Name: SHRUTI KATYAL

Batch: 7

Employee Code: 655382

**Project Management Setup:**

In Azure DevOps, you can use the Azure Boards service to manage user stories, epics, and sprints for your projects using Agile or Scrum methodologies. Here's are some steps for creation of user stories, epics, and sprints:

**1. Create Epics:**

* **Step 1:** Sign in to your Azure DevOps account.
* **Step 2:** Navigate to your project.
* **Step 3:** Go to the "Boards" tab in the left sidebar.
* **Step 4:** Click on "Work items" to open the work items list.
* **Step 5:** Click on "New Work Item" and select "Epic."
* **Step 6:** Fill in the required information for the epic, such as title, description, and any other relevant details.
* **Step 7:** Save the epic.

**2. Create User Stories:**

* **Step 1:** Follow the same steps as above until step 5.
* **Step 2:** Select "User Story" instead of "Epic" in the "New Work Item" menu.
* **Step 3:** Fill in the details for the user story, including title, description, acceptance criteria, and any other relevant information.
* **Step 4:** Associate the user story with the appropriate epic using the "Parent" field.
* **Step 5:** Save the user story.

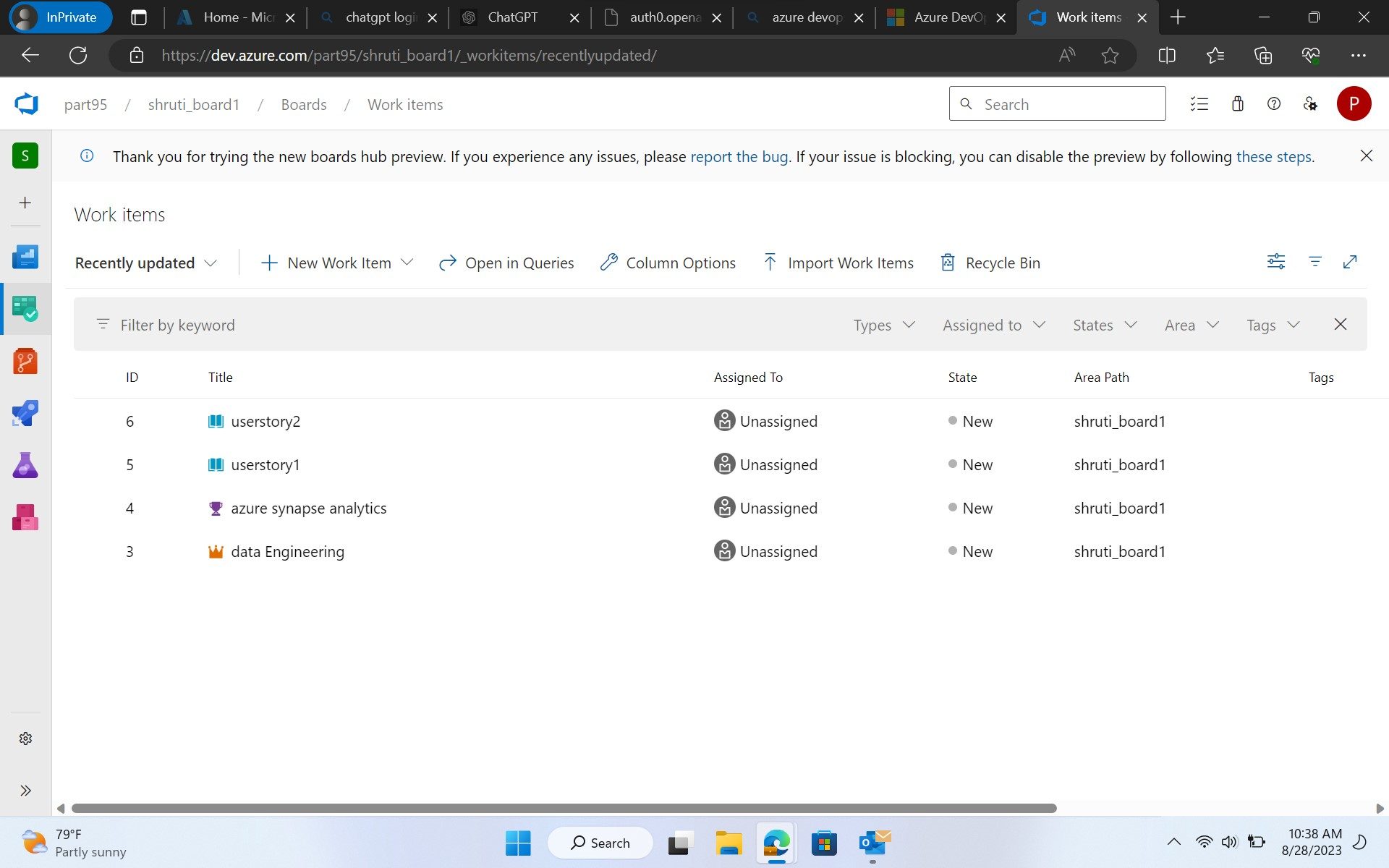
**3. Plan Sprints:**

* **Step 1:** In the "Boards" tab, click on "Sprints" to manage your sprints.
* **Step 2:** Click on "New sprint" to create a new sprint.
* **Step 3:** Provide a name for the sprint and specify the start and end dates.
* **Step 4:** Define the sprint goals and select the user stories and work items that you want to include in the sprint.
* **Step 5:** Save the sprint.
* **Step 6:** Once the sprint is planned, you can move user stories and tasks into the "Active" column to indicate that they are part of the current sprint.

