# Architecture of the system

## Design and workflow

A screenshot of a computer

Description automatically generated with medium confidence

1. Developer commits changes to the git repo. After the commit or after the developer branch is merged to the main branch the pipeline defined in ‘azure-pipelines-full-app.yml’ would be trigered (For better control it is a good practice disable direct commits to the main branch, instead branches should be used).
2. The triggered pipeline would compile the code using the maven wrapper (mvnw), then some unit tests would run. For the tests to complete the pipeline would provision a temporary MySql database on azure. Once the tests are done the a docker image would be created and pushed to the docker hub. After that a cleanup task would remove the temporary resources created for the pipeline run.
3. The infrastraructure provisioning part would be done by the pipeline defined in ‘azure-pipelines-create-infra.yml’. It would fetch ARM templates from the git and deploy the reosurces that would host and monitor the webapp. This pipeline should provison a app service plan, an app service instance for a webapp and a Mysql server for the purpose of hosting the database.
4. In addtion to that two additional resources an application insights instance and a log analytics instance for monitoring and alerting purposes.

Graphical user interface, text, application, email

Description automatically generated

1. The pipeline wil also deploy some monitoring metrics and alerting rules for the application serivce plan and the app service. How to configure their parameter values would be explained in the setup process.

## Technologies adapted and choices made

1. GitHub for source control: This is one of the widely used git solutions around and gives you the option to create public repos by default unlike other solutions. It has a rich set of features and backed by a large community.
2. Azure pipelines: Good set of features available and GitHub has a extension that you can install to conect the repo to azure pipelines.
3. DokerHub: There are other options like azure container registory available, but dockerhub is free and you can have the images public. Further solutions like azure container registory requires provisioing, incures cost and need authentication to work. In that regard DockerHub is simple.
4. Azure App service plan: Different skus and plans supported, can set autoscale feature based on the demand. This can not ony scale up, but a higher skus has a scale out option with multiple instances.
5. Azure webapp for containers: With app service plan and a docker image is to hosted this appears to be obvious option. Furthermore it can be configured for monitoring and alerting rules.
6. Appcation insights: In addition to inbuilt monitoring and alertig in azure webapps this has additional features for webapps and can be configured easily.
7. If logs needed to processed from the webap, log analytics can be used for that. Furthermore an application insights instance can also be connected to a log analytics workspace.
8. Mysql: it is a one of the widely used databses for around. Useully free and azure has a PaaS solution for Mysql, can be installed using a ARM template.

# Project Setup information

The below sections will include information about the git repo structure, setting up the devops project, setting up the pipelines and setting the parameters for monitoring and alerting rules.

## GitHub project and repo structure

The GitHub repo for the solution can be found [here](https://github.com/lakpj/contoso-payroll). You should be able to clone or download it to any location.

Graphical user interface, text, application, email

Description automatically generated

The solution is related to ‘full-app’ and ‘infrastructure’ folders inside the repo.

You can ignore the ‘backend’ and ‘frontend’ folders as they are some test folders to break the spring-boot and react code into two containers.

Both ‘full-app’ and ‘infrastructure’ folders containers for pipeline yml files related to creating the docker image and provisioning infrastructure.

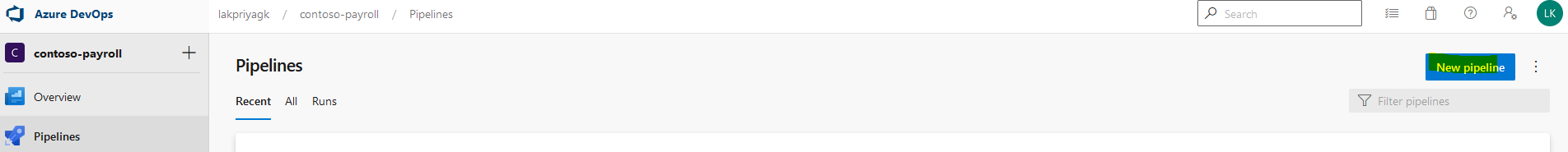
## Setting up the azure devops project

If you do not have an azure devops project, please follow this [link](https://docs.microsoft.com/en-us/azure/devops/organizations/projects/create-project?view=azure-devops&tabs=browser) to create a devops organization and a project.

