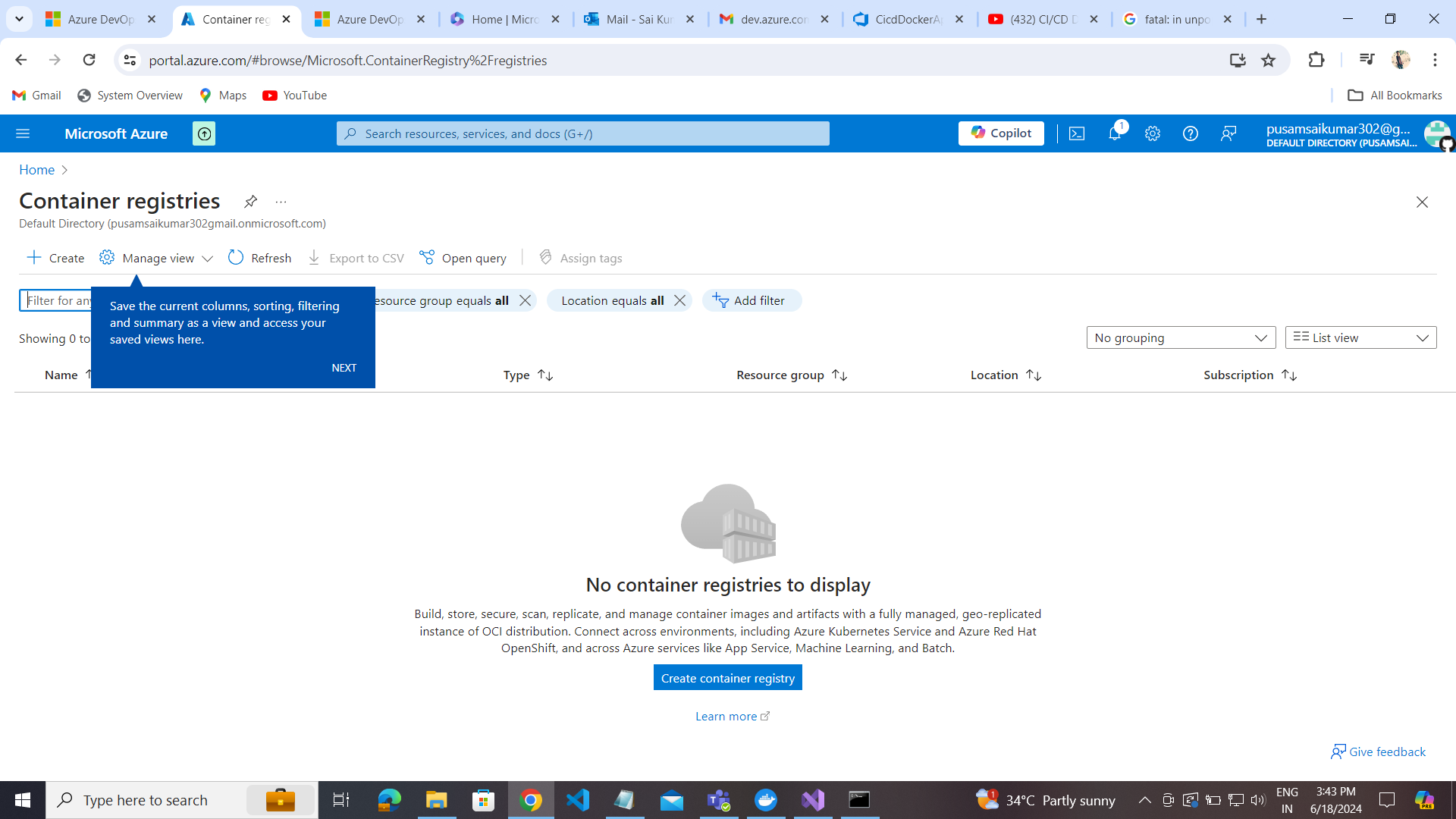
AZURE CICD PIPE LINES WITH AZURE DEVOPS

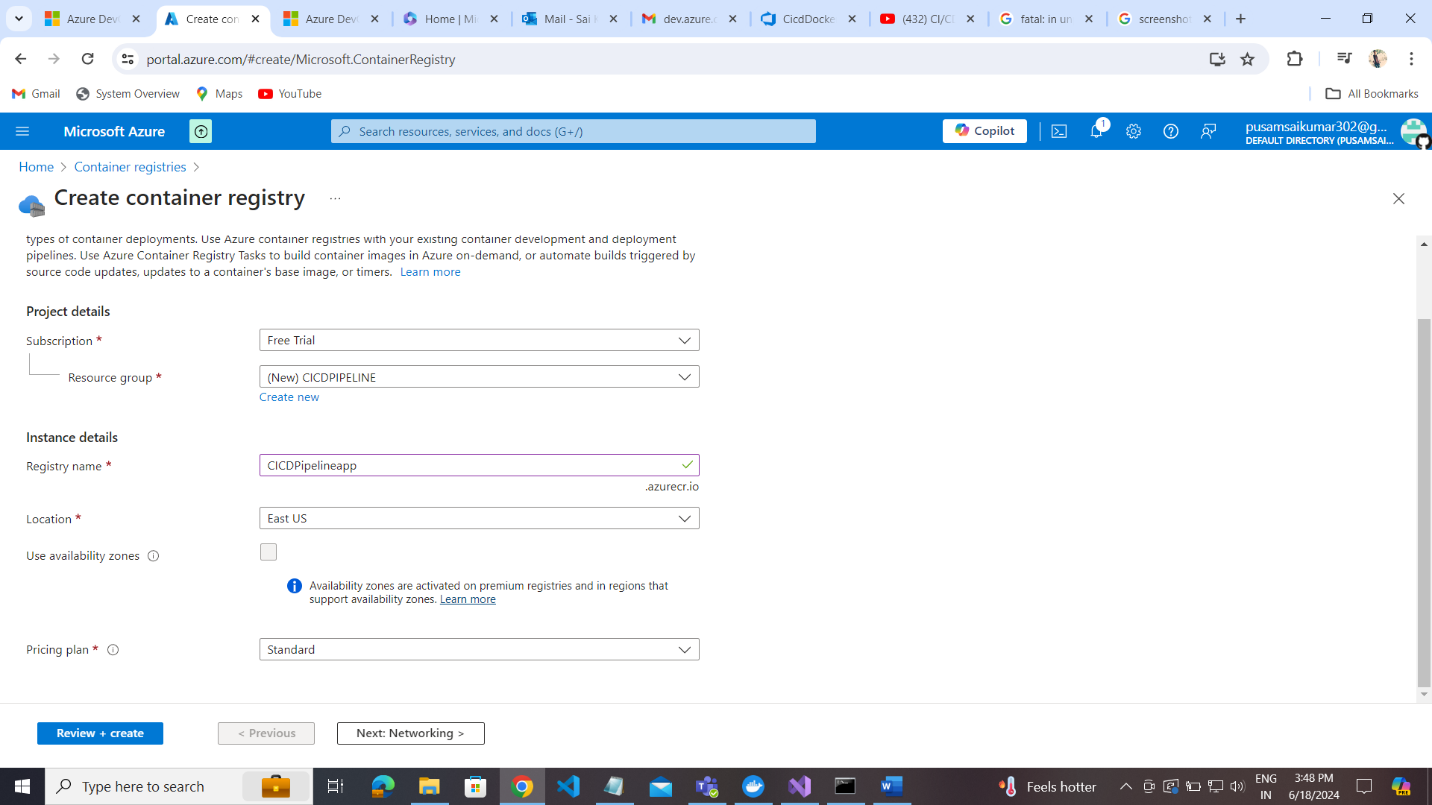
Azure provides comprehensive support for Dockerized applications, including Azure App Service and Azure Kubernetes Service (AKS).

