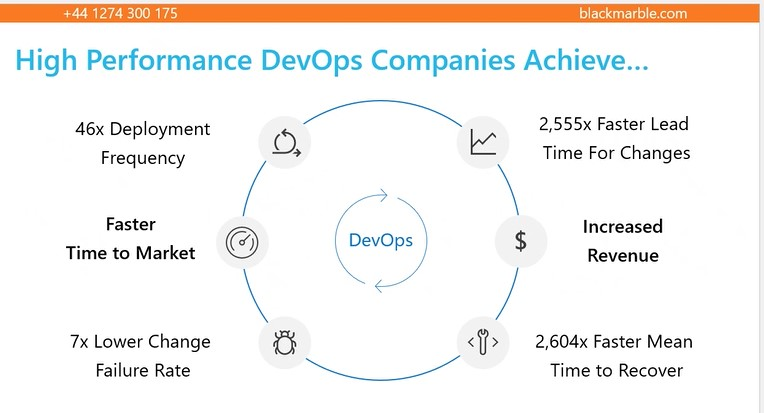
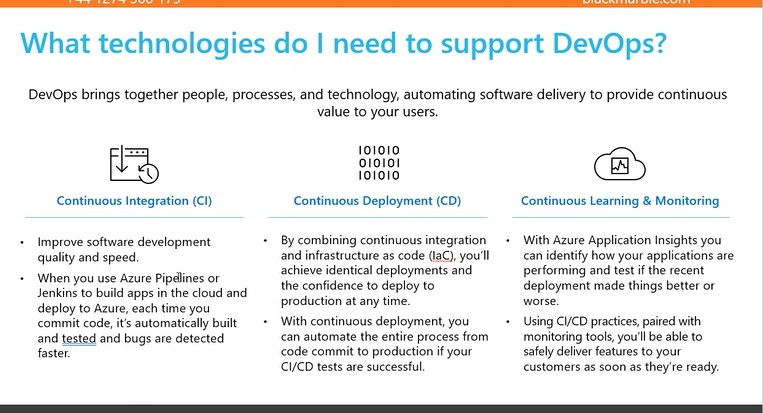
Azure DevOps

Development Operations

* Developing ETL pipelines, images, generating new data, data transformations, etc..
* DevOps: union of people, process, and products
* It helps in continuous integration and delivery





Collaboration and Feature Branch

* Take a copy of the code from the collaboration branch and create a new branch and name it accordingly
* This creates a feature branch for individual developers
* Each developer can make changes in their own feature branches
* To commit the changes to the main branch(collaboration branch), they make a request which is verified by the admin
* If the changes are valid and meet the requirement, they are committed to the main branch and going further everyone can see those changes in it

Epic, Feature, Story, Tasks,

Sprint

* Goal-oriented
* Time-bounded
* Sprint timespan – 2 to 4 weeks

Example

* Epic
  + Create user authentication for a web application
* Features
  + Designing
  + Authentication
  + Verification
  + Testing
  + Production
* User Story
  + Home page
  + Integration

\*Note: if tasks in a user story can’t be completed in the designated sprint, they are moved to backlog

Organization settings

* Overview, projects, users, billing, global notification, usage, extensions, Azure Active Directory
* Click on billing -> set up billing -> upon validating the subscription -> save -> change the MS Hosted CI/CD to 1 -> save

Inside the project

* Wiki
  + About the project
  + All the project related documents to be placed under wiki

Add Priorities

* 1, 2, 3, 4
* Low -> High

Risk

* 1- High
* 2- Medium
* 3- High

Effort

* How long it is going to take
* 10 – meaning 10 hours

Acceptance Criteria

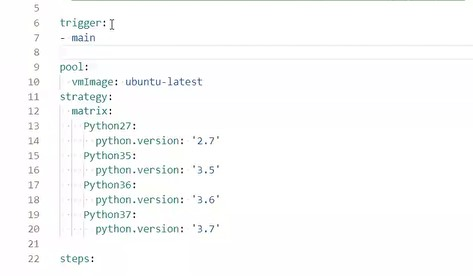
* Description about the foremost requirements that are absolutely needed from the developer
* Only once the acceptance criteria is completed can the user story be considered closed
* The user story is present for every user story