A screenshot of a computer

Description automatically generated

1.

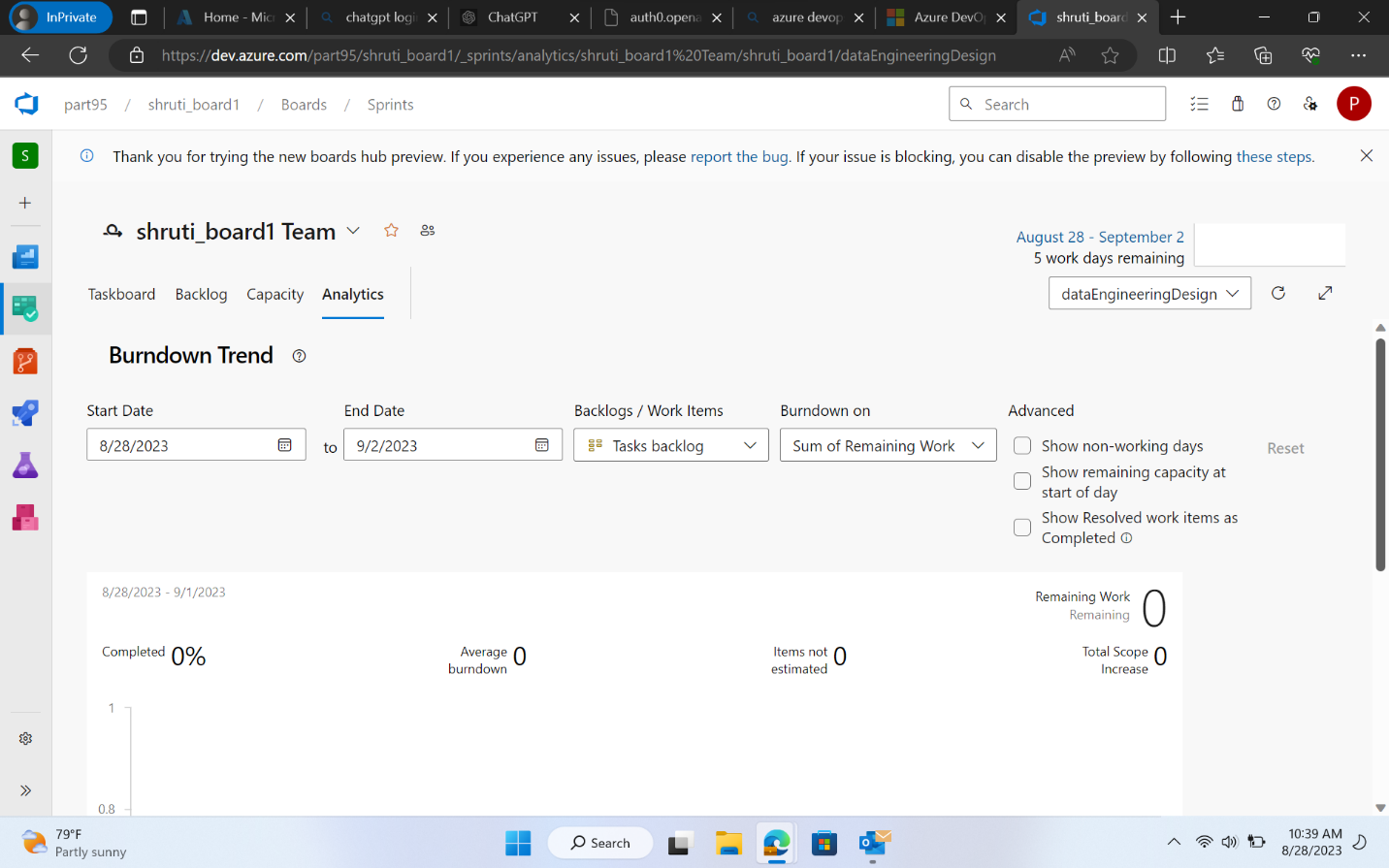


2.

A screenshot of a computer

Description automatically generated

3.



4.

