Continuous Integration ,Continuous Delivery & Continuous Deployment

Introduction : To develop the UdaPeople product, a revolutionary concept in Human Resources which promises to help small businesses care better for their most valuable resource: their people.

Continuous Integration: Merging all Developer's developed code into code repository like GITLAB, Bitbucket.

- All developers will work on the development activity and will commit the code to the code repository to different feature branches. By doing so, we avoid integration challenges.
- The developer's changes are validated by creating a build running automated tests against the build. By this we avoid cost which results in finding the defects at earlier stage and "Less bugs in production and less time in testing“.

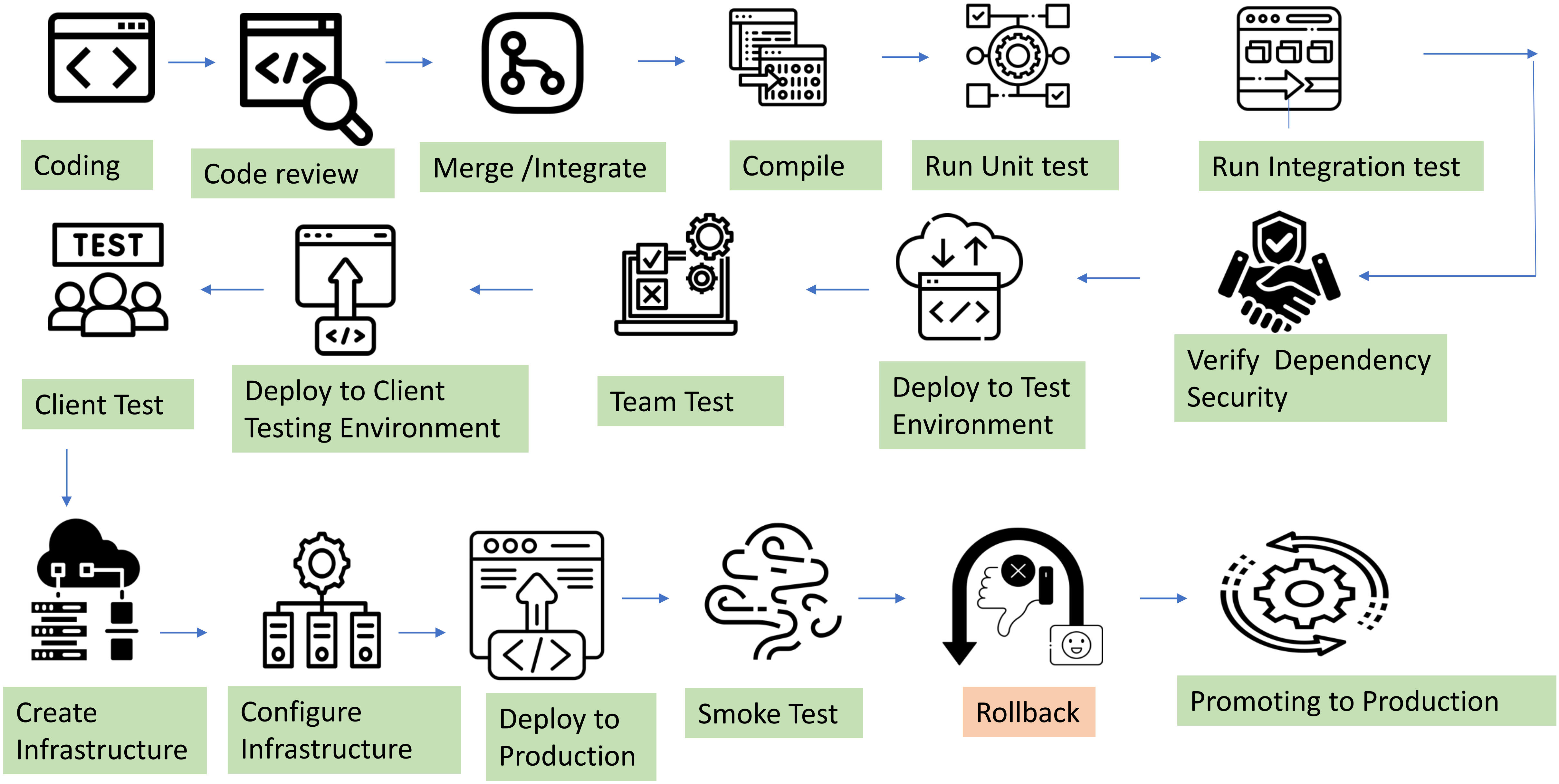
Continuous delivery :

- Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.

Continuous deployment

- Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

COMPLETE PICTURE OF CI/CD PROCESS



Coding

Code review

Merge /Integrate

Compile

Run Unit test

Run Integration test

Verify Dependency Security

Deploy to Test Environment

Team Test

Deploy to Client Testing Environment

Client Test

Create Infrastructure

Configure Infrastructure

Deploy to Production

Smoke Test

Rollback

Promoting to Production

Benefits of CI/CD:

- **Less developer time on issues from new developer code:**

For Continuous integration we need a source code repository where have all the developed code with history of changes or version of code, so that we wont lose any developed code with new features .we avoid integration challenges

- **Less bugs in production and less time in testing:**

The developer's changes are validated by creating a build running automated tests against the build.

by this we avoid cost which results in finding the defects at earlier stage

- **Prevent embarrassing or costly security holes:**

Application Security plays an important role and running vulnerable test will prevent us form costly security holes.

- **Less human error, Faster deployments:**

creating infrastructure is the vital role in CI/CD, where we create an infrastructure to deploy the application where new feature is rolled for customers . Here without human intervention we can create infrastructure which is less human error and results in faster deployments

- **Less infrastructure costs from unused resources:**

we can go with cloud where we pay for what we use. So that we can purchase servers based on the demand and load which will result in less infrastructure cost.

- **New value-generating features released more quickly:**

We can deploy and test the applications without human intervention where the new features are released to customers which will generate revenue. And take less time to market.