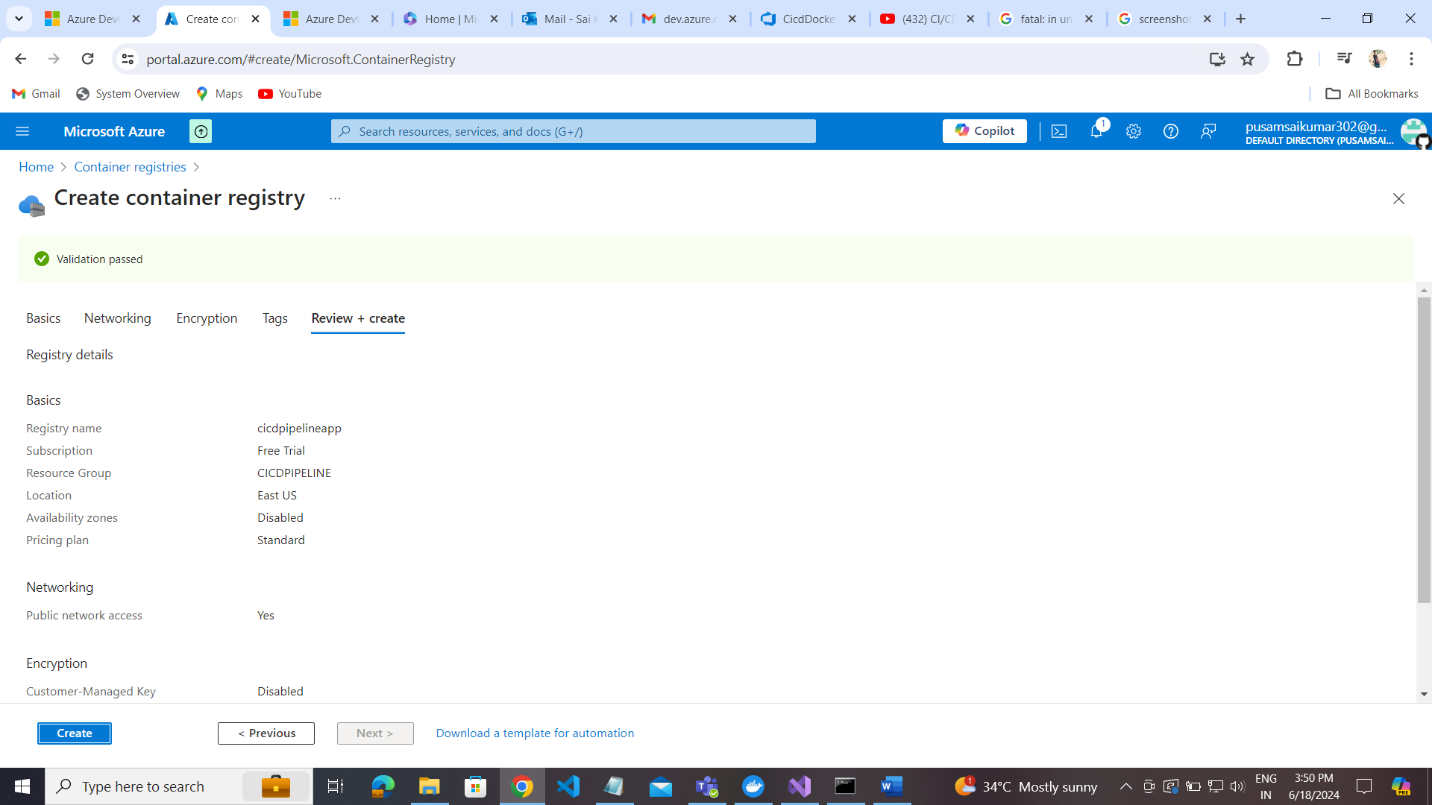
| **Property** | **Description** | **Example** |
| --- | --- | --- |
| **Code Deployment** | Use Azure App Service for Containers or Azure Kubernetes Service for more control. | az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name myApp --deployment-container-image-name mydockerimage |
| **Framework Update** | Update Docker image and push to Azure Container Registry (ACR). | docker build -t mydockerimage . <br> docker push myacr.azurecr.io/mydockerimage |
| **Database Timeout** | Configure connection strings and retry policies in Azure SQL Database. | az sql server firewall-rule create --resource-group myResourceGroup --server myServer --name AllowYourIp --start-ip-address <your\_ip> --end-ip-address <your\_ip> |
| **Disaster Recovery** | Use Azure Site Recovery and configure geo-redundant backups. | az backup vault create --resource-group myResourceGroup --name myBackupVault |
| **Bug Tracking** | Integrate with Azure DevOps for continuous integration and monitoring. | az pipelines create --name myPipeline --repository-url https://github.com/your-repo.git --branch main --yaml-path azure-pipelines.yml |

