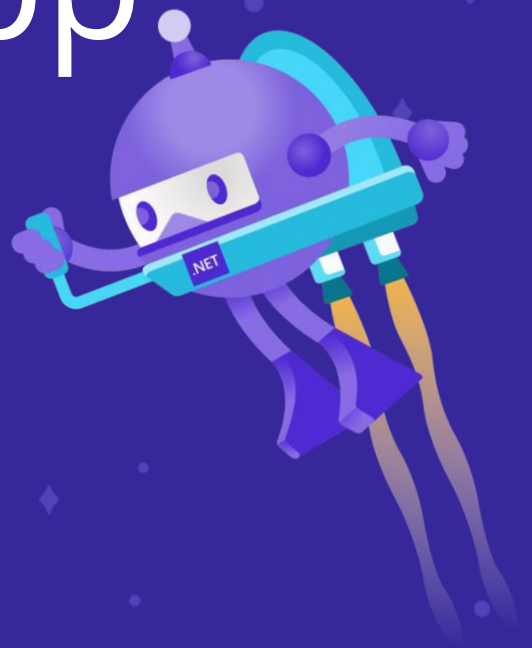
Build and run an Azure Static Web App with a backend API

Vaibhav Gujral

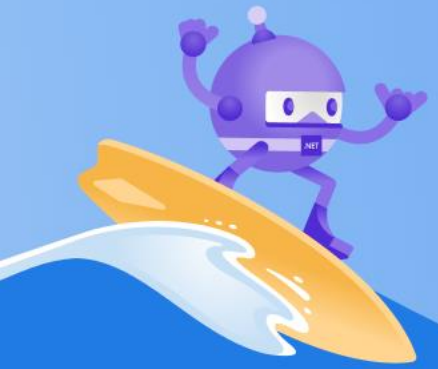
Microsoft Azure MVP

https://vaibhavgujral.com/



About me

- Cloud Architect at Kiewit Corp
- Microsoft Azure MVP
- Microsoft Certified Trainer
- Microsoft Certified Azure Solutions Architect
- Organizer, Omaha Azure User Group
- Blog: https://vaibhavgujral.com/
- Twitter: @vabgujral



Running static web sites in Azure

Azure offers few options to run static web sites –

1. Azure Storage – Supported by General Purpose V2 storage accounts
   - Accessible over http
   - Cost effective
   - Integration with Azure CDN is required for some common tasks
   - Few known challenges like
     - You cannot set default documents,
     - Custom Error pages cannot be returned

2. Azure App Services – Highly scalable web hosting service
   - Loaded with features like load balancing, auto-scaling and security
   - Not necessarily, is the best choice to run static web sites

What are Azure Static Web Apps?

Azure Static Web Apps* is an Azure service that automatically builds and deploys full stack web applications to Azure from a GitHub repository.

Ideal for static web apps built using libraries and frameworks like Angular, React, Vue and Blazor

Can be created through Azure portal, Azure CLI or Visual Studio Code

Two pieces to it:

- Static Content – which is served through geographically distributed locations
- Serverless API endpoints – using Azure functions

* - Azure Static Web Apps are currently in public preview

What are Azure Static Web Apps?

Features:

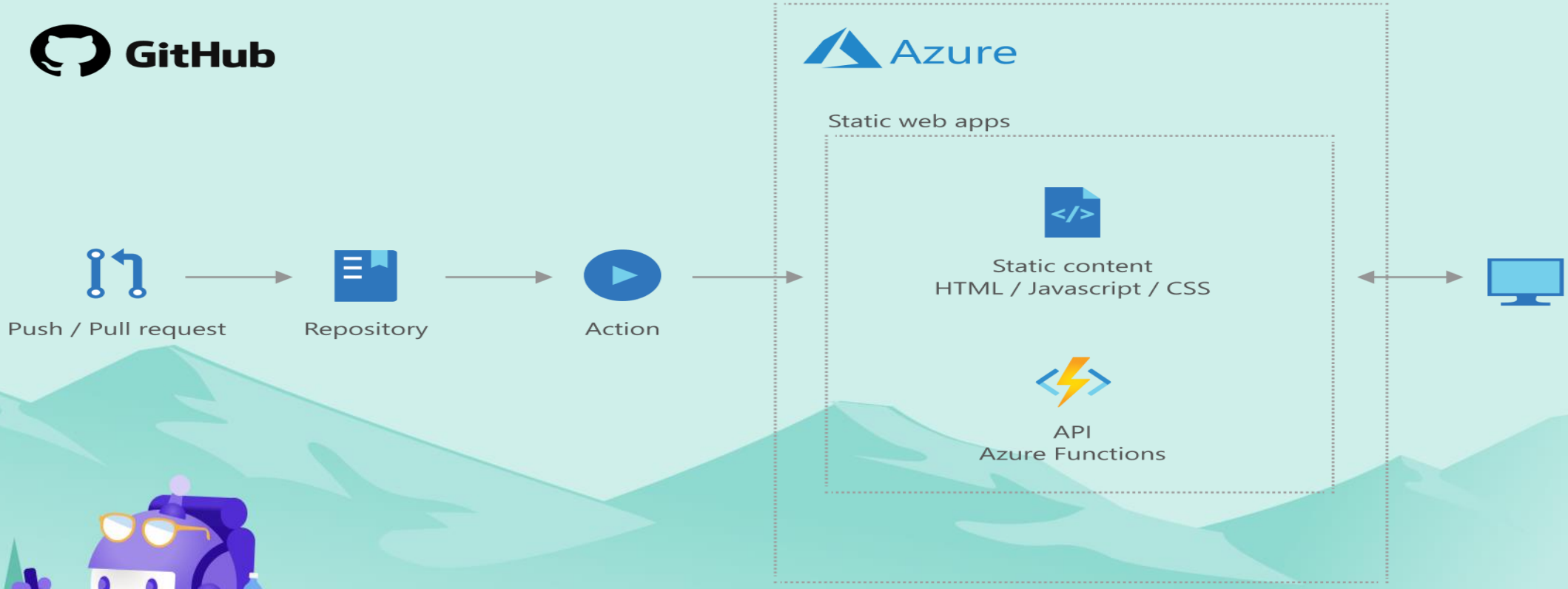
- Supports hosting static content like HTML, CSS, JavaScript and Images
- Supports popular web application frameworks like Vue.js, React, Angular, Svelte, and Blazor
- Supports static site frameworks like Gatsby, Hugo, VuePress
- Integrated API support provided by Azure Functions
- First-class GitHub Integration (yes, no support for Azure DevOps as of now)

What are Azure Static Web Apps?

Features:

- Globally distributed static content
- Free SSL certificates
- Custom domains
- Seamless security model
- Integration with many authentication providers
- Custom Authorization role definitions and assignments
- Auto-generated pre-production versions for every pull request

What are Azure Static Web Apps?



Demo

Azure Static Web Apps

API support

Azure Static Web Apps provides serverless API endpoints via Azure Functions

Limitations –

1. The API route prefix must be api
2. API must be a JavaScript, TypeScript, Python or C# Azure Function App
3. Route rules for API functions only support redirects and securing routes with roles
4. Only HTTP triggers are supported
5. Logs are only available with Application Insights

Routing

Routes for the static content and api are defined as an array of rules in *routes.json* file

routes.json file must exist at the root of the build artifact directory

Framework / Library	Location
Angular	assets
React	public
Svelte	public
Vue	public
Blazor	wwwroot

Routing

Routes in static web apps lets you define two things:

- Back-end routing rules
- Authorization roles

Rules are executed in the sequence as they appear in the array

Each rule supports one or more following properties:

Route Property	Description
route	The route pattern requested by the caller
serve	File or path returned from the request
allowedRoles	An array of role names
statusCode	The HTTP status code response for the request

Authentication and Authorization

Use Invitations to associate users with roles and authorize users to routes based on rules defined in *routes.json*

Following providers are supported:

- Azure Active Directory
- GitHub
- Facebook
- Google
- Twitter

Two built-in authorization roles:

- anonymous
- authenticated

User information can be accessed in Azure Static Web apps (/.auth/me)

routes.json

{} routes.json > [ ] routes

```
{
    "routes": [
      {
        "route": "/profile",
        "allowedRoles": ["authenticated"]
      },
      {
        "route": "/api/admin",
        "allowedRoles": ["administrator"]
      },
      {
        "route": "/customers/contoso",
        "allowedRoles": ["administrator", "customers_contoso"]
      },
      {
        "route": "/login",
        "serve": "/.auth/login/github"
      },
      {
        "route": "/logout",
        "serve": "/.auth/logout"
      },
      {
        "route": "/calendar/*",
        "serve": "/calendar.html"
      }
    ],
    "platformErrorOverrides": [
      {
        "errorType": "NotFound",
        "serve": "/custom-404.html"
      }
    ],
    "defaultHeaders": {
      "content-security-policy": "default-src https: 'unsafe-eval' 'unsafe-inline'; object-src 'none'"
    },
    "mimeTypes": {
        "custom": "text/html"
    }
```

Environments

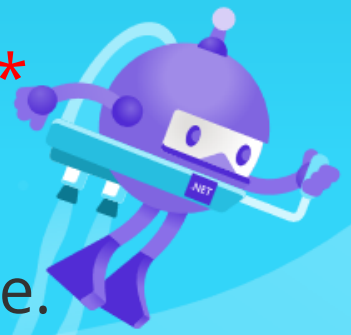
Pre-production environments in ASWA provides a full-fledged staging environment

When a pull request is created against a branch that the Actions workflow watches, a pre-production environment is built

Once you merge your pull request, the changes are merged to the production environment

Multiple pre-production environments can co-exist at a time*

Staged versions of your application are currently accessible publicly by their URL, even if your GitHub repository is private.

