**Azure Web App Auto-Deployment via GitHub Actions**

**LinkedIn Video Reference:** [Video LinkedIn](https://www.linkedin.com/feed/update/urn:li:activity:7191082240196419584/?originTrackingId=Kmo4qM2hSvW7ICWRWOqSCw%3D%3D)

**Azure resources being leveraged:**

1. Azure App Service
2. Azure Web Apps
3. Azure Managed Identity
4. Azure subscription

**Other resources:**

1. Github Desktop, Actions, and secrets

**Steps towards deployment (high-level):**

1. Create resource group, create an app service, connect deployment method for web app to GitHub repository
   1. Resource groups allow us to contain multiple resources together for solutions
   2. App service allows us to build, deploy, and scaling web apps. We publish code via a github connection, and select a region for the resource
      1. Configure runtime stack, OS, region, pricing plan for hardware (Premium V3)
   3. In azure deployment center, we can connect github repo to app service and enable continuous deployment, app gets updated when we push changes to the branch
      1. Configure source, org, repo, branch, and preview workflow .yml produced
2. Create startup scripts to run on deployment via startup command parameter, installs packages and updates packages, start gunicorn server to run flask application
   1. The Gunicorn "Green Unicorn" is a Python Web Server Gateway Interface HTTP server. It is a pre-fork worker model, ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with several web frameworks, simply implemented, light on server resources and fast.
      1. --bind 0.0.0.0 and –timeout is 600 and –chdir
      2. Set name of initial script in start command argument in Azure settings
3. Configure workflow to create virtual environment, install packages, etc. before deploying
   1. Add steps in workflow to run system commands before packaging and deployment
4. Update Azure configuration with keys and secrets for APIs
   1. Openai, pinecone, other API keys and soon identity secret
5. Configure workflow file to log into Azure CLI via service principal
   1. Add step to log into CLI
   2. Configure service principle by running command to create managed identity and output JSON containing credentials of identity
   3. Update Azure credentials in GitHub Actions secrets in settings for repo
6. Update app.py file to run specified host and port
   1. Run on 0.0.0.0 and port 8000 to ensure app listens to all incoming connections
   2. To enhance, add log in method or verification to provide credentials or API keys
7. Commit change to push changes and trigger deployment to Azure