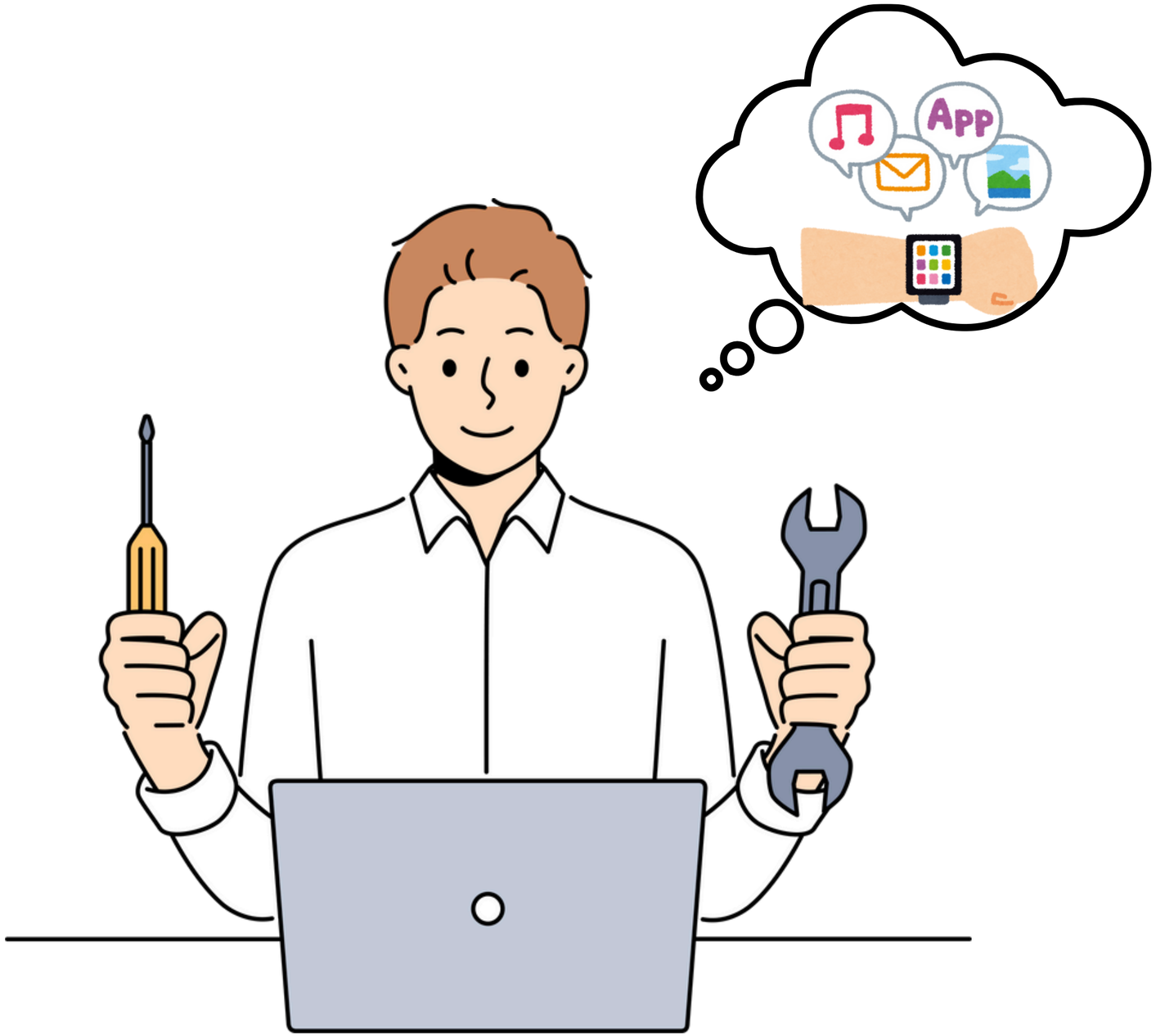
SOFTWARE ARCHIECTURES

WHAT IS AN ARCHITECTURE

LETS PLAN TO DESIGN THE BLUE PRINT FOR APP



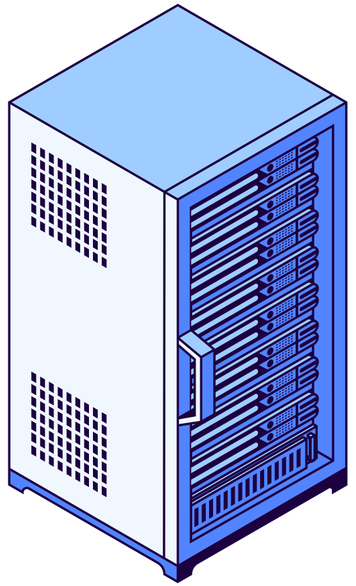
TYPES OF ARCHITECTURES

- ONE-TIER
- TWO-TIER
- THREE-TIER
- N-TIER

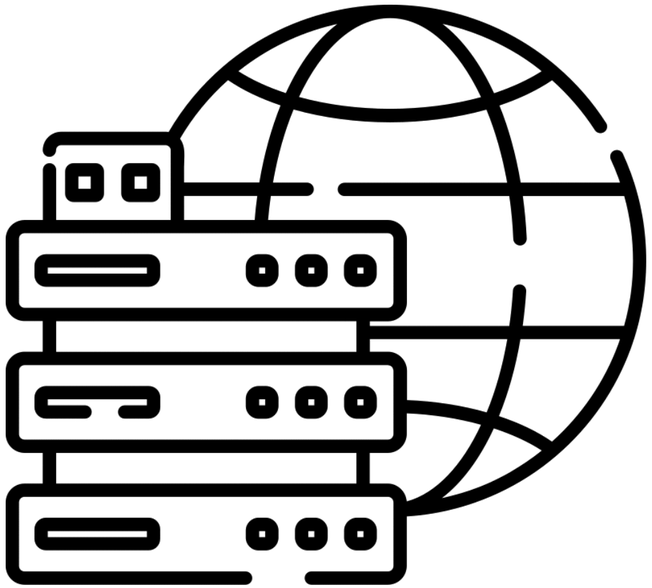
PROVIDES SERVICES TO USER

USERS

