**Azure Devops**

Azure DevOps provides several tools you can use for better team collaboration. It also has tools for automated build processes, testing, version control, and package management.

Azure Boards. These are agile tools that help us plan, track, and discuss our work, even with other teams.

Azure Pipelines. These will let us build, test, and deploy with CI/CD that works with any language, platform, and cloud.

Azure Test Plans. These are manual and exploratory testing tools.

Azure Repos. These provide unlimited, cloud-hosted private, and public Git repos.

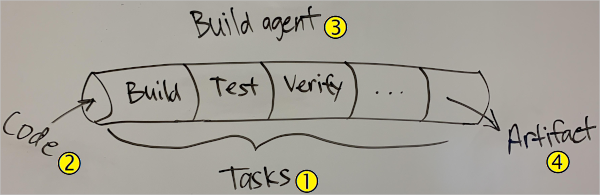
Azure Artifacts. These let us create, host, and share packages.

**Azure Pipelines**

Microsoft Azure Pipelines is a cloud service that you can use to automatically build, test, and deploy your code project. You can also make it available to other users. And it works with just about any language or project type.

**Continuous integration (CI)** is the process of automating the build and testing of code every time a team member commits changes to version control.

CI encourages developers to share their code and unit tests by merging their changes into a shared version control repository after every small task completion. Committing code triggers an automated build system to grab the latest code from the shared repository and to build, test, and validate the full main branch (also known as the trunk).



CI is the process of automating the build and testing of code every time a team member commits changes to version control. I know we don't do automated testing yet, but give it time.

* A **pipeline** defines the continuous integration process for the app. It's made up of steps called tasks  . It can be thought of as a script that defines how your build, test, and deployment steps are run. I'm going to try to map our scripts to tasks.
* The pipeline runs when you submit code changes . You can configure the pipeline to run automatically, or you can run it manually. You connect your pipeline to a source repository like GitHub, Bitbucket, or Subversion. We'll use GitHub. One of our tasks for this sprint is to start using GitHub.
* A build agent builds or deploys the code. When your build or deployment runs, the system begins one or more jobs. An agent is installable software that runs one build or deployment job at a time. Because we're using Azure Pipelines, we can use a Microsoft-hosted agent. With Microsoft-hosted agents, maintenance and upgrades are taken care of for us. Each time we run a pipeline, we'll get a fresh virtual machine. There are several virtual machine images to choose from, including Ubuntu 20.04, which is what we use.
* The final product of the pipeline is a **build artifact** . Think of an artifact as the smallest compiled unit that we need to test or deploy the app. For example, an artifact can be:

A Java or .NET app packaged into a .jar or .zip file.

A C++ or JavaScript library.

A virtual machine, cloud, or Docker image.

**Build agents**

A build agent is a piece of installable software that runs one build or deployment job at a time.

Differences between implementing **hosted and private agents**

You can use either a Microsoft-hosted or a private agent. What are the differences?

With Microsoft-hosted agents, maintenance and upgrades are taken care of for you. Each time you run a pipeline, you get a fresh virtual machine. The virtual machine is discarded after one use.

An agent that you set up and manage on your own to run build and deployment jobs is a self-hosted agent. You can use self-hosted agents in Azure Pipelines. Self-hosted agents give you more control and let you install any software you need for your builds and deployments.

You can install the agent on Linux, macOS, or Windows machines. You can also install an agent on a Linux Docker container. After you've installed the agent on a machine, you can install any other software on that machine as required by your build or deployment jobs.

**Agent pools**

An agent pool defines the sharing boundary for all agents in that pool. In Azure Pipelines, agent pools are scoped to the Azure DevOps organization so you can share an agent pool across projects.

A project agent pool provides access to an organization agent pool. When you create a build or release pipeline, you specify which pool it uses. Pools are scoped to your project so you can only use them across build and release pipelines within a project.

**Agent queues**

If you are a project team member, you create and manage agent build queues from the agent pools tab in project settings.

Service endpoints for integration with third-party systems

Service endpoints are a way for Azure DevOps to connect to external systems or services. They are a bundle of securely stored properties that includes but is not limited to:

Service name

Description

Server URL

Certificates or tokens

User names and passwords

Extensions are then able to access the service endpoint to get the stored details to perform the necessary operations on that service.

**Concurrent pipelines**

You can run concurrent pipelines (also called parallel jobs) in Azure Pipelines. One parallel job in Azure Pipeline lets you run a single build or release job at any given time. This rule is true whether you run the job on Microsoft-hosted or self-hosted agents. Parallel jobs are purchased at the organization level, and they are shared by all projects in an organization.