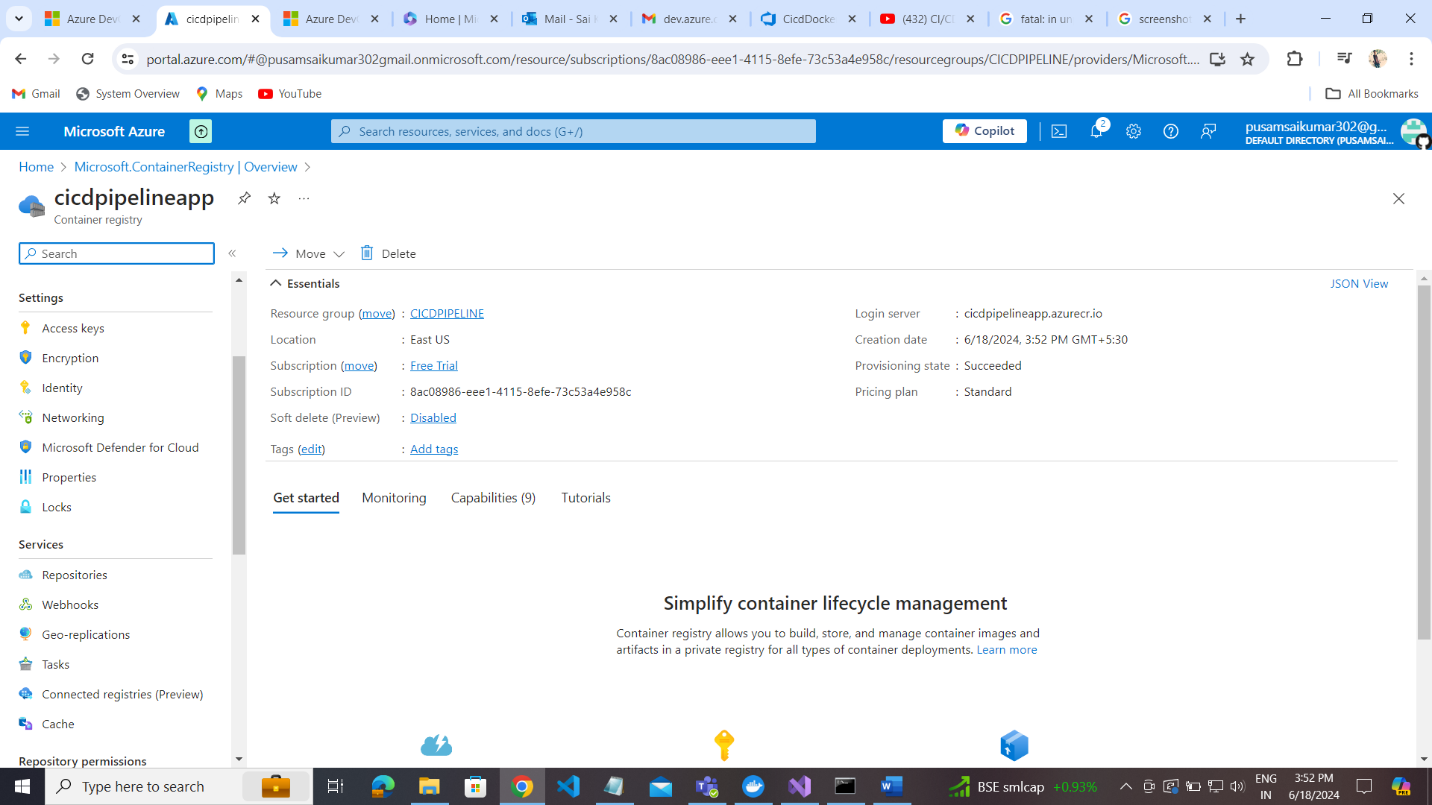
AZURE DEVOPS WITH DOCKER CONTAINER

1. CREATE DOCKER CONTAINER IN AZURE:

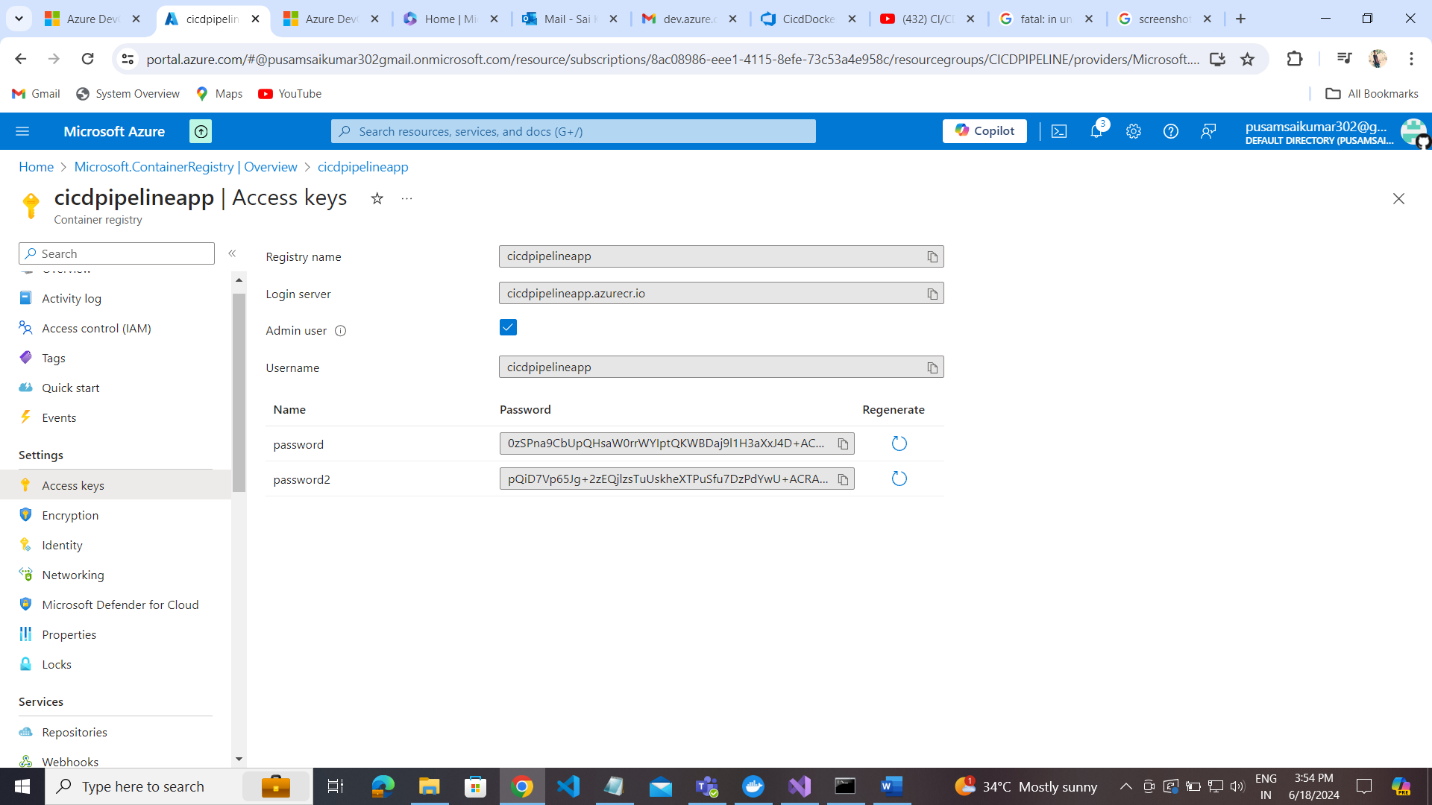




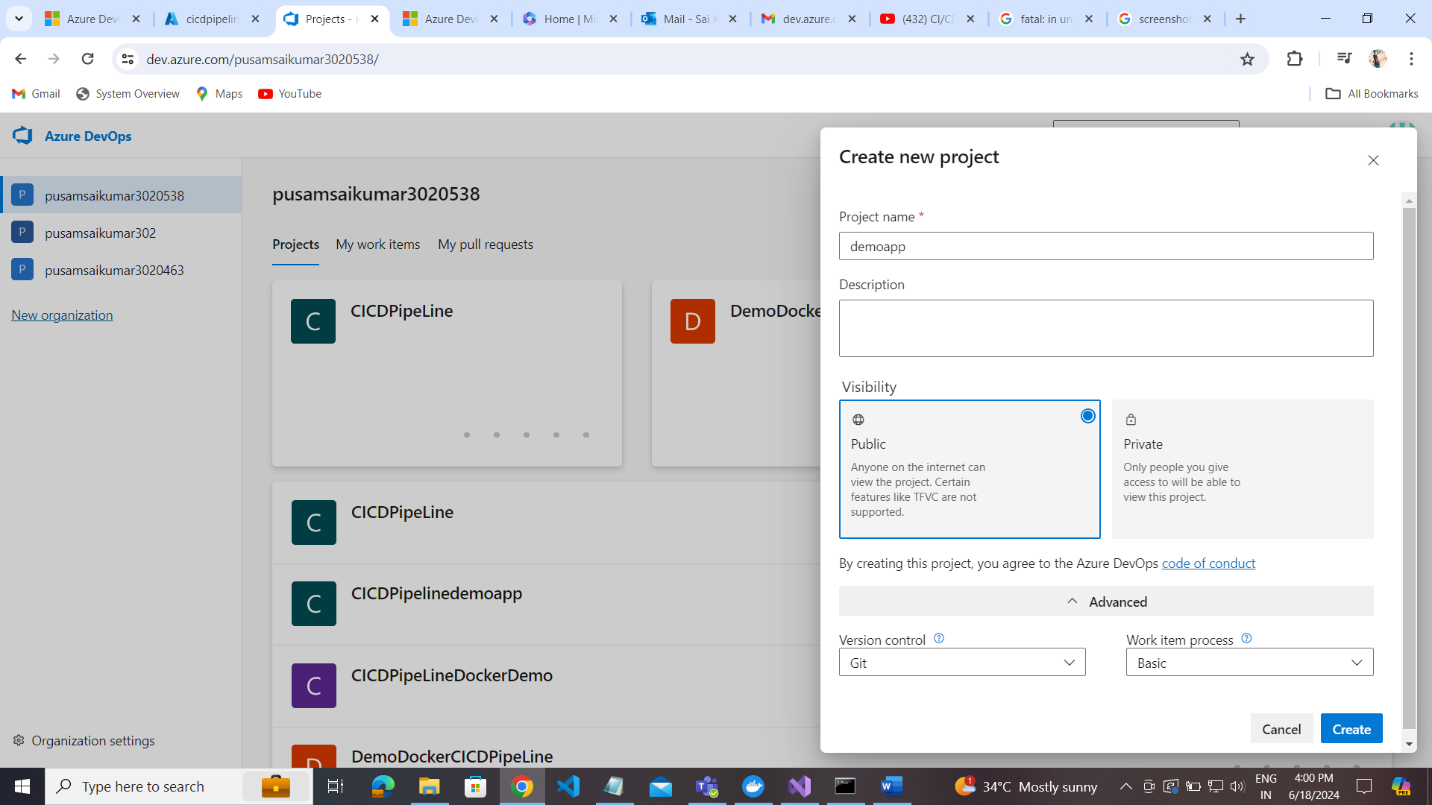


