1], objects, properties, group.httpVerb, group.name, group.id);
      }
      else {
        this.add(group.name, group.description || '', group.longDescription || '',
          ['', ''], [['']], [['']], '', '', group.id);
      }
    }
    recurse(group.items);
  });
};
recurse(groups);
```

// function that returns the paths and queries of an operation as an object

```
addParams(parameters: FieldModel[]): string[][] {
```

```
  const paths: string[] = [];
  const queries: string[] = [];
  parameters.forEach(parameter => {
    if(parameter.in === "path") {
      paths.push(parameter.name);
    }
    if(parameter.in === "query") {
      queries.push(parameter.name);
    }
  });
  return [paths, queries];
}
```

## SearchStore.ts

```
getDeepFields(field: FieldModel, temp: ReturnObj): ReturnObj | any {
  // if a field has a schema with other fields
  if(field.schema.fields !== undefined) {
    field.schema.fields.forEach(f => {
      temp = this.getDeepFields(f, temp);
    })
    // adds the field's name to the array of objects
    if(field.name !== undefined) {
      temp.objects.push(field.name);
    }
    return temp;
  }
  // if a field's schema doesn't have fields but the field's schema does have items in it
  if(field.schema.items !== undefined && field.schema.items.fields !== undefined) {
    field.schema.items.fields.forEach(f => {
      temp = this.getDeepFields(f, temp);
    });
    // adds the field's name to the array of objects
    if(field.name !== undefined) {
      temp.objects.push(field.name);
    }
    return temp;
  }
  // this is where we would like to add properties
  if(field.name !== undefined) {
    temp.properties.push(field.name);
    return temp;
  }
}
```



```
const regex = /(POST|GET|PATCH|DELETE) | ([a-z\\.\-\\{\}\\+]) | (TITLE|PATH|QUERY|PROPERTY|OBJECT|ENDPOINT)\\[(.+?)\\]/g;
```

adding a single term to a query

```
query.term("foo")
```

adding a single term to a query and specifying search fields, term boost and automatic trailing wildcard

```
query.term("foo", {  
  fields: ["title"],  
  boost: 10,  
  wildcard: lunr.Query.wildcard.TRAILING  
})
```

```
queryObject.term([searchItems[count], {  
  fields: [field]  
}])
```

'get' was a word filtered out of lunr by default (caused us some trouble)

Q PATH[uuid]

- GET** /storage/volumes/{volume.uuid}/metrics  
Deprecated in v9.9
- DEL** /storage/volumes/{uuid}  
Introduced in v9.8
- GET** /storage/volumes/{uuid}  
Introduced in v9.7
- PATCH** /storage/volumes/{uuid}  
Introduced in v9.7

Q GET PATH[uuid]

- GET** /storage/volumes/{volume.uuid}/metrics  
Deprecated in v9.9
- GET** /storage/volumes/{uuid}  
Introduced in v9.7

26

# Testing

□ Debug code using Inspect element + console in web browser

```
▼ Array(18)
  ► 0: GroupModel {items: Array(0), id: "section/Using-the-ONTAP-REST-API-online-documentation", type: "section"...}
  ► 1: GroupModel {items: Array(0), id: "section/Features-for-all-ONTAP-APIs", type: "section", ...}
  ► 2: GroupModel {items: Array(0), id: "tag/cloud", type: "tag", ...}
  ► 3: GroupModel {items: Array(0), id: "tag/cluster", type: "tag", ...}
  ► 4: GroupModel {items: Array(0), id: "tag/networking", type: "tag", ...}
  ► 5: GroupModel {items: Array(0), id: "tag/svm", type: "tag", ...}
  ► 6: GroupModel {items: Array(0), id: "tag/name-services", type: "tag", ...}
  ► 7: GroupModel {items: Array(0), id: "tag/SAN", type: "tag", ...}
  ► 8: GroupModel {items: Array(0), id: "tag/NVMe", type: "tag", ...}
  ► 9: GroupModel {items: Array(0), id: "tag/application", type: "tag", ...}
  ► 10: GroupModel {items: Array(0), id: "tag/support", type: "tag", ...}
  ► 11: GroupModel {items: Array(0), id: "tag/security", type: "tag", ...}
  ► 12: GroupModel {items: Array(7), id: "tag/storage", type: "tag", ...}
  ► 13: GroupModel {items: Array(0), id: "tag/snapmirror", type: "tag", ...}
  ► 14: GroupModel {items: Array(0), id: "tag/ndmp", type: "tag", ...}
  ► 15: GroupModel {items: Array(0), id: "tag/snaplock", type: "tag", ...}
  ► 16: GroupModel {items: Array(0), id: "tag/NAS", type: "tag", ...}
  ► 17: GroupModel {items: Array(24), id: "tag/models", type: "tag", ...}
    length: 18
    __proto__: Array(0)
```

```
▼ 12: GroupModel
  absoluteIdx: 12
  active: (...)
  depth: 1
  description: "Manages physical and logical storage"
  expanded: (...)
  externalDocs: undefined
  id: "tag/storage"
  ▼ items: Array(7)
    ► 0: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 1: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 2: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 3: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 4: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 5: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    ► 6: OperationModel {parser: OpenAPIParser, operationSpec: {...}, options: RedocNormalizedOptions, type: "..."}
    length: 7
    ► __proto__: Array(0)
  level: 1
  longDescription: "## Overview#The ONTAP storage APIs can be used to manage physical and logical storage..."
  name: "storage"
  parent: undefined
  type: "tag"
  ► Symbol(mobx administration): ObservableObjectAdministration {target_: GroupModel, values_: Map(2), name_...}
  Symbol(mobx-applied-decorators): true
  ► get active: f ()
  ► set active: f (v)
  ► get expanded: f ()
  ► set expanded: f (v)
```

# Future Goals

- ❑ Ensure that search functionality indexes all required sections
  - ❑ Parameters, object definitions, responses
  - ❑ Maximize search performance
- ❑ Customize styling
  - ❑ Make it more user friendly
  - ❑ Fit it to the NetApp color scheme
  - ❑ Investigate built in Redoc theming options
- ❑ Automate Version Tracking

**POST**

**/storage/volumes**

**Introduced in v9.6**

**POST**

**/storage/volumes**

Introduced in v9.6

**GET**

**/storage/volumes/{uuid}**

Introduced in v9.7

**PATCH**

**/storage/volumes/{uuid}**

Introduced in v9.7

**DEL**

**/storage/volumes/{uuid}**

Introduced in v9.8

**GET**

**/storage/volumes  
/{volume.uuid}/metrics**

Deprecated in v9.9

# Thank you

Special thanks to Anuradha Kulkarni,  
Sami Benbourenane, and Brian  
Kinkade for the guidance they have  
given us so far, and we are excited to  
continue finishing this project with  
your help.

The slide features a dark blue background with abstract, thin, light-colored geometric lines in the corners. In the top right, there is a complex arrangement of overlapping lines forming a series of triangles and polygons. In the bottom left, a similar but simpler geometric pattern is visible. The central text 'Questions?' is rendered in a light-colored, sans-serif font.

# Questions?