**Microsoft-hosted CI/CD**

If you want to run your builds and releases on machines that Microsoft manages, use Microsoft-hosted parallel jobs. Your jobs run on the pool of hosted agents. Microsoft provides a free tier of service by default for every organization.

If you want Azure Pipelines to orchestrate your builds and releases, but use your own machines to run them, use self-hosted parallel jobs. You start by deploying agents on your machines. You can register any number of these self-hosted agents in your organization. Microsoft charges based on the number of jobs you want to run at a time, not the number of agents registered.

Plan a strategy for concurrent pipelines

Here are some steps to take to plan for concurrent pipelines.

Determine how many parallel jobs you need

**Github**

github repository

git:local repository

github:central repository hosted on cloud

**Create a repository**

1.Unique name

2.description

3.choose visibility:public or private

4.choose three files:

**readme**:This is where you can write a long description for your project.

**.gitignore**:Choose which files not to track from a list of templates

**licenses**:A license tells others what they can and can't do with your code.

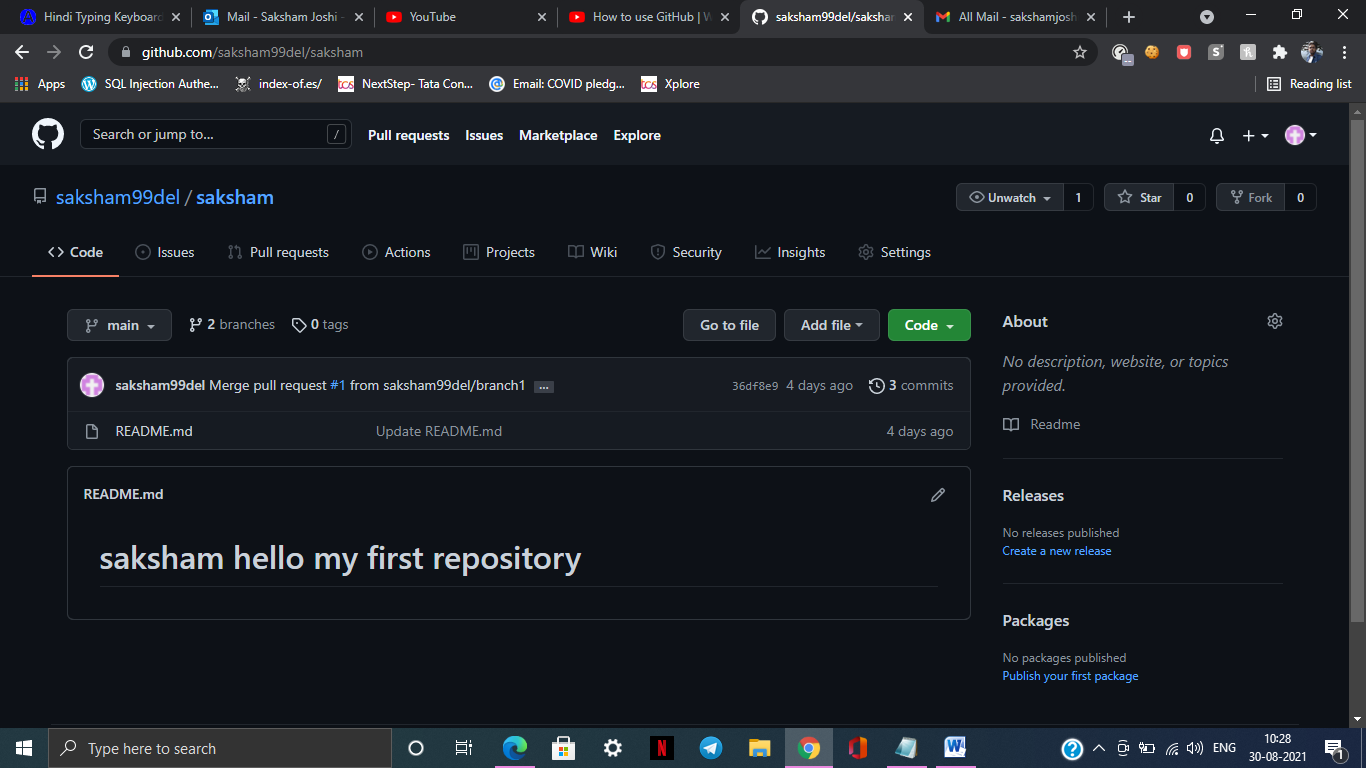
**master branch**:branch where main code resides

**branch**:add some new features to the project and merge with master branch

**commit changes**:record changes,git assigns a unique id called hash or sha to every change

**open and merge pull request**:notifies developers about changes in branch pushed to repository

Creating my own repository and branch



Merge Pull Request