Repository

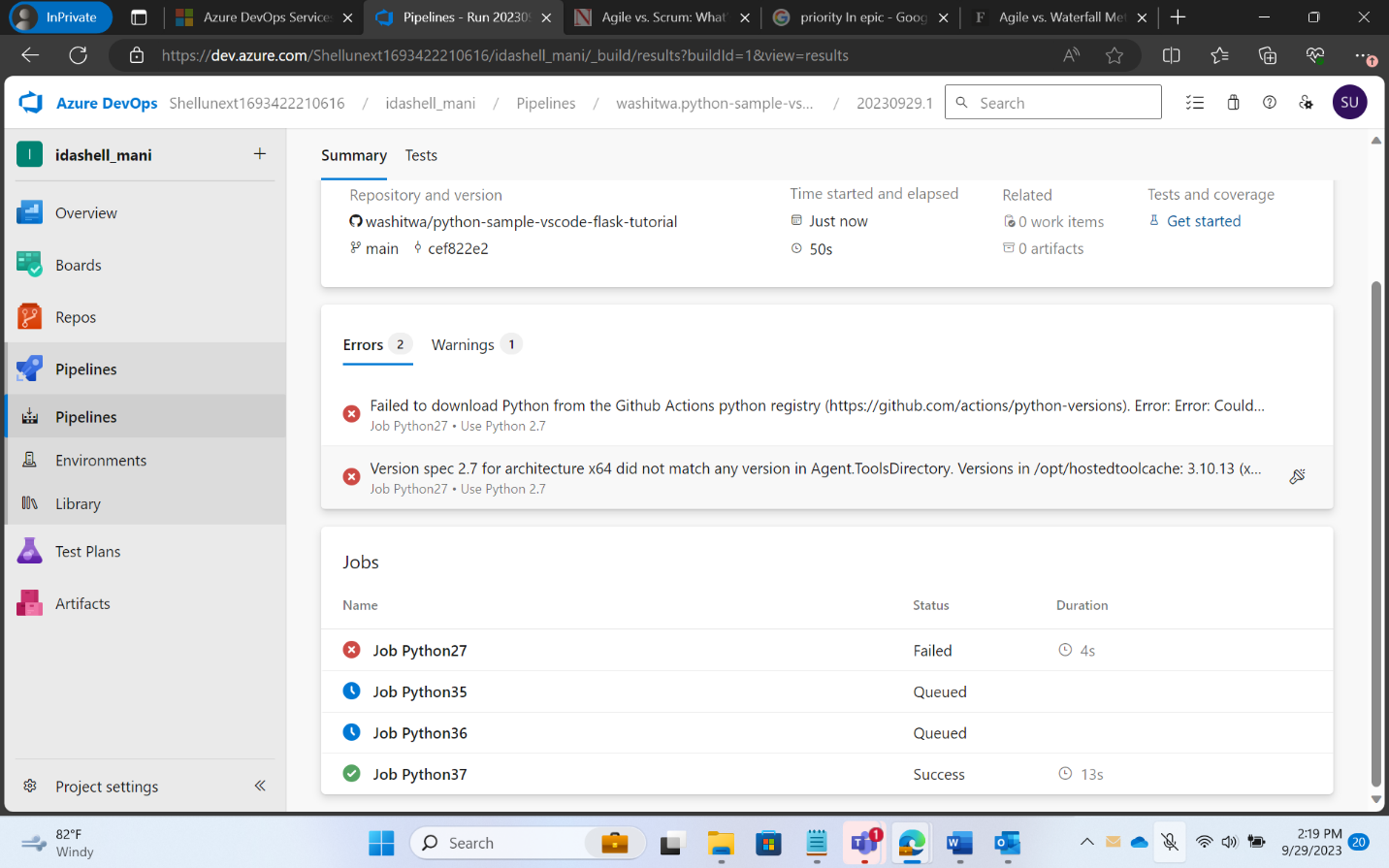
* Project -> Repos -> Files -> Repository\_Name
* Inside the repo, create a config.json file
* It is not necessary to get the code from the main branch
* Any other feature branch can be used to get the data if needed
* Create a new repo
* In the main repo, add a config.json file
* Create a feature branch pointing to the main branch
* Make some changes to the file present in the feature branch
* Create a pull request
* Approve the request
* Revert the operation by going to the pull requests option

Pipeline

* Scenario:
  + Main branch is present
  + To work on the code changes, a feature branch is also present
  + Upon making changes, pull request is done
  + The main branch is updated using merge
  + To automate the process of continuous deployment, a pipeline is created
  + The continuous deployment ensures that the testing process happens with integration
* Inside the pipeline



* + The trigger suggests that when any changes are made in the main branch, the pipeline should start running
  + Create the pipeline with the Flask project in GitHub and then save and run
  + Python Flask 3.7 application is running



Case Study

* Only 2 days
* Second day second half, presentation
* Each and every team, one case study
* Project 1: Alternative Fuel
* Reimbursement