***2. Getting started with azure devops portal***

* Google with azure devops and click on start free
* Using [spraveen879@gmail.com](mailto:spraveen879@gmail.com) for login.
* Once logged in, it will ask you to create a project. Create a project as below.

A screenshot of a computer

Description automatically generated

***3. Getting started with deployment basics***

**Application server/ Web server**

Application server is specifically designed to run applications. It hands http requests and sends response calls over http protocol.

Ex : We have the code for rahulshettyacademy in our local. This application server will run this code and display the UI when a request is sent from a user on <http://www.rahulshettyacademy.com>. Example is apache tomcat.

**Hosted server**

Hosted servers are nothing but physical machines where application servers are hosted.

For ex, we can download apache tomcat from our local and run the code for rahulshettyacademy. So our local laptop/cpu is the hosted server since it hosts the application server

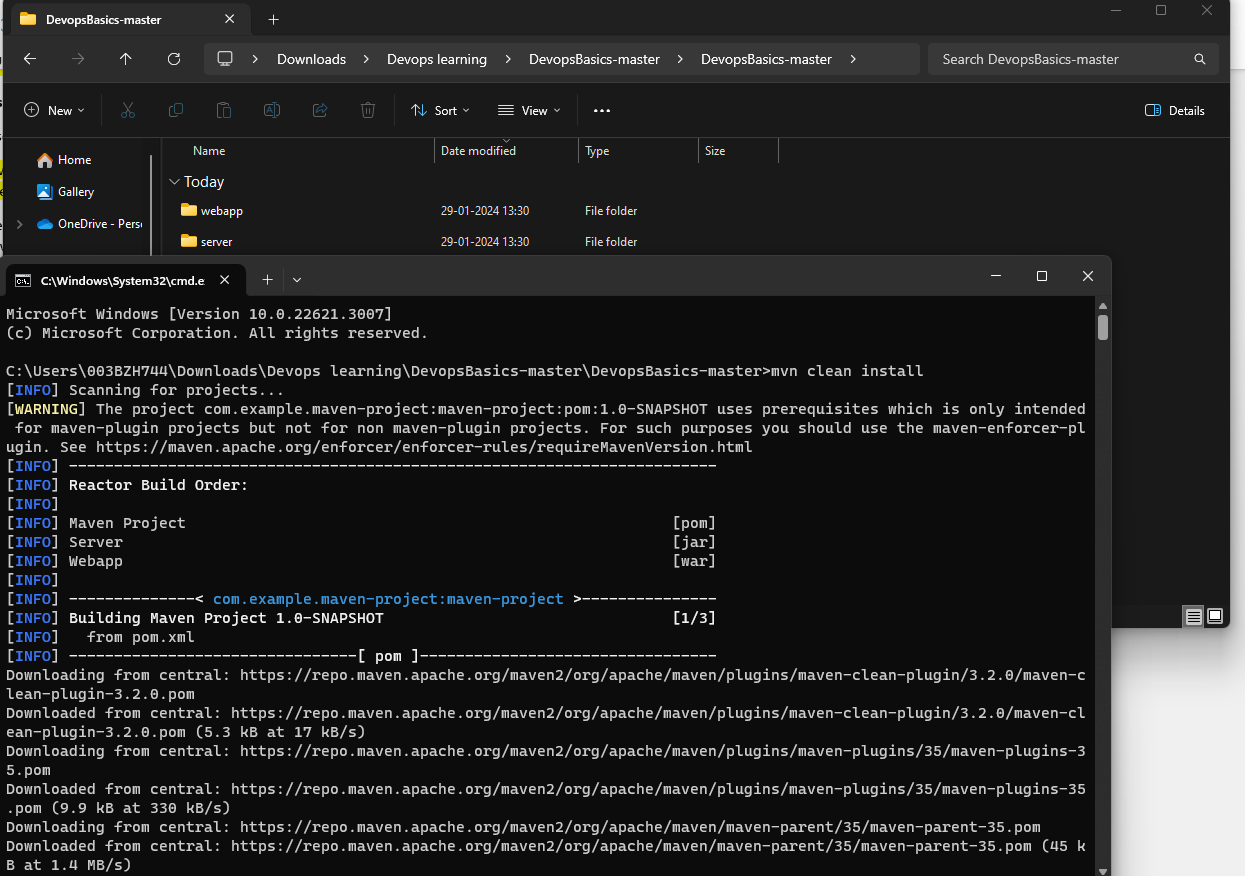
Now every computer will have a unique ip address. Ex : 199.323.423.66. Now I am starting the port in 8080. So the unique address where I am hosting the code for rahulshettyacademy is 199.323.423.66:8080.