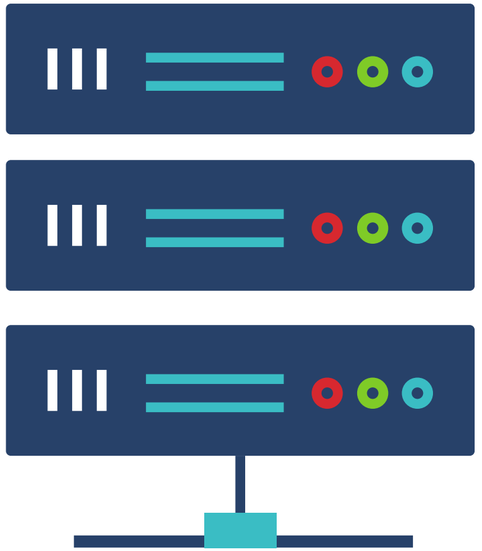
SERVER



SERVER TYPES

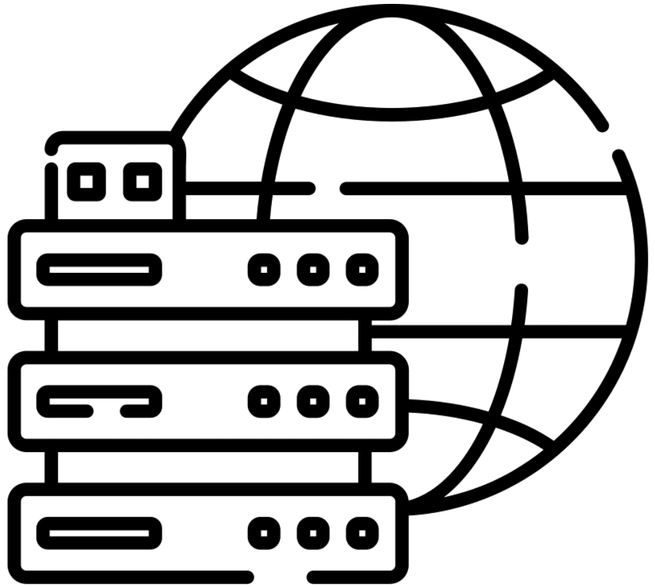


WEB SERVER



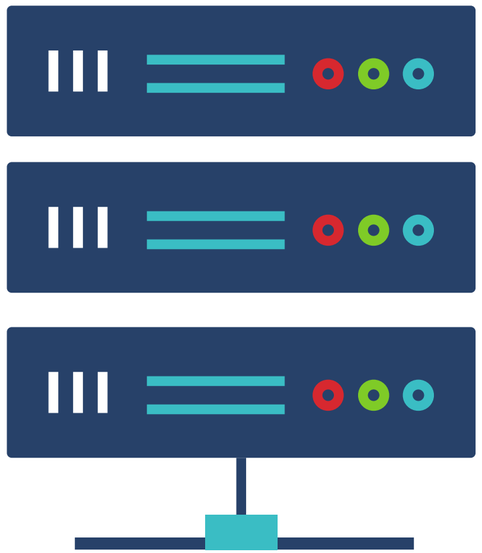
APP SERVER

DB SERVER



AKA : the presentation layer
Purpose : to show the app
Who : UI/UX (front-end dev)
What : Web Technologies
EX : html, css, js