**Git Hub repository and feature branching:**

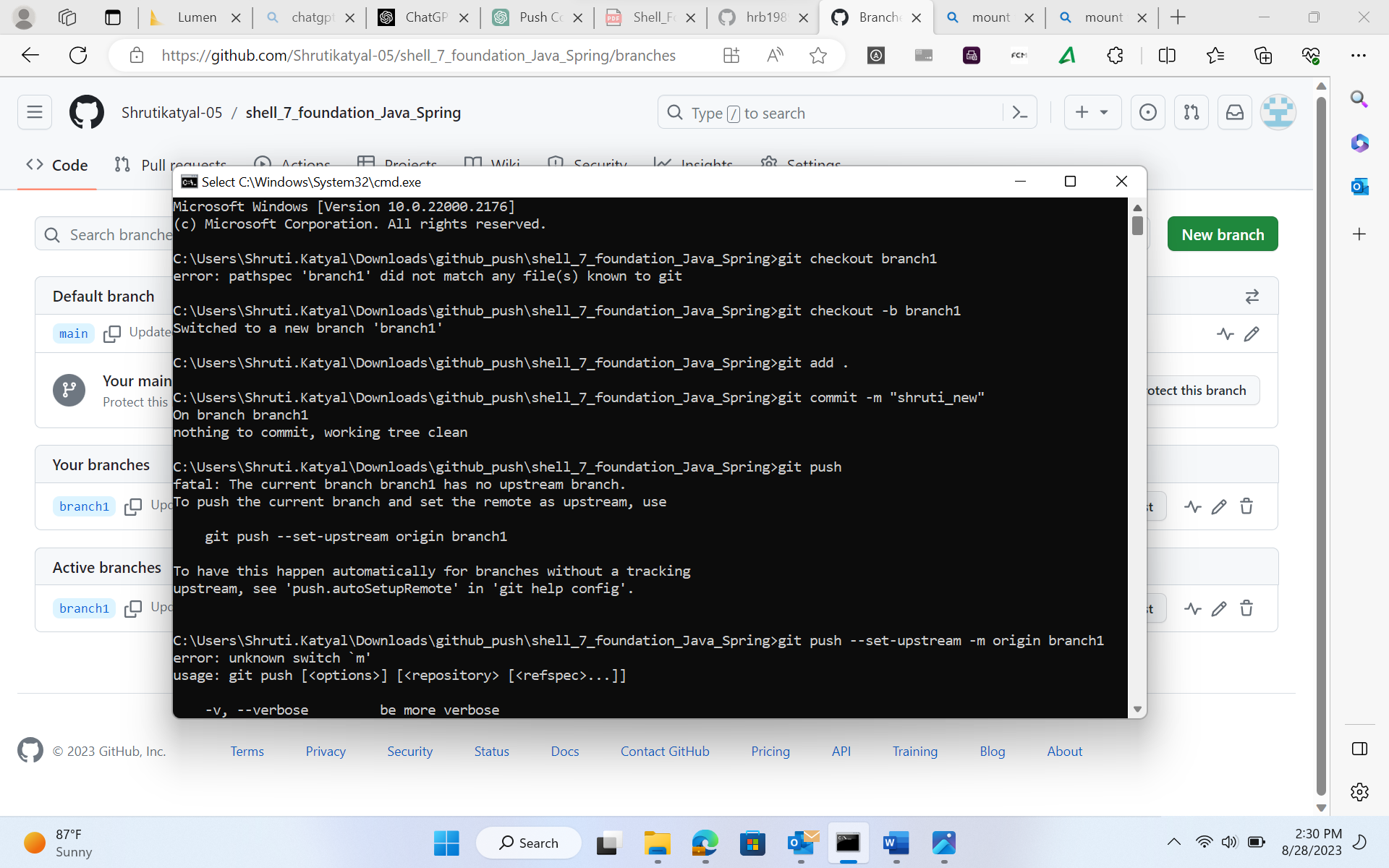
Step1: First fork the repository

Step 2: Create a new branch assign it any name hence feature branch gets created

**A screenshot of a computer

Description automatically generated**

**1.**



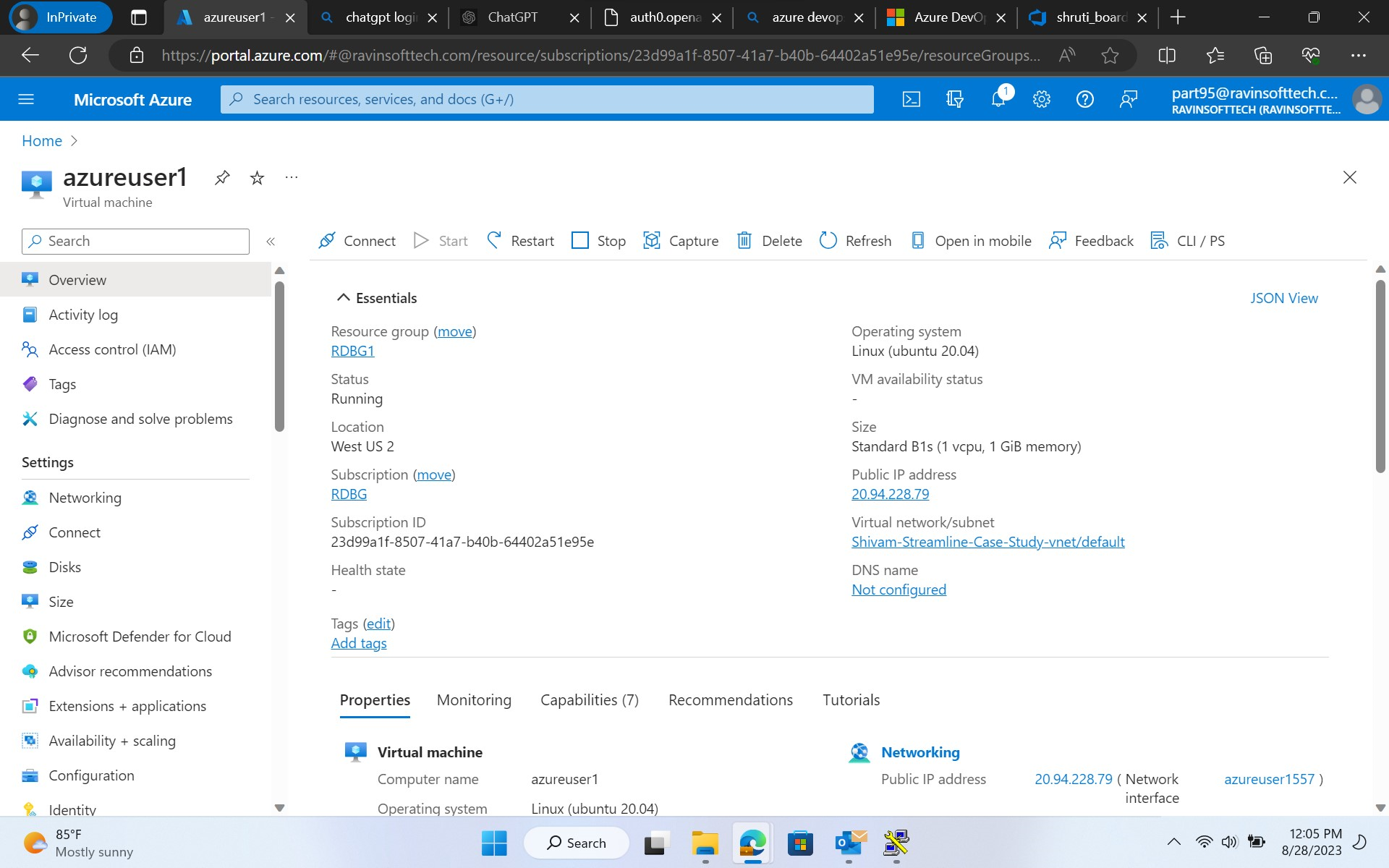
**2.**

**Azure Creating a virtual machine (VM) in Microsoft Azure involves a few steps. Here's a general outline of the process:**

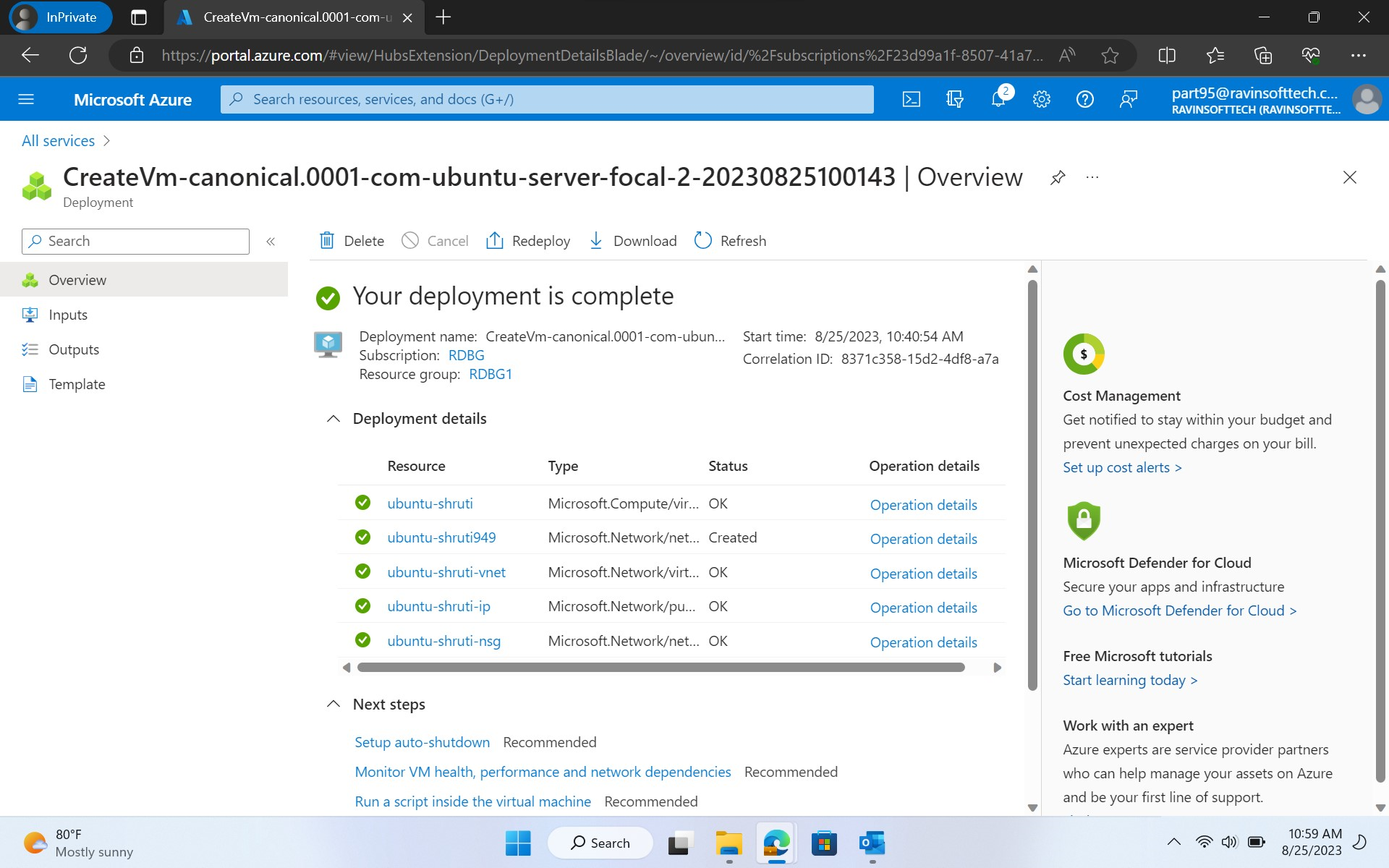
1. Sign in to Azure Portal:
   * Open your web browser and navigate to the [Azure Portal](https://portal.azure.com/).
   * Sign in using your Azure account credentials.
2. Navigate to Virtual Machines:
   * In the Azure Portal, click on "Create a resource" or the "+" button on the left-hand menu.
   * Search for "Virtual Machine" and select "Virtual Machine" from the results.
3. Basic Configuration:
   * In the "Basics" tab of the VM creation wizard, provide the following information:
     + Subscription: Choose your subscription.
     + Resource group: Create a new one or use an existing group to organize your resources.
     + Virtual machine name: Choose a unique name.
     + Region: Choose the data center location.
     + Availability options: Select your preference.
     + Image: Choose an operating system image (Windows, Linux, etc.).
     + Size: Choose the VM size based on your requirements.
4. Administrator Account:
   * Provide an administrator username and password for the VM.
   * You can also configure SSH authentication settings if you're creating a Linux VM.
5. Inbound Port Rules:
   * Configure inbound port rules to allow traffic to the VM.
   * You can select predefined rules for common services or create custom rules.
6. Disks:
   * Configure the OS disk and additional data disks if needed.
   * Choose the disk type (standard or premium) and size.
7. Networking:
   * Configure the networking settings for the VM.
   * Choose a virtual network, subnet, and public IP configuration.
8. Management:
   * Configure management options, including auto-shutdown settings.
   * You can also enable boot diagnostics and guest OS diagnostics.
9. Advanced:
   * Configure additional settings like extensions, tags, etc.
10. Review and Create:
    * Review your configuration settings on the "Review + create" tab.
    * If everything looks correct, click "Create" to start the VM deployment process.