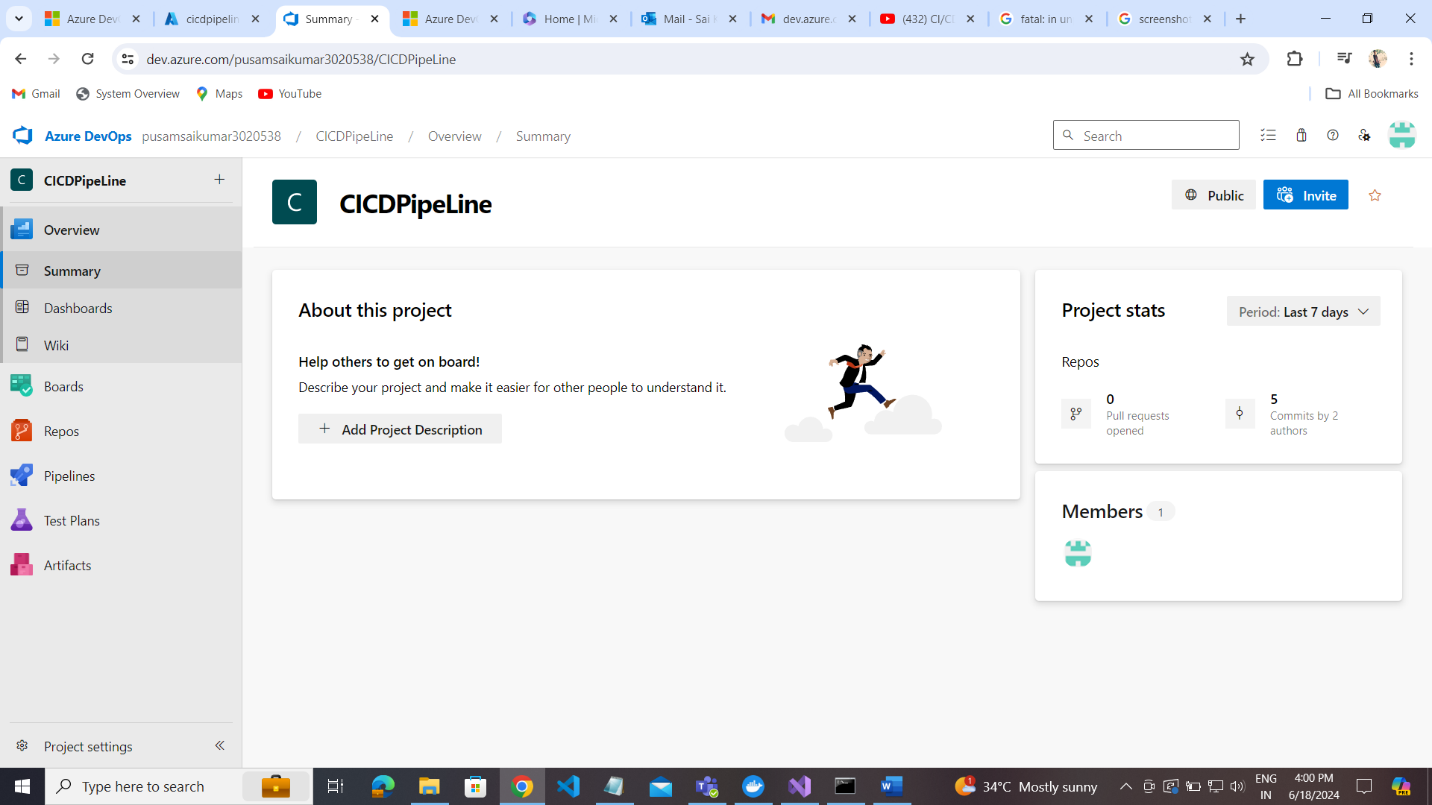
Go to Repositories  


Go Access Keys to Enable Admin User

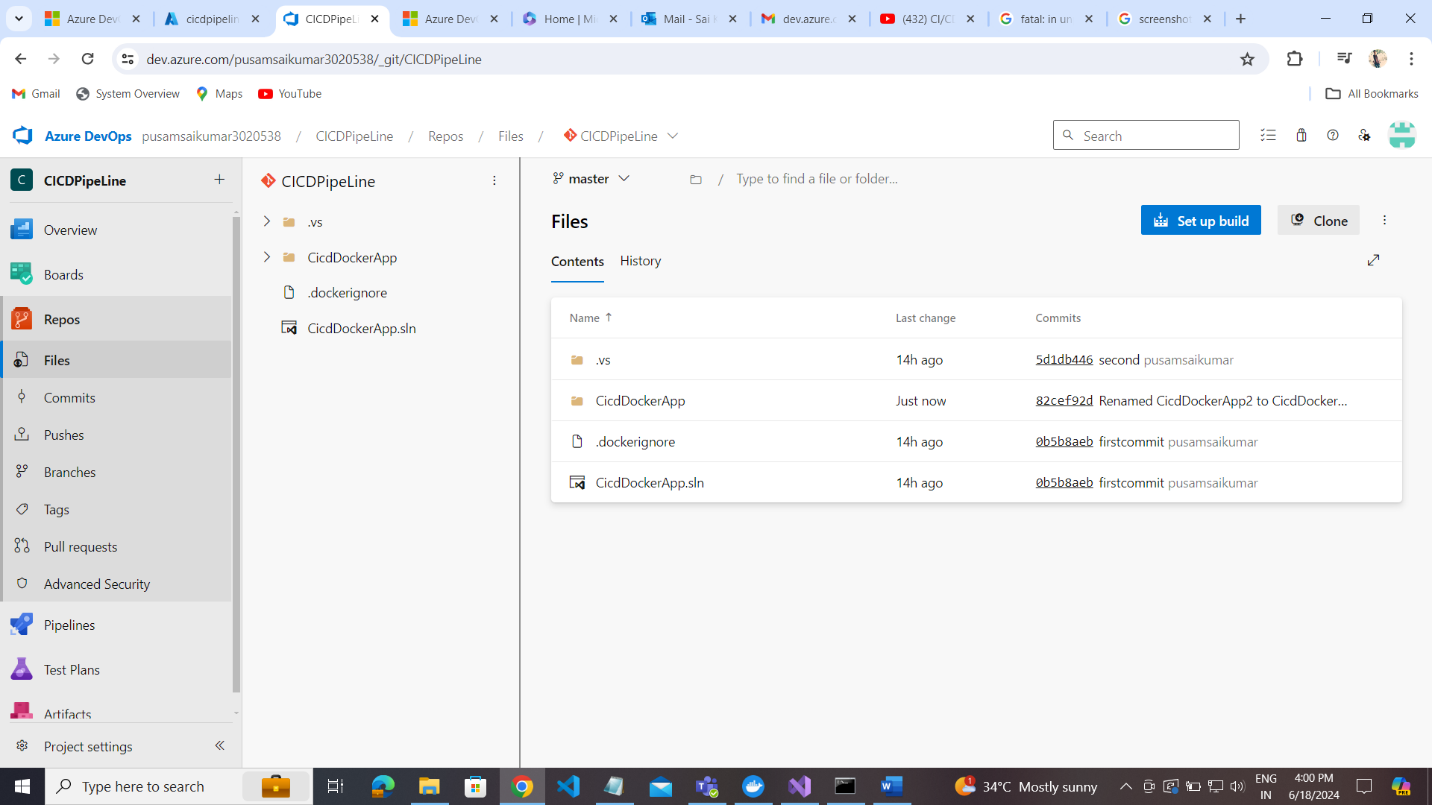


Create the pipeline in azure devops:  
create project:

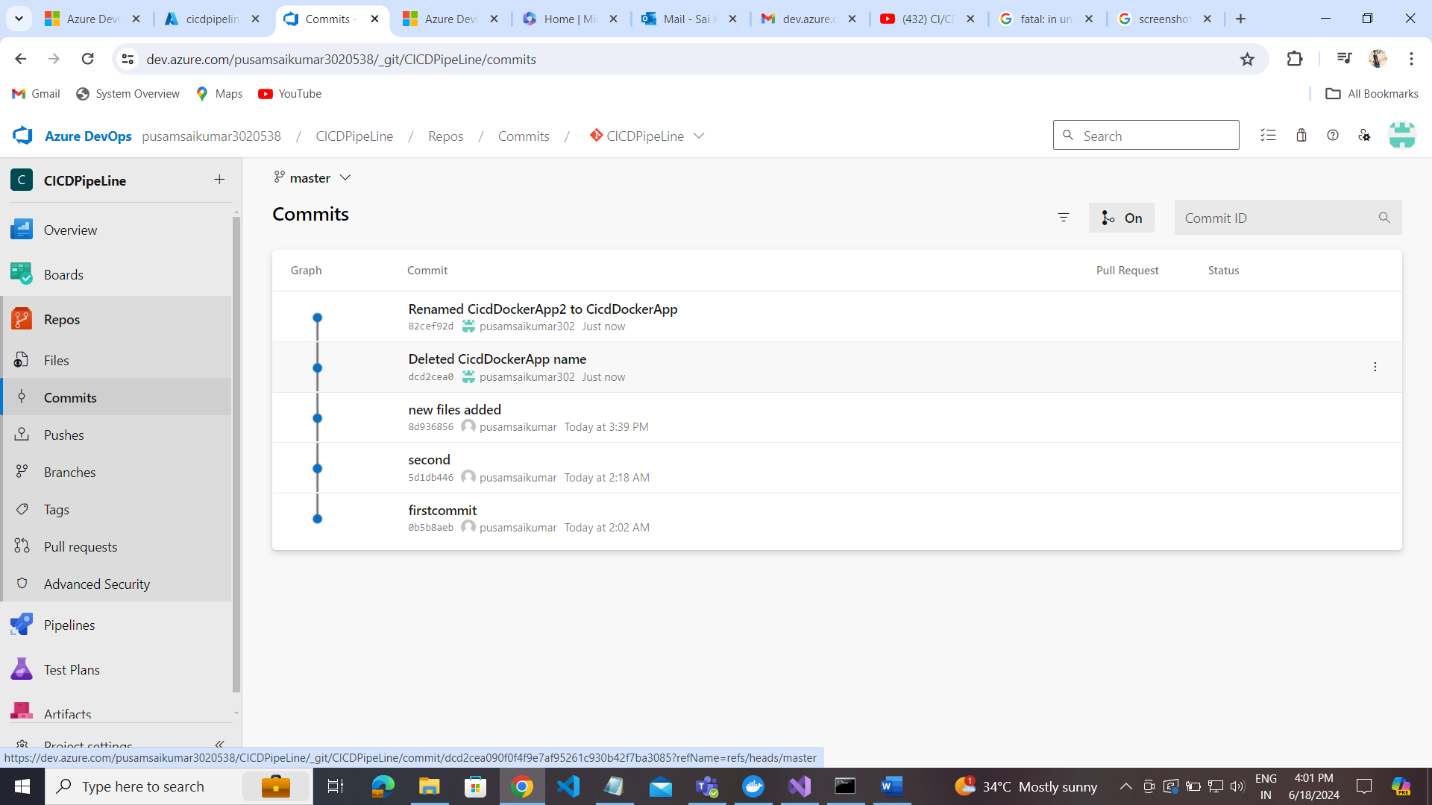




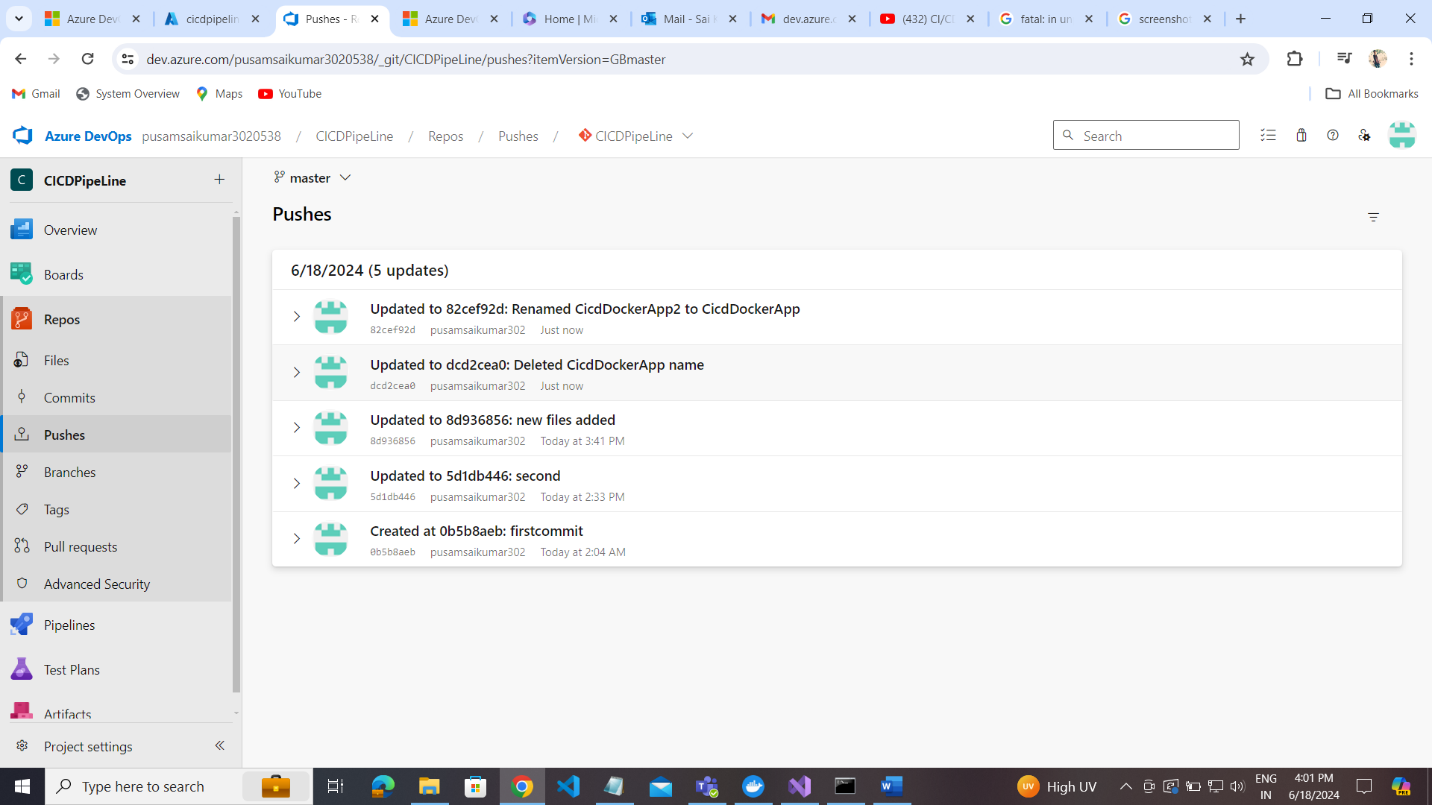
Files added into master branch



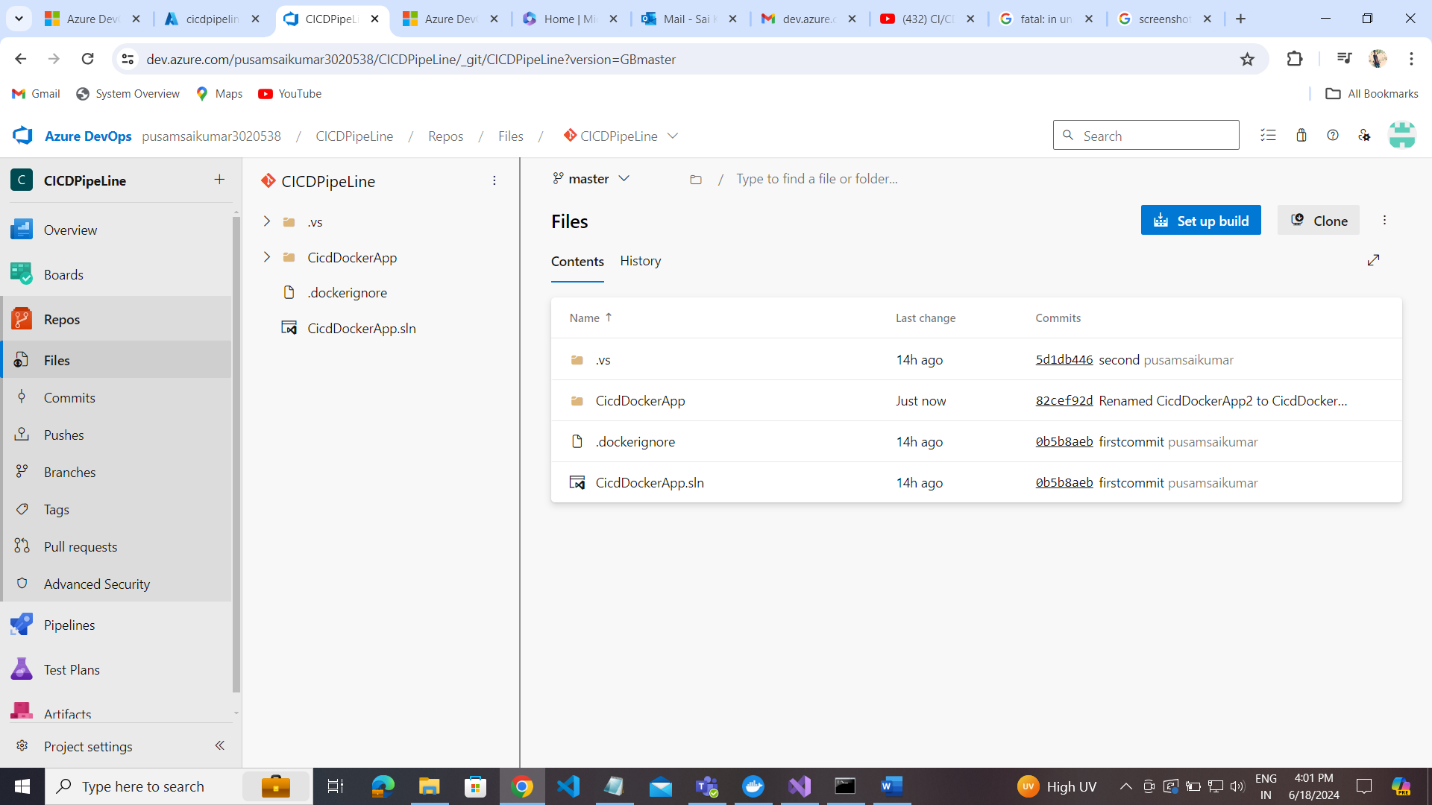
Commits :



Push codes:

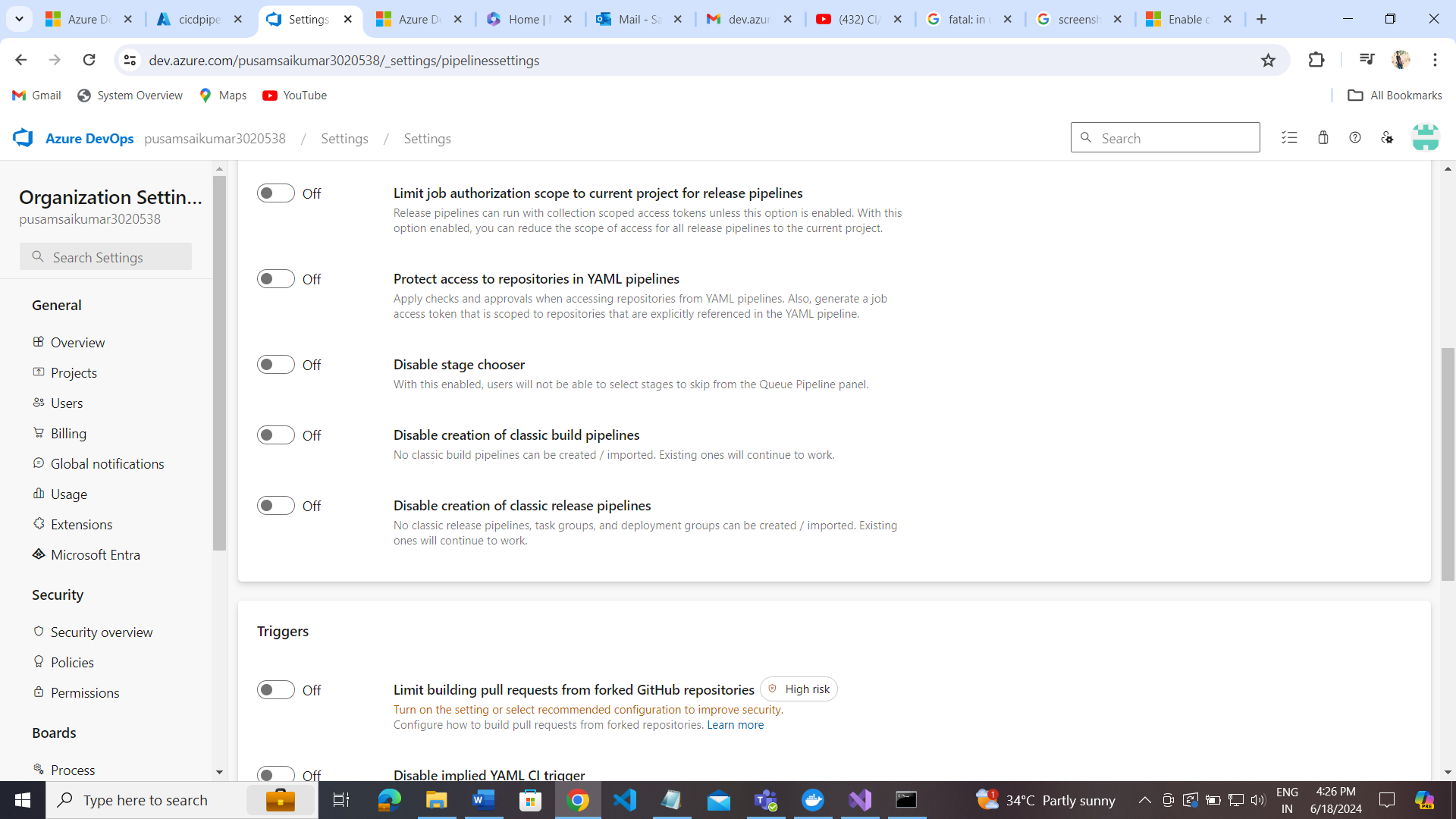


Master branch :



Create pipe line then go to pipeline sections:

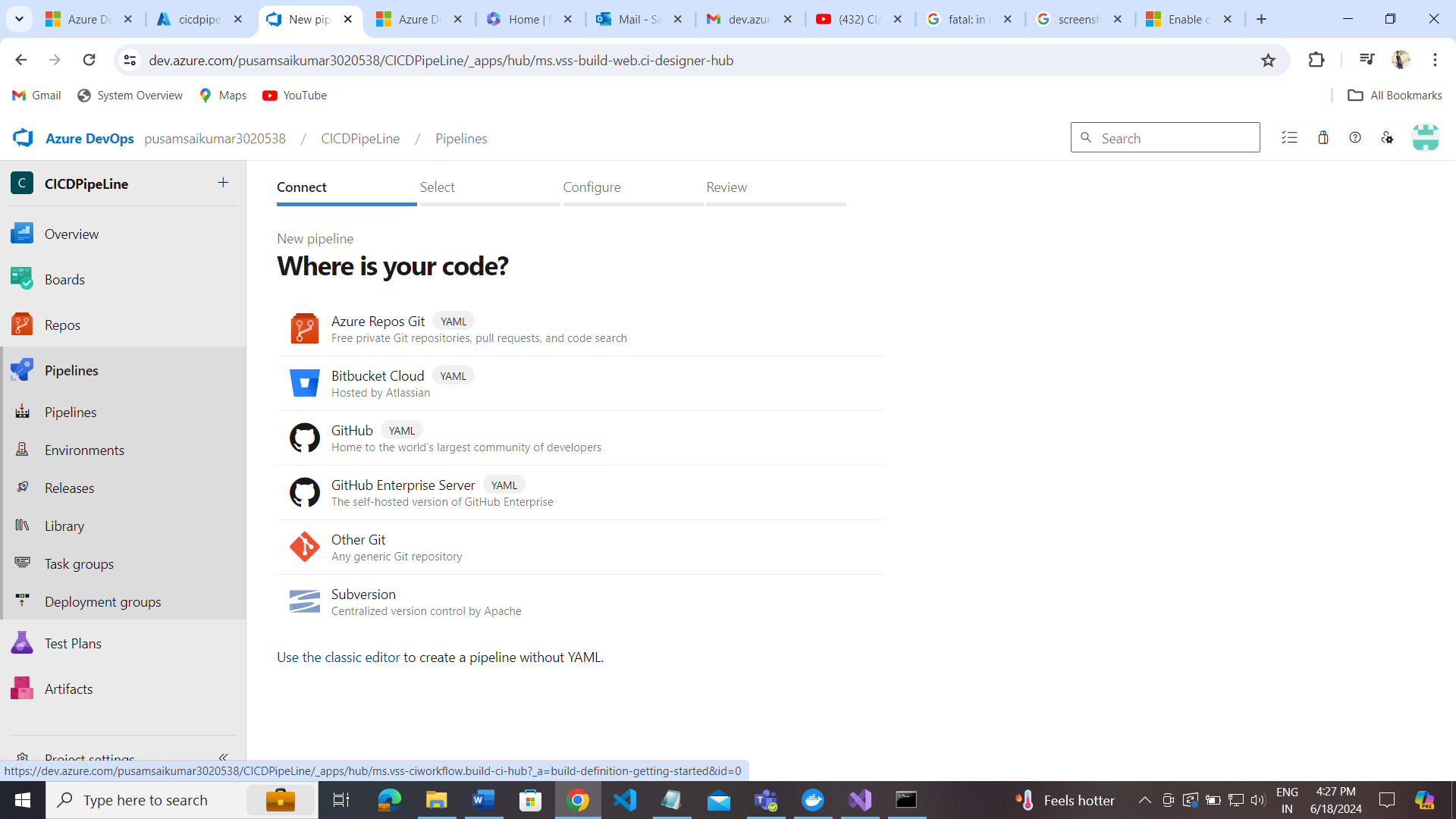
To enable the classic editor for build and release pipelines in Azure DevOps, you can try these steps:



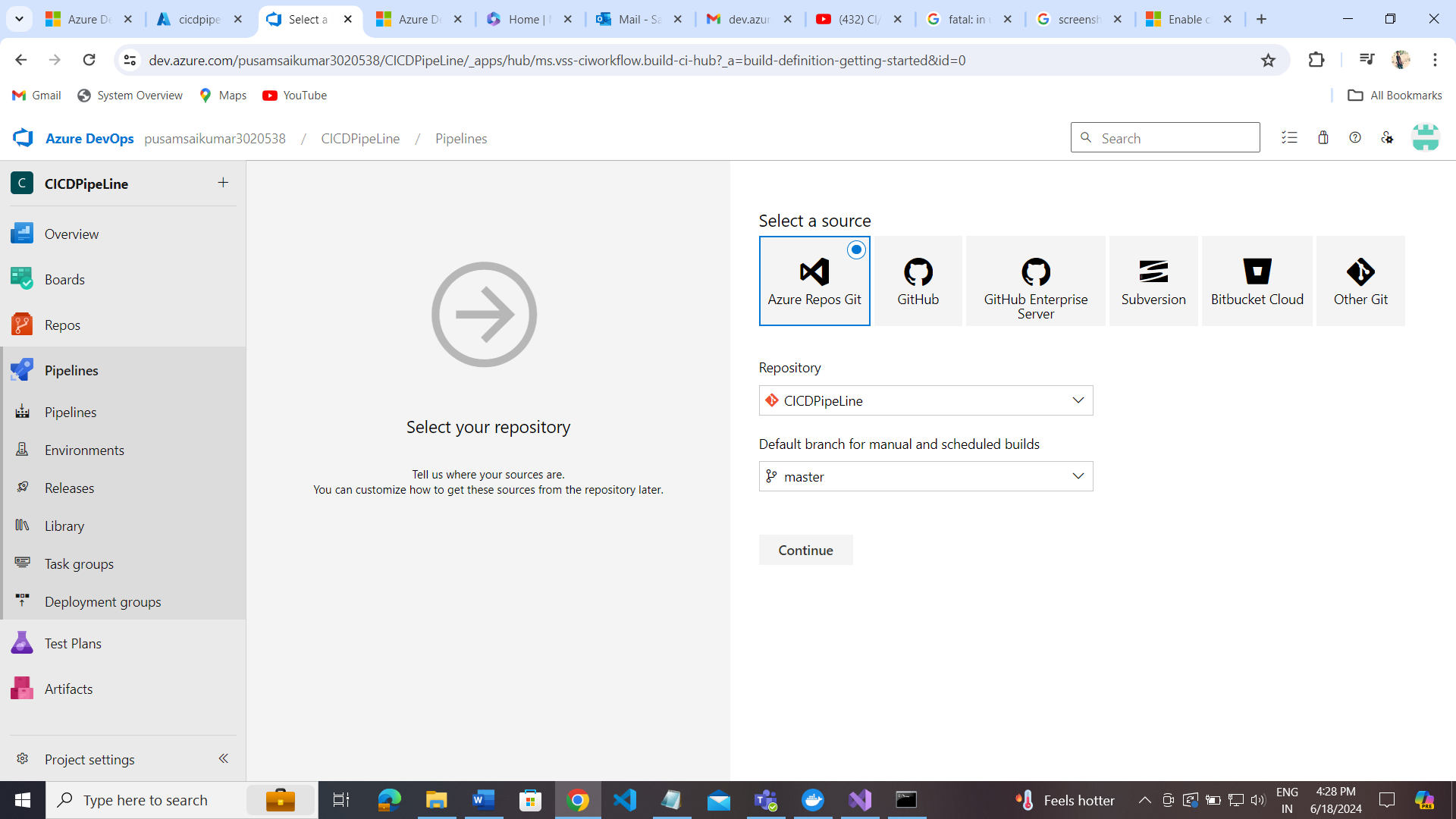
1. Go to Organization Settings of Azure Devops
2. Select Pipelines
3. Select Settings
4. Turn off the settings

5.You should now see the Classic Editor option

Now go to pipeline sections:

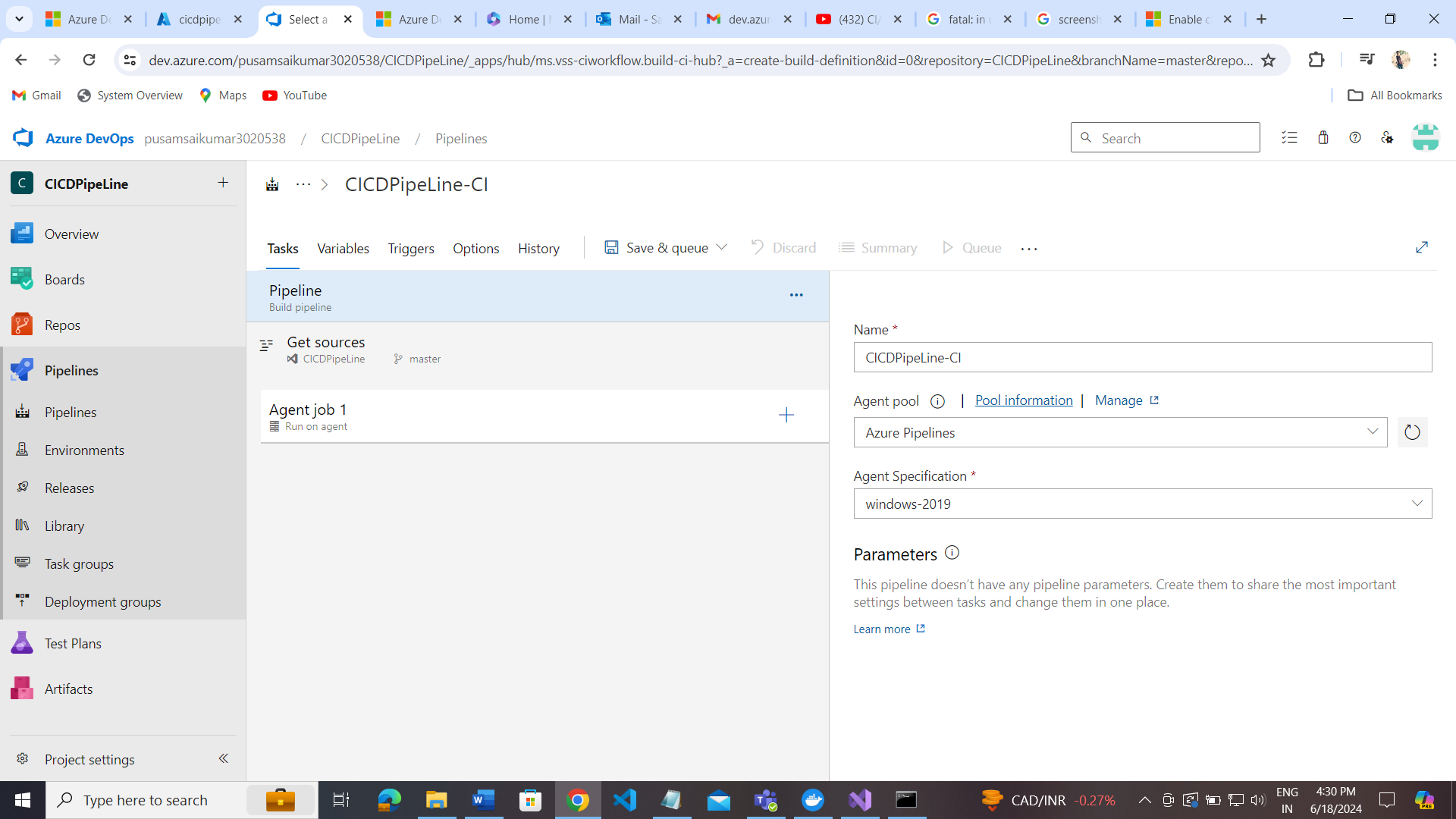


Select classic editor and

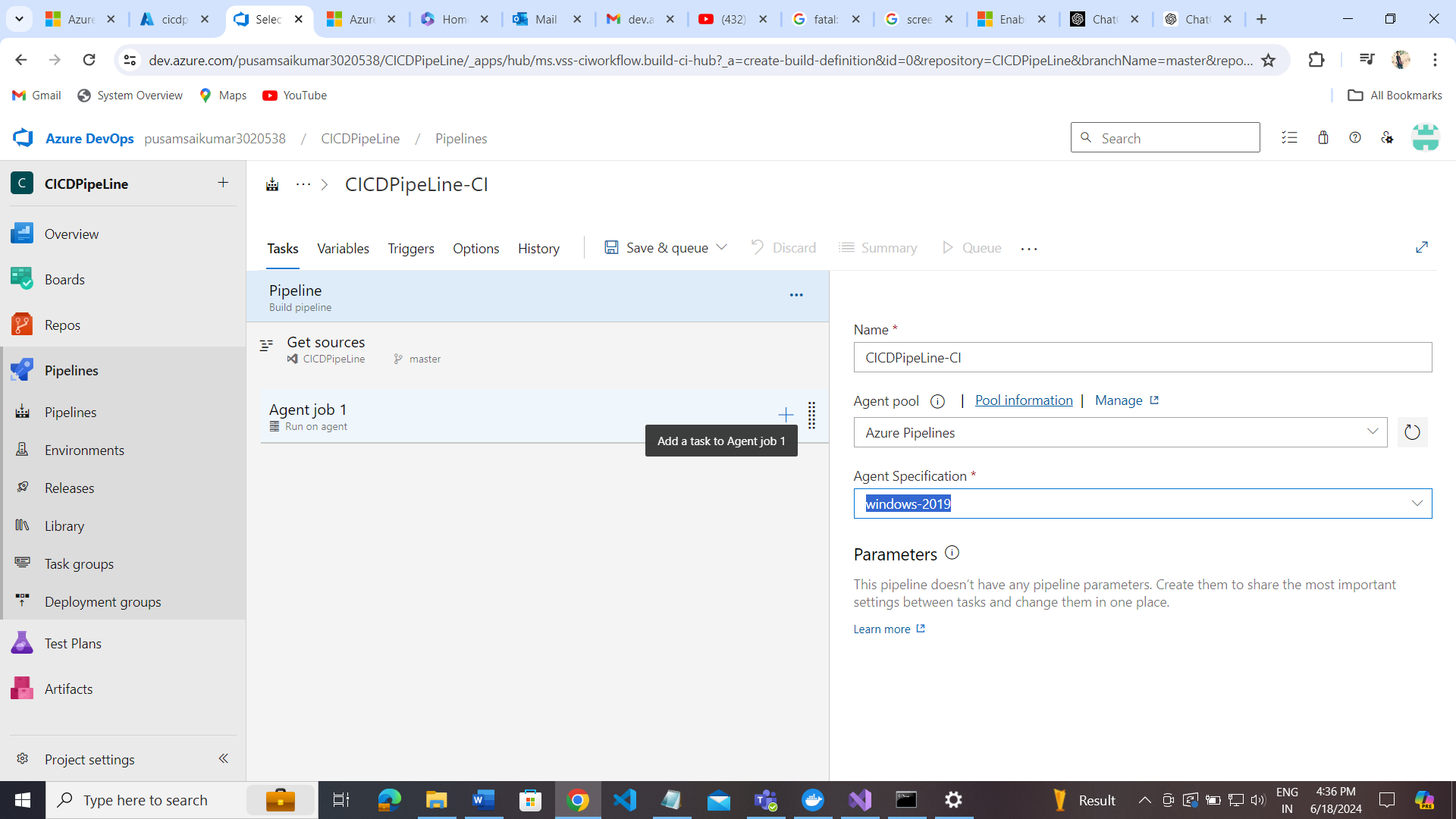


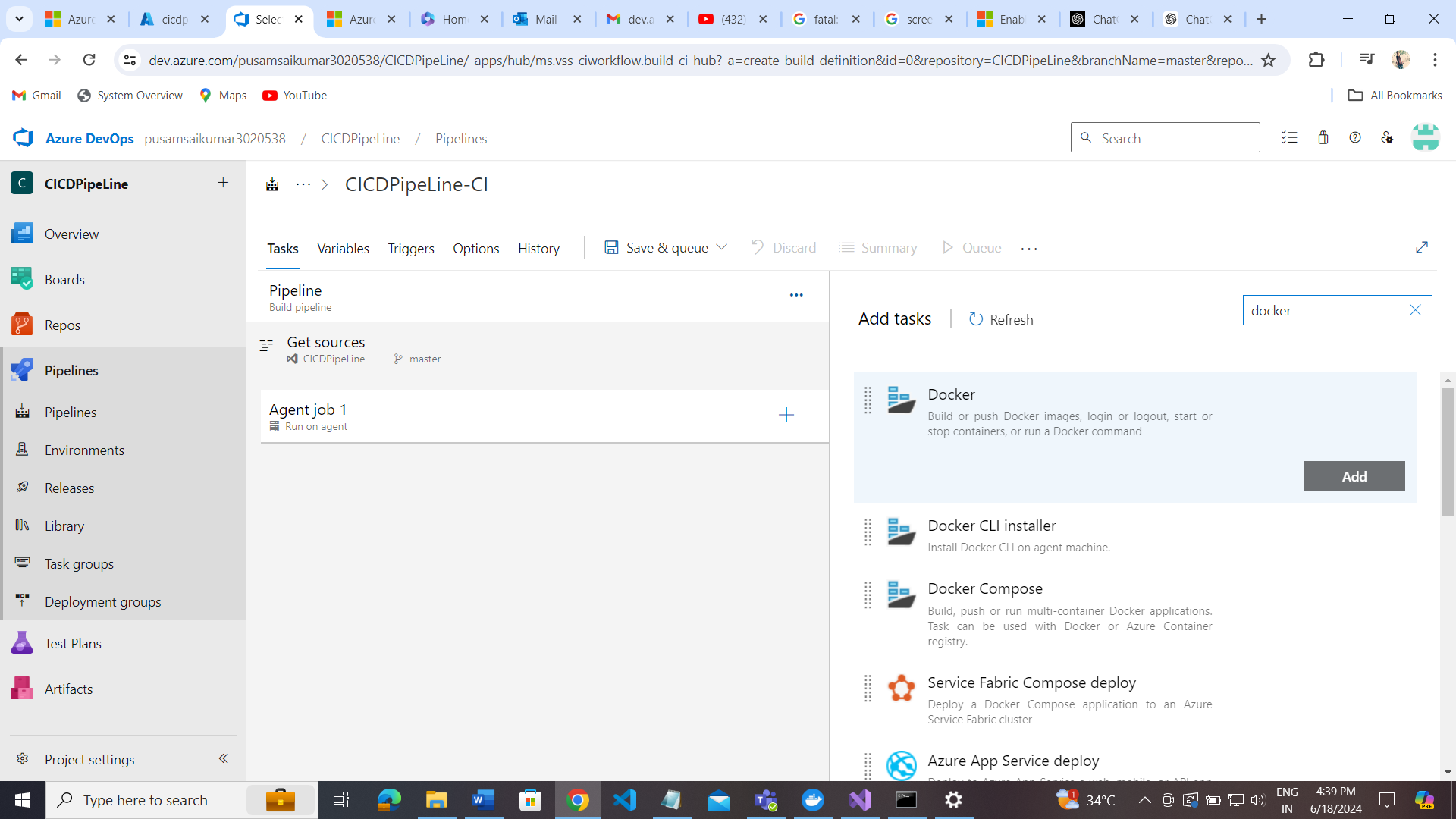
Click on continue>

Select empty job:



Add the specific version of ASP.NET CORE -🡪 click on the + icon in Agent job



Add docker -> to build docker image and container.  


Add app service connection to byclicking add

Benefits of Azure Pipelines for Dockerized ASP.NET Core 6 Deployment

Seamless Integration with Azure and Other Services:

Azure Services: Native integration with Azure Kubernetes Service (AKS), Azure Container Registry (ACR), and other Azure services.

Third-party Services: Support for GitHub, GitLab, Bitbucket, and more.

Scalability and Reliability:

Elastic Build and Deployment: Scales up or down based on your workload.

High Availability: Managed by Azure ensuring high uptime.

Flexibility and Customization:

Pipeline as Code: Define pipelines using YAML, allowing for version control and easy modifications.

Custom Build Agents: Use custom build agents for specific build environments or dependencies.

Extensive Ecosystem:

Marketplace Extensions: Access to a vast number of extensions from the Azure DevOps Marketplace.

Security and Compliance:

Secure Access: Role-based access control, integration with Azure Active Directory (AAD).

Compliance: Meets industry standards and compliance requirements.

End-to-End DevOps:

Complete DevOps Lifecycle: Covers everything from code repository management, build, test, release, and monitor.

Key Features in Specific Areas

Code Deployment

Automated Deployment: CI/CD pipelines automate the build, test, and deployment processes.

Multi-Stage Pipelines: Supports multiple stages, including build, test, and deploy phases.

Framework Update:

Automated Testing: Run automated tests for updated frameworks to ensure compatibility.

Dependency Management: Integrates with tools like NuGet for managing dependencies.

Database Management

Infrastructure as Code (IaC): Use ARM templates, Terraform, or other IaC tools for managing database infrastructure.

Deployment Scripts: Automate database migrations and updates using scripts or tools like EF Core migrations.

Disaster Recovery Plan:

Backup and Restore: Automate backups using Azure Backup and configure automated restore processes.

Geo-Redundancy: Use Azure’s geo-redundant storage options for critical data.

Bug Tracking

Azure Boards: Integrated with Azure Boards for tracking work items, bugs, and feature requests.

Integration with Other Tools: Connects with other bug tracking tools like Jira.

Pricing Comparison

| CI/CD Tool | Pricing Model | Free Tier | Notable Features |
| --- | --- | --- | --- |
| Azure Pipelines | Pay-as-you-go, per parallel job | Free tier with 1 free Microsoft-hosted CI/CD parallel job (1800 min/month) | Integration with Azure services, YAML pipelines, rich ecosystem |
| GitHub Actions | Pay-as-you-go, per job | Free tier with 2000 minutes/month for public and private repos | Seamless GitHub integration, Marketplace actions, YAML workflows |
| GitLab CI/CD | Included with GitLab subscription, SaaS options | Free tier with 400 CI/CD minutes per month | Integrated with GitLab, auto DevOps, Docker support |
| CircleCI | Pay-as-you-go, per usage | Free tier with 2500 credits/month | Orbs (reusable packages), Docker layer caching, parallelism |
| Travis CI | Pay-as-you-go, per parallel job | Free for open-source projects | Easy setup, GitHub integration, support for multiple languages |

Considerations for Selecting Azure Pipelines

Integration Needs: If you are heavily using Azure services, Azure Pipelines offers the most seamless integration.

Cost Management: Evaluate the cost based on your pipeline usage and compare it with the available free tiers and pricing models of other CI/CD tools.

Feature Requirements: Consider the features you need such as YAML support, custom build agents, and marketplace extensions.