WEB SERVER



APP SERVER

AKA : Business/Logic layer

Purpose : to use the app

Who : Backend Dev

What : Programming languages

Ex : java, python, c, c++, .net

DB SERVER

AKA : data server

Purpose : to store & retrieve data

Who : DBA/DBD

What : DB languages

Ex : mysql, sql, postgres, oracle --

ONE TIER ARCHITECTURE
stand-alone apps

WEB LAYER

APP LAYER

DB LAYER

TWO TIER ARCHITECTURE

client-server

WEB LAYER

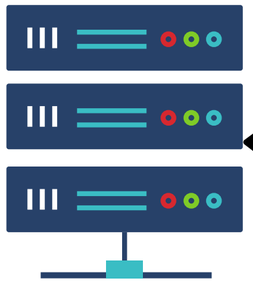
APP LAYER

DB LAYER

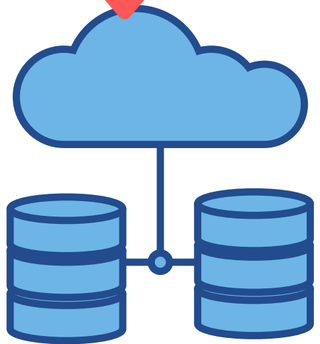
THREE TIER ARCHITECTURE

web apps

WEB LAYER



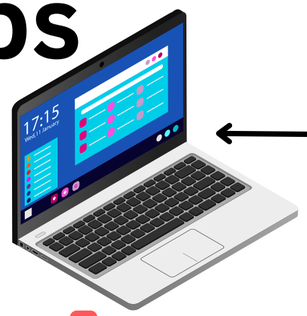
APP LAYER



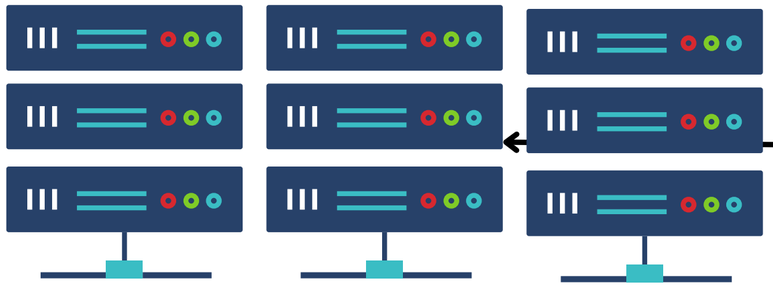
DB LAYER

N-TIER ARCHITECTURE

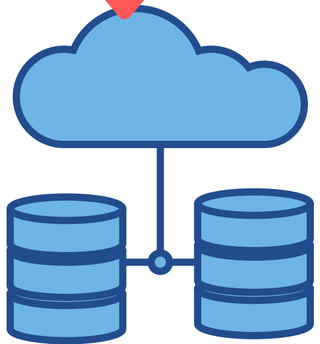
distributed apps



WEB LAYER



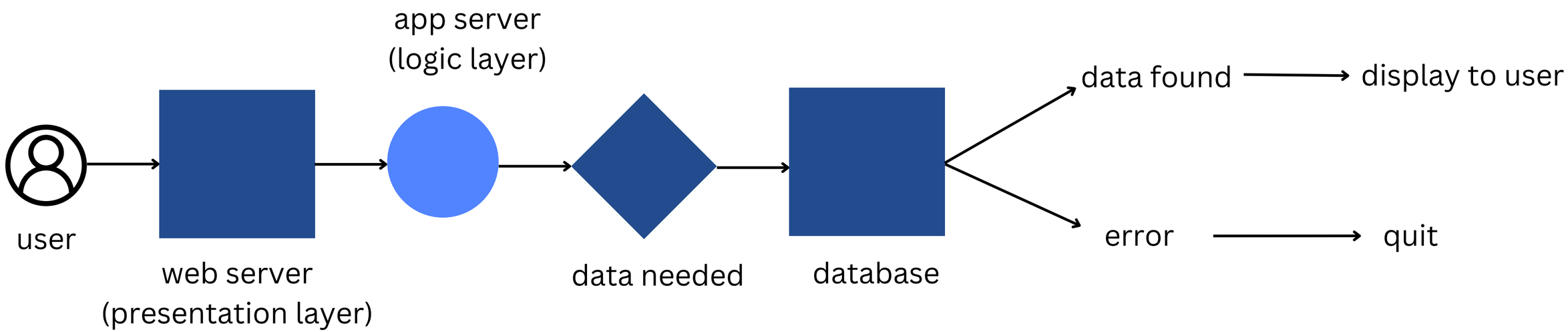
APP LAYER



DB LAYER

Here we will use docker-compose to deploy 3 tier application, because it is flexible to create multiple containers at a time.

3 TIER APP ARCHITECTURE



TOOLS TO DEPLOY THE APPLICATION

Developers will write the code from **frontend** and **backend.** They use **GIT** to track the changes and **GitHub** to push the code

We can use **CI/CD** tools called **jenkins** to automate the many tasks like code, build, test & deploy.

To pack the application into containers, we use **Docker.**

ENVIRONMENT SETUP

- **Development (Dev)**: Its like a playground for our developers, developers actively work on building and testing new features and fixes the bugs.

- **Quality Assurance (QA)**: the application is tested for functionality, performance, and bugs using the latest code changes. It's more stable than the dev environment and closely resembles production settings. QA engineers run various tests, such as unit, integration, and regression testing, to ensure that the software works correctly and meets requirements before moving to staging or production.

- **Production (Prod)**: the application is fully deployed and accessible to end users.

DEPLOY ON ENVIRONMENT

- The new code is first deployed to the **Dev environment**. Here, developers check if everything works as expected.
- Next, the code moves to **QA**. Testers perform different types of testing to ensure there are no bugs or performance issues.
- Once QA testing is complete and the code is stable, it’s reviewed for **final approval** before going live.
- The code is deployed to the **Production environment**. This is where users can access the updated application.