Now the application can be accessed in the ip 199.323.423.66:8080 from anywhere. Now we can buy domain name from go daddy. Now when you google “rahulshettyacademy.com”, in the back end 199.323.423.66:8080 will get hit and website would be displayed for a user.

Now we have a code in our machine. We will not copy the code in apache tomcat. If we use java, for ex, we can export it as a war file. That war file will be placed on the apache tomcat server and that will be hosted on the www.

***5. Deploying hello world code app in tomcat server***

* Download and install apache tomcat server.
* Now go to eclipse and open the folder where your code is kept.
* Now open the folder from command prompt and run “mvn clean install”
* Now what happens is it will run all the unit test cases in the project and once these unit cases are passed, it will create a war file. If there are any compile issues, it will not create .war file.
* Always, the war file gets created in the target folder.
* Now copy the war file and paste it in webapps folder in tomcat local.



We are taking Rahul Shetty git folder from lecture 4, hitting command prompt with mvn clean install. Now, the war file is generated as below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Copying this war file to webapps in apache tomcat file.

* Now, start the apache local host by clicking on startup.bat in bin folder in apache tomcat folder.
* When port 8080 is already used by some other process, we need to kill that and make apache tomcat use this port. (See google or video for the process)

A screenshot of a computer

Description automatically generated

* Now, open localhost:8080 in chrome. Apache tomcat should open.

A screenshot of a computer

Description automatically generated

* Since we copied the .war file and copied in webapp folder, open the link localhost:8080/webapp.
* Now, the UI displays in the URL. Now, we will work on doing the same through azure.

***6. What is continuous Integration and continuous Delivery***

**Continuous Integration** : If developer makes any changes in the code and pushes to github, automatically build should get triggered. Build is nothing but the execution of mvn test install command here. This will run unit tests and once its successful and generate war file.

Ex : When a code is pushed, git alerts azure that code is pushed. Azure will trigger the build(that is it will run mvn test install which in turn runs test cases and once successful, .war file is generated)

CI is the process of automating the build and testing of code every time a team member pushes his code to repo.

**Continuous Delivery** : This is the extension of CI where after triggering the build, CD will also deploy the build to prod/stg etc. That extra step of deploying the code + CI is named as Continuous delivery.

A close-up of a diagram

Description automatically generated

* Whenever code is pushed to git, the automated flow(Azure) gets notified on the code push and does the build process(Run unit cases, generate war file) - CI
* Now, azure will take war file and deploy into stg or test environment. In stg or test, automation tests should run(See integ tests, load tests and other tests)
* Once that is run, it will push that war file to production environment. – CD

***8. What are build pipelines ? What should you know about it ?***

**Build pipelines** : These pipeline jobs clone the code from assigned Repositories and build the project based on instructions provided in the yaml file. Typically we use build pipelines to generate artifacts (Here artifacts means the war files)

A screenshot of a video game

Description automatically generated

* Click on create pipeline.
* Click on github in the options to select the repo.
* It will ask you to login. Login with your creds
* Now, to get the code which Rahul explains in the demo, go to git hub and create a new repository.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Instead of creating a new one, click on import a repo.
* Give this and import : <https://github.com/rahulshettyacademy/DevopsBasics>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Now if you see in Azure, all the repos will be visible.
* Now click on devopsrepo\_learning and proceed to configure.

A screenshot of a computer

Description automatically generated

* Now whatever azure has to do for a build pipeline will be provided in a yaml file.