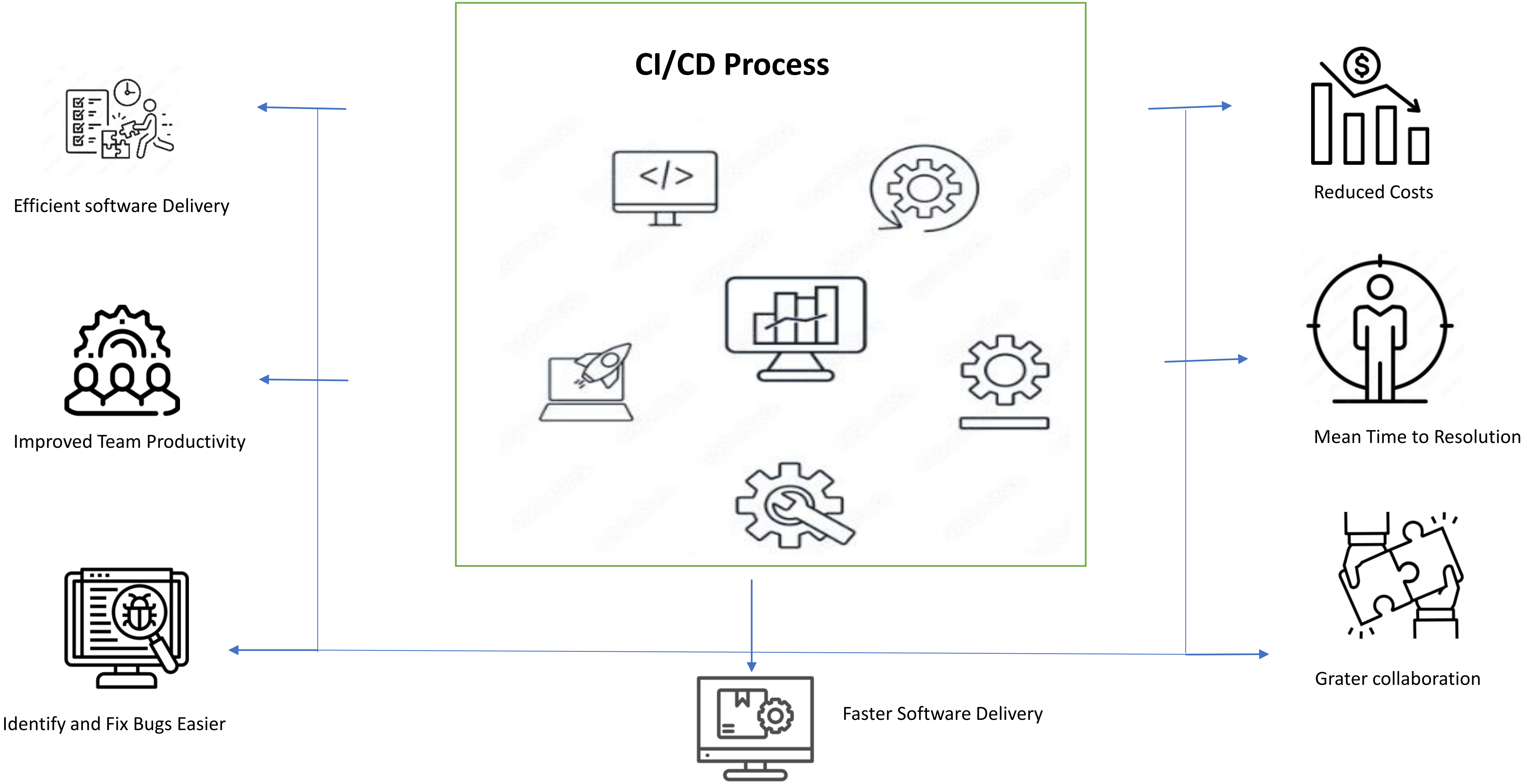
- **Reduced downtime from a deploy-related crash or major bug:**

smoke tests will help us to reduce the down time from deploying-related crashes or major but in the application which results to save project revenue

- **Quick undo to return production to working state:**

If there is any failure in the production deployment we will go to previous version of production code base , so that the down time to customers are less which saves project revenue.

BUSINESS BENEFITS OF CI/CD



CI/CD Process

Efficient software Delivery

Improved Team Productivity

Identify and Fix Bugs Easier

Faster Software Delivery

Reduced Costs

Mean Time to Resolution

Grater collaboration

Removing *Human* Error

- Before we implement CI/CD almost everything requires human intervention. Can you imagine a world without human error? Neither can I, but with CI/CD, we can reduce it!

Stage	Before CI/CD	After CI/CD
Coding	Human	Human
Code Review	Human, Subjective, Inconsistent	Human/CI - Static Analysis
Compile/Lint	Human	CI
Merge/Integrate	Human	CI
Run Unit Tests	Human, Hit or Miss, Easily Bought Off with Pressure	CI
Run Integration Tests	Human, Hit or Miss, Easily Bought Off with Pressure	CI
Verify Dependency Security	Human, Often Not Done	CI
Deploy to Test Env	Human, Problematic, Missed Steps	CD
Team Test	Human, Time Consuming	CD - Automated Acceptance Tests
Deploy to Client Test Env	Human, Problematic, Missed Steps	CD
Client Test	Human, Often Unnecessary If Pre-Development Activities are On Point	Human - Maybe Not Needed If We Can Build Confidence
Create Infrastructure	Human, Problematic, Missed Steps, Stressful	CD
Deploy to Production	Human, Problematic, Missed Steps, Stressful	CD
Smoke Test in Prod	Human, Inconsistent	Automated Smoke Tests
Rollbacks	Human, Problematic, Missed Steps, Stressful	CD
Promoting Production	Human, Problematic, Missed Steps, Stressful	CD
Celebrate!	Human	Human