**VM configuration:**

Creation of Virtual Machine



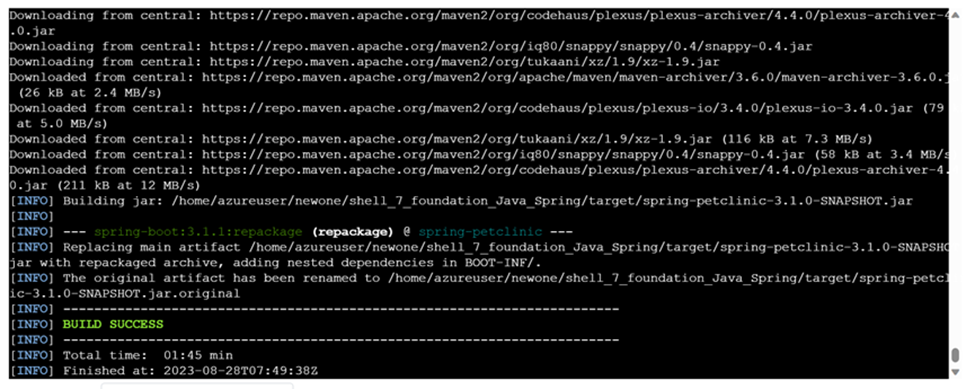
**1.**



**2.**

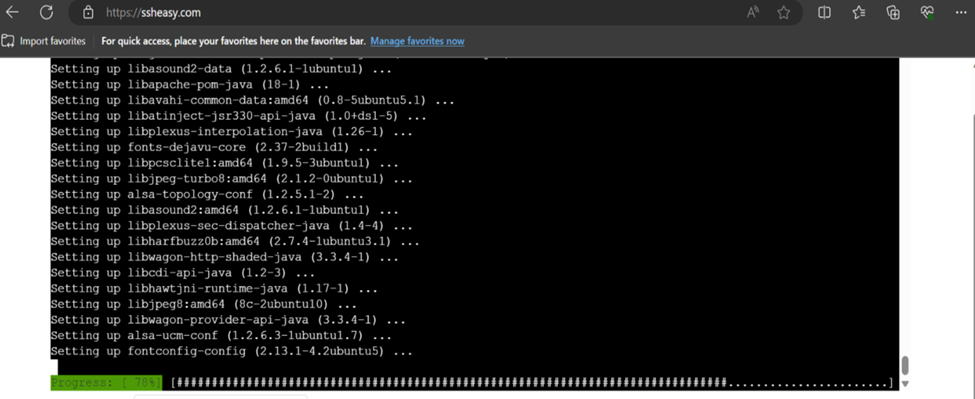
**Maven based build process:**

The Maven build process is a standardized and widely-used approach for managing and building Java projects. Maven helps you automate the building, testing, and packaging of your project, along with handling dependencies and generating documentation.