A screenshot of a phone

Description automatically generated

* Now among the list, select maven.

A screenshot of a computer

Description automatically generated

* Now by default this yaml file will be generated.
* This yaml file will act on our repository. Whenever there is a push, the process starts based on whatever you give in yaml.

***9. Understanding build pipeline configuration and logs***

A screenshot of a computer

Description automatically generated

* Click on save and run in the yaml file. What it does is, it will save this yaml file to the repository selected.
* So whenever you push a code, azure will look for the yaml file in the repo. Based on the instructions given in the yaml file, the pipeline will proceed.
* Example : run in ubuntu machine, look for pom.xml, goals : package means it will download all the dependencies for the project and run the command ‘mvn clean install’

A screenshot of a computer

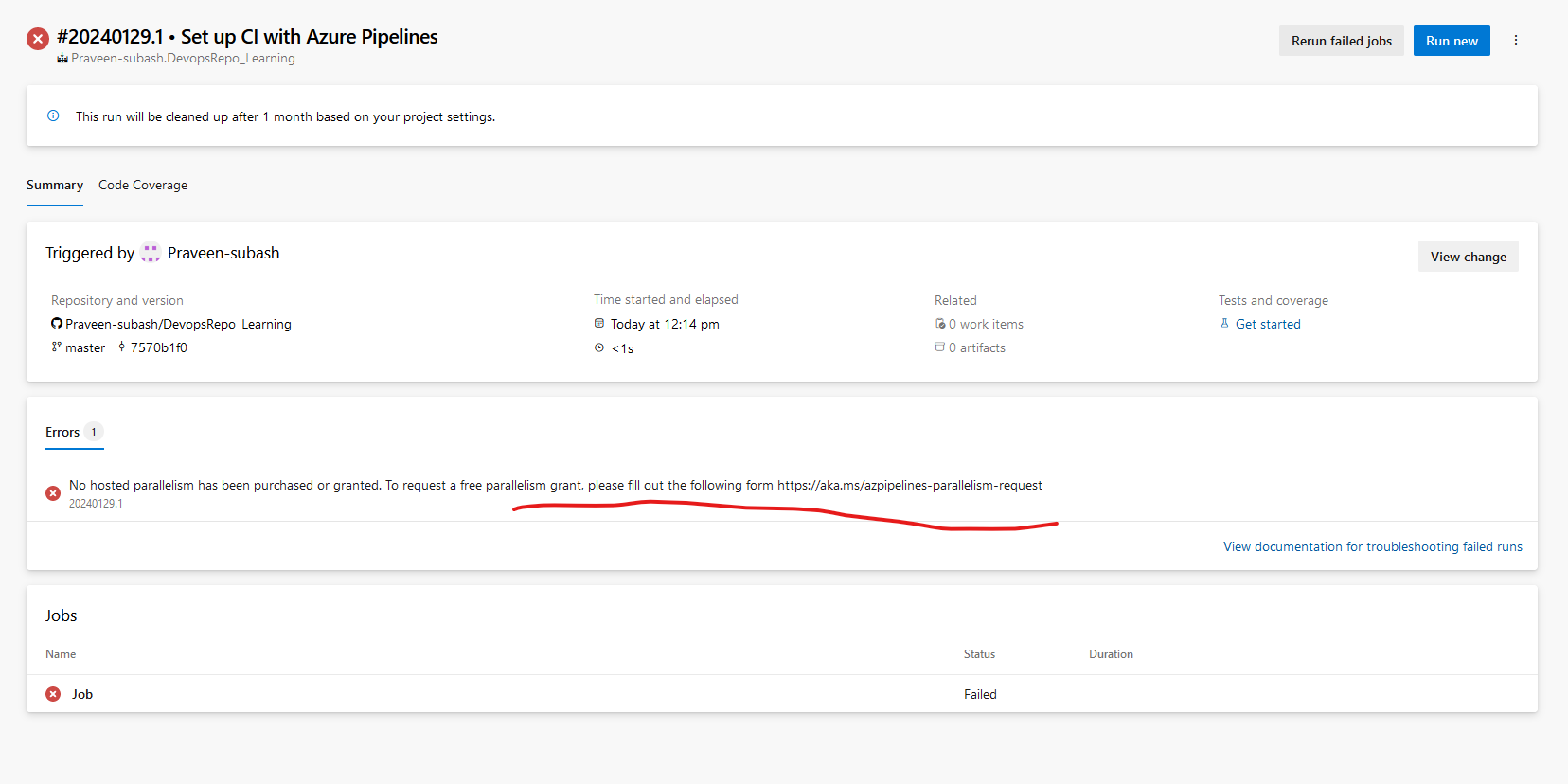
Description automatically generated

* Click on save and run.

A screenshot of a computer

Description automatically generated

* This yaml file is saved to the repo now.



The pipeline is run now and we are getting this error. We need to submit a request to Microsoft to enable us.

<https://intellipaat.com/community/45138/what-is-the-azure-devops-organization>

A screenshot of a computer

Description automatically generated

After one day, the request is approved

A screenshot of a computer

Description automatically generated

Now if you trigger the build, job will run successfully as shown below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Previously, we would have run this in our local. Now this is being run in ubuntu machine as specified in the yaml file.