If you setup the git repo in GitHub, you would need to install the azure pipelines extension from the GitHub marketplace. If you do not have the setup, you can follow this [article](https://www.azuredevopslabs.com/labs/vstsextend/github-azurepipelines/) to do the needful.

## Setting up the build pipelines

1. In the azure pipeline project under Pipeline section, ‘New Pipeline’ can be used to import the pipelines.



1. Select the location where you have placed the pipeline yml files. Location outside ‘Azure Repo Git’ may prompt login.

Graphical user interface, text, application

Description automatically generated

1. Then select the repository

Graphical user interface, text, application, email

Description automatically generated

1. Select an existing pipeline yaml file

Graphical user interface, text, application, email

Description automatically generated

1. Then pick the branch and pipeline yaml relative path from the drop downs.

Graphical user interface, text, application, email

Description automatically generated

Review the pipeline content and save the pipeline

Graphical user interface, application

Description automatically generated

1. Once the pipeline is imported and saved. You can edit the pipeline and change the pipeline name or move it to a structure you want. ‘Rename/move’



1. For the pipelines to run then you need to configure the service connections to the docker registry and the azure subscription you want the resources to be deployed.

This is general documentation on creating [service connections](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml) for azure pipelines

The following screen shots show the steps for service connection for the docker registry.

* Go to ‘service connections’ from the ‘project settings’ page

Graphical user interface, application

Description automatically generated

* From ‘New connections’ select ‘Docker registry’ option.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

* Enter docker registry credentials and verify if the connection is made properly.
* Then you need to create another service connection to the subscription. For that select ‘Azure Resource Manager’ option and the automatic option where it will create a service principle in the tenant where your subscription is connected.

NOTE: For this to work the user you have connected to should have access to create service principal objects in the azure tenant. If not the service connection creation attempt would fail.

Graphical user interface, text, application, email

Description automatically generated

* Now you should see the list of service connections created under ‘service connections’ from the ‘project settings’ page

## Setting up variable groups for the build

For the pipelines to run you would require the above created service connections and few other values to be included as variable group values. This would help the pipelines run without doing any change to the yml content.

This is the general steps to create azure pipeline variable groups.

To create a Variable Group, follow the steps below:

1. In the left panel, click and expand Pipelines.
2. Under Pipelines, click Library.
3. Click the Variable group button.
4. Enter a name for the variable group in the Variable Group Name field.
5. Use the Description field to enter information about the variable group.
6. Click the Add button to create a new variable for the group.
7. Fill in the variable Name and Value.

NOTE: For this setup the variable group name should be ‘ContosoPayroll’, if not pipeline yaml content needs to be changed

Graphical user interface, text, application, email

Description automatically generated

All together 7 variables should be defined in the ‘ContosoPayroll’ variable group.

The names should be,

|  |  |
| --- | --- |
| **Variable group name** | **Purpose or default value** |
| ARM\_SERVICE\_CONNECTION | Service connection name for the subscription created earlier |
| ARM\_TEMPLATE\_BASE\_URI | Should be in the format https://raw.githubusercontent.com/<github-user>/<github-repo>/main |
| DOCKER\_REGISTORY | lakpriyagk/contoso-payroll-app |
| DOCKER\_SERVICE\_CONNECTION | Service connection name for the docker registry created earlier |
| MYSQL\_PWD | User password for Mysql database. Used for this labs ‘Password1234!’, the same should be used in infrastucture\MySqldb\azuredeploy.parameters.json. |
| MYSQL\_SERVER\_NAME | contoso-pay-db-server.mysql.database.azure.com |
| MYSQL\_USER\_NAME | sqladmin@contoso-pay-db-server |

When running the pipeline for the first time you grant permission to the pipeline to use the variable group created.

Text

Description automatically generated

## Setting the parameters for builds

Before running the pipeline defined in ‘azure-pipelines-create-infra.yml’, it is possible to change the deploy parameters for the azure resources.

Text, application

Description automatically generated

NOTE: The information for the Mysql database should not be changed.

For the alerting rule emails, it is required to add a valid action group email address to the file at infrastucture\AppMonitoring\azuredeploy.parameters.json

Text

Description automatically generated

## Cleaning up Resources in the subscription

If you want to clean up the resources created, then you can import the pipeline defined in ‘azure-pipelines-remove-infra.yml’.

When running the pipeline do enter the resource group in the subscription you want to remove, then pipeline should clean it.

Graphical user interface, text, application, email

Description automatically generated