**Prometheus monitoring integration:**

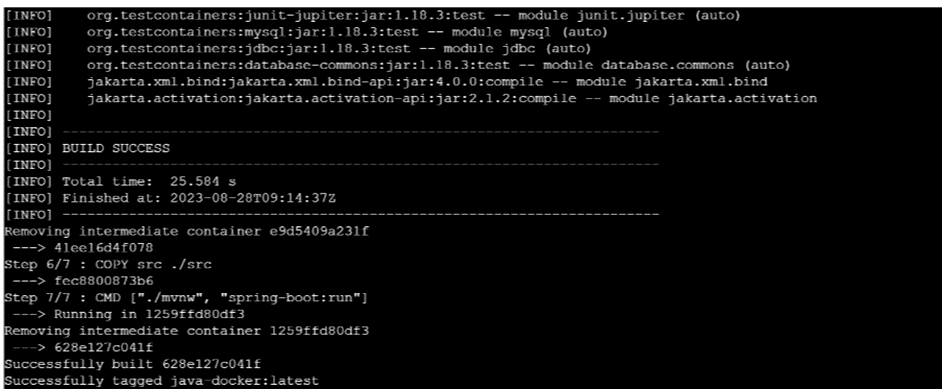
1. Prometheus monitoring integration refers to the process of setting up and configuring Prometheus to collect, store, and analyze metrics from various sources within an environment. **Installation**: Install Prometheus on an Azure VM.
2. **Configuration**: Create a **prometheus.yml** file to define scrape targets and settings.
3. **Scrape Targets**:
   * **Local Environment**: Use Node Exporter to gather VM hardware and OS metrics.
   * **Spring Boot App**: Instrument the app with Prometheus client libraries to expose custom metrics via an HTTP endpoint.
4. **PromQL Queries**: Use PromQL to query and analyze collected metrics.
5. **Alerting Rules**: Define alerting rules with PromQL for triggering alerts based on specific conditions.
6. **Optional Visualization**: Consider using tools like Grafana for advanced data visualization.
7. **Optional Service Discovery**: Enable dynamic monitoring by setting up service discovery for scaling applications.

****

Top of Form

**Docker Containerization:**

Docker containerization is a technology that allows you to package, distribute, and run applications and their dependencies in isolated environments called containers.