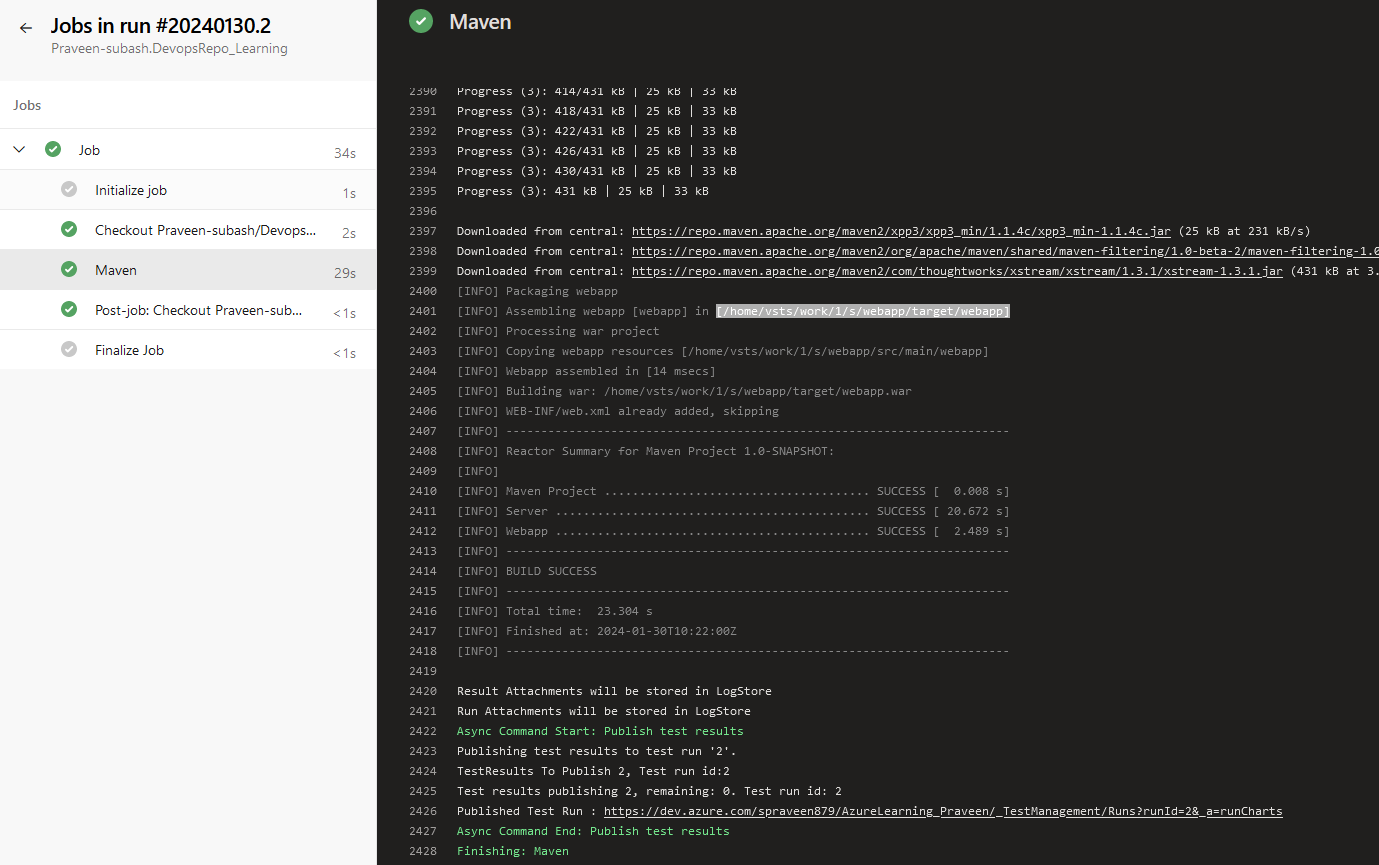
A screenshot of a computer

Description automatically generated

* As a next step, this is fetching everything from the repo we specified in the azure config.



* This entire project is checked out to folder s (We will be speaking about this later)



* Since we have specified type as package, maven will download all the dependencies and execute mvn clean install.
* See the line 2401 ; Assembling webapp in home/vsts/work/1/s/webapp/target/webapp
* This is the folder in the ubuntu machine. It will create a folder called home in that it will store it in s folder(s means source code)
* Now war file is created. That warfile must be placed in apache tomcat. Currently it is sitting in some other ubuntu machine. Our goal is to bring that war file and bring it to our build project. Only then we can deploy it in test/stg/prod environments.

***10. Copying build artifacts from the project to azure staging directory***

* We need to bring that war file from s folder of the ubuntu machine.
* Note that like s folder inside work/1 there is a folder called a. a stands for artefacts and s stands for source code.
* We need to bring the war file from s folder to a folder. We have access only to a folder. That is the reason why we need to bring the war file from s to a folder.

A computer screen shot of a computer

Description automatically generated

* Now search for this copy files. This will be used to copy the war files

A screenshot of a computer

Description automatically generated

* You don’t have to type work/1/a etc.. Just type ${build.artifactstagingdirectory}. This is the a folder. So we are targeting to copy the war files to this folder.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* We are looking for war files from the root directory and placing them in target folder.