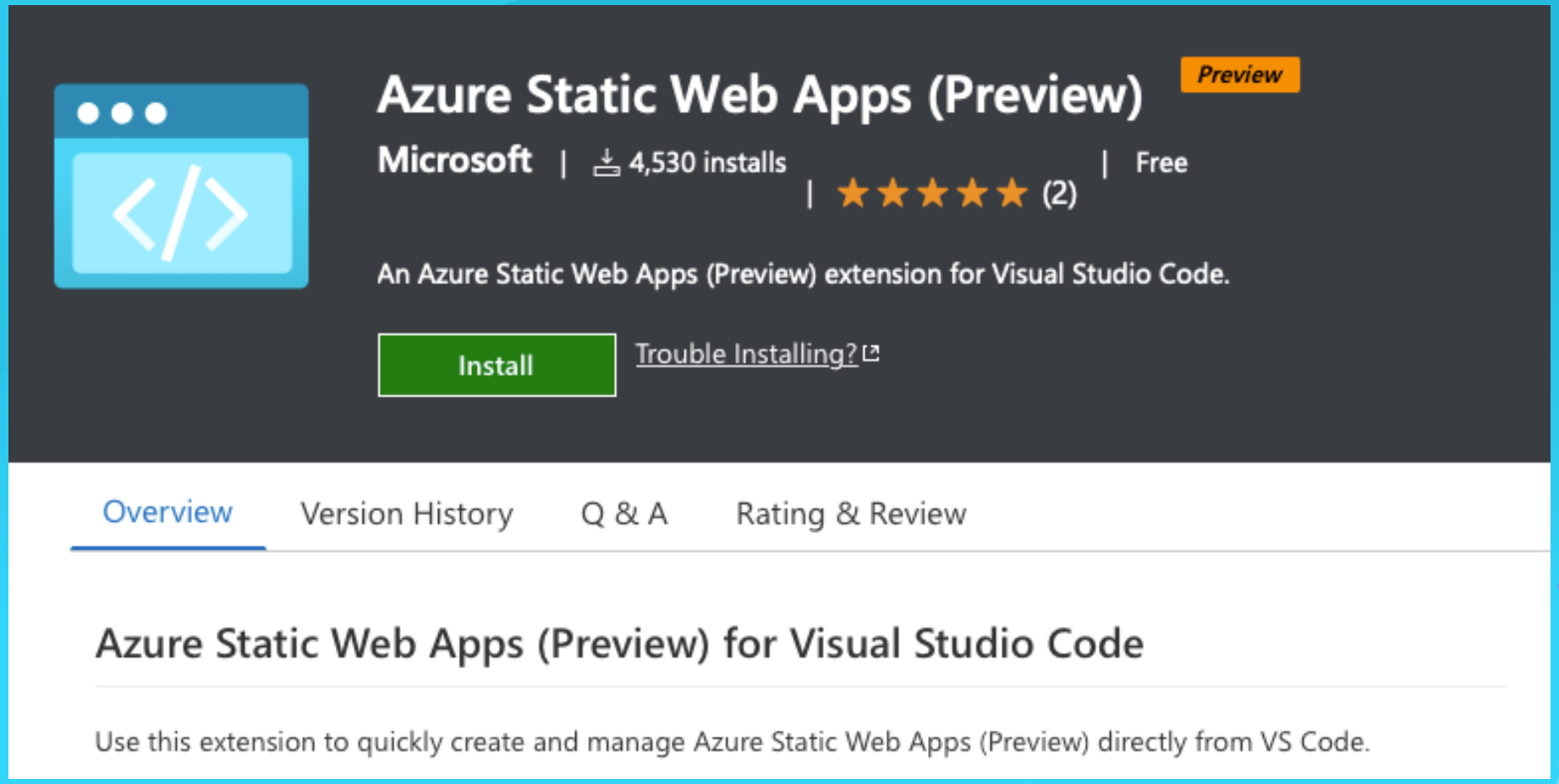


* - During preview, only one staging environment is allowed

Demo

Azure Static Web Apps

Azure Static Web Apps (Preview) extension for Visual Studio Code



https://aka.ms/savscode

References

- Azure Static Web Apps Documentation: https://docs.microsoft.com/en-us/azure/static-web-apps/
- GitHub starter: https://github.com/staticwebdev/blazor-starter/generate
- Blazor: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor
- Blazor and C# APIs - https://docs.microsoft.com/en-us/azure/static-web-apps/deploy-blazor
- GitHub Actions: https://docs.github.com/en/actions
- Visual Studio Code extension - https://aka.ms/savscode

Thanks for joining!