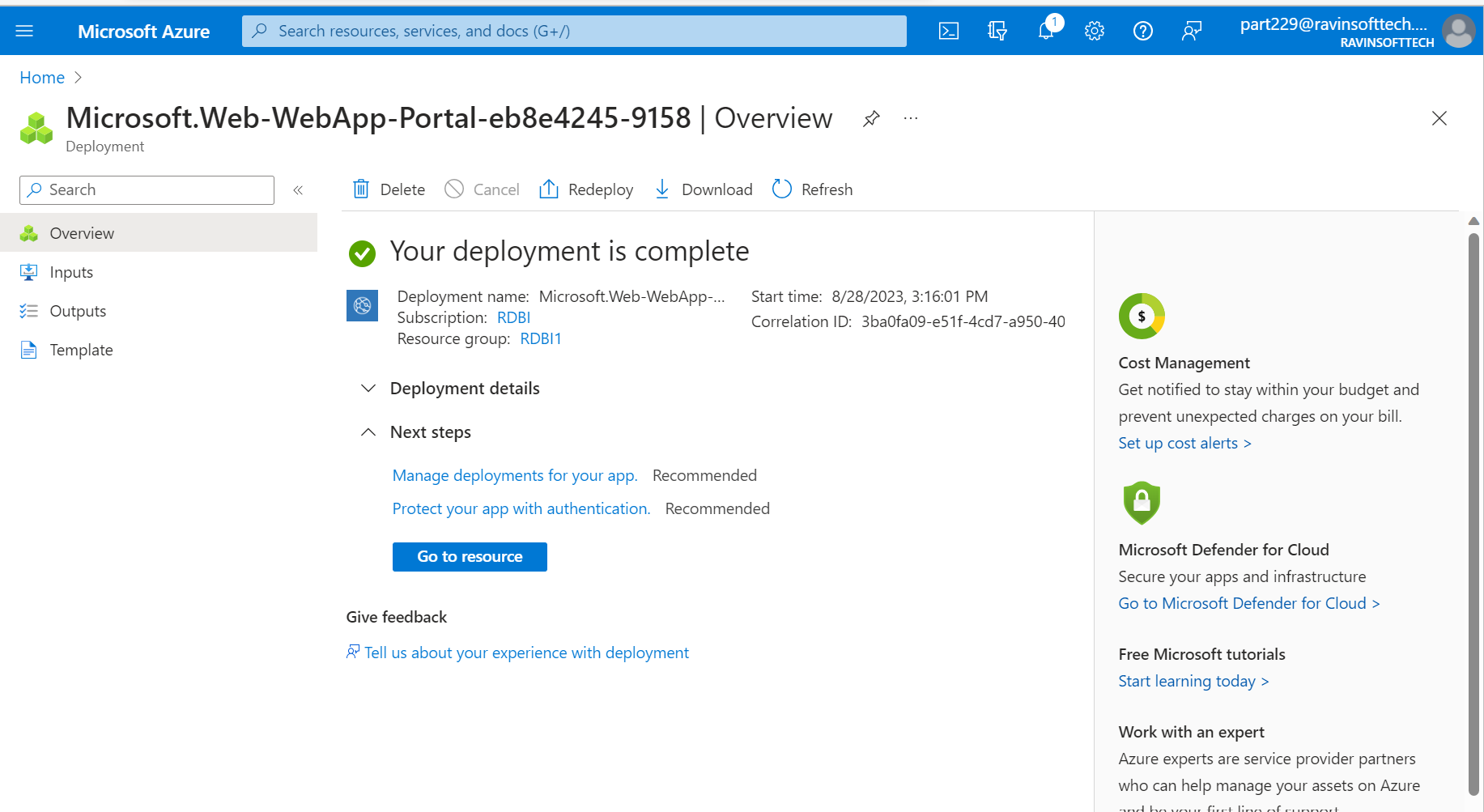
**Azure Web App Configuration:**

steps to configure an Azure Web App:

1. Create: Set up the web app with a unique name, OS, and runtime.
2. Deployment: Choose source control or Docker for deployment.
3. Domains: Configure custom domains and SSL for security.
4. Scaling: Adjust scaling based on traffic, enable autoscaling.
5. Settings: Configure app-specific environment variables.
6. Connections: Set up database and service connections.
7. Auth: Implement authentication and authorization.
8. Monitoring: Configure performance tracking and errors.
9. Backup: Set backup policies to safeguard data.
10. Slots: Use slots for testing and seamless updates.
11. Apply: Review and save your changes.