* Click on add and the yaml file will be updated as below
* \*\*/\*.war' is the correct content which indicates folder and subfolder. Change it

A screenshot of a computer

Description automatically generated

* Now we need to publish this war file in our build. Like bringing it from a folder to our build. For that search for the task publish

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* We are bringing that war file to our azure pipeline.

A screenshot of a computer

Description automatically generated

* Commit the yaml file

***11. Apply continuous integration to build pipeline for every commit***

***A screenshot of a computer

Description automatically generated***

* Now after the pipeline is run, it says 1 published. Previously, it would have been 0 artifacts.
* When you click on that, you will see the war file.

A screenshot of a computer

Description automatically generated

Here war file is copied from s to a folder

A screenshot of a computer

Description automatically generated

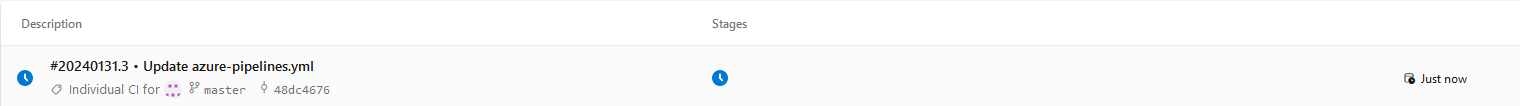
Here war file is copied from a folder to our build

Now as you have seen, whenever there is a commit in the code, this pipeline would have run. This is due to the yaml file having the command trigger : master. So whenever a change is made in master, automatically pipeline will run.

A black rectangular object with a black stripe

Description automatically generated

Ex I am going to git and adding a comment and committing it.



Suddenly build is triggered and updated war file is applied.