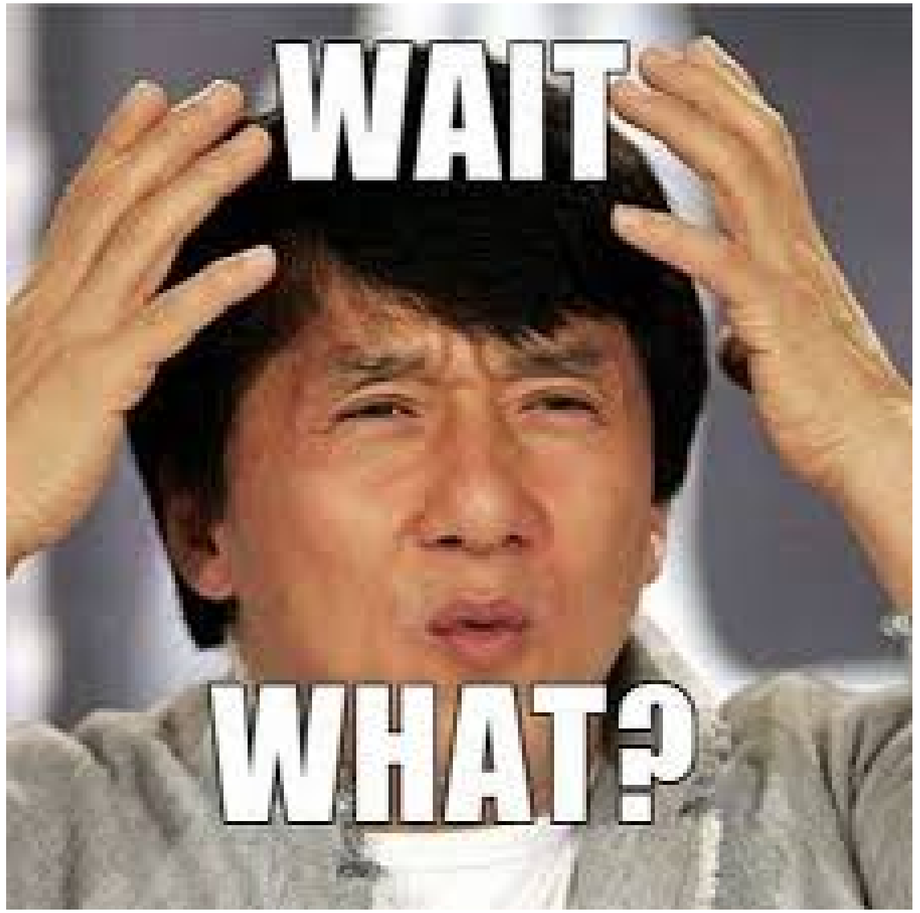
REPEATUUUU

LOCAL DEPLOYMENT

- Lets deploy the application locally, if it works fine then we can deploy write pipelines to automate.
- Install docker & docker compose on EC2 instance and deploy it by writing the Dockerfile and creating the containers manually.
- If containers are working well, then write the docker-compose file and check it.
- If everything works fine, then we can move to pipelines

LOCAL PIPELINE

- Create a local pipeline to test the application on our servers.
- Pipeline must contains the below
  - Code
  - CQA
  - quality gates
  - npm install
  - dependencies check
  - docker-image build
  - trivy scan
  - tag & push
  - docker-compose up -d
- Once the pipeline is success, then create a pipeline for dev team



Why for dev team, because we have to make sure that the application works smoothly at every env and to minimize the chances of errors in the prod env

DEV PIPELINE

For the dev pipelines, the application is tested by developers. This is where new features are added, bugs are fixed, and code changes are made.

STAGES:

- code
- cqa
- quality gates
- unit test
- Building a dockerfile
- create container on dev env
- Notify Team

QA PIPELINE

In the QA stage, testers check if everything works as expected. Trying to find bugs, issues, or performance problems.

STAGES:

- code
- cqa
- quality gates
- unit test
- Building a dockerfile
- Deploy to Test/QA Environment
- Integration Testing
- Functional Testing
- Regression Testing
- Performance Testing
- Security Testing
- User Acceptance Testing (UAT)
- Generate Test Reports
- Notify Team

FINAL PIPELINE

The final version of the application is deployed for real users.

STAGES:

- code
- cqa
- quality gates
- unit test
- Building a Dockerfile
- Push Image to Registry
- Approval Step
- Deploy to Production
- Post-Deployment Smoke Tests
- Monitoring & Alerts
- Notify Team