**A screenshot of a computer

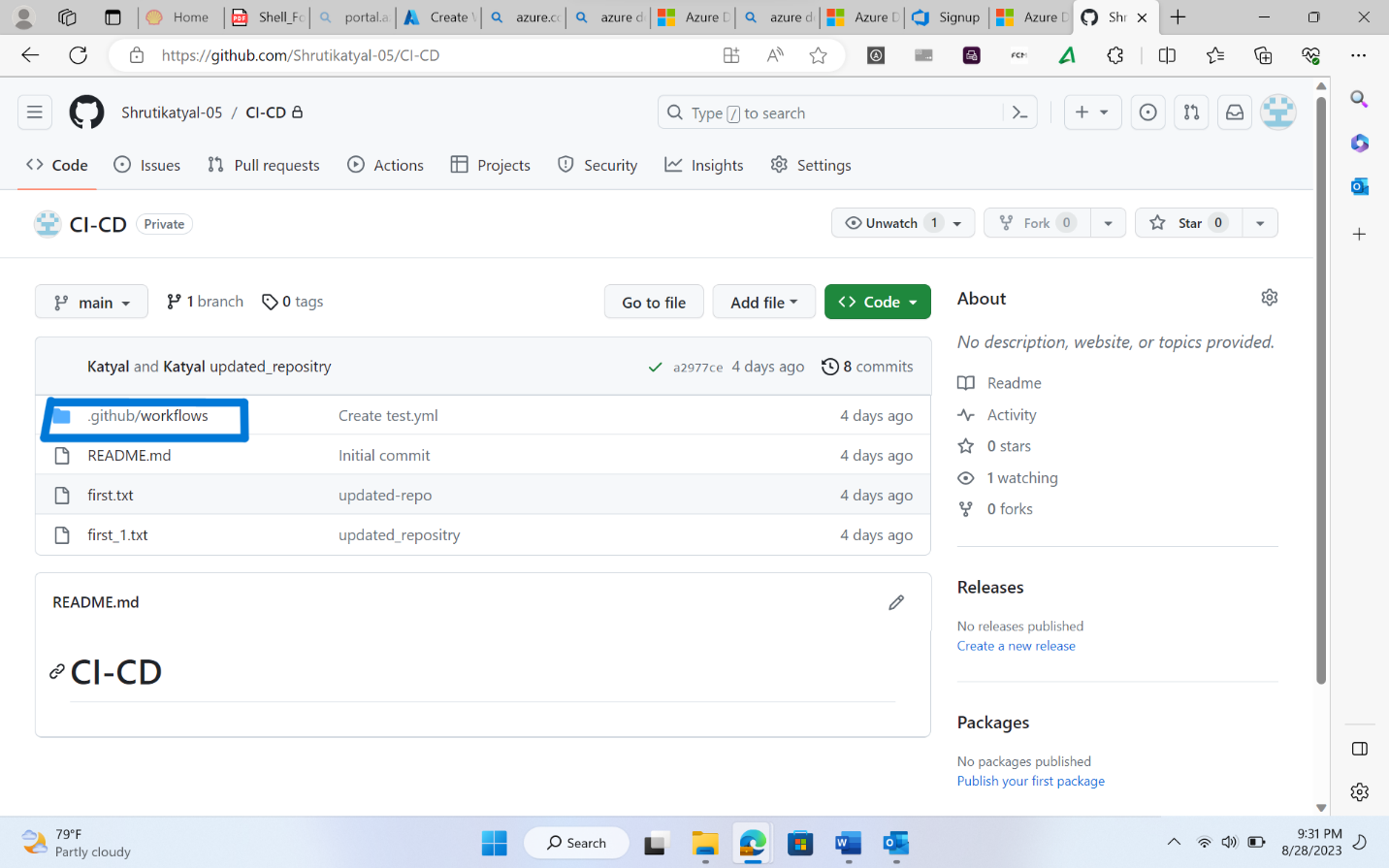
Description automatically generated**

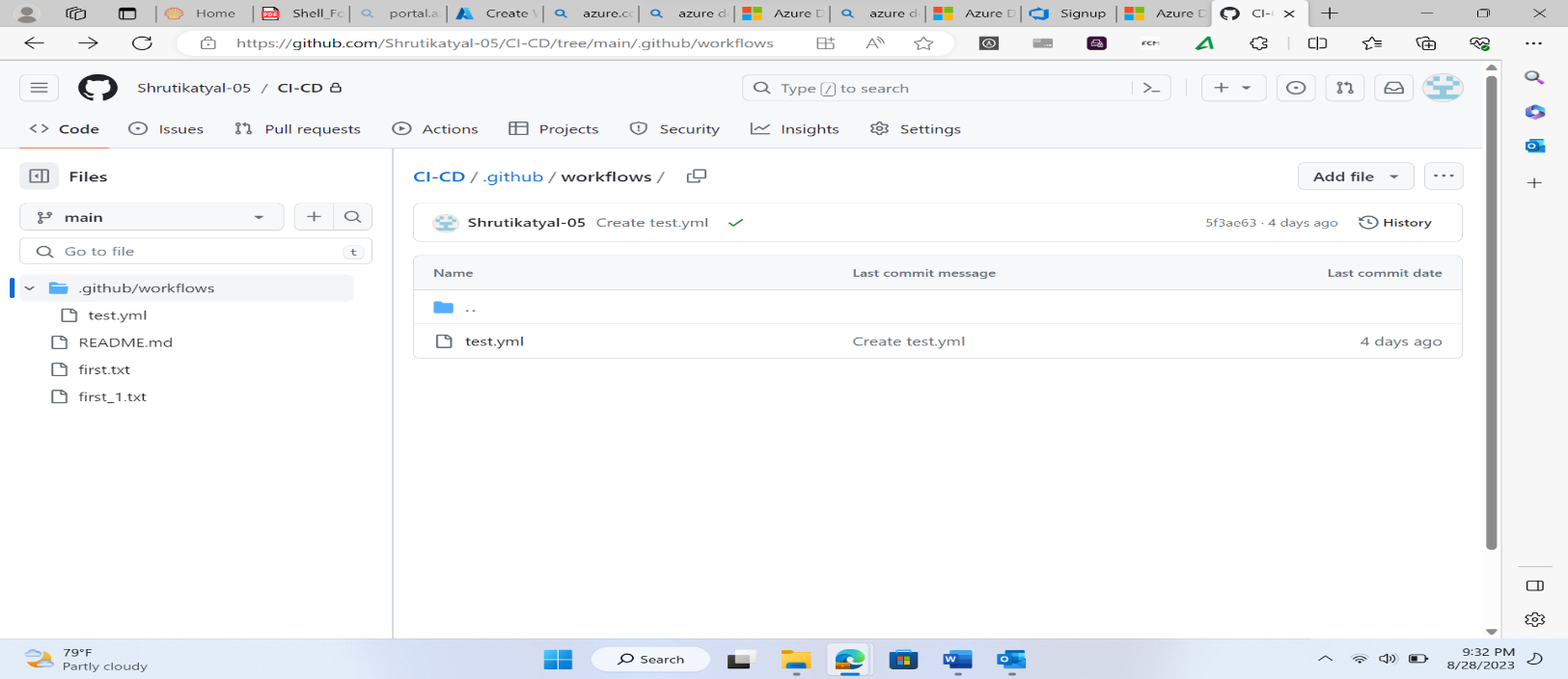


**GitHub Actions Workflow Implementation:**

Implementing a GitHub Actions workflow:

1. Create File: Make .github/workflows dir, add workflow.yml.
2. Define Steps: Write YAML for your workflow:
   * Specify trigger event (e.g., push).
   * Define jobs with steps (e.g., checkout, build, test).
3. Features:
   * Use existing or custom actions.
   * Manage secrets and variables.
   * Set up testing on different environments.
4. Push and Test:
   * Commit and push YAML.
   * GitHub Actions auto-triggers workflow.
5. Results:
   * Check workflow status in "Actions" tab.
   * Inspect logs and artifacts for details.
6. Advanced:
   * Create workflows for PRs, releases.
   * Deploy apps, integrate services.
   * Customize based on project